UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Luiz Fernando Carvalho

Um Ecossistema para Detecção e Mitigação de Anomalias em Redes Definidas por Software

Campinas

2018

UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Luiz Fernando Carvalho

Um Ecossistema para Detecção e Mitigação de Anomalias em Redes Definidas por Software

Tese apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Doutor em Engenharia Elétrica, na área de Telecomunicações e Telemática.

Orientador: Prof. Dr. Leonardo de Souza Mendes

Coorientador: Prof. Dr. Mario Lemes Proença Junior

Este exemplar corresponde à versão final da tese defendida pelo aluno Luiz Fernando Carvalho, e orientada pelo Prof. Dr. Leonardo de Souza Mendes

Campinas

2018

Agência(s) de fomento e nº(s) de processo(s): Não se aplica.

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Rose Meire da Silva - CRB 8/5974

C253e

Carvalho, Luiz Fernando, 1989-
Um ecossistema para detecção e mitigação de anomalias em redes definidas por software / Luiz Fernando Carvalho. – Campinas, SP : [s.n.], 2018.

Orientador: Leonardo de Souza Mendes.
Coorientador: Mario Lemes Proença Junior.
Tese (doutorado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Rede de computadores. 2. Detecção de anomalias. 3. Redes definidas por software (Tecnologia de rede de computador). I. Mendes, Leonardo de Souza, 1961-. II. Proença Junior, Mario Lemes. III. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. IV. Título.

Informações para Biblioteca Digital

Título em outro idioma: An ecosystem for anomaly detection and mitigation in software-defined networking
Palavras-chave em inglês:
Computer networks
Anomaly detection
Software defined-Networking
Área de concentração: Telecomunicações e Telemática
Titulação: Doutor em Engenharia Elétrica
Banca examinadora:
Leonardo de Souza Mendes [Orientador]
Alexandre de Aguiar Amaral
Elieser Botelho Manhas Junior
André Marcelo Panhan
Gean Davis Breda
Data de defesa: 16-08-2018
Programa de Pós-Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE DOUTORADO

Candidato: Luiz Fernando Carvalho **RA:** 160966
Data de Defesa: 16 de agosto de 2018

Título da Tese: Um Ecossistema para Detecção e Mitigação de Anomalias em Redes Definidas por *Software*

Prof. Dr. Leonardo de Souza Mendes (Presidente, FEEC/UNICAMP)
Prof. Dr. Alexandre de Aguiar Amaral (IFC)
Prof. Dr. Elieser Botelho Manhas Junior (UEL)
Prof. Dr. André Marcelo Panhan (IFSP)
Dr. Gean Davis Breda (Consultoria)

A ata de defesa, com as respectivas assinaturas dos membros da Comissão Julgadora, encontra-se no processo de vida acadêmica do aluno.

Agradecimentos

A Deus, primeiramente, por tudo que me proporciona. As palavras são insuficientes para expressar minha gratidão.

Aos meus pais, Maria e Altair, que sempre priorizaram meus estudos e nunca pouparam esforços para que eu atingisse os meus objetivos.

Ao meu orientador Prof. Leonardo de Souza Mendes pela confiança, orientação e pela oportunidade concedida.

Ao meu co-orientador Prof. Mario Lemes Proença Junior, grande responsável pela minha escolha em seguir a carreira acadêmica. Obrigado por mostrar e ser um exemplo de que quem trabalha duro e se dedica sempre colhe frutos. Agradeço também pela confiança e incentivo.

A minha amiga e namorada Jeniffer pela paciência, apoio e cumplicidade ao longo desse processo.

Aos amigos do grupo de rede da UEL: Marcos, Gilberto, Anderson e Eduardo. Mesmo que distantes, sempre estiveram presentes ao longo deste trabalho. Acredito que a superação de cada deadline, de cada noite em claro, de horas e horas de reunião não chegou ao fim. Fazemos o que fazemos. Fomos treinados para fazer. Gostamos de fazer.

Aos novos integrantes do grupo de redes da UEL, Cinara e Matheus. A troca de conhecimento mútuo foi muito importante.

Ao Prof. Alexandre de A. Amaral que, antes mesmo de me tornar calouro na graduação, foi sempre solícito. Saiba que estou seguindo seus passos, pois você escolheu o melhor caminho.

Agradeço a todos os professores que, direta ou indiretamente, contribuíram para a construção deste trabalho. Não posso deixar de mencionar os professores Bruno B. Zarpelão, Sylvio Barbon e Taufik Abrão. As conversas e conselhos certamente me tornaram um professor e pesquisador melhor.

Resumo

A massificação dos recursos computacionais e o aumento das taxas de transmissão das redes de comunicação têm contribuído para o surgimento de serviços cada vez mais sofisticados. Mais do que nunca, as novas tecnologias precisam apresentar soluções dinâmicas e extensíveis para se adequarem às necessidades dos usuários. Entretanto, essa demanda acrescenta também dificuldades ao gerenciamento da rede, tornando essa atividade mais complexa e onerosa. Nesse contexto, Redes Definidas por *Software* (*Software-defined Networking*, SDN) surgem como um novo paradigma que propicia maior controle sobre a infraestrutura de rede e sobre o comportamento de cada equipamento que a compõe. Embora a SDN forneça maior controle sobre o fluxo de tráfego, sua segurança e garantia da disponibilidade de seus serviços permanecem um desafio. Dessa maneira, nesta tese é apresentado um ecossistema baseado em SDN voltado à solução desses problemas. A proposta monitora o tráfego de rede e detecta proativamente anomalias que podem prejudicar o funcionamento adequado da rede. Além da identificação dos eventos anômalos, a abordagem proposta também conta com rotinas que propiciam a mitigação automática dos efeitos das anomalias. O ecossistema foi submetido a diversos testes para comprovação de sua eficiência. Os resultados demonstram que o ecossistema proposto consegue alcançar taxas de detecção mais altas em comparação com outras abordagens. Além disso, a análise de desempenho mostra que a abordagem pode contribuir eficientemente para a resiliência e disponibilidade da rede gerenciada.

Palavras-chaves: Redes de computadores; Detecção de anomalias; Redes Definidas por *Software*.

Abstract

The popularization of computing resources and the increase in the transmission rates of communication networks have contributed to the emergence of increasingly sophisticated services. More than ever, new technologies need to present dynamic and extensible solutions to fit the needs of users. However, this demand also adds difficulties to the network management, making this activity more complex and costly. In this context, Software-defined Networking (SDN) appears as a new paradigm that provides greater control over the network infrastructure and the behavior of each equipment that compose it. Although SDN provides greater control over the flow of traffic, its security and assurance of the availability of its services still challenging. In this manner, this thesis presents an ecosystem based on SDN aimed to solve these problems. The proposal monitors network traffic and proactively detects anomalies that may impair the proper functioning of the network. Besides the identification of the anomalous events, the proposed approach also has routines that allow automatic mitigation of the anomalies effects. The ecosystem has undergone several tests to prove its efficiency. The results demonstrate that the proposed ecosystem can achieve higher detection rates compared to other approaches. In addition, performance analysis shows that the approach can contribute effectively to the resilience and availability of the managed network.

Keywords: Computer networks; Anomaly detection; Software defined-Networking.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ACO	*Ant Colony Optimization*
ACODS	*Ant Colony Optimization for Digital Signature*
API	*Application Programming Interface*
ARIMA	*Autoregressive Integrated Moving Average*
ARP	*Address Resolution Protocol*
ASIC	*Application Specific Integrated Circuit*
AUC	*Area Under the Curve*
BYOD	*Bring Your Own Device*
CkNN	*Correlational k-Nearest Neighbor*
CPU	*Central Process Unit*
DDoS	*Distributed Denial of Service*
DoS	*Denial of Service*
ForCES	*Forwarding and Control Element Separation protocol*
FTP	*File Transfer Protocol*
HIDS	*Host-based Intrusion Detection System*
ICMP	*Internet Control Message Protocol*
IDS	*Intrusion Detection System*
IETF	*Internet Engineering Task Force*
IP	*Internet Protocol*
IPFIX	*Internet Protocol Flow Information Export*
IPv4	*Internet Protocol version 4*
IPv6	*Internet Protocol version 6*
ISO	*International Organization for Standardization*

ISP	*Internet Service Provider*
MAC	*Media Access Control*
MIB	*Management Information Base*
MPLS	*Multiprotocol Label Switching*
NBI	*Northbound Interface*
NFV	*Network Functions Virtualization*
NIDS	*Network-based Intrusion Detection System*
NMSE	*Normalized Mean Square Error*
ONF	*Open Networking Foundation*
OVS	*Open vSwitch*
OVSDB	*Open vSwitch Database*
OXM	*OpenFlow Extensible Match*
PCA	*Principal Component Analysis*
POF	*Protocol Oblivious Forwarding*
PTT	Ponto de troca de tráfego
QoS	*Quality of Service*
RF	*Random Forest*
RFC	*Request for Comments*
RLM	Regressão Logística Multinomial
RoC	*Receiver Operating Characteristic*
SBI	*Southbound Interface*
SDN	*Software-defined Networking*
SNMP	*Simple Network Management Protocol*
SOM	*Self-Organizing Maps*
SVM	*Support Vector Machine*

TCAM	*Ternary Content-Addressable Memory*
TCP	*Transmission Control Protocol*
TLS	*Transport Layer Security*
TTL	*Time to live*
UDP	*User Datagram Protocol*
VLAN	*Virtual Local Area Network*

Lista de símbolos

Ω	Conjunto de informações do tráfego enviadas para o controlador
F_t	6-tupla contendo os atributos do tráfego analisados no intervalo t
$H(\cdot)$	Entropia de Shannon
$\mathbf{d}$	Assinatura digital do tráfego
J	Função objetivo usada no ACODS
$\mathcal{C}$	Conjunto de *cluster*
$\mathcal{G}(\mathcal{V}, \mathcal{E})$	Grafo $\mathcal{G}$ com conjunto de vértices $\mathcal{V}$ e arestas $\mathcal{E}$
$\mathbf{x}_i$	i-ésima 6-tupla do tráfego histórico
K	Número de *clusters*
$\bar{\mathbf{x}}^{(j)}$	centro (centróide) do cluster C_j
$dist\,(\mathbf{x}_i, \bar{\mathbf{x}})$	Distância Euclidiana entre $\mathbf{x}_i$ e $\bar{\mathbf{x}}_i$
τ	Trilha de feromônio
L	Número de soluções mais promissoras criadas a cada iteração
ρ	Taxa de evaporação do feromônio
I	Número total de iteração
E	Total de elementos a serem clusterizados
δ	Valor da medida Índice Dunn
$s\,(i)$	Valor da medida Silhouette para a amostra i
Θ_t	Erro relativo entre o tráfego esperado e a atual medição no instante t
w	Tamanho da janela analisada
$\mathbf{b}$	Vetor de pesos associados a cada atributo de fluxo

Sumário

1 Introdução

Embora a conveniência fornecida pelas redes de comunicação instigue continuamente a demanda por serviços cada vez mais sofisticados, a capacidade da Internet está rapidamente se tornando insuficiente para os padrões de tráfego resultantes das novas modalidades de serviços emergentes [1]. Destaca-se o crescimento do número de dispositivos móveis conectados à rede devido à Internet das Coisas (*Internet of Things*, IoT), computação ubíqua e o surgimento de novas tecnologias como Internet Tátil (*Tactile Internet*). Estima-se que entre 20 e 46 bilhões de dispositivos estejam conectados à Internet até 2020 [2]. Além disso, é previsto que o tráfego gerado mensalmente em 2021 seja o triplo do observado no mesmo período em 2016 [3]. Nesse cenário, apesar dos investimentos frequentes nas tecnologias de comunicação, a insatisfação com a capacidade das redes tradicionais em adaptar-se às mudanças exigidas por essas tecnologias se tornará cada vez mais evidente [4].

A arquitetura de rede tradicional utilizada atualmente é a mesma que foi desenvolvida há algumas décadas, criada para comunicação entre clientes e servidores, em uma época cuja quantidade de tráfego e de conexões não eram preocupações fundamentais [5]. No entanto, as redes devem ser cada vez mais rápidas e flexíveis para suportar as necessidades de clientes inseridos em ambientes altamente dinâmicos com mudanças constantes de aplicações e equipamentos [4]. Isso fez com que as redes se tornassem sistemas complexos, constituídos por diversos elementos heterogêneos que mantêm uma interação ininterrupta entre si.

A estrutura rígida torna o gerenciamento das redes uma tarefa árdua à medida que elas são ampliadas. Parte disso decorre do fato de que, para expressar as políticas dos níveis de serviços acordados e para a manutenção de QoS (*Quality of Service*), operadores de rede necessitam configurar cada dispositivo separadamente, como uma coleção de diferentes *switches*, roteadores, *middleboxes*[1], entre outros, usando comandos de baixo nível, específico de cada fornecedor de equipamentos [7]. Mudanças em tais políticas, geralmente, exigem que a equipe de gerência concentre esforços em fazer diversas alterações manualmente nos componentes de rede. Por utilizar pessoal especializado e requerer tempo considerável para realização dessa tarefa, é, portanto, difícil de gerenciar políticas em um ambiente de redes que sofrem constantes mudanças e inovações [8].

Nesse contexto, Redes Definidas por *Softwares* (do inglês, *Software-defined*

[1] Qualquer dispositivo que transforma, inspeciona, filtra ou manipula o tráfego para outros fins que não o encaminhamento de pacotes [6].

Networking – SDN) surgem como uma arquitetura alternativa e promissora, permitindo flexibilidade e escalabilidade sem precedentes, tanto na configuração quanto na implantação de serviços. Embora SDN seja proposta como candidata à arquitetura da próxima geração da Internet, empresas como Google já a adotaram em seus *data centers* internos [9]. A maior característica desse novo paradigma é a desvinculação do plano de dados em relação ao plano de controle da rede, além da transferência desse último para uma aplicação centralizada, chamada controlador. O plano de controle é responsável pela inteligência da rede, ou seja, define rotas e regras de manipulação dos fluxos de dados transmitidos. O plano de dados tem a função de encaminhar as informações até o seu destino, seguindo as características definidas pelo plano de controle. Com essa separação, vários dispositivos de rede podem compartilhar o mesmo controlador. Uma grande vantagem é que, caso ocorra a necessidade de uma mudança de política ou o estabelecimento de qualidade de serviço, somente o plano de controle deverá ser alterado e não mais cada equipamento da rede [10].

Infelizmente, o crescimento das redes tem sido acompanhado pelo aumento de anomalias e a arquitetura SDN não é invulnerável a essas ameaças. Grande parte dos trabalhos da literatura é destinada à detecção de anomalias em que o controlador é alvo de massivos ataques de negação de serviço, oriundos de agentes externos à rede [11, 12, 13]. Ainda, para contenção dos efeitos desse ataque, regras simples são instauradas no *switch* de entrada da rede, com o intuito de bloquear o tráfego proveniente de fontes suspeitas [14] ou interromper completamente a comunicação com o serviço comprometido [15, 16]. Uma desvantagem observada em tais abordagens deriva do processo de mitigação dos ataques, em que regras de encaminhamento, fixadas por operadores humanos, tendem a prejudicar os usuários por conta de sua inflexibilidade e falta de adaptação em relação ao comportamento normal do tráfego. Tais soluções geralmente limitam a taxa de requisições destinadas ao controlador, excluindo aquelas consideradas excedentes, o que também acarreta na eliminação de requisições legítimas.

Outra limitação das abordagens anteriormente citadas é o pressuposto de que a rede está isenta de comportamentos maliciosos provenientes de seu tráfego interno. Um *host* ou *switch* SDN comprometido pode contribuir para uma variedade de ataques. Algo que pode levar a essa situação é o chamado *Bring Your Own Device* (BYOD), uma tendência recente advinda do ambiente de negócios em que, na busca pelo aumento da produtividade dos empregados e redução dos custos de tecnologia da empresa, é permitido aos funcionários utilizarem seus dispositivos pessoais para acessarem os recursos da empresa para o trabalho [17]. Um dispositivo infectado pode disseminar aplicações maliciosas para outros equipamentos, usando as falhas presentes nos *softwares* do sistema ou até mesmo prejudicar a infraestrutura da rede. Por exemplo, um atacante pode preencher

erroneamente a tabela de encaminhamento de um *switch* para informar falsas topologias (lógicas ou físicas) ou deixar o controlador em um estado imprevisível [18]. Além disso, mensagens de controle, manipuladas pelo atacante, podem tornar o plano de dados instável e até mesmo desconectá-lo do plano de controle, inutilizando parte ou toda a rede. Esses ataques já foram demonstrados em trabalhos anteriores [19]. Dispositivos SDN comprometidos também podem levar a uma variedade de ataques do tipo *man-at-the-end* (MATE), ARP-*Poisoning* e sequestro do tráfego (*Hijacking*).

Juntamente com as mudanças introduzidas pelo paradigma SDN, faz-se necessário adaptar o modelo de gerenciamento aplicado atualmente a fim de garantir confiabilidade, resiliência e disponibilidade da rede. Para tanto, primeiramente, é necessário que o comportamento do tráfego de rede seja constantemente monitorado, visando uma detecção antecipada das divergências em relação ao comportamento esperado, possibilitando uma rápida tomada de decisões para solução de problemas. O monitoramento completo das redes de larga escala é uma tarefa quase impraticável para ser realizada de forma manual por um administrador de rede. As altas velocidades de transmissão, aliadas ao crescente número de segmentos monitorados, tornam essa tarefa ainda mais complicada. Dessa maneira, um sistema de monitoramento de tráfego destinado à detecção de anomalias deve operar de forma proativa, evitando a interrupção dos serviços prestados, de forma a assegurar que a rede não sofra solução de continuidade.

O objetivo principal desta tese é a criação de um ecossistema capaz de auxiliar o administrador na gerência da segurança de redes SDN. A proposta é destinada à detecção de eventos anômalos que violam políticas e objetivos definidos pela gerência, disparando medidas apropriadas para solucioná-las de forma automática. Seu uso justifica-se pela mínima intervenção do administrador de rede, propiciando maior agilidade ao processo de detecção e autorreparo, diminuição do capital destinado a manter uma equipe especializada e redução de erros que podem ser inseridos em razão da operação humana nesse ambiente.

O ecossistema é dividido em quatro módulos fundamentais e complementares. O primeiro é responsável pela coleta e armazenamento dos fluxos do tráfego por meio do protocolo OpenFlow, o qual tem sido amplamente usado para coletar estatísticas de fluxo de rede de forma passiva ou medir, de forma ativa, a latência mediante a injeção de pacotes de análise no tráfego. O segundo módulo caracteriza o perfil do tráfego com objetivo de criar uma assinatura digital do comportamento normal do tráfego. Nesse módulo também ocorre a detecção de anomalias e, para isto, é realizada a comparação dos fluxos coletados em tempo real com o perfil do tráfego previamente estabelecido. O terceiro módulo tem o objetivo de mitigar o efeito da anomalia na rede, isto é, neutralizar os eventos prejudiciais. Por fim, no último módulo são gerados relatórios sobre ataques identificados para fins de

validação e auditoria, em que os resultados da atuação dos módulos são utilizados como avaliador do desempenho do ecossistema, propiciando a melhoria contínua dos parâmetros usados na detecção e mitigação.

As principais contribuições dessa tese são:

- Desenvolvimento e aplicação do ecossistema para detecção de anomalias em redes SDN;
- Solução para diagnóstico de anomalias em tempo real;
- Formalização de um mecanismo de detecção de anomalias do tipo DDoS, *port scan*, e *flash crowd*, em tempo real, baseada na regressão logística multinomial e na assinatura normal do tráfego em redes SDN;
- Acionamento automático de procedimentos de mitigação em resposta aos eventos anômalos detectados;
- Testes para validação do modelo proposto em ambientes usando controladores SDN POX e Floodlight.

O restante deste trabalho está organizado da seguinte maneira: o capítulo 2 apresenta fundamentos referentes à área de gerência de redes, bem como o estudo de detecção de anomalias. Os conceitos e terminologias que permeiam a SDN são abordados no capítulo 3. O capítulo 4 apresenta trabalhos relacionados ao tema de pesquisa desenvolvido. O capítulo 5 detalha a proposta de detecção de anomalias desta tese. O processo de avaliação e os resultados obtidos com a solução proposta são discutidos no capítulo 6. Por fim, o capítulo 7 apresenta as considerações finais acerca do trabalho e as futuras direções de pesquisa.

2 Detecção de Anomalias em Redes de Computadores

A ideia de detecção de anomalias é amplamente difundida em várias áreas de pesquisas, pois proporciona a criação de mecanismos capazes de identificar eventos – geralmente críticos - que exigem contramedidas imediatas. De acordo com Chandola *et al.* [20], anomalias podem ser genericamente definidas como padrões em dados que não estão em conformidade com uma noção bem definida de normalidade; por exemplo, o comportamento anômalo em transações de cartões de crédito pode indicar atividades fraudulentas; leituras anômalas de um sensor de uma nave espacial podem significar uma falha em algum componente e imagens de ressonância magnética podem indicar a presença de problemas relacionados à saúde [21].

As redes ainda podem ser afetadas por eventos incomuns ou inesperados, mesmo com os constantes investimentos e esforços rumo à modernização. Tais eventos podem estar ligados à complexidade de gerenciamento e configuração, o que resulta em falhas e degradação do desempenho. Além disso, devido à sua vital importância nas transações comerciais e financeiras, interação social e demais serviços prestados, as redes frequentemente se tornaram alvos de atividades mal-intencionadas a fim de causar prejuízos às empresas e às pessoas. Esses eventos podem ser considerados como anomalias, pois configuram alterações atípicas do tráfego e do comportamento da rede [22].

2.1 Tipos de Anomalias

Uma das primeiras tarefas na detecção de anomalias é a identificação e definição corretas da declaração do problema. Isso significa que deve existir conhecimento prévio, mesmo que limitado, sobre o tipo de anomalia com que se deseja lidar. Existem vários tipos de anomalias no tráfego de rede, os quais podem ser categorizados respeitando duas propriedades relevantes: baseado em sua natureza e de acordo com o seu aspecto causal (distingue-se dependendo de sua causa, em relação ao aspecto malicioso ou não malicioso) [23].

2.1.1 Anomalias categorizadas por sua natureza

Um evento que difere do padrão pode ser considerado ou não uma anomalia dependendo do contexto no qual ele foi encontrado ou como ele ocorreu. Esse aspecto

pode direcionar como a análise manipulará e entenderá as anomalias detectadas. Com base em sua natureza, as anomalias podem ser categorizadas em três grupos [21, 24]:

- Anomalias pontuais: Uma anomalia pontual é o desvio de uma instância de dados individual em relação ao comportamento usual. Essas anomalias são as mais simples e, por isso, são o foco da maioria das pesquisas [20]. Como exemplo de anormalidade dessa natureza, pode-se citar um evento isolado em que um usuário tenta acessar um servidor restrito. Caso essa tentativa não esteja atrelada a um contexto específico ou a outros acontecimentos de natureza atípica, é uma anomalia pontual [25].

- Anomalias contextuais: Também chamadas de anomalias condicionais, são eventos considerados anômalos dependendo do contexto em que são encontrados. Dois conjuntos de atributos definem um contexto (ou a condição) para ser uma anomalia, sendo que ambos devem ser especificados durante a formulação do problema. Os atributos contextuais definem o contexto (ou ambiente); por exemplo, coordenadas geográficas em dados espaciais ou tempo em dados de séries temporais especificam a localização ou posição de uma instância, respectivamente. Por outro lado, os atributos comportamentais denotam os recursos não contextuais de uma instância, ou seja, indicadores que determinam se uma instância é ou não anômala no contexto [20, 26]. Considere uma série temporal que descreve a média de bits/s de tráfego em um conjunto de dias (atributo contextual), em que todos os dias, à 0h, o servidor faz um *backup* regular (atributo comportamental). Embora o *backup* gere um valor discrepante na série, ele pode não ser anormal, pois é um comportamento legítimo resultante do *backup* tipicamente realizado. No entanto, uma discrepância semelhante no volume do tráfego às 12h pode ser considerado uma anomalia contextual.

- Anomalias coletivas: Quando uma coleção de instâncias de dados se comporta de maneira anômala em relação ao conjunto de dados, esse grupo caracteriza uma anomalia coletiva.

As instâncias de dados individuais em uma anomalia coletiva podem não ser anomalias, a menos que ocorram continuamente por um período de tempo prolongado. Um exemplo de anomalia coletiva pode ser dado assumindo que um grupo de usuários interaja com um servidor Web, em que um usuário, que supostamente não possua os privilégios adequados, execute a seguinte sequência de operação: (i) acessa um diretório local no servidor; (ii) transfere um arquivo executável para esse diretório; e, (iii) executa um *script*. Enquanto o acesso dos usuários não privilegiados é restrito apenas aos diretórios públicos no servidor, para simplesmente ler e fazer transferência de arquivos não executáveis, as

atividades desse usuário em particular não estão em conformidade e indicam a ocorrência de uma anomalia coletiva.

Anomalias coletivas podem ocorrer apenas em conjuntos de dados cujas instâncias estão relacionadas, ao passo que anomalias pontuais ocorrem em qualquer conjunto de dados. Ademais, a incidência de anomalias contextuais depende da disponibilidade de atributos de contexto. Uma anomalia pontual ou uma anomalia coletiva também pode ser uma anomalia contextual se analisada em relação a um contexto específico [20].

2.1.2 Anomalias categorizadas por seu aspecto casual

O aspecto causal distingue as anomalias de acordo com a motivação ou circunstância em que elas ocorrem. Dessa maneira, as anomalias podem ser classificadas como maliciosas ou apenas eventos de comportamento não usual. Baseada nessas características, Barford *et al.* [27] e Marnerides *et al.* [28] esclarecem que uma anomalia nem sempre está relacionada a um ataque cujo objetivo é prejudicar os sistemas ou roubar informações. Para uma melhor compreensão, as anomalias podem ser agrupadas de acordo com quatro categorias.

O primeiro grupo é definido pelas anomalias operacionais, que surgem pela má configuração ou falha de equipamentos. Travamentos de servidores, falta de energia, congestionamento de tráfego quando o limite da infraestrutura é atingido, configuração inadequada de recursos ou mudanças significativas no comportamento da rede, causados pela imposição de limites de taxa de transferência ou adição de novos equipamentos são exemplos dessa categoria de anomalia [29]. Essas anomalias podem ser observadas por mudanças quase súbitas nos atributos que mensuram o volume do tráfego transmitido, por exemplo, bits por segundo [30].

O segundo grupo é constituído por anomalias geradas pelo uso legítimo dos recursos, mas que são discrepantes em relação à normalidade. Um evento típico dessa categoria é o *flash crowd*, o qual resulta em um grande volume de tráfego quando um número crescente de usuários tenta acessar um recurso específico da rede. Geralmente, esses eventos estão relacionados às requisições feitas a servidores Web [31]. *Flash crowd* podem ocorrer quando um resultado de concurso é publicado, quando um site de comércio eletrônico anuncia uma promoção ou até mesmo devido ao lançamento de um *software*.

Analogamente, *alpha flows* também caracterizam anomalias que não têm a intenção de afetar a operação normal ou comprometer a segurança da rede. A diferença entre *flash crowd* e *alpha flows* reside na quantidade de nós origem que tentam acessar o recurso de rede. Enquanto no *flash crowd* existe um grande número de fontes realizando requisições ao destino, no *alpha flow* esse volume de dados é transferido apenas de um nó

emissor para o destino. Embora esses eventos não sejam mal-intencionados, se não houver tempo suficiente para reagir e fornecer os recursos necessários para lidar com a demanda requisitada, podem levar a uma falha completa no serviço Web, impactando no acesso de outros recursos disponíveis, tais como correio eletrônico e *File Transfer Protocol* (FTP).

Anomalias de medição do tráfego compõem o terceiro grupo e também são exemplos de eventos que não têm necessariamente ligação com atividades maliciosas. Essas anomalias estão relacionadas a problemas com a infraestrutura de coleta ou com o próprio processo de aquisição dos dados. Exemplos incluem a perda de dados causada pela sobrecarga do equipamento monitorado e a falha na comunicação durante a coleta.

O quarto tipo de anomalia é conhecido como abuso de rede (ou ataque de rede). Cada anomalia é um conjunto de ações maliciosas que visam espionar, interromper, degradar ou destruir informações e serviços de sistemas de redes de computadores, comprometendo sua integridade, confidencialidade ou disponibilidade. Vários tipos e classes de ataques existentes atualmente podem variar de simples *spam* de e-mail a ataques de invasão em infraestruturas de redes críticas.

Na categoria de espionagem, dois métodos tradicionais podem ser usados para sondagem de uma rede e de seus dispositivos. O primeiro deles, *port scan*, é uma técnica usada para identificar portas abertas ou serviços disponíveis em um *host*. Uma vez que esse evento anômalo promove o reconhecimento de vulnerabilidades do dispositivo que sofrerá o ataque, geralmente é seguido de um ataque mais complexo. O segundo ataque dessa categoria, *network scan*, é mais abrangente e tem como objetivo descobrir dispositivos que estão ativos na rede, coletando suas informações, tais como o sistema operacional utilizado e os serviços oferecidos por eles [32].

Worms são programas maliciosos com capacidade de autorreplicação que se propagam pela rede e infectam dispositivos explorando falhas na segurança em serviços amplamente usados. *Worms* são anomalias que podem degradar o desempenho do sistema e da rede por meio da saturação de enlaces com transmissão de tráfego malicioso associado à sua propagação. Dessa maneira, é comum que os *worms* sejam notados pelo usuário apenas quando a replicação descontrolada consome recursos do sistema, retardando ou interrompendo outras tarefas [33]. Tradicionalmente, para que possa se autorreplicar e infectar outros dispositivos, esses *softwares* devem conhecer os possíveis hospedeiros e suas vulnerabilidades. Portanto, *worms* são exemplos de anomalias que sucedem uma ação de espionagem.

Com relação às anomalias que visam interromper um recurso oferecido pela rede, têm-se como protagonistas os ataques de negação de serviço. Como o próprio nome sugere, esses ataques têm como objetivo garantir que a infraestrutura da rede seja afe-

tada negativamente de alguma forma. Os ataques de negação de serviço não envolvem a invadisão do sistema alvo. Em vez disso, o atacante tenta esgotar os recursos disponíveis da rede ou serviço para que usuários legítimos não consigam obter acesso [34]. As vítimas de ataques dessa natureza geralmente são usuários que usufruem das funcionalidades de servidores Web de organizações como bancos, comércio, grandes corporações ou organizações governamentais. Embora esses ataques geralmente não resultem no roubo ou perda de informações significativas ou outros ativos, eles podem custar às vítimas muito tempo e prejuízos financeiros. *Denial of Service* (DoS) e *Distributed Denial of Service* (DDoS) são exemplos de ataques de negação de serviço [21, 31].

Os ataques do tipo DoS são desencadeados por uma única fonte atacante, responsável por gerar e enviar um grande número de requisições a um recurso de rede [35]. Por sua vez, os ataques DDoS aplicam uma abordagem distribuída para sua disseminação e, por conta disso, possuem duas fases durante sua execução. Durante a primeira fase, o atacante compromete máquinas da Internet e instala *softwares* especializados para que esses *hosts* contribuam no ataque. Dá-se o nome de zumbis às máquinas infectadas, uma vez que elas passam a agir sob o controle do atacante. Na segunda fase, cada *host* zumbi recebe ordens para iniciar o ataque e suas requisições serão somadas aos demais *hosts* infectados para sobrecarregar o alvo.

A atuação dos ataques de negação de serviço gera uma inundação (*flooding*) de requisições destinadas ao alvo. Por esse motivo, os principais representantes desse tipo de ataque são: SYN *flooding*, UDP *flooding* e ICMP *flooding*. O ataque SYN *flooding* opera sobre o método de estabelecimento de conexões do protocolo TCP (*Transmission Control Protocol*), ou seja, o *handshake* de três vias (3-*way handshake*). Durante o *handshake*, o primeiro passo é executado pelo cliente. Ele solicita a conexão com o servidor por meio de um mensagem SYN (*synchronization*). No segundo passo, o servidor confirma a solicitação respondendo à mensagem com um SYN-ACK (*acknowledge*) e aguarda a resposta do cliente. O terceiro passo finaliza o *handshake* quando o cliente envia um ACK ao servidor. Para a realização do ataque, o cliente não efetua o terceiro passo. Ao enviar a mensagem no segundo passo, o servidor reserva os recursos para o cliente e fica na espera por sua confirmação. Dada a quantidade finita de recursos, o servidor fica sobrecarregado quando um número elevado de conexões não são completadas. Esse ataque pode também utilizar endereços IP forjados (IP *spoofing*) para que o atacante não receba as mensagens SYN-ACK das suas falsas solicitações de conexão.

O ataque UDP *flooding* explora a forma de comunicação não orientada a conexão do protocolo UDP (*User Datagram Protocol*). Normalmente, o atacante sobrecarrega o alvo enviando-lhe pacotes IP com datagramas UDP para diferentes portas. Ao receber o pacote, o alvo busca por aplicações associadas a essas portas. Quando identifica que

nenhuma aplicação está aguardando pacotes na porta, envia ao emissor um pacote ICMP para informá-lo que o destino não pode ser alcançado. Quando efetuado em larga escala, o UDP *flooding* pode fazer com que o alvo gaste seus recursos apenas respondendo aos pacotes do ataque, impedindo que usuários legítimos possam ser servidos. Para não receber os pacotes ICMP como respostas das solicitações, nesse ataque pode-se também utilizar a técnica de mascarar o endereço IP dos pacotes enviados ao alvo.

No ataque ICMP *flooding* uma grande quantidade de mensagens ICMP do tipo *Echo Request* é recebida pelo alvo. Como resultado, o alvo responde utilizando a mensagem ICMP *Echo Replay*, ficando sobrecarregado quando grande quantidade de requisições simultâneas são direcionadas a ele. Outra consequência é que, devido ao tráfego de mensagens ICMP, o caminho entre o atacante e o alvo pode ficar congestionado.

2.2 Fonte de Dados

O progresso no desenvolvimento de tecnologias de redes implica na crescente complexidade do seu gerenciamento, aumentando a responsabilidade do administrador em detectar anomalias e problemas para que não causem nenhum impacto significativo sobre a qualidade ou a interrupção dos serviços fornecidos aos usuários. Com essa finalidade, a gerência de redes aborda várias questões para assegurar a confiabilidade, integridade e disponibilidade da comunicação. Reconhecendo essa situação, a organização internacional de normalização ISO (*International Organization for Standardization*) desenvolveu um padrão para o gerenciamento de redes. Segundo Hunt [36], esse modelo pode ser dividido em cinco áreas de atuação:

- Gestão de falhas: Necessária para detectar a deterioração das condições operacionais do sistema, erros em aplicações ou ainda falha de um componente de *hardware*. Essa área também pode incluir as atividades de isolamento e correções de falhas.
- Gestão de contabilidade: Desejável para registro do uso da rede por parte de seus usuários. Pode ser utilizada para organizar o uso de recursos para aplicações e operadores apropriados.
- Gestão de configuração: Tarefas ligadas a essa área devem manter registros de *softwares* e *hardwares*. Além disso, devem proporcionar informações sobre manutenção, inclusão e atualização de relacionamentos entre os dispositivos durante a operação da rede.
- Gestão de desempenho: Indispensável para testes, monitoramentos, bem como para fornecimento de estatísticas como vazão da rede, tempo de resposta e disponibilidade

de equipamentos. Estas informações também têm a finalidade de assegurar que a rede atue de acordo com a qualidade de serviço firmada pelos usuários.

- Gestão de segurança: Trata da integridade dos dados trafegados e também do funcionamento correto da rede. Para esse fim, agrega atividades como monitoramento constante dos recursos disponíveis e restrição de acesso para evitar o uso incorreto dos recursos pelos usuários legítimos e agentes sem autorização.

Ao levar em conta o pleno domínio das cinco áreas de gerenciamento, pressupõe-se que o gerente tenha informações suficientes dos eventos que acontecem na rede. Além das áreas destacadas anteriormente, a gerência de redes divide suas atividades em duas categorias distintas. A primeira é o monitoramento, responsável pela observação constante de todos os eventos da rede e tem como objetivo contabilizá-los. A segunda está relacionada ao controle que utiliza as informações do monitoramento para ajustar os parâmetros de rede e para garantir seu desempenho [37].

O monitoramento pode ser realizado de forma passiva, analisando apenas o conteúdo do tráfego da rede sem interferir no seu fluxo de dados. Em geral, essa abordagem consegue examinar uma grande quantidade de informações sem prejudicar a operação da rede. Por meio dessa técnica, podem ser extraídas informações em relação ao tráfego, como quantidade de bits e pacotes, ou até mesmo propriedades específicas tal qual dados do cabeçalho de pacotes transmitidos. Em contrapartida, no monitoramento ativo ocorre a inserção e análise de pacotes de teste na rede. Esses pacotes, denominados *probes* (sondas), são enviados por dispositivos que realizam medições fim a fim no intuito de conseguir informações e características do tráfego, tais como perda e atraso de pacotes. A dificuldade desse procedimento é adequar o volume de *probes* usadas para realização de testes e aferições desejadas sem o acometimento dos recursos de rede.

Após a escolha de qual abordagem de monitoramento deve ser aplicada, a definição da fonte de dados se torna um fator crucial. Um sistema de detecção de anomalias eficaz deve conter uma fonte de dados que forneça informações adequadas e precisas para o monitoramento das atividades da rede e que auxiliem no reconhecimento de comportamentos anômalos do tráfego.

O protocolo SNMP (*Simple Network Management Protocol*) é tradicionalmente uma das ferramentas mais utilizadas durante as últimas décadas para capturar informações do tráfego de rede. Sua ampla adoção é explicada pela sua simplicidade, por ser suportado por grande parte dos equipamentos de rede, além das diversas ferramentas disponíveis no mercado que o utilizam, dentre as quais pode-se mencionar Cacti[1] e NTop[2].

[1] https://www.cacti.net/
[2] https://www.ntop.org/

Esse protocolo tem uma estrutura cliente-servidor que atua sobre o protocolo UDP, embora o transporte sobre TCP também seja possível. Normalmente, o agente coleta, em tempo real, dados do dispositivo no qual ele reside. O gerente controla os agentes para acompanhar o estado dos recursos monitorados. Dessa forma, pode-se definir o gerente como uma entidade que assume o papel operacional de gerar requisições para recuperar e modificar informações, cabendo a ele manipular as repostas às requisições transmitidas pelos agentes. Já o agente é considerado o elemento responsável por receber as requisições, providenciar o conteúdo exigido e encaminhá-lo ao gerente. As informações trocadas entre o agente e o gerente são armazenadas na MIB (*Management Information Base*). Esse repositório dispõe de uma lista de variáveis, denominadas de objetos gerenciados, e os seus respectivos valores.

Os protocolos de exportação e análise de fluxos de pacotes surgiram como uma alternativa ao SNMP para a coleta de informações sobre o tráfego de rede. O desenvolvimento desses novos métodos de aquisição de informações se deu pelo surgimento de novos serviços e aumento da complexidade das redes. Esses fatores somados culminaram na necessidade de análise de novos elementos e atributos gerenciáveis que apresentassem de forma mais detalhada os eventos de redes, comportamento de aplicações e de usuários. Assim, fluxos de pacotes extrapolam as informações contidas nos objetos SNMP, agregando uma vasta gama de dados que propiciam ao administrador obter maior precisão no monitoramento e diagnóstico de anomalias.

2.2.1 Fluxos IP

A mais simples das abordagens de aquisição de dados da rede é a coleta de pacotes em uma interface de rede por meio de ferramentas como Wireshark[3] e tcpdump[4]. Esse método permite uma inspeção profunda do tráfego, uma vez que cada pacote transmitido na rede é armazenado integralmente. Entretanto, esse mecanismo não é parcimonioso, pois o processamento de todos os pacotes resulta em alta demanda computacional. Dessa maneira, a coleta de pacotes torna-se proibitiva em ambientes de redes com enlaces com grande capacidade de transmissão [28].

Uma solução para as desvantagens da captura de pacotes é a coleta e análise de fluxos IP. Ela utiliza agrupamentos de pacotes com sessões de transmissões unidirecionais e que compartilham propriedades em um determinado intervalo de tempo. Essas características compartilhadas incluem o protocolo de transporte, endereço de origem e destino e portas de origem e destino [38]. A cada agrupamento de pacotes é dado o nome de fluxo e, desde que possam ser identificados pelas características comuns, os fluxos podem ser

[3] https://www.wireshark.org/
[4] https://www.tcpdump.org/

relacionados a uma aplicação, equipamento ou usuário.

A coleta de fluxos IP em redes de computadores produz um monitoramento passivo e pode ser dividida em dois grupos principais: a coleta nativa e a adaptada. Na coleta nativa os dispositivos de rede tais como *switches* ou roteadores possuem suporte à geração e exportação de fluxos IP. Uma vez configurado, o dispositivo analisa o tráfego e realiza o processo de medição (análise e montagem de fluxos). Nesse processo, ao receber um pacote, o dispositivo verifica os atributos e, caso haja algum fluxo ativo com atributos semelhantes (IP, portas, protocolo, etc.), as informações do pacote são agregadas ao fluxo correspondente. Caso contrário, um novo fluxo é criado. Eventualmente, os fluxos são exportados para uma máquina coletora (*flow collector*), responsável pelo processamento e armazenamento dos fluxos recebidos. Os fluxos quando exportados possuem diversos atributos como os campos do cabeçalho IP, que compõem a identidade do fluxo, diferenciando-o dos demais; contadores de bytes e pacotes, entre outros.

A coleta adaptada ocorre quando os dispositivos de rede não possuem suporte à exportação de fluxos. Uma alternativa possível nessa situação é o espelhamento de uma porta desse equipamento para um servidor que trata os pacotes espelhados e transforma-os em fluxo e, logo em seguida, transmite o fluxo para a máquina coletora.

Qualquer que seja o modo de coleta, o processo de criação e manipulação do tráfego antes dos fluxos serem devidamente armazenados pela máquina coletora segue as três etapas seguintes:

- Observação: Captura dos pacotes que trafegam pelo dispositivo;
- Medição: Análise dos pacotes para criação dos fluxos que os representam;
- Exportação: Envio dos fluxos ao dispositivo coletor para armazenamento persistente.

Ao longo dos anos diversas tecnologias foram criadas no intuito de coletar informações baseadas em fluxos IP. A empresa Cisco Systems[5] foi a pioneira nesse aspecto por meio da introdução do protocolo proprietário NetFlow em 1996, o qual se tornou um padrão pela adoção em diversos dispositivos de vários fornecedores. Servindo para o mesmo propósito, o sFlow [39] foi apresentado pela empresa InMon[6]. O seu nome é o acrônimo de *Sampling Flow*, indicando que sua maior diferença em relação ao NetFlow é a capacidade de coletar o tráfego por meio de amostragem. Desde sua criação, o objetivo do sFlow é ser um protocolo simples e escalável, tendo sua aplicação voltada às redes que operam com taxas de transmissão elevadas. O IP *Flow Information Export* (IPFIX), definido na RFC 3917, foi padronizado pela IETF (*Internet Engineering Task Force*), baseado na

[5] https://www.cisco.com
[6] https://inmon.com

versão 9 do NetFlow [40]. Seu objetivo sempre foi a padronização da forma com que a exportação de fluxos IP deveria ser realizada. Por se tratar de um padrão, o IPFIX é flexível o bastante para ser utilizado em dispositivos de diferentes fabricantes, prezando pelo desempenho e funcionalidade.

O protocolo OpenFlow [41], largamente utilizado em redes definidas por *software*, pode ser utilizado para aquisição de estatísticas do tráfego, ainda que não tenha sido elaborado especificamente para coleta e exportação de fluxos. Assim, o monitoramento do tráfego em SDN é relativamente fácil, pois o controlador, uma entidade centralizadora que detém o controle da rede, mantém, sobre ela, uma visão global, sendo capaz de requisitar estatísticas sobre os fluxos de qualquer dispositivo a qualquer momento [42].

O uso do OpenFlow para monitoramento da rede veio acompanhado com o crescente interesse em SDN. A combinação dessas duas tecnologias foi um atrativo para Tootoonchian *et al.* [43] na concepção da ferramenta OpenTM, um sistema de estimação da matriz de tráfego. Com a proposta, os autores conseguiram representar o volume de tráfego transmitido entre os pares origem-destino na rede de maneira mais eficiente que as técnicas existentes de estimativa de matriz de tráfego em redes IP tradicionais. Trabalhos como [44] e [45] também conseguiram comprovar as capacidades OpenFlow por meio do monitoramento passivo. Deve-se notar, entretanto, que o escopo desses trabalhos é limitado a redes onde a manutenção de contadores de fluxo é perfeitamente tratável. Para ambientes de rede de larga escala, a aquisição de dados usando OpenFlow pode introduzir uma sobrecarga no controlador. Nesse caso, os protocolos próprios de exportação de fluxos são mais indicados [15]. Mais informações sobre o protocolo OpenFlow são apresentadas na seção 3.2.3.

2.3 Detecção de Anomalias e Intrusão

Os sistemas de detecção de intrusão (*Intrusion Detection System* – IDS) são mecanismos de defesa destinados ao monitoramento, detecção e análise de atividades maliciosas na rede ou em um *host*. Os IDS têm como objetivo, portanto, garantir a segurança, confiabilidade e disponibilidade dos recursos monitorados, podendo agir em conjunto com outros componentes que têm o mesmo propósito. Um *firewall*, por exemplo, é comumente, a primeira linha defensiva em uma rede e um IDS é usado quando há evidência de uma anomalia que o *firewall* não conseguiu interromper ou atenuar. O IDS funciona então como a segunda linha de defesa.

Embora o termo "intrusão" seja difundido no âmbito de segurança, os IDS nem sempre são desenvolvidos com o intuito de identificar intrusos. Muitas vezes eles são capazes apenas de reconhecer evidências de uma atividade que difere do comportamento

normal, seja durante ou após a sua ocorrência [20, 46]. Independente da natureza de uma anomalia, maliciosa ou não intencional, é de suma importância que o IDS tenha conhecimento sobre ela e produza um diagnóstico preciso, a fim de que medidas de mitigação possam ser aplicadas ou que o desenvolvimento de defesas mais efetivas seja realizado [47].

Os IDS podem ser classificados de diferentes maneiras. Dependendo de onde eles são implantados, três classificações são possíveis: *Network-based* IDS (NIDS), *Host-based* IDS (HIDS) e uma versão híbrida. Um NIDS é desenvolvido para a detecção de atividades anômalas no tráfego para proteger todos os nós que compõem a rede. Mais especificamente, o NIDS captura o tráfego em segmentos de rede específicos por meio de sensores e, posteriormente, analisa as atividades de aplicativos e protocolos para identificar incidentes suspeitos [48]. O tipo de anomalia mais comum detectada por essa abordagem é a pontual, embora as informações do tráfego também possam ser modeladas sequencialmente (*e.g.*, temporalmente) para o reconhecimento de anomalias coletivas [49]. Dentre as vantagens desse tipo de IDS, destaca-se a capacidade de monitorar o tráfego de rede de entrada e saída. Além disso, permite que uma grande rede possa ser monitorada com apenas alguns sensores instalados, desde que estejam bem posicionados.

Um HIDS é configurado para operar em *hosts* específicos, por exemplo, computadores pessoais ou servidores. Seu foco é monitorar eventos no *host* e detectar atividades suspeitas locais, ou seja, ataques realizados por usuários da máquina monitorada ou ameaças que podem comprometer o *host* em que o IDS opera. O HIDS avalia a segurança do *host* utilizando informações do *log* do sistema ou de aplicações, detecção de estouro de *buffer*, monitoramento de chamadas do sistema, uso indevido ou abuso de privilégios, entre outros [23]. Uma vantagem do HIDS é a possibilidade de detectar ataques baseados em protocolos criptografados. Dependendo de onde a criptografia reside dentro da pilha de protocolos, pode deixar um NIDS negligente a certos ataques. O IDS baseado em *host* não possui essa limitação. No momento em que o sistema de intrusão baseado em *host* analisa o tráfego de entrada, o fluxo de dados já foi descriptografado.

Nos IDS híbridos o objetivo é tornar a detecção de intrusão uma atividade consistente, robusta e equilibrada quanto à eficiência e desempenho. A combinação de NIDS e HIDS caracteriza IDS híbridos, cujo monitoramento é realizado tanto no tráfego de dados da rede como em informações coletadas dos próprios terminais.

Os sistemas de detecção de intrusão ainda podem ser classificados de acordo com o tipo de detecção realizada. Sob essa perspectiva, os IDS podem ser divididos em duas categorias: baseadas em assinatura e baseado em anomalias [21]. Um sistema de detecção baseado em assinaturas possui uma base de dados com a descrição das características dos eventos anômalos passíveis de serem detectados. O tráfego da rede é monitorado constantemente e quando uma situação semelhante a um registro desta base é

encontrada, uma possível ameaça é identificada [24]. Por outro lado, a detecção baseada em anomalias não necessita de conhecimento prévio sobre as características dos eventos a serem detectadas. Para tanto, essa abordagem constrói um perfil normal de tráfego, baseado em seu histórico, e as anomalias são identificadas a partir de desvios em relação a esse padrão [50, 51].

Ambas as metodologias, baseadas em assinatura e detecção de anomalia, apresentam suas vantagens e desvantagens. A detecção baseada em assinatura é eficiente na detecção de ameaças conhecidas e tende a gerar menor taxa de falsos positivos que as abordagens baseadas em anomalia [48, 52]. Sua principal desvantagem é a necessidade de que uma assinatura deve ser definida para cada possível investida que o atacante pode empregar contra a rede. Dessa maneira, esse método não é efetivo no reconhecimento de novos tipos de ataques ou variações de ataques conhecidos. Além disso, a base de dados contendo as assinaturas deve ser constantemente atualizada [53].

IDS que caracterizam o comportamento normal do tráfego apresentam duas grandes vantagens sobre os sistemas de detecção baseados em assinatura. A primeira diz respeito à capacidade de detectar ataques desconhecidos. Esta vantagem é devida à propriedade desses sistemas modelarem o funcionamento normal de uma rede e detectar desvios a partir dele [54]. A segunda vantagem é que os perfis de atividade normal são personalizados para cada rede, tornando difícil para um atacante saber quais as atividades que podem ser realizadas sem que ele seja detectado. No entanto, essa abordagem também apresenta algumas desvantagens. Aprender o comportamento normal não é uma tarefa trivial, já que a questão "normalidade" pode mudar constantemente, ainda mais em ambientes de redes. Além disso, os IDS que detectam anomalias podem apresentar altas taxas de falso positivos, bem como dificuldade de determinar qual evento específico desencadeou os indicativos de anomalias [24].

2.4 Considerações sobre o capítulo

Como ponto central deste capítulo, foram apresentados fundamentos e conceitos que compreendem a detecção de anomalias e intrusão em redes de computadores. Nesse âmbito, a Tabela 2.1 sintetiza o que foi discorrido sobre esse tema. Nela são listados os tipos de anomalias, caracterizados por sua natureza e causa; fontes de dados, discriminadas pelo tipo de protocolo de coleta; e tipos de sistema de detecção de anomalia e intrusão, categorizados pela sua localização e pela forma de detecção realizada.

Congestionamento, falhas, qualidade inaceitável de serviço e ataques maliciosos são exemplos de eventos anômalos, pois destoam do comportamento usual e geram a necessidade de medidas corretivas para amenizar ou corrigir os prejuízos provocados.

Tabela 2.1 – Resumo dos conceitos apresentados no capítulo.

Anomalias	Categorizadas por sua natureza	Anomalia pontual Anomalia contextual Anomalia coletiva
	Categorizadas por seu aspecto casual	Anomalia operacional Anomalia de uso legítimo Anomalia de coleta Anomalia de abuso de rede
Fonte de dados	SNMP	Objetos gerenciados (contadores)
	Coleta de pacotes	Pacotes inteiros
	Fluxos IP	Campos do cabeçalho compartilhado por uma sequência de pacotes transmitidos, juntamente com estatísticas dos pacotes dessa sequência
Detector de Anomalias	Localização	*Host-based* IDS (HIDS) *Network-based* IDS (NIDS) Híbrido
	Tipo de detecção	Baseado em assinatura Baseado em anomalias

Porém, antes que rotinas em resposta a esses eventos sejam disparadas, é preciso identificar corretamente o tipo de anomalia que se deseja solucionar. A fase de detecção das anomalias conta com o monitoramento constante do tráfego, o que por si só não é uma atividade simples. A complexidade dessa tarefa advém do próprio comportamento dinâmico da rede, definido pela interação de diferentes dispositivos, vários segmentos, diferentes configurações e recursos. Desse modo, a fim de auxiliar o processo de gerência de redes, tornando menos complexo o seu controle, diversas ferramentas e protocolos foram criados.

O desenvolvimento de técnicas e modelos que contribuíssem com a gerência das redes, durante muito tempo, foi quase exclusivamente focado sobre o protocolo SNMP. Desde a sua introdução, esse protocolo teve como objetivo fornecer maior visibilidade do tráfego da rede para os seus gerentes. Mesmo com a ampla aceitação, o SNMP foi confrontado ao longo das décadas com muitas mudanças no comportamento e uso das redes, grande parte ocasionada pelo surgimento de novas aplicações e serviços. Os protocolos que utilizam a análise de fluxos surgiram como uma alternativa, na qual os gerentes de rede poderiam obter informações para compreender melhor a composição do tráfego, não somente a utilização de um enlace ou de um número restrito de objetos SNMP.

Colocando em prática as cinco áreas de gestão propostas pela ISO, o gerente pode ter controle sobre a rede. Dessa forma, ele é provido com informações relevantes que

o auxilie no monitoramento constante do comportamento do tráfego, facilitando a análise e o reconhecimento dos mais variados eventos que prejudicam o funcionamento correto da rede. Entretanto, devido ao montante de informações conferidas a partir desse processo, é inviável que se realize diagnósticos manualmente e se crie relatórios sobre o estado da rede, pois esta tarefa requer cada dia mais recurso humano especializado e maior esforço para ser realizada.

O presente trabalho apresenta um NIDS que utiliza a abordagem de detecção de anomalias com o objetivo de auxiliar os administradores na gerência de redes. Mais especificamente, esta proposta tem a finalidade de reconhecer automaticamente desvios comportamentais de fluxos IP que condizem com eventos que possam afetar a segurança e integridade da rede, fazendo isso com alta taxa de acertos e com uma taxa de falsos alarmes reduzida. Além disso, rotinas de mitigação são empregadas para que as consequências das anomalias detectadas sejam contidas.

A proposta é idealizada para atuar em ambientes de redes definidas por *software*, um novo paradigma de comunicação. No próximo capítulo são apresentados conceitos e características que envolvem o funcionamento desse novo paradigma, assim como as diferenças em relação às redes tradicionais.

3 Software-defined Networking (SDN)

3.1 Abordagem tradicional de redes e origem da SDN

Nas redes tradicionais, grande parte da comunicação é realizada por roteadores e *switches*, que dispõem de *Application Specific Integrated Circuit* (ASIC), cujos *hardware* e *software* são projetados especificamente para o controle e transmissão de dados [55]. Isso torna as redes verticalmente integradas. O plano de controle, que decide como lidar com o tráfego da rede e o plano de dados, o qual encaminha o tráfego de acordo com as decisões estipuladas pelo plano de controle, são agrupados dentro do dispositivo de rede. O acoplamento desses planos foi bem aceito durante a fase de criação da Internet e nas décadas que sucederam. A ideia parecia ser a mais aplicável para manter a resiliência e o desempenho da rede [56]. Entretanto, a Internet desenvolveu-se e novas tecnologias de comunicação surgiram, como computação em nuvem, Internet das Coisas e virtualização de funções de rede (*Network Functions Virtualization* - NFV). Atrelado ao aparecimento dessas tecnologias, a necessidade de redes com maior largura de banda, maior acessibilidade e gerenciamento dinâmico se tornou uma questão crítica.

As redes tradicionais são centradas em *hardware*, que sofrem de deficiências significativas em relação à pesquisa e a inovações, fiabilidade, extensibilidade e flexibilidade. Por exemplo, a transição do IPv4 para IPv6, iniciada há mais de uma década, ainda continua sem ser efetivada completamente, mesmo sendo apenas uma atualização de protocolo [57]. Essa estrutura pouco flexível torna o monitoramento e configuração das redes uma atividade onerosa. Os gerentes precisam garantir que uma grande quantidade de dispositivos heterogêneos tenha alta disponibilidade e possa se comunicar para atender às necessidades dos clientes.

Além disso, uma vez que os dispositivos e *softwares* que os acompanham são soluções proprietárias, operadores e terceiros são impedidos de desenvolver novas funcionalidades que atendam às suas demandas sem serem restringidos por fornecedores de dispositivos. Essas características dificultam a implementação de tarefas de gerenciamento de rede. Para tentar contornar esse problema, pesquisas foram realizadas no intuito de investir em iniciativas que levassem à implantação de redes com maiores recursos de programação, de forma que novas tecnologias pudessem ser inseridas na rede gradualmente. A iniciativa mais bem-sucedida da flexibilização da rede é dada pela definição do protocolo OpenFlow. Usando esse protocolo, os dispositivos de encaminhamento podem ser manipulados por uma interface de programação simples que permite estender o acesso à lógica

de controle do *hardware*. Assim, o encaminhamento se mantém eficiente, pois a consulta à tabela de encaminhamento continua sendo realizada pelo *hardware*, mas a decisão sobre como o tráfego flui é transferida para um nível superior, onde diferentes funcionalidades podem ser implementadas. Essa estrutura permite que a rede seja controlada de forma extensível por meio de aplicações, expressas em *softwares* personalizados.

Antes do surgimento do OpenFlow, as ideias subjacentes à SDN (*Software-defined Networking*) enfrentavam um conflito entre a visão de redes totalmente programáveis e o pragmatismo que permitiriam a sua implantação no mundo real. O OpenFlow conseguiu um equilíbrio entre esses dois objetivos, permitindo que mais funções pudessem ser executadas comparado com o que controladores de rotas anteriores permitiam e, fazendo isso, em *hardware* já existente [58].

A junção dos planos de controle e de dados, bem como sua incorporação nos elementos de rede, torna difícil o desenvolvimento e a implantação de novos recursos como, por exemplo, algoritmos de roteamento e balanceamento de carga do tráfego. Essa dificuldade é pautada na necessidade de modificação do plano de controle de todos os dispositivos de rede por meio da instalação de novos *firmwares* e, em alguns casos, atualização de *hardware* [59]. É exatamente na natureza estática desses dispositivos, de não permitirem a configuração detalhada do seu plano de controle, que a rede definida por *software* se posiciona como uma alternativa promissora. O objetivo final da SDN é fornecer ao usuário um gerenciamento flexível das funções de encaminhamento dos elementos de rede [60]. Para tanto, a SDN centraliza a inteligência do plano de controle, a fim de que os dispositivos mantenham sua função de encaminhamento (plano de dados), mas permitam que suas ações (funcionalidades de comutação e roteamento) sejam guiadas pelo plano de controle gerenciado pelo administrador da rede.

3.2 Arquitetura SDN

O termo Redes Definidas por *Software* (do inglês, *Software-defined Networking* - SDN) surgiu para representar as ideias e estudos em relação ao OpenFlow na universidade de Stanford. Embora tanto SDN como OpenFlow tenham iniciado como experimentos acadêmicos, ganharam interesse da indústria ao longo dos anos. Muitos fabricantes de *switches*, como Acatel-Lucent, Cisco, IBM e Juniper Networks, já incorporaram a API (*Application Programming Interface*) do OpenFlow em seus equipamentos [61]. O potencial da SDN é tão grande que induziu um aglomerado de empresas formado por Google, Yahoo, Microsoft, Facebook, Verizon, entre outros, a fundarem a *Open Networking Foundation*[1] (ONF), cujo principal objetivo é promover a adoção da SDN através do

[1] https://www.opennetworking.org/

desenvolvimento de padrões abertos.

Os materiais ONF TR-502 [62] e TR-521 [63], elaborados pela ONF, indicam a composição da arquitetura SDN e como cada um de seus elementos deve interagir. Essa arquitetura pode ser caracterizada pelos seguintes aspectos:

- Os planos de controle e dados são desacoplados. A funcionalidade de controle é removida dos dispositivos de rede, que se tornarão simplesmente elementos de encaminhamento de pacotes. A lógica de controle é atribuída a uma entidade externa, chamada controlador SDN. O controlador é uma plataforma de *software* que fornece os recursos essenciais e abstrações para facilitar a programação dos dispositivos de encaminhamento. Seu objetivo é, portanto, semelhante ao de um sistema operacional tradicional;

- O controle da rede é logicamente centralizado. Nesse caso, o termo "logicamente" indica que o controle se comporta como uma única entidade, independente da sua possível implementação em forma distribuída. O princípio de controle centralizado sugere que os recursos da rede podem ser usados de forma mais eficiente quando vistos de uma perspectiva mais ampla, ou seja, a visão do todo. Um controlador SDN centralizado pode orquestrar recursos que abrangem uma série de entidades subordinadas (por exemplo, roteadores, *switches* e *hosts*) em diferentes sub-redes ou da rede toda. Assim, ele oferece melhor abstração para seus clientes do que se pudesse apenas abstrair subconjuntos de entidades subordinadas individuais;

- A rede é programável através de aplicativos em execução no controlador, o qual interage com os dispositivos do plano de dados. Essa é uma característica fundamental da SDN, considerada como sua principal vantagem em relação à arquitetura tradicional de rede;

- O quarto aspecto diz respeito à manutenção de interfaces públicas e abertas para contribuição da comunidade. Esse aspecto é relevante para manter a competitividade entre as soluções criadas, uma vez que é mais provável que interfaces não proprietárias tenham maior suporte.

Como definida originalmente, SDN se refere a uma arquitetura alternativa às redes tradicionais, cujo objetivo é melhorar a utilização de recursos, simplificar o gerenciamento, reduzir os custos operacionais e promover a evolução das tecnologias de interconexão. Com o uso da SDN, o administrador pode alterar as regras de encaminhamento do plano de dados quando necessário, priorizando ou mesmo bloqueando tipos específicos de tráfego. Isso é feito por meio da programação do plano de controle que, ao contrário da

configuração dos dispositivos, utiliza uma linguagem de alto nível, mais intuitiva [9]. Um benefício dessa característica é a criação de aplicações que podem reagir automaticamente às mudanças espúrias do estado da rede para assegurar que as políticas sejam mantidas. O desenvolvimento de serviços e aplicativos mais sofisticados pode ser simplificado por conta da centralização da lógica de controle em um controlador com conhecimento global do estado da rede. Desse modo, todas as aplicações podem tirar proveito das mesmas informações de rede providas pela visão global, levando às políticas mais consistentes e eficazes.

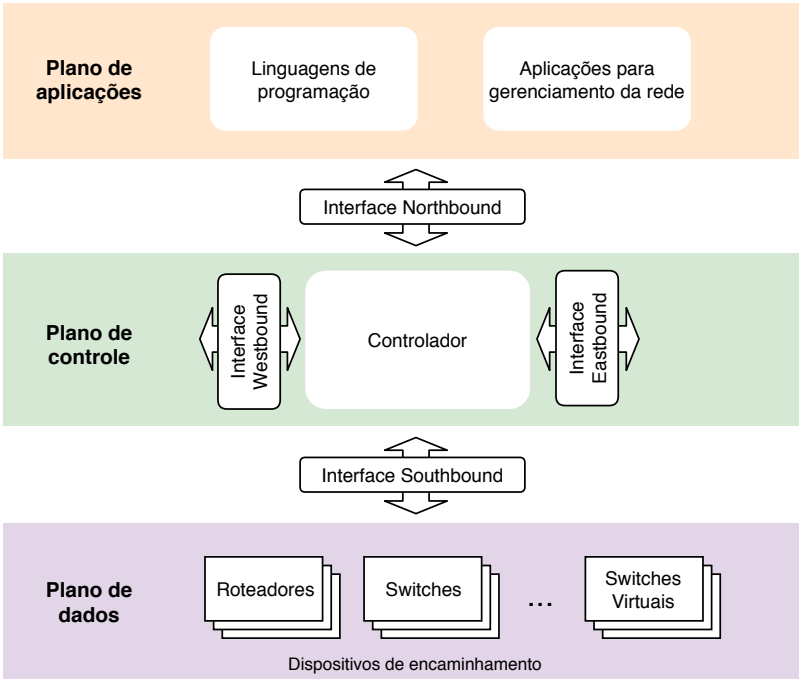

Figura 3.1 – Arquitetura SDN.

A arquitetura SDN pode ser representada por várias camadas, cada qual com sua função. Segundo Masoudi e Ghaffari [55], a estrutura SDN consiste em três partes principais. No nível mais baixo, encontra-se o plano de dados; no nível mais alto, o plano de aplicações, também conhecido como plano de gerência, e o plano de controle está entre eles. A comunicação entre o controlador e o plano de dados é mantida via *Southbound* interface (SBI), localizada nos *switches* SDN e a comunicação entre aplicações e o controlador é realizada pela *Northbound* interface (NBI), que está no plano de controle, como demonstrado na Figura 3.1. Usando a divisão entre os planos de controle e de dados, as aplicações seguem seu propósito específico, como segurança, *Quality of Service* (QoS), engenharia de tráfego e soluções para medição e monitoramento do tráfego.

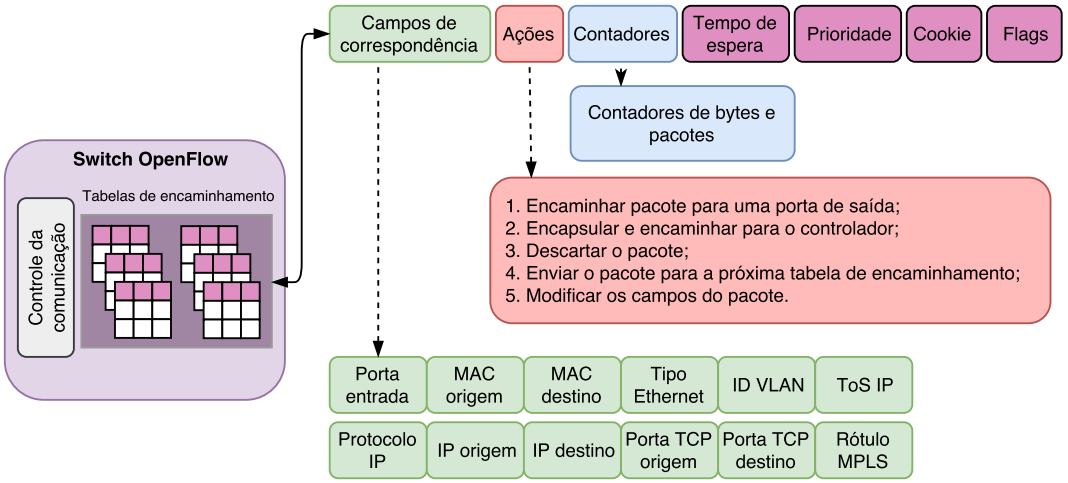

Figura 3.2 – Principais componentes de uma entrada de fluxo em uma tabela de fluxo.

3.2.1 Plano de dados

Existem dois elementos principais na infraestrutura SDN, o(s) controlador(es) e os dispositivos de encaminhamento. Os dispositivos de encaminhamento são *hardware* ou *softwares* com a propriedade de transmissão de pacotes de dados e, diferentemente do que ocorria nas redes tradicionais, não são capazes de tomar decisões autônomas sobre o roteamento do tráfego. Um aspecto importante é que essa infraestrutura é criada sob interfaces de padrões abertos (*e.g.*, protocolo OpenFlow), cruciais para manter a interoperabilidade entre os diferentes dispositivos do plano de dados, bem como a comunicação desses com o controlador.

O envio de pacotes continua sendo baseado nas tabelas de encaminhamento, armazenadas no próprio dispositivo. Regras de como pacotes específicos devem ser transmitidos podem ser acrescentadas, atualizadas e removidas por um controlador usando mensagens OpenFlow tanto de maneira reativa (em resposta a pacotes) quanto proativamente. Para que isso seja possível, o dispositivo deve fornecer suporte ao protocolo, caracterizando-o como um equipamento OpenFlow-*enabled* ou simplesmente *switch* OpenFlow. O caminho de dados dentro de um *switch* OpenFlow consiste de um *pipeline*, formado por uma ou mais tabelas de fluxos ligadas, que possuem entradas configuradas pelo controlador. Cada entrada de fluxo possui sete partes [64], como pode ser visto na Figura 3.2. Nela é apresentada a definição usada na versão 1.5.1 do protocolo OpenFlow, lançada em 2015. Nessa versão, cada entrada da tabela de encaminhamento contém:

- **Campos de correspondência** (*match fields*): usados para confrontar com as informações de identificação dos pacotes para decidir a ação a ser tomada sobre eles. Consiste na porta de ingresso no *switch* e cabeçalho de pacotes IP. Opcionalmente outros campos, como metadados fornecidos por outra tabela do *switch*, podem ser

utilizados;

- **Prioridade**: indica a precedência atribuída ao pacote que combina com os campos de correspondência. O pacote é comparado com entradas de fluxo na tabela de fluxo e somente a entrada de prioridade mais alta que corresponde ao pacote deve ser selecionada;
- **Contadores**: mantêm estatísticas dos pacotes que combinam com os campos de correspondência;
- **Instruções**: são aplicadas aos pacotes que combinam com os campos de correspondência;
- **Tempos de espera** (*Timeouts*): quantidade máxima (*hard timeout*) de tempo ou tempo ocioso (*idle timeout*), antes que o fluxo expire e seja removido da tabela de encaminhamento;
- ***Cookie***: pode ser usado pelo controlador para filtrar entradas na tabela de *switches*;
- ***Flags***: alteram a forma como as entradas de fluxo são gerenciadas. Por exemplo, quando a *flag* `OFPFF_SEND_FLOW_REM` está ativa, uma mensagem é enviada ao controlador para notificar a remoção da entrada correspondente.

Dentro de um *switch* OpenFlow, um caminho através de uma sequência (*pipeline*) de tabelas de fluxo define como os pacotes devem ser manipulados. A estrutura interna do *switch* é representada na Figura 3.3. Quando o *switch* recebe um pacote, ele executa as funções apresentadas na Figura 3.4. As informações do cabeçalho de pacote e porta de ingresso são comparadas com os campos de correspondência das entradas de fluxo da primeira tabela. Caso não seja encontrada uma correspondência, a busca procede para a tabela seguinte. Quando nenhuma regra é encontrada para esse pacote em todo o

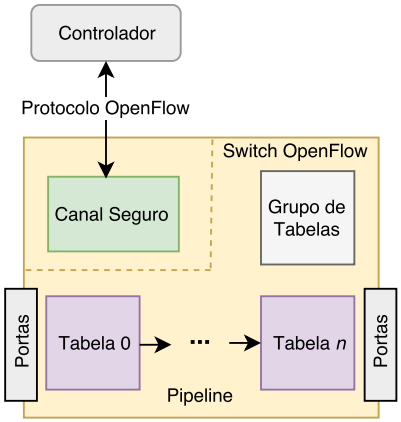

Figura 3.3 – Principais componentes do *switch* OpenFlow. Adaptado de [65].

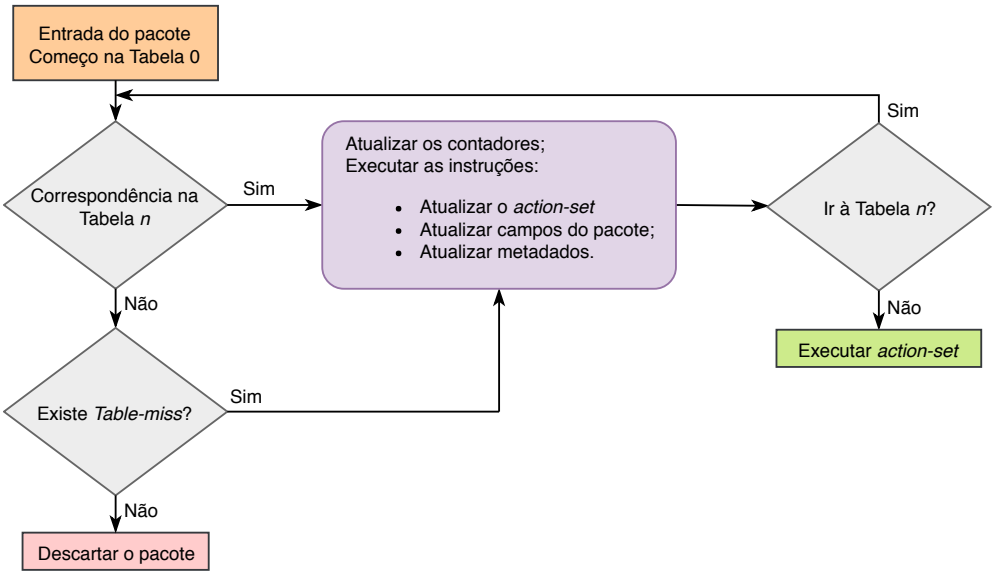

Figura 3.4 – Fluxograma simplificado detalhando o fluxo de pacotes através de um switch OpenFlow. Adaptado de [65].

pipeline de tabelas de encaminhamento tem-se o chamado *table-miss*. Se nenhuma regra padrão foi definida para lidar com essa situação, o pacote pode ser descartado. Entretanto, a alternativa mais empregada é instalar nas tabelas de encaminhamento uma entrada contendo uma regra padrão que obriga o *switch* a enviar o pacote ao controlador e esse, por sua vez, deverá responder com mensagens OpenFlow indicando a ação a ser tomada com o pacote em questão e todos os demais que pertencem ao mesmo fluxo. Essa resposta é decodificada e armazenada na tabela de encaminhamento como uma nova entrada. Uma entrada *table-miss* tem prioridade zero e é criada omitindo-se todos os campos de correspondência. Assim, sob qualquer pacote que não combine com nenhuma regra específica é aplicada a ação definida pela entrada *table-miss*.

Como mencionado anteriormente, as entradas de fluxo têm um conjunto de instruções que são executadas quando um pacote combina com a regra de correspondência dessa entrada. Essas instruções podem alterar o pacote, o conjunto de ações (*action set*) que pode ser aplicado ao pacote e ainda modificar o processamento do *pipeline*. As instruções que *switches* OpenFlow podem suportar são:

- ***Clear-Actions***: Elimina todas as ações do *action set* imediatamente. Essa instrução é necessária apenas às entradas de fluxo *table-miss*, sendo opcional para outras entradas de fluxos;
- ***Write-Actions***: Ações que vão se adicionando no *action set* e só serão executadas ao final da última tabela;
- ***Goto-Table***: Indica a próxima tabela que dará sequência ao processamento do

pacote;

- ***Apply-Actions*** (opcional): Indica que as ações devem ser executadas imediatamente, sem modificar o conjunto de instruções. Essa instrução pode ser usada para modificar o pacote entre uma tabela e outra;
- ***Write-Metadata*** (opcional): Permite a transmissão de informações adicionais entre tabelas do *pipeline*;
- ***Stat-Trigger*** (opcional): Gera um evento para o controlador caso alguma estatística do fluxo ultrapasse um limiar.

Um *switch* deve rejeitar uma entrada de fluxo se não for possível executar as instruções ou parte das instruções associadas a essa entrada [64]. Além disso, ele não precisa oferecer suporte para todos os tipos de ações e, por isso, o controlador pode indagá-lo para descobrir quais delas o *switch* consegue executar. A seguir são listadas todas as ações que podem ser realizadas pelo *switch* na medida em que seja necessário.

- ***Output***: Essa ação encaminha um pacote para uma porta específica do *switch*, onde se inicia o processamento de egresso;
- ***Group***: Processa o pacote por meio de um grupo específico. A interpretação exata depende do tipo de grupo. Nessa ação, diferentes entradas de fluxos na mesma ou em diferentes tabelas podem ter a mesma ação de grupo;
- ***Drop***: Não há ação explícita para representar o descarte. Em vez disso, os pacotes cujo *actions set* não tem ação de saída (*output*) definida e nenhuma ação de grupo (*group*) devem ser descartados;
- ***Set-Queue*** (opcional): Indica qual fila de saída irá processar o pacote, permitindo ainda aplicar políticas de priorização de tráfego;
- ***Set-Field*** (opcional): Usado para reescrever uma variedade de campos do cabeçalho de pacotes;
- ***Push-Tag/Pop-Tag*** (opcional): Permite o manuseio de *tags*, tanto para identificação em VLAN (*Virtual Local Area Network*) quanto para rótulo MPLS (*Multi-Protocol Label Switching*);
- ***Change-TTL*** (opcional): Provê funcionalidades como modificação e cópia dos valores de *time to live* (TTL) do protocolo IPv4 e MPLS assim como o *Hop limit* do IPv6.

3.2.2 Interface *southbound* e o protocolo OpenFlow

As interfaces *southbound* são API que garantem a comunicação entre o plano de controle e o plano de dados. A padronização dessas interfaces promove a interoperabilidade, permitindo o desenvolvimento de dispositivos de rede independente de soluções proprietárias. O protocolo OpenFlow é o padrão de *southbound* aberto mais aceito e implantado para SDN. Além de oferecer uma especificação comum para o desenvolvimento de dispositivos, intermedia a comunicação entre esses dispositivos e o controlador.

3.2.3 Protocolo OpenFlow

O protocolo OpenFlow é capaz de entregar três tipos distintos de informação para o controlador SDN. Primeiro, mensagens baseadas em eventos são enviadas pelos dispositivos do plano de dados quando ocorrem mudanças em enlaces ou portas. O segundo tipo corresponde às mensagens denominadas de *packet-in*, as quais são enviadas para o controlador quando o dispositivo não sabe como lidar com um pacote recém-chegado, ou seja, não existe nenhuma entrada em sua tabela de fluxos que informe qual ação deve ser tomada (por exemplo, primeiro pacote de um fluxo). Essa mensagem pode ainda ser encaminhada ao controlador quando a ação que exige esse envio está presente no *action set* da entrada da tabela ao qual o pacote corresponde. As duas mensagens anteriores são enviadas para o controlador de maneira proativa. A terceira informação encaminhada por meio do protocolo OpenFlow é realizada de forma passiva, ou seja, ocorre mediante a solicitação do controlador. Essa informação compreende as estatísticas de fluxos que são geradas pelos *switches* e roteadores e repassados para o plano de controle.

As arquiteturas baseadas em OpenFlow possuem capacidades particulares que podem ser utilizadas por pesquisadores para experimentar novas ideias e testar novas aplicações. Esses recursos incluem análise de tráfego baseada em *software*, controle centralizado, atualização dinâmica de regras de encaminhamento e abstração de fluxo [55]. Os aplicativos baseados em OpenFlow foram propostos para facilitar a configuração de uma rede, simplificar o seu gerenciamento e adicionar recursos de segurança, virtualizar redes e *data centers* e implantar sistemas móveis.

Desde a sua introdução, o OpenFlow vem passando por diversas atualizações para tornar-se cada vez mais estável e robusto. Algumas das principais modificações entre as versões lançadas são mostradas a seguir:

- Versão 1.0 [66]: lançada em 2009, é a primeira especificação que possui suporte oficial dos fabricantes de dispositivos de rede. O protocolo suporta uma tabela com entradas de fluxo composta por três componentes: campos do cabeçalho (*Header*

Fields), contadores e ações. Os campos do cabeçalho contêm 12 elementos para serem comparados com as informações dos pacotes. Nas versões seguintes do OpenFlow, esse campo foi renomeado para campos de correspondência (*Match Fields*), visto que ele não era usado unicamente para os campos do cabeçalho, mas também utilizava porta de ingresso no *switch* e metadados. De forma análoga, o campo ações passou a chamar-se instruções.

- Versão 1.1 [67]: liberada em 2011, introduziu o processamento de múltiplas tabelas de fluxos. Também foi adicionado suporte ao protocolo MPLS. Nessa versão também foi adicionado o recurso de grupos de tabelas.

- Versão 1.2 [68]: ainda em 2011, a ONF adicionou suporte ao protocolo IPv6. Também ficou estabelecido que os *switches* poderiam se conectar a vários controladores simultaneamente. Essa característica contribuiu para o balanceamento de carga entre os controladores e melhorou a recuperação de falhas no plano de controle. A adição de novos campos de correspondência foi facilitada com os recursos de extensibilidade introduzidos no OpenFlow versão 1.2 por meio do *OpenFlow Extensible Match* (OXM). Com o OXM, a entrada de fluxo possuía campos extensíveis para identificar o fluxo, permitindo que novos critérios de correspondência pudessem ser adotados de maneira rápida e fácil.

- Versão 1.3 [69]: em 2012 o OpenFlow incorporou suporte às extensões do cabeçalho IPv6. Nesta versão foi incluída uma tabela denominada *Meter Table* para melhorar a questão do QoS no *switch* OpenFlow. Como outro destaque, a entrada do tipo *table-miss* foi definida. Nas versões anteriores do protocolo, quando um pacote não atendia os campos de correspondência de nenhuma entrada, era descartado ou enviado para o controlador em uma mensagem *packet-in*. Com a introdução da *table-miss*, o processamento de tais pacotes torna-se mais flexível.

- Versão 1.4 [65]: lançada em 2013, permitiu a sincronização de tabelas no *pipeline* de processamento. Quando duas tabelas estão sincronizadas e a tabela de origem sofre uma alteração (adição de uma nova entrada, modificação ou remoção), essa mudança também é empregada na tabela sincronizada. O uso mais comum da sincronização é para o aprendizado do endereço MAC (*Media Access Control*) e para o encaminhamento. A primeira tabela tem a função de aprendizado, adicionando uma entrada à tabela com o endereço e porta de origem. A segunda provê a funcionalidade de encaminhamento, em que o endereço de destino é pesquisado e, caso seja encontrado, o pacote seguirá para a porta do *switch* indicada pela entrada correspondente. Outro recurso interessante foi denominado *Bundle*, o qual foi definido como uma sequência de modificações solicitadas pelo controlador que é aplicada com uma única ope-

ração OpenFlow. Caso todas as modificações tenham êxito, todas serão mantidas, porém, se ocorrerem erros, nenhuma delas será aplicada. Portanto, o *Bundle* garante a característica de atomicidade das operações realizadas nos *switches*.

- Versão 1.5 [70]: lançada em 2014, estende o conceito de *Bundle* por meio do *Schedule Bundle*, que corresponde ao agendamento da execução das operações do *Bundle*. A mensagem de *Bundle*, enviada pelo controlador aos *switches*, inclui um tempo de execução que determina quando tais operações devem ser efetivadas. Quando o *switch* recebe a mensagem, aplica as operações o mais próximo possível do tempo preestabelecido na mensagem recebida.

3.2.4 Mensagens OpenFlow

O OpenFlow cria uma interface de comunicação que conecta cada dispositivo de encaminhamento do plano de dados ao controlador. Por meio dessa interface, o controlador pode configurar e gerenciar os dispositivos, ser notificado sobre eventos que ocorrem no plano de dados e enviar pacotes aos dispositivos de encaminhamento. Todas as mensagens trocadas entre os planos devem ser formatadas de acordo com as especificações do protocolo OpenFlow e podem ser transmitidas por meio de um canal seguro usando o protocolo TLS (*Transport Layer Security*) [71]. Caso a criptografia não seja aplicada, a comunicação ocorre diretamente por meio do protocolo TCP [64]. A implementação da comunicação entre o plano de dados e o OpenFlow é específica para cada equipamento, mas o canal OpenFlow tem que ser padrão. Dessa maneira, o OpenFlow suporta três tipos de mensagens: controlador para *switch*, assíncrona e simétrica. Cada um dos tipos possuí diversos subtipos, que servem para os mais variados propósitos.

3.2.4.1 Mensagens controlador para *switch*

Essas mensagens são iniciadas pelo controlador e podem ou não exigir uma resposta do equipamento receptor da mensagem. Dentre essas mensagens, destacam-se os subtipos:

- ***Features***: O controlador solicita a identificação e os recursos básicos de um *switch*. O *switch* deve responder com uma mensagem *Features replay*, informando sua identificação e suas capacidades. Isso geralmente é realizado durante o estabelecimento do canal OpenFlow;
- ***Configuration***: O controlador pode definir ou solicitar parâmetros de configuração aos equipamentos;

- ***Modify-state:*** Mensagens de modificação de estado são enviadas pelo controlador para gerenciar o estado dos equipamentos. Sua finalidade é adicionar, excluir e modificar entradas de fluxos ou de grupos nas tabelas de encaminhamento, assim como definir propriedades das portas dos dispositivos;

- ***Read-State***: Usadas pelo controlador para coletar informações dos equipamentos como configuração e estatísticas (*e.g.*, portas e fluxos);

- ***Packet-out***: Mensagem utilizada para indicar ao *switch* a porta de saída que deverá ser utilizada para encaminhar o pacote, o qual foi recebido pelo controlador por meio de uma mensagem *packet-in*. A mensagem *packet-out* deve conter o pacote completo ou a referência do pacote armazenado no *switch*. A mensagem também deve conter uma lista de ações a serem aplicadas na ordem em que são especificadas. Uma lista vazia de ações indica o descarte do pacote;

- ***Role-Request***: Mensagens usadas pelo controlador para definir sua função no canal OpenFlow bem como definir ou requerer um identificador que o diferencie dos demais controladores. Geralmente essa funcionalidade é utilizada quando um *switch* se conecta a um plano de controle com múltiplos controladores.

3.2.4.2 Mensagens assíncronas

O *switch* pode enviar mensagens ao controlador mesmo quando não solicitado. Essas mensagens são nomeadas de assíncronas e transmitidas para indicar a chegada de um pacote ou a mudança de estado do *switch*. Os principais tipos dessas mensagens são listados a seguir:

- ***Packet-in***: O *switch* transfere ao controlador a responsabilidade de definir o encaminhamento de um pacote recém-chegado. Nesse caso, o pacote apresenta correspondência com uma entrada de fluxo na tabela de encaminhamento cuja regra é transmitir o pacote por uma porta reservada ao controlador ou o pacote tem correspondência com a regra *table-miss*. Um evento do tipo *packet-in* pode ser configurado para armazenar pacotes em *buffers*, caso o *switch* possua memória suficiente. Nesse caso, apenas uma fração do cabeçalho do pacote e um identificador do *buffer* onde está armazenado são passados na mensagem. A quantidade de bytes do pacote armazenado que deve ser incluída na mensagem é variável, porém o padrão é utilizar 128 bytes [64]. *Switches* que não possuem armazenamento interno suficiente são instruídos a não realizarem o armazenamento e devem enviar o pacote todo para o controlador;

- ***Flow-Removed***: Informa ao controlador sobre a remoção de uma entrada em uma das tabelas de fluxo. Somente entradas cuja *flag* `OFPFF_SEND_FLOW_REM` está ativa geram o evento *Flow-Removed*. A mensagem pode ser resultado da solicitação de remoção de fluxo feita pelo controlador, do processo de expiração do fluxo quando um dos tempos de espera (*idle timeout* ou *hard timeout*) é excedido ou quando o dispositivo remove fluxos para liberar recursos usados;

- ***Port-status***: usada pelo *switch* para informar ao controlador que uma mudança em uma de suas portas ocorreu. O objetivo é informar mudanças ocasionadas por uma ação de configuração no equipamento como, por exemplo, desativação realizada por um usuário, ou por um evento de mudança de estado como um enlace inativo.

3.2.4.3 Mensagens simétricas

Mensagens simétricas podem ser enviadas tanto pelo controlador quanto pelos *switches* sem nenhuma solicitação prévia. Mensagens dessa categoria incluem:

- ***Hello***: são trocadas entre o *switch* e o controlador na inicialização da conexão;

- ***Echo***: mensagem que pode ser transmitida pelo *switch* ou pelo controlador e deve ser respondido pela outra parte com um *Echo reply*. Ela pode ser usada para indicar a latência, largura de banda e o estado da conexão;

- ***Error***: são usadas pelo controlador ou *switch* para notificar problemas para a outra parte. Geralmente as mensagens de erros são enviadas pelo *switch* para indicar falhas em uma solicitação iniciada pelo controlador;

- ***Experimenter***: Reservado para recursos destinados a futuras versões do OpenFlow na qual poderão ser oferecidas funcionalidades adicionais.

3.2.5 Outras interfaces *southbound*

O OpenFlow não é a única interface *southbound* para SDN. Outras API propostas são ForCES (*Forwarding and Control Element Separation protocol*) [72], Open vSwitch Database (OVSDB) [73], POF (*Protocol Oblivious Forwarding*) [74] e OpFlex [75]. O protocolo ForCES proporciona maior flexibilidade para as redes tradicionais sem a necessidade de mudança da arquitetura atual, nem mesmo a adição de uma entidade de controle logicamente centralizada. As lógicas de controle e de encaminhamento são desacopladas, mas podem estar inseridas no mesmo dispositivo de rede. ForCES foi definido e mantido pelo grupo de trabalho de mesmo nome no IETF, o qual teve seu estado alterado para concluído em 2015.

O Open vSwitch[2] (OVS) é um *software* de código aberto que implementa um *switch* virtual multicamada, cujo propósito é o encaminhamento de tráfego entre diferentes máquinas virtuais no mesmo *host* físico e também a transmissão do tráfego entre as máquinas virtuais e a rede física. Uma interface desenvolvida especificamente para melhorar a capacidade de gerenciamento do OVS é o OVSDB. Em um ambiente virtualizado, o OpenFlow pode ser utilizado para o gerenciamento das entradas de fluxos enquanto o OVSDB configura o comportamento dos *switches*. Essa combinação permite que o OVSDB se concentre em atividade como criação e manipulação de várias instâncias de *switches* virtuais, configuração de políticas de QoS, gerenciamento de filas e incorporação de interfaces aos *switches* [56].

Um dos maiores objetivos do POF é otimizar o encaminhamento das informações no plano de dados. Um dos problemas enfrentados pelos *switches* OpenFlow é que eles precisam entender o cabeçalho do protocolo para analisar os pacotes e executar a correspondência com a tabela de fluxo, o que pode causar sérios problemas de incompatibilidade quando novos protocolos adicionam ou removem campos do cabeçalho. Essa análise representa um fardo significativo para os dispositivos do plano de dados. Isso pode ser comprovado pela própria evolução do OpenFlow, em que sua primeira versão poderia utilizar até 12 campos para a correspondência, enquanto na versão 1.3 esse valor ultrapassava 40. O POF introduz um conjunto de instruções independente de protocolo, permitindo que um operador de rede defina a pilha de protocolos e o procedimento de processamento de pacotes de uma maneira muito mais flexível do que as especificações OpenFlow atuais. Esse conjunto de instruções é denominado POF-FIS (POF - *Flow Instruction Set*) e seu uso faz com que o dispositivo de rede não precise conhecer nada a respeito do pacote antecipadamente [76].

Mais recentemente, a empresa Cisco desenvolveu sua própria interface *southbound* com o intuito de facilitar a comunicação entre o controlador e plano de dados. Enquanto o OpenFlow centraliza toda a funcionalidade de controle da rede no controlador, o OpFlex foca principalmente em políticas, as quais, em parte, podem residir nos próprios dispositivos de encaminhamento, como ocorre no ForCES. A ideia é diminuir a potencialidade do controlador se tornar um gargalo e, consequentemente, aumentar a resiliência e disponibilidade da infraestrutura de rede.

3.2.6 Plano de Controle

O plano de controle em SDN atua como uma camada intermediária entre as aplicações e o plano de dados e, do ponto de vista prático, pode ser comparado a um sistema operacional [56]. Tradicionalmente, um sistema operacional provê abstração, ser-

[2] http://www.openvswitch.org

viços essenciais e uma API comum para desenvolvedores. A arquitetura SDN utiliza esses recursos para facilitar o gerenciamento da rede e descomplicar a resolução de problemas por meio do controle logicamente centralizado fornecido por um ou mais controladores. As funções essenciais de rede como manutenção das informações da topologia, descoberta de novos dispositivos conectados e configuração dos dispositivos de encaminhamento são exemplos de serviços desempenhados pelo plano de controle. Além disso, a API provida pelos controladores permite aos administradores não se preocuparem com detalhes de baixo nível de *switches* e roteadores quando forem estabelecer ou modificar políticas.

O controlador corresponde a uma entidade de *software* que possui controle exclusivo sobre um conjunto de recursos do plano de dados [62]. É também considerado um elemento crítico na arquitetura SDN, pois suporta a lógica de controle para gerar e manter a configuração da rede com base nas políticas definidas pelos operadores. Assim, a robustez do controlador é importante uma vez que lida diretamente com o desempenho da rede. Devido à sua importância, muitas pesquisas sobre o aprimoramento de projetos de controladores para melhoria de consistência, escalabilidade, flexibilidade, latência e disponibilidade têm sido constantemente desenvolvidas [55].

Os controladores podem ser categorizados em muitos aspectos. Do ponto de vista da arquitetura utilizada, o critério mais importante é se o controlador é centralizado ou distribuído. Um controlador centralizado é uma única entidade que gerencia todos os dispositivos de encaminhamento do plano de dados [77]. Como alguns representantes dessa categoria tem-se os controladores POX[3], Beacon [78] e Floodlight[4], todos desenvolvidos concomitantemente a fim de oferecer o desempenho requerido por empresas e *data centers*.

Em um estudo apresentado por Erickson [78], foi verificado que um único controlador Beacon, executado em um servidor com 12 núcleos de processamento, foi capaz de suportar 12,8 milhões de novos fluxos por segundo com uma latência média de 24,7 microssegundos. De maneira similar, Tootoonchian *et al.* [79] conduziram avaliações quanto à estabilidade dos controladores NOX-MT, Maestro e Bacon. Uma rede contendo 100 mil *hosts* e 256 *switches* foi emulada para a realização dos testes. Os resultados indicaram que todos os controladores conseguiram lidar com pelo menos 50 mil novos fluxos por segundo. Um único controlador é capaz de lidar com um número surpreendente de novas solicitações de fluxo, e deve ser capaz de gerenciar redes de tamanho considerável. Além disso, os novos controladores em desenvolvimento têm como alvo o paralelismo de múltiplos núcleos de processamento (*multithread*) de poderosos servidores para garantir a escalabilidade até grandes cargas de trabalho de *data center* (cerca de 20 milhões de solicitações por segundo e até 5 mil *switches*) [80]. No entanto, o uso de apenas um

[3] https://github.com/noxrepo/pox
[4] http://www.projectfloodlight.org/

controlador, naturalmente, representa um ponto de falha bem como um gargalo quando utilizado em redes excessivamente grandes. Desse modo, vários controladores podem ser usados de forma colaborativa para reduzir a latência e aumentar a resiliência.

Contrário ao modelo centralizado, o plano de controle distribuído tenta atender aos requisitos de qualquer ambiente, de redes pequenas às grandes. Um controlador distribuído pode ser uma coleção centralizada de nós (*cluster*) ou um conjunto de elementos fisicamente separados. Enquanto a primeira configuração oferece baixa latência para estruturas que suportam grande volume de dados, por exemplo *data centers*, a segunda é mais tolerante a diferentes tipos de falhas [77]. Ainda, esses múltiplos controladores podem ser implementados de duas maneiras distintas. Na abordagem de plano de dados logicamente centralizado, os controladores sincronizam sua visão local sobre a rede uns com os outros. Dessa maneira, todos os controladores mantêm uma visão global e consistente da rede para tomar as melhores decisões de encaminhamento. No caso do plano de dados distribuídos fisicamente, não é necessário que cada controlador tenha conhecimento de todo o ambiente ao qual está inserido. Uma visão local já é suficiente para que ele desempenhe o seu papel no plano de controle.

Tabela 3.1 – Comparação entre controladores.

Controlador	Categoria	NBI API	Linguagem	Desenvolvedor
Beacon	Centralizado	REST API	Java	Univ. Stanford
Floodlight	Centralizado	RESTful API	Java	Big Switch Networks
Maestro	Centralizado	REST API	Java	Univ. Rice
NOX	Centralizado	ad-hoc API	C++	Nicira
ONOS	Distribuído	RESTful API	Java	ON.Lab
OpenDaylight	Distribuído	REST, RESTCONF	Java	Linux Foundation
POX	Centralizado	ad-hoc API	Python	Nicira
RYU	Centralizado	adhoc API	Python	NTT

A Tabela 3.1 apresenta a síntese de alguns dos controladores mais populares do mercado. As informações destacam a categoria em relação à arquitetura, tipo de interface de comunicação, linguagem de programação usada e a entidade responsável pela manutenção do desenvolvimento de cada controlador.

3.2.6.1 Interfaces Eastbound e Westbound

As interfaces *eastbound* e *westbound* são API usadas por controladores distribuídos. Suas principais funções incluem a comunicação entre os controladores, algoritmos para manutenção da consistência dos dados, recursos de monitoramento e notificação. Como exemplo, as interfaces *eastbound* e *westbound* são usadas para verificar se um con-

trolador está em funcionamento ou notificar a transferência do controle de um conjunto de dispositivos do plano de dados para outro controlador.

Para garantir a compatibilidade e a interoperabilidade entre os diferentes controladores, é necessário que essas interfaces também sejam definidas por meio de padrões. Assim, as soluções propostas podem ser usadas de maneira orquestrada e interoperável a fim de construir planos de controle distribuídos mais escalonáveis e confiáveis. Pensando dessa forma, a arquitetura SDN estabeleceu requisitos comuns para coordenar comportamentos entre os controladores e trocar informações de controle em vários domínios SDN.

3.2.7 Interface Northbound

A interface *northbound* (NBI) situa-se entre o plano de aplicações e o controlador. Ela provê a abstração necessária dos conjuntos de instruções de baixo nível usados pela SBI para programar os dispositivos de encaminhamento. O objetivo da NBI é facilitar o desenvolvimento de aplicações para a gerência e controle da rede, fornecendo aos programadores uma API, isto é, uma interface comum para o desenvolvimento de aplicações [81].

Contrário às API da SBI, as API da NBI ainda não possuem especificação padrão, pois tanto os controladores quanto os aplicativos existentes possuem recursos diversos, o que é difícil de unificar [81, 82]. A padronização da NBI é imprescindível para manter a compatibilidade com diferentes controladores. Essa situação levou os controladores como Floodlight e OpenDaylight[5] a proporem sua própria API, cada uma com suas definições. Para resolver essa questão, a *Open Networking Foundation* (ONF) possui um grupo de trabalho dedicado à padronização de várias API. O objetivo é criar soluções que possam ser estendidas, que forneçam estabilidade e que possam ser portáveis para os variados controladores e serviços de rede. A estratégia adotada é a definição de várias API em diferentes níveis de abstração para permitir a programação detalhada do comportamento da rede. Além disso, as instituições fornecedoras de controladores são convidadas a ajudarem no desenvolvimento por meio de suas extensões, o que acelera a inovação do padrão SDN.

Como nenhum padrão foi estabelecido, diversas API para NBI foram propostas na literatura. Frenetic [83] é uma linguagem de alto nível para programar *switches* por meio de uma linguagem de consulta declarativa para classificar e agregar tráfego de rede. Essa ferramenta também fornece uma biblioteca funcional para descrever políticas de encaminhamento de pacotes. Pyretic [84] é outra API com o mesmo propósito. Além de proporcionar todas as facilidades da Frenetic, seu foco é a construção de aplicações

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

modulares, as quais permitem que os programadores criem com maior facilidade as aplicações dependentes umas das outras. Já a solução proposta pela API denominada Procera [85] aplica os princípios da programação reativa funcional [86], a qual serve como base para construção de um *framework* voltado ao controle da rede.

3.2.8 Plano de Aplicações

As aplicações de rede implementam a lógica de controle que será traduzida em comandos, que por sua vez, serão instalados no plano de dados a fim de instruir como os dispositivos de rede devem operar [56]. Por exemplo, uma aplicação desenvolvida para roteamento de pacotes deve definir o caminho pelo qual o pacote deve seguir da origem até o destino. Para isso, a aplicação deve determinar o caminho a ser usado e instruir o controlador a instalar regras de encaminhamento em todos os dispositivos de rede compreendidos no caminho escolhido.

Ao desacoplar os planos de controle e dados, as redes programáveis permitem o controle personalizado, a oportunidade de eliminar as *middleboxes*, bem como a facilidade no desenvolvimento e a implantação de novos serviços e protocolos de rede. Diferentes ambientes podem se beneficiar da programação de suas atividades, desde redes domésticas até pontos de troca de tráfego (PTT) [80]. Essa diversificação também é acompanhada pela variedade de aplicações de redes desenvolvidas. Tais aplicações não se limitam apenas a desempenhar as funcionalidades tradicionais de rede como roteamento e balanceamento de carga, mas também exploram novas abordagens, por exemplo, redução do consumo de energia [87, 88], confiabilidade do plano de dados, monitoramento de QoS [89], virtualização de rede [90, 91], gerenciamento de mobilidade em redes sem fio [92], entre outros.

3.3 Considerações sobre o capítulo

As redes de computadores são formadas por uma grande quantidade de equipamentos, como roteadores, *switches* e diferentes tipos de *middleboxes*, com vários protocolos complexos regendo o funcionamento deles. Para que esses dispositivos possam funcionar corretamente, os administradores de rede devem configurar regras e políticas relacionadas à rede com o objetivo de atender às particularidades das diversas aplicações. Porém, eles precisam transformar essas regras e políticas de alto nível em diversos comandos não padronizados, estabelecidos pelos diversos fabricantes existentes. Como resultado, o gerenciamento se torna desafiador e propenso a erros, uma vez que é preciso lidar com as particularidades dos dispositivos de encaminhamento, obrigando os operadores a se con-

centrarem também em detalhes de baixo nível enquanto implementam as funcionalidades da rede.

A ideia de tornar a configuração dos equipamentos mais simples culminou no surgimento da arquitetura SDN. Com essa nova abordagem, o gerenciamento se tornou mais versátil, uma vez que promoveu a desvinculação do plano de dados e do plano de controle, os quais eram verticalmente integrados nos dispositivos de rede. Com a transferência da lógica de encaminhamento para o controlador, os operadores puderam gerenciar as políticas de maneira mais simples, tendo uma visão global da rede e aplicando as políticas necessárias usando linguagens de programação tradicionais.

Nos últimos anos, a arquitetura SDN tem atraído a atenção da academia e da indústria. Muito do esforço para sua adoção é destinado à padronização de interfaces de comunicação e a criação de soluções de código aberto, como o protocolo OpenFlow e controladores. Por se tratar de uma tecnologia recente, ainda existem benefícios a serem explorados e importantes desafios a serem superados.

O próximo capítulo apresenta trabalhos da literatura direcionados à manutenção da segurança e gerência de redes de computadores. Nele são discutidas soluções voltadas às redes tradicionais, bem como abordagens desenvolvidas especificamente para SDN. O capítulo expõe também a mitigação dos eventos anômalos usando a programabilidade da rede e como a abordagem proposta nesta tese se diferencia do estado da arte.

4 Trabalhos Relacionados

Este capítulo apresenta uma descrição de conceitos e artigos relacionados ao presente trabalho. Visando a compreensão das soluções apresentadas, é realizada uma exposição abrangendo os seguintes tópicos: detecção de anomalias em redes tradicionais usando fluxos IP, detecção de anomalias e mitigação dos efeitos resultantes dos eventos anômalos em SDN.

4.1 Detecção de Anomalias

Além da classificação genérica baseada em assinatura ou baseada em anomalias, as técnicas empregadas na detecção de ameaças à segurança e ao bom funcionamento da rede podem ser categorizadas de acordo com a classe de algoritmo utilizada. Essas abordagens incluem, entre outras, a detecção de anomalia principalmente por meio de ferramentas estatísticas [93], métodos baseados em processamento de sinais [94], técnicas que utilizam teoria da informação [95] e algoritmos de aprendizado de máquina [96].

São considerados como métodos estatísticos os esquemas matemáticos que propiciam a captura de características, ocorrências e tendências temporais. As informações adquiridas permitem reconhecer propriedades sobre o tráfego, como por exemplo, comportamentos que diferem da regularidade observada. Em muitos casos, os métodos estatísticos são empregados em conjunto com as demais classes de algoritmos para a caracterização e previsão do comportamento da rede, de modo a revelar as tendências anômalas.

Principal Component Analysis (PCA) é uma técnica estatística amplamente utilizada para detecção de anomalias em redes de computadores. Lakhina *et al.* [97], pioneiros nesse campo, abordaram o problema de diagnóstico de anomalia no tráfego de rede usando o PCA para separar com eficiência as medições de tráfego em subespaços normais e anômalos. A ideia principal é que o PCA produz um conjunto reduzido de k variáveis (componentes principais ou k-subespaço) que corresponde ao comportamento normal do tráfego de rede, enquanto o subespaço restante de m componentes ($m = n - k$) consiste em anomalias ou ruído. Em seguida, cada nova medição de tráfego é projetada em ambos os subespaços para que diferentes limites possam ser definidos para classificar essas medidas como normais ou anômalas.

Fernandes *et al.* [50] propuseram uma abordagem de detecção de anomalias de volume utilizando PCA, denominada *Principal Component Analysis for Digital Signature and Anomaly Detection* (PCADS-AD). A proposta dos autores é dividida em duas

etapas. Na primeira, o PCADS-AD analisa dados históricos do tráfego para identificação de características mais significativas, como tendências e padrões de uso. A base histórica então é reduzida para um novo conjunto capaz de caracterizar de maneira eficiente o comportamento normal do tráfego, ou seja, uma assinatura do comportamento esperado. Na segunda fase, a assinatura é confrontada com o tráfego coletado em intervalos de um minuto, por meio de uma janela deslizante. As anomalias são identificadas quando ocorrem discrepâncias nos atributos bits, pacotes ou fluxos transmitidos por segundo. Além disso, é exibido um relatório com os IPs e Portas de origem e destino presentes no momento da ocorrência da anomalia, de modo que auxilie o gerente de rede a resolver o problema.

Huang *et al.* [98] discutiram a hipótese de que o uso de uma matriz de covariância combinada com a redução de dimensionalidade produz um resultado mais satisfatório para a detecção de anomalias do que outros métodos aplicados tradicionalmente. A motivação para tal abordagem vem do fato de que as diferenças estruturais entre as matrizes de covariância do tráfego observado e o normal são mais relevantes para a detecção de anomalias do que a estrutura dos dados de treinamento. Tipos distintos de desvios na matriz de covariância do tráfego observado são fortes indícios de ocorrência de anomalias. Com relação à redução de dimensionalidade, eles apontaram que a escolha de k componentes principais, como no PCA, é ineficiente em capturar mudanças em tempo real. O número correto de atributos do tráfego a serem analisados para detecção efetiva de anomalias frequentemente varia durante períodos de tempo diferentes, e ainda, depende da estrutura do tráfego normal e do tráfego observado. Portanto, para superar as limitações das abordagens baseadas em variância, os autores criaram um método de redução de dimensionalidade baseada na distância entre os subespaços.

Outra abordagem estatística é utilizada no trabalho desenvolvido por Assis *et al.* [99]. Os autores aprimoraram o tradicional método de previsão Holt-Winters para detecção de anomalias, criando o *Holt-Winters for Digital Signature* (HWDS). Sete atributos dos fluxos IP são analisados nessa proposta: bits/s, pacotes/s, fluxos/s, endereços IP de origem e destino e portas de origem e destino. O método gera uma assinatura para cada atributo, descrevendo o comportamento normal ou esperado da rede. Uma janela temporal de um minuto é usada para reconhecimento dos eventos anômalos. Para detectar uma anomalia devem ser observadas divergências entre as assinaturas geradas e o tráfego coletado durante a janela de análise. Os autores propõem o uso de dois níveis de anomalias. O primeiro nível caracteriza-se pela emissão de alertas – avisos de menor prioridade – quando um desvio de comportamento é apresentado por um dos atributos durante o monitoramento do tráfego. Alarmes constituem o segundo nível, em que o administrador é notificado quando vários atributos destoam de suas assinaturas.

Técnicas baseadas em processamento de sinal são usadas para representar o

tráfego de rede como componentes de sinal que podem ser processados de modo independente. Tipicamente, um sinal é convertido a partir do domínio do tempo (ou espaço) para o domínio da frequência, no qual pode então ser dividido nas suas partes de alta, média e baixa frequência. Dessa forma, falhas de equipamento e ataques maliciosos podem ser identificados por meio da filtragem dos componentes de baixa frequência que representam a parte do sinal previsível, ou normal [100]. Isto permite o isolamento de sinais anômalos nos dados de tráfego por meio da análise das propriedades do sinal, como observado em [101, 102].

As abordagens que utilizam teoria da informação baseiam-se na teoria de probabilidade e estatísticas para quantificação da informação. Uma das principais propriedades desse tipo de classe de algoritmo é o cálculo da entropia, a qual descreve o grau de incerteza associada a um valor de uma variável aleatória. A entropia está ligada ao fato que um evento suspeito é capaz de causar um distúrbio significativo em alguns atributos do tráfego como endereços IP de origem e destino, bem como portas de origem e destino [103].

O trabalho apresentado por Bhuyan *et al.* [104] apresenta uma abordagem de detecção de anomalias baseadas em valores discrepantes (*outlier detection*), que tem seu foco no uso da entropia generalizada e informação mútua para criar uma técnica de seleção de atributos do tráfego. Por meio da seleção, um subconjunto de atributos relevantes e não redundantes é aplicado na detecção, o que aumenta a taxa de acerto da solução proposta bem como a torna mais parcimoniosa.

Berezinski *et al.* [105] introduziram um detector de anomalia de rede, baseado na entropia de Shannon, para detectar *botnet* de *malwares*. A abordagem cria um perfil de rede, que armazena os valores mínimos e máximos de aceitáveis para a entropia em uma janela de tempo de 5 minutos. Esses valores foram utilizados para comparação com a entropia observada ao longo do monitoramento da rede. Isto define um limiar, assim, a dispersão ou concentração anormal para diferentes distribuições de características pode ser identificada. Por fim, os autores utilizaram classificadores populares, como árvores de decisão e redes bayesianas, para classificar as anomalias.

David *et al.* [106] propuseram uma detecção aprimorada de ataques de negação de serviço por meio do cálculo da entropia e do uso da análise de fluxos. Os autores agregam os fluxos observados em um único, considerando a contagem de fluxo de cada conexão em um determinado intervalo de tempo, em vez de levar a contagem de pacotes de cada conexão. O segundo passo é basicamente o cálculo da entropia rápida da contagem de fluxo para cada conexão. Finalmente, um limiar adaptativo é gerado com base na entropia rápida e na média e nos desvios padrão das contagens de fluxo. A atualização constante do limite em relação à condição do padrão de tráfego melhora a precisão da detecção,

enquanto o uso de entropia rápida reduz o tempo de processamento computacional.

Amaral *et al.* [107] propuseram um sistema de detecção de anomalias que utiliza propriedades de fluxos IP e uma representação por meio de grafos para realizar uma inspeção profunda do tráfego, em que é possível verificar a troca de informações entre dispositivos ativos na rede. A detecção é realizada por meio da entropia de Tsallis, a generalização da entropia de Shannon. Além disso, o sistema proposto permite que o administrador da rede crie métricas para monitorar e adquirir informações detalhadas sobre o equipamento de rede, serviços e usuários.

4.2 Detecção de anomalias em SDN

Embora redes definidas por *software* permitam novos aplicativos de rede e controle mais flexível em ambientes de rede dinâmicos, a segurança ainda é uma preocupação importante, pois ainda não é um recurso inerente à arquitetura SDN. Cada vez mais, espera-se que os atuais esquemas de segurança funcionem em tempo real para enfrentar um espectro de ameaças, inspecionar grandes volumes de tráfego e fornecer uma identificação eficiente de várias anomalias de rede. Assim, esta seção apresenta alguns esforços desenvolvidos na área de detecção de anomalias relacionadas à SDN para atender a essa demanda.

Aleroud e Alsmadi [11] classificaram os eventos anômalos que potencialmente comprometem a arquitetura SDN em três categorias: (i) ataques ao plano de controle, (ii) comprometimento da comunicação entre os planos de controle e plano de dados e (iii) ameaças projetadas para atacar o equipamento do plano de dados. Um ataque de inundação pode cobrir, direta ou indiretamente, essas três categorias devido ao volume de tráfego e ao crescente número de solicitações de conexão. O uso excessivo desses recursos de rede pode sobrecarregar o controlador, bem como ocupar todas as entradas de tabela de encaminhamento dos dispositivos no plano de dados com fluxos de conexões ilegítimas. Desta forma, apesar do extenso trabalho na segurança da SDN, a maioria das abordagens foi projetada para conter ataques de inundações (por exemplo, ataques DDoS). Tendo isso como base, os autores apresentaram um modelo para detecção de ataques do tipo DDoS que afetam a comunicação entre os *switches* e o controlador. O tráfego é agregado e classificado de acordo com predefinições dispostas em um grafo. Uma vez que técnicas de *machine learning* e *data mining* podem conduzir a muitos falsos positivos, os autores desenvolveram um modelo baseado em um contexto relacional. Diferente do contexto individual, o contexto relacional só existe quando dois nós do grafo têm características comuns. Por exemplo, se dois alertas pertencem a categorias similares, eles devem estar fortemente conectados. A detecção de ataques utiliza a técnica de classificação k-NN,

sendo um vértice classificado pela maioria dos seus k vizinhos. Embora a taxa de detecção dos eventos já conhecidos pela abordagem seja alta, o método proposto pelos autores torna-se incapaz de reconhecer variações do mesmo ataque, visto que é utilizada uma detecção baseada em assinatura. Por esse mesmo motivo, as técnicas que fazem parte da proposta devem ser alimentadas como uma grande base de dados rotulada durante o seu treinamento, a qual nem sempre é possível de se obter.

Um dos primeiros trabalhos sobre detecção de anomalias baseada em análise de fluxo de pacotes utilizando conceitos SDN foi relatado por Braga *et al.* [108]. O protocolo OpenFlow foi utilizado para interagir diretamente com o plano de dados e coletar estatística do tráfego dos ativos de rede. Nessa abordagem, cada pacote é classificado como anômalo ou normal de acordo com uma rede neural *Self-Organizing Maps* (SOM) com topologia de vizinhança descrita por uma função gaussiana. Apesar dos resultados promissores, o trabalho é restrito à detecção de ataques *Distributed Denial of Service* (DDoS). Kreutz *et al.* [7] também se concentraram em explorar os benefícios da SDN para defender o mesmo tipo de anomalia. No entanto, suas vítimas residem no ambiente de rede tradicional, o que torna suas soluções inadequadas para SDN.

Ainda sobre a detecção de ataques do tipo negação de serviço, Ha *et al.* [12] afirmaram que a identificação desses ataques requer processamento e inspeção detalhada de um grande volume de tráfego. Caso o poder computacional dos detectores não seja suficiente, isso pode representar um gargalo, resultando no comprometimento de outros serviços. Em busca da solução, os autores apresentaram uma estratégia de amostragem de tráfego para redes SDN que mantém o volume agregado total amostrado abaixo da capacidade de processamento de um IDS (*Intrusion Detection System*). A proposta é formulada através de um problema de otimização que visa encontrar uma taxa de amostragem adequada para cada *switch*.

Mousavi e St-Lilaire [109] propuseram um mecanismo baseado em entropia para detectar um ataque DDoS. Em caso de ataque, a entropia diminui avaliando a aleatoriedade dos endereços IP de destino dos pacotes recebidos. Xiao *et al.* [110] desenvolveram uma abordagem de detecção efetiva baseada na classificação de tráfego com análise de correlação (CkNN). Embora a taxa de detecção de eventos anômalos conhecidos seja alta, esses métodos tornam-se incapazes de reconhecer variações do mesmo ataque, porque a detecção baseada em assinatura é usada. Por essa mesma razão, as técnicas devem ser treinadas com um grande conjunto de dados rotulado.

Uma abordagem que rastreia os equipamentos responsáveis por encaminhar uma anomalia pela rede monitorada foi proposta por Francois e Festor [111]. A aplicação desenvolvida busca detectar quais são os pontos de entrada (primeiros equipamentos da rede) no caminho de um tráfego anômalo. Para esse propósito, é criado um grafo direci-

onado que descreve a topologia da rede, contendo os roteadores e *switches* como vértices e os enlaces de comunicação como arestas. Após descoberto o alvo da anomalia, uma busca recursiva é realizada na estrutura do grafo, ao mesmo tempo que se busca correspondências do tráfego anômalo nas tabelas de encaminhamento definida pelo protocolo Openflow. Segundo os autores, medidas como controlar o tráfego desses equipamentos ou mesmo interromper a comunicação definindo regras no plano de controle podem amenizar as consequências da anomalia até que ela seja definitivamente solucionada.

Diversos estudos empregam alterações na infraestrutura do SDN para realizar a detecção de anomalias. Wang *et al.* [112] propuseram um modelo denominado NetFuse como um mecanismo de proteção contra sobrecarga gerada pelo tráfego em data centers baseados em OpenFlow. A fim de proteger a rede contra os efeitos do tráfego malicioso, NetFuse é implementado entre os dispositivos de rede e o controlador SDN, criando uma camada adicional. Para detectar o excesso de tráfego gerado pelo ataque, um algoritmo de agregação de fluxos é empregado, cuja função é identificar grupos de fluxos suspeitos. Joldzic *et al.* [16] construiram uma solução alternativa, baseada em uma topologia de rede SDN formada por três camadas para abrigar o IDS. A mais externa, situada no *gateway* da rede, contém um *switch* OpenFlow responsável por dividir o tráfego de entrada e encaminhar as partes geradas à camada intermediária. A segunda camada é composta por diversos dispositivos denominados *processors*, os quais realizam a detecção do ataque. Na terceira camada, o tráfego gerado pelos *processors* são agregados e enviados para o interior da rede por meio de um switch OpenFlow. Duas desvantagens são observadas com essas abordagem. A primeira é a de que os autores assumem que a rede interna é livre de anomalias, o que pode se tornar um problema quando anomalias partindo da própria rede tentam inviabilizar o controlador SDN. A segunda desvantagem é resultado da divisão do tráfego entre os *processors* que pode levar ao mascaramento de ataques, uma vez que, por conta do balanceamento de carga, cada um deles pode analisar partes desconexas do mesmo ataque.

Cheng *et al.* [113] aplicaram um conjunto de máquinas virtuais ligado aos switches próximos à borda da rede SDN para aumentar a escalabilidade da rede e ajustar a quantidade de dados que chegam ao plano de controle durante um ataque. Cada grupo implementa um esquema de filtragem baseado em dois níveis para proteção do controlador. Os fluxos são diferenciados entre anômalos e legítimos logo no primeiro estágio. Para isso, a probabilidade de cada fluxo de ser legítimo é calculada comparando-se as características de tráfego recentemente medidas a um modelo de referência do comportamento normal da rede. O segundo estágio analisa os fluxos caracterizados como anômalos para tentar reduzir a quantidade de falsos positivos. A proposta foi chamada de SDNShield, pois blinda o controlador SDN contra ataques massivos de negação de serviço.

4.3 Mitigação de Eventos Anômalos

A natureza centralizada das redes SDN é uma vulnerabilidade para o sistema, uma vez que os atacantes podem lançar ataques contra o controlador. Soluções existentes limitam a taxa de requisições destinadas a ele, excluindo as consideradas excedentes, o que também acarreta na eliminação de requisições legítimas. A fim de contornarem esse problema, Wei e Fung [13] propuseram FlowRanger, um *buffer* de priorização para controlador lidar com requisições de roteamento baseado em sua probabilidade de ataque, que deriva de um valor de confiança da origem solicitadora. Baseado no valor de confiança, FlowRanger classifica as requisições em múltiplas filas de *buffers* com diferentes prioridades. Dessa maneira, baixa prioridade é atribuída às requisições maliciosas.

Giots *et al.* [15] aplicam um modelo de detecção de anomalia baseado em assinaturas em uma simulação de rede SDN. A proposta apresenta atividades que são divididas em três etapas. A primeira corresponde à coleta dos dados estatísticos que serão usados para o monitoramento da rede. Para tanto, os autores comparam a coleta realizada pelo protocolo OpenFlow e sFlow, em que este último é aplicado com amostragem para diminuir a granularidade dos fluxos na ocorrência de um ataque capaz de comprometer o desempenho da rede. Os dados coletados correspondem aos campos do cabeçalho dos pacotes: endereço IP (origem e destino) e portas (origem e destino). Na segunda fase é utilizado um método de detecção baseado em entropia. Mudanças súbitas nos valores desses atributos, que correspondam à assinatura de uma anomalia, são identificadas. Após o reconhecimento dos endereços IP e portas usadas para disseminação da anomalia, a terceira e última etapa tem por finalidade mitigar os efeitos do evento anômalo detectado. Usando o próprio protocolo OpenFlow, os *switches* recebem a instrução de descartar os pacotes destinados ao alvo do ataque.

Sahay *et al.* [114] apresentaram ArOMA, um *framework* para mitigação de ataques do tipo DDoS. A ideia dos autores é integrar diversos módulos, voltados à contenção do ataque, que estão distribuídos entre o ISP (*Internet Service Provider*) e seus clientes. O monitoramento do tráfego e a detecção de anomalias são realizados por módulos específicos implantados no lado do usuário, enquanto o mecanismo de mitigação é executado pelo ISP. Como o foco do trabalho é a mitigação, o reconhecimento do ataque e sua notificação ao ISP é de responsabilidade do usuário. Quando um fluxo suspeito é identificado, o controlador da rede ao qual o cliente está inserido encapsula em uma mensagem de alerta contendo o identificador do fluxo e a envia para o controlador do ISP. As contramedidas são aplicadas por políticas dentro do ISP para evitar a propagação do DDoS.

Um sistema de detecção e mitigação de ataques do tipo DDoS, próprio para

ambientes de computação em nuvem que utilizam estrutura SDN foi apresentado por [9]. Nomeado de DaMask, esse sistema é divido em dois módulos. O primeiro é responsável pela detecção do ataque, realizando análises estatísticas. Utiliza uma estrutura em forma de grafo para armazenar os padrões de tráfego conhecidos e, quando um comportamento incomum é observado, esse grafo determina se conexões maliciosas estão de fato ocorrendo. O segundo módulo, voltado especificamente para o ambiente de redes dinâmicas, executa contramedidas e gera registros sobre os ataques detectados. Com propósito semelhante, no mecanismo proposto por Cui *et al.* [14], uma vez que um ataque DDoS é detectado, a atenuação da anomalia ocorre por meio da inserção de entradas na tabela de fluxos do *switch* reconhecido o primeiro no caminho de propagação da anomalia. Essas entradas possuem regras que descartam pacotes cujo endereço e porta de destino correspondam ao do alvo do ataque.

4.4 Considerações sobre o capítulo

Neste capítulo foram apresentadas diversas abordagens sobre detecção de anomalias em redes tradicionais e em redes SDN, bem como trabalhos onde são abordadas técnicas sobre mitigação de anomalias. Embora os trabalhos apresentados demonstrem peculiaridades e características distintas, compartilham a existência de um conceito subjacente de normalidade. A noção de comportamento normal da rede é geralmente fornecida por um modelo formal que expressa as relações entre as variáveis envolvidas na dinâmica do monitoramento do tráfego. Por conseguinte, um evento é classificado como anômalo quando o grau de desvio em relação ao perfil de comportamento característico do tráfego, especificado pelo modelo de normalidade, é elevado e ultrapassa um limiar previamente estabelecido ou dinamicamente calculado.

As soluções apresentadas para detecção de anomalias em SDN focam, na sua grande maioria, em um tipo particular de ataque ou requerem que a infraestrutura da rede seja alterada. Embora consigam bons resultados, essas abordagens se tornam inflexíveis. O ecossistema proposto nesta tese é apresentado no capítulo seguinte e difere desses estudos em vários aspectos. Ele não requer nenhuma alteração na infraestrutura SDN e pode detectar e classificar uma variedade de anomalias para escolher a estratégia de mitigação mais adequada. Além disso, o ecossistema fornece relatórios gráficos ao administrador da rede sobre os detalhes das anomalias detectadas e a eficiência das contramedidas tomadas contra elas.

5 Ecossistema Proposto

O ecossistema apresentado para detecção e mitigação de anomalias em ambientes específicos de SDN é baseado nas seguintes propriedades:

- Abordagem autonômica: A proposta permite a automação do monitoramento do tráfego da rede e da detecção de eventos anômalos;
- Configuração/*Design* modular: Embora tarefas como coleta do tráfego, detecção de anomalias, mitigação dos ataques e geração de relatórios sejam complementares, elas são projetadas para serem independentes. Essa característica permite que cada um dos módulos seja alterado sem causar interferência em outro;
- Duas fases de monitoramento: Ao contrário das abordagens que analisam todo o tráfego agregado da rede, o sistema proposto monitora o comportamento dos fluxos em cada *switch*. A primeira fase consiste em observar e comparar o comportamento do tráfego em relação ao esperado, isto é, seu perfil normal. Se um desvio de comportamento é detectado, a segunda fase inicia um monitoramento ostensivo para identificar o atacante e os alvos sob ataque;
- Inteligência voltada à mitigação dos efeitos das anomalias: A abordagem apresentada baseia-se no tipo de anomalia detectada para tomar decisões que neutralizem o evento anômalo e conduzam a rede ao seu estado normal. Para que isso seja possível, o controlador manipula a tabela de encaminhamento dos equipamentos de rede, instalando regras de fluxo de acordo com a classificação atribuída a cada fluxo.

5.1 Visão Geral

A Figura 5.1 mostra as partes do ecossistema e como elas estão relacionadas. O plano de dados consiste em vários dispositivos de encaminhamento (por exemplo, *switches*), que transmitem informações ao seu destino seguindo as regras definidas pelo controlador. Requisições, instalação e atualização de regras de encaminhamento ocorrem por meio da troca de mensagens entre o controlador e o plano de dados. Além disso, o controlador pode usar alguns aplicativos implementados pelo administrador para gerenciar regras de encaminhamento na rede. Usando esse recurso, um conjunto de rotinas relacionadas à detecção e mitigação de anomalias é definido. As rotinas são agrupadas em quatro módulos: coleta do tráfego, detecção de anomalias, mitigação e geração de relatórios.

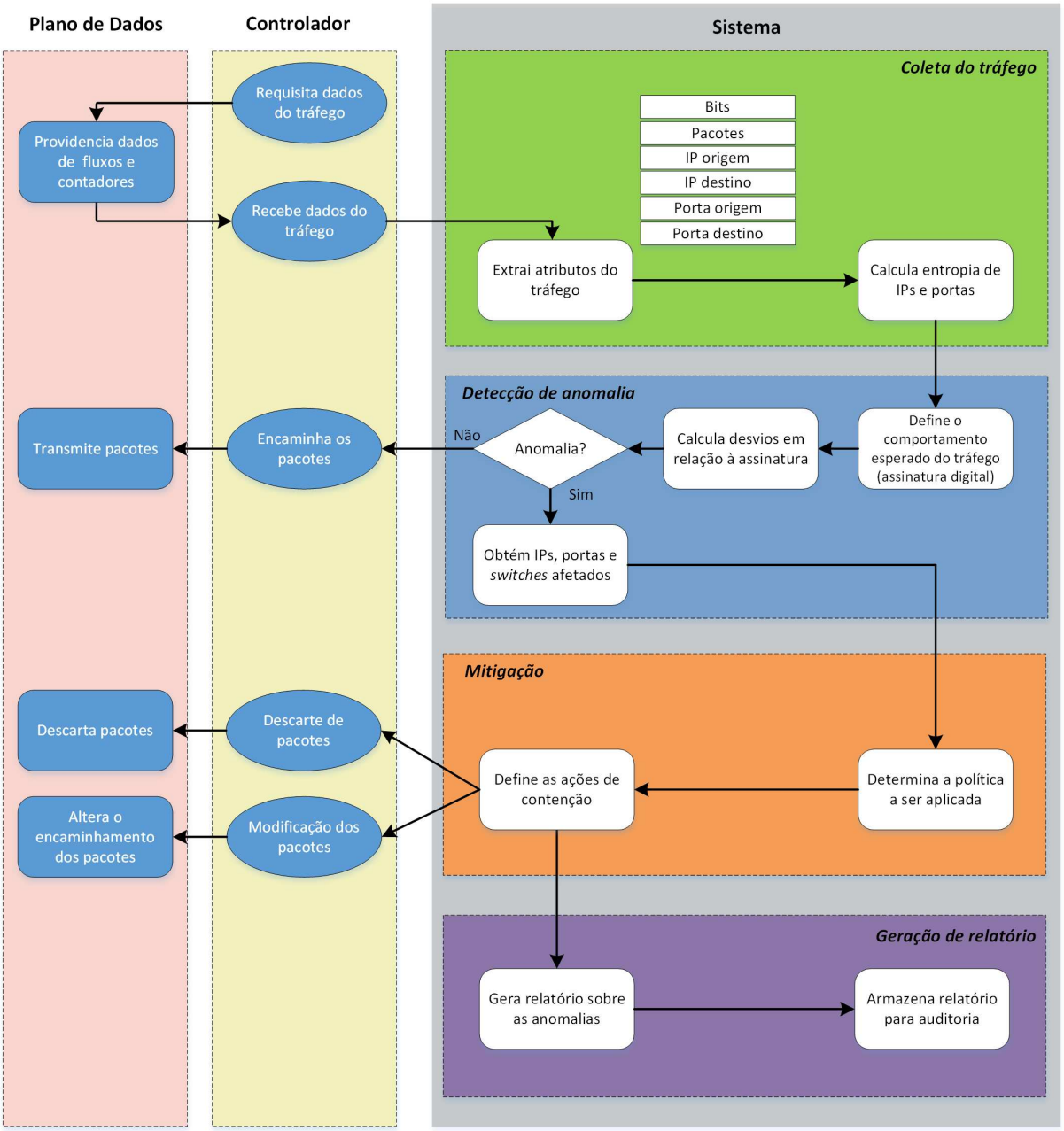

Figura 5.1 – Visão geral do ecossistema proposto.

O controlador envia periodicamente mensagens de solicitação de dados de fluxos para os dispositivos no plano de dados. Consequentemente, os dispositivos respondem com trechos de conteúdo das tabelas de fluxos. Cada fragmento contém uma parte das entradas de fluxo (por exemplo, endereços IP, portas, protocolos) junto com os contadores de pacote e bits de cada fluxo. Depois de obter a resposta, o controlador encaminha as mensagens para o módulo de coleta do tráfego.

A etapa seguinte realiza a extração dos dados necessários para o monitoramento da rede e pré-processa essas informações antes de transmiti-las ao módulo de detecção. No módulo de detecção é verificado se o tráfego recém-coletado difere do esperado.

Se uma anomalia for detectada, o tipo de evento, bem como os dispositivos, os endereços IP e as portas envolvidas na suspeita de comunicação mal-intencionada serão reportados ao módulo de mitigação.

Os endereços IP e informações de portas são usados para compor as mensagens enviadas ao plano de dados para aplicação das rotinas de contenção de anomalias, em que o tipo de evento determina a escolha de uma rotina de mitigação apropriada. As rotinas de mitigação incluem o descarte de pacotes de um determinado fluxo ou de um endereço IP de origem, ou a modificação de entradas de fluxo para alterar o encaminhamento de pacotes pertencentes a um fluxo específico. Por fim, o módulo de relatório armazena informações sobre ameaças detectadas, políticas de mitigação acionadas e endereços IP e portas envolvidas. Esses dados são disponibilizados para fins de validação e auditoria, em que a retroalimentação dos resultados do sistema pode ser usada pelo administrador para avaliação do desempenho do ecossistema, propiciando a melhoria contínua dos parâmetros usados na detecção e mitigação. As próximas seções detalham cada um dos módulos que compreendem o ecossistema proposto.

5.2 Módulo de Coleta do Tráfego

Este módulo é responsável pela aquisição de dados do tráfego, fundamental para monitoramento constante da rede e pré-requisito para a detecção de anomalias. A coleta de informações é feita usando o protocolo OpenFlow. Ele fornece uma interface comum para controlar como os pacotes são encaminhados, como as tabelas de encaminhamento interna dos *switches*, que compõem o plano de dados, são gerenciadas e como mensagens de configuração dos equipamentos devem ser compostas [108]. Além disso, ele utiliza o conceito de fluxos para identificar o tráfego de rede com base em regras previamente estabelecidas, programadas no controlador SDN. Os fluxos são definidos como uma sequência de pacotes com sessões de transmissão unidirecionais que compartilham características como protocolo de transporte, endereços IP e portas tanto de origem quanto de destino. Como os fluxos são identificados por características comuns, eles podem estar relacionados a um aplicativo, dispositivo de rede ou usuário. Como resultado, os administradores ou sistemas de gerenciamento automatizados podem definir como o tráfego deve fluir através dos dispositivos de rede.

Diversos estudos na literatura exploram a detecção de anomalias utilizando análise de tráfego em intervalos fixos de cinco minutos [107, 115, 116]. No entanto, o monitoramento do tráfego transcorrido nesse intervalo torna-se cada vez mais inviável por conta do aumento contínuo das taxas de transmissão. Por exemplo, uma rede que opera a 10 Gbps pode ter terabits de informações comprometidas durante o intervalo

de cinco minutos. Assim, neste trabalho, a cada trinta segundos os dados do tráfego são solicitadas aos *switches* do plano de dados, processadas e enviadas ao módulo de detecção de anomalias. O objetivo da redução da janela de análise é fazer com que as respostas a eventos anômalos, dadas pelo acionamento de contra medidas, sejam efetivadas em um curto período de tempo. Dessa maneira, o sistema garante detecção rápida e mitigação de ataques que, de outra forma, se propagariam ao longo desse período.

O controlador envia periodicamente solicitações para coleta de estatísticas de todas as entradas de fluxo nos *switches* por meio de mensagens do tipo *Read-State*. Mais especificamente, o módulo de coleta constrói uma mensagem *ofp_flow_stats_request* para requisição de informações de cada fluxo armazenado nas tabelas de encaminhamento. Tal mensagem é repassada ao controlador e esse, por sua vez, dispara a solicitação para os *switches* no plano de dados.

Ao receber essa solicitação, cada *switch* responde usando mensagens OpenFlow para o controlador solicitante com partes do conteúdo das tabelas de fluxo. Definimos $\Omega = \{\omega_1, \omega_2, \ldots, \omega_n\}$ como o conjunto de informações do tráfego enviadas por um *switch* para o controlador e cada ω_i contém as seguintes propriedades do fluxo i:

- ω_i^{bits}: bits recebidos, representando o total de bits transportados pelo fluxo;
- $\omega_i^{pacotes}$: quantidade de pacotes recebidos que pertencem ao fluxo;
- ω_i^{srcIP}: endereço IP de origem;
- ω_i^{dstIP}: endereço IP de destino;
- $\omega_i^{srcPort}$: porta de origem;
- $\omega_i^{dstPort}$: porta de destino;
- ω_i^{proto}: protocolo da camada de transporte.

Trabalhos anteriores [117, 118] realizaram extensas análises estatísticas para traçar o perfil normal do tráfego em enlaces de alta velocidade. Os resultados mostraram que a natureza observada do tráfego é dinâmica e diversificada, e também que a quantidade de informações transmitidas muda com o tempo e/ou o dia. Para compreender melhor essa composição diversificada do tráfego, alguns atributos dos fluxos foram selecionados para estudar seus aspectos comportamentais. Os atributos são extraídos a cada intervalo de tempo do conjunto Ω, e consistem em seis propriedades, formando a 6-tupla $F = \langle$bits, pacotes, H(srcIP), H(dstIP), H(srcPort), H(dstPort)$\rangle$, como mostra a Tabela 5.1. O processo de comunicação entre controlador e o *switch*, bem como a atuação do módulo para geração das estatísticas do tráfego são apresentados na Figura 5.2.

Tabela 5.1 – Atributos do tráfego extraídos do conjunto Ω que compõem F.

Atributo	Descrição
bits	Número de bits transmitidos
pacotes	Número de pacotes transmitidos
H(srcIP)	Entropia de IP de origem
H(dstIP)	Entropia de IP de destino
H(srcPort)	Entropia de porta de origem
H(dstPort)	Entropia de porta de destino

A etapa de extração dos atributos relevantes do tráfego é necessária para garantir que sejam utilizadas somente as características que auxiliam na abordagem de detecção de anomalias, evitando solução de continuidade e aumentando a resiliência das redes. Os dois primeiros atributos, bits e pacotes transmitidos, são relativos ao volume do tráfego e podem ser obtidos por bits $= \sum_{i=1}^{|\Omega|} \omega_i^{bits}$ e pacotes $= \sum_{i=1}^{|\Omega|} \omega_i^{pacotes}$, respectivamente. Esses atributos já eram usados para monitoramento da rede e detecção de anomalias baseada na gerência de objetos SNMP [119, 120]. Porém, com a introdução do monitoramento baseado em fluxos IP, uma gama maior de atributos que proporcionam maior riqueza de detalhes do tráfego pode ser obtida.

Os quatro últimos atributos são métricas qualitativas extraídas do cabeçalho de pacote que pertencem a cada ω_i. O uso desses atributos proporciona dois benefícios para o monitoramento da rede e detecção de anomalias. Primeiro, endereços IP e portas fornecem informações para reconhecer dispositivos e serviços que funcionam de maneira suspeita. Por esse motivo, são utilizados no módulo de mitigação para a contenção das anomalias identificadas. Além disso, esses atributos são indicadores sensíveis de mudan-

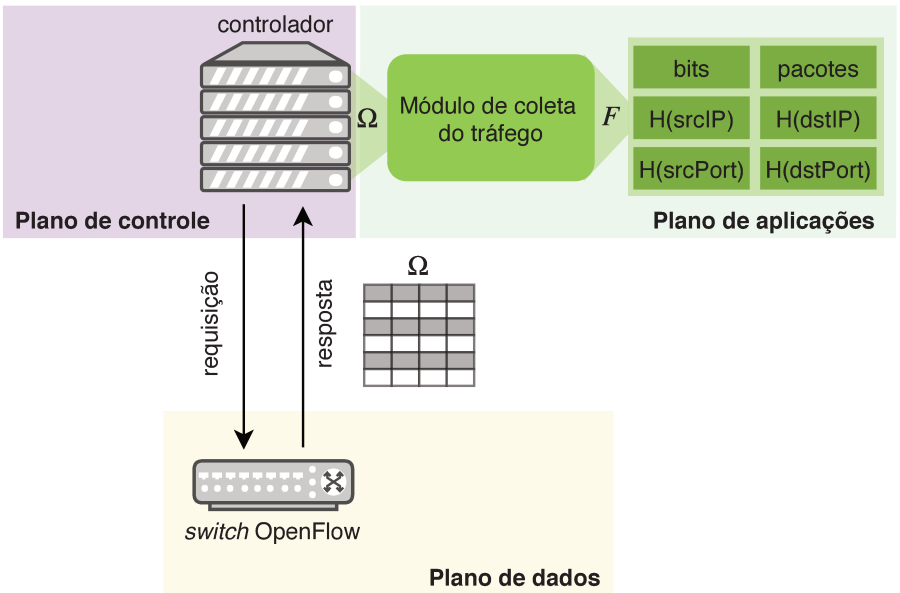

Figura 5.2 – Coleta do tráfego e extração dos atributos relevantes.

ças no comportamento do tráfego, principalmente na ocorrência de alterações abruptas ou acentuadas [97]. Por exemplo, a distribuição dos endereços IP de origem se torna mais dispersa durante um ataque DDoS ou a distribuição de portas pode ser alterada quando elas são verificadas para encontrar vulnerabilidades durante um *port scan*. Ainda, quando uma grande quantidade de informações é recebida de um determinado endereço, a distribuição torna-se mais concentrada para o atributo endereço IP de origem. Por meio de uma análise conjunta entre os atributos qualitativos e quantitativos (bits e pacotes), pode-se verificar a ocorrência de anomalias que geram um volume expressivo no tráfego, bem como aquelas que causam pequenas variações no comportamento da rede.

A análise do comportamento dos atributos de volume ao longo de vários intervalos de tempo é trivial, como em uma série temporal, porém o mesmo não acontece com os atributos qualitativos. Uma forma de transformar as informações de endereços IP e portas em dados quantitativos é a aplicação do cálculo da entropia sobre eles. Neste trabalho é empregada a entropia de Shannon [121], que determina o grau de dispersão ou concentração dos atributos qualitativos dos fluxos.

Dado um conjunto de medidas do atributo $A = \{a_1, \ldots, a_n\}$, em que a_i representa o número de ocorrências da amostra i, a entropia H para o atributo A é definida como:

$$H(A) = -\sum_{i=1}^{n} p_i \log_2 p_i, \tag{5.1}$$

em que $p_i = a_i/n$ é a probabilidade de ocorrência de a_i. Quando a distribuição das amostras está concentrada, o valor de entropia é mínimo e é zero quando todas as amostras são idênticas, isto é, $a_1 = a_2 = ... = a_n$. Por outro lado, quanto mais próximo de $\log_2(n)$, maior é o grau de dispersão, atingindo esse valor quando $p_i = 1/n$ para todo i.

Diversos trabalhos foram realizados com o objetivo de detectar anomalias em redes tradicionais baseados em análise de fluxos e no cálculo da entropia. Assis *et al.* [99] utilizaram o método Holt-Winters para esse propósito. Fernandes *et al.* [118] identificaram ataques por meio da análise de componentes principais do tráfego. Já Pena *et al.* [122] e Hammamoto *et al.* [123] atingiram altas taxas de detecção usando ARIMA (*Autoregressive Integrated Moving Average*) e Algoritmo Genético, respectivamente.

5.3 Módulo de Detecção de Anomalias

Do ponto de vista teórico, o problema de detecção de anomalias neste trabalho pode ser definido como segue. Uma sequência de pontos $\mathcal{F}$ em $\mathbb{R}^{|F|}$ consiste em um conjunto de medições das propriedades do tráfego $\{F_t\}_{t=1}^{T}$, regulado por uma distribuição de probabilidade P. Essas medições são resultados do processamento realizado sobre as

respostas à requisição de estatísticas do tráfego feitas periodicamente pelo controlador. Embora os valores das medições sejam consequência de eventos específicos dentro do espaço de eventos possíveis S (comportamentos do tráfego), o mapeamento $f : S \rightarrow \mathbb{R}^{|F|}$ pode não ser conhecido a princípio. Inicialmente, supõe-se que S pode ser dividido em dois subespaços correspondentes às condições normais e anômalas do tráfego de rede. Essa definição pode ser estendida para mais subespaços, em que se mantém o subespaço normal e o anômalo é dividido entre os diversos tipos de eventos incomuns que podem afetar o tráfego da rede.

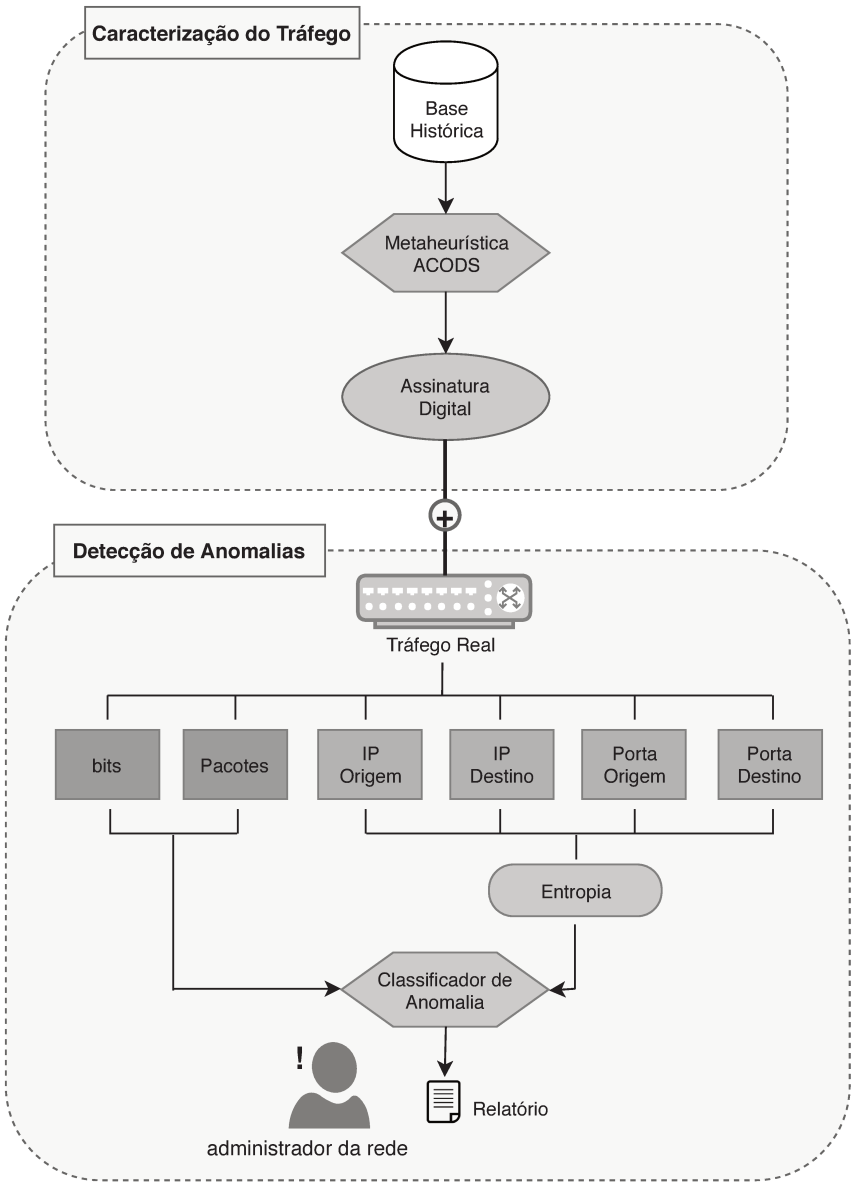

Figura 5.3 – Funcionamento do módulo de detecção.

Desse modo, a detecção de anomalias pode ser decomposta em duas atividades. A primeira corresponde à separação do espaço S entre subespaços normal e anômalos. A segunda tarefa consiste em inferir a associação, ou pertinência, de F_t a um desses subespaços. Para a realização da primeira atividade, um perfil que descreve o comportamento

normal do tráfego é gerado. Um grande desafio encontrado na caracterização desse comportamento é a definição de um modelo que corresponda adequadamente ao conceito de normalidade, visto que o tráfego é dinâmico e se altera de acordo com o padrão de uso de seus usuários [51]. Já para a identificação de qual subespaço F_t pertence, primeiramente observa-se se ele apresenta o comportamento normal esperado. Em caso negativo, os desvios em relação ao comportamento normal serão usados para definir a associação de F_t a um subespaço anômalo.

A Figura 5.3 apresenta o esquema usado na detecção de anomalias. A primeira parte do módulo, destinada à caracterização do tráfego, utiliza o tráfego histórico para a geração da assinatura do comportamento normal da rede. Na segunda parte do módulo, a assinatura gerada é comparada com o tráfego real, recém coletado do plano de dados. Um classificador é usado para o reconhecimento de comportamentos incomuns do tráfego. Ele analisa o desvio comportamental dos atributos de fluxos durante a ocorrência de eventos anômalos e classifica-os de acordo com as anomalias previamente conhecidas pelo ecossistema.

5.3.1 Caracterização do Tráfego

O diagnóstico de anomalia de tráfego constitui um aspecto importante na gestão de redes para torná-las resilientes, uma vez que permite a identificação e classificação de comportamentos que possam vir a comprometer o funcionamento correto da rede [119]. Entre as diferentes metodologias destinadas a esse propósito, destaca-se a criação de um perfil de comportamento normal da rede que proporciona monitoramento contínuo do tráfego [24, 123]. Esse perfil, também chamado de assinatura, tem como propósito demarcar os limites em que uma amostra coletada do tráfego deixa de ser usual e passa a apresentar características anômalas.

O uso cotidiano da rede é identificado por ciclos diários de onde se extraem características de comportamento, de acordo com o período do dia ou dia da semana [124]. O mecanismo responsável pela caracterização do comportamento deve ser capaz de reconhecer tais peculiaridades e de lidar com elas para criação do padrão normal. A abordagem desse projeto reconhece esses comportamentos e suas características para criar um perfil de tráfego normal, denominado assinatura digital do tráfego. A assinatura de normalidade é ajustada automaticamente ao comportamento do tráfego, introduzindo novos comportamentos à medida que eles ocorrem. Assumindo que um dado evento esteja presente em grande parte do histórico e se mantém regularmente, ele será adicionado ao perfil normal. Caso contrário, poderia ser reconhecido como um evento esporádico e sua inclusão na assinatura implicaria em inúmeros falsos alarmes durante a detecção de anomalias.

O tráfego histórico de cada *switch* OpenFlow é analisado a fim de se realizar a caracterização do comportamento usual da rede. Esse processo acontece para cada um dos atributos de fluxos apresentados na Tabela 5.1, e o resultado é um perfil de tráfego representado pelo vetor $\mathbf{d}$:

$$\mathbf{d} = \begin{bmatrix} \mathbf{d}_1 \\ \mathbf{d}_2 \\ \vdots \\ \mathbf{d}_n \end{bmatrix},$$

em que variável n indica o total de períodos de análise de um dia. Cada elemento do vetor contém as estatíticas do tráfego de um intervalo de análise, calculadas para cada atributo do fluxo. Portanto, durante o intervalo t, o valor esperado dos atributos é descrito por:

$$\begin{array}{rcccccc} & \text{bits} & \text{pacotes} & \text{H(srcIP)} & \text{H(dstIP)} & \text{H(srcPort)} & \text{H(dstPort)} \\ \mathbf{d}_t = & [d_1 & d_2 & d_3 & d_4 & d_5 & d_6] \end{array}$$

Após garantir que todos os atributos do fluxo coletados sejam representados por valores quantitativos, a extração do comportamento normal é iniciada. A criação do perfil de tráfego, que fornece a base para a detecção de anomalias, baseia-se na técnica de clusterização (agrupamento). A clusterização é uma ferramenta de mineração de dados utilizada para descobrir e quantificar semelhanças entre amostras de uma base de dados. Por meio da aplicação de um critério bem definido, procura-se agrupar as amostras de tal forma a minimizar a variância entre os elementos de um conjunto, denominado *cluster*, e maximizá-la em relação aos demais grupos. Como resultado do agrupamento, tem-se como resposta as amostras da base de dados dispostas em grupos, em que cada amostra é mais semelhante aos demais elementos do seu grupo do que qualquer outro.

5.3.1.1 Clusterização

Em relação à estrutura final do agrupamento gerado, a categorização dos algoritmos de clusterização pode ser dividida entre: agrupamento exclusivo e não-exclusivo. Um agrupamento exclusivo gera uma partição única do conjunto de dados. O termo partição remete à teoria de conjuntos e pode ser definida como uma divisão de um determinado conjunto em outros subconjuntos menores. Para que o agrupamento exclusivo seja efetivo, algumas regras devem ser respeitadas. Dado o conjunto de dados $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, uma partição de $\mathbf{X}$ em K *clusters* pode ser definida como $\mathcal{C} = \{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_K\}$ com

$K \leq n$, tal que [125]:

$$\bigcup_{j=1}^{K} = \mathbf{X} \tag{5.2}$$

$$\mathbf{C}_j \neq \emptyset \tag{5.3}$$

$$\mathbf{C}_j \bigcap \mathbf{C}_h = \emptyset, h \leq K \text{ e } j \neq h. \tag{5.4}$$

A equação (5.2) indica que todos os elementos do conjunto de dados participam do processo de agrupamento. A equação (5.3) determina que nenhum *cluster* deve ser vazio, ou seja, cada subconjunto gerado deve conter pelo menos uma amostra da base de dados. Por fim, a última regra ressalta que cada amostra pertença exclusivamente a um grupo, como mostra a equação (5.4). Essa característica também faz com que os agrupamentos do tipo exclusivo sejam conhecidos como *hard*, devido à inflexibilidade de uma amostra pertencer a dois ou mais subconjuntos diferentes. Um agrupamento não exclusivo, por sua vez, pode associar um mesmo objeto a vários *clusters* ou pode atribuir a cada um dos objetos um grau de pertinência em relação a cada um dos *clusters*. Como o intuito desse trabalho é utilizar a clusterização para minerar o tráfego histórico a fim de reconhecer padrões e tendências que indiquem o comportamento usual do tráfego, a abordagem de agrupamento exclusivo é mais apropriada. Dessa maneira, pequenas flutuações em relação ao tráfego normal, como anomalias pontuais e coletivas, podem ser reconhecidas e excluídas da assinatura gerada, evitando erros de classificação quando a assinatura estiver sendo utilizada para detecção.

Respeitando as três regras do agrupamento exclusivo, a clusterização ajuda a promover o conhecimento sobre o conjunto de dados utilizando aprendizado não-supervisionado, ou seja, a extração de padrões, comportamentos e características dos dados é realizada sem a necessidade de conhecimento prévio como, por exemplo, rótulos das amostras (*e.g.*, anômalo ou normal) [126].

Devido à importância da estratégia de agrupamento em vários campos de pesquisas, diversos algoritmos foram propostos na literatura para resolver o problema de clusterização [127, 128]. A criação de uma partição dos dados pode apresentar complexidade exponencial quando o número de *clusters* é grande e o encontro de uma solução viável torna-se um problema NP-difícil. Para solucionar essa inconveniência, modelos heurísticos têm se tornado cada vez mais populares dentre as técnicas de clusterização. Abordagens baseadas em métodos heurísticos utilizam mecanismos que exploram amplamente o espaço de soluções, fazendo com que as chances do encontro de um resultado ótimo sejam maiores [120]. Outro aspecto importante dos modelos heurísticos é a capacidade de convergência

dos resultados mais rápida do que por métodos convencionais, por exemplo, algoritmos determinísticos. Essa propriedade é mais evidente ao passo que o problema analisado se torna mais complexo computacionalmente.

Para a criação da assinatura do tráfego normal, o processo de clusterização utiliza-se de uma modificação da metaheurística *Ant Colony Optimization* [129]. As alterações ao algoritmo tradicional produziram a técnica chamada *Ant Colony Optimization for Digital Signature* (ACODS) [130], a qual cria partições das amostras do tráfego histórico em cada intervalo de análise para a descoberta do comportamento usual. De acordo com Jiang e Papavassiliou [100], o hábito das formigas de viver em grupos é essencialmente similar ao agrupamento de dados. Algoritmos baseados em comportamento de formigas têm vantagens naturais na aplicação da análise de *cluster*, tais como auto-organização, flexibilidade, robustez, ausência de necessidade de informação prévia e descentralização. Além desses benefícios, o ACODS possui mecanismos que dificultam a construção de soluções de ótimo local, que representam um grande problema em vários algoritmos de clusterização.

5.3.2 *Ant Colony Optimization for Digital Signature*

Deneubourg *et al.* [131] observaram, usando experimentos controlados, que um conjunto de formigas poderia encontrar o caminho mais curto entre sua colônia e uma fonte de alimento, marcando o caminho com uma substância chamada feromônio. Esse comportamento inspirou Marco Dorigo na criação da metaheurística *Ant Colony Optimization* (ACO) para resolver problemas de otimização combinatória.

A metaheurística ACO faz parte de um grupo de algoritmos chamado *Swarm Intelligence*, que é biologicamente inspirado em comportamentos de uma população de agentes assíncronos globais, que cooperam para encontrar uma solução ideal. Cada um desses indivíduos tem inteligência limitada, mas quando eles interagem entre si e com o ambiente, são capazes de realizar tarefas difíceis, como encontrar o caminho mais curto para uma fonte de alimento, organizar seu ninho, sincronizar seus movimentos como uma entidade coerente, única e com alta velocidade. Essa conquista é mais representativa, uma vez que as atividades são realizadas sem a presença de uma entidade centralizada para controlar o movimento (por exemplo, a rainha da colmeia) [132]. Mesmo que cada agente tenha a capacidade de construir uma solução viável, assim como uma formiga verdadeira possa encontrar um caminho entre a colônia e a fonte de alimento, soluções com maior qualidade são alcançadas através da cooperação entre indivíduos de toda a colônia [129].

Durante a busca por comida, as formigas começam a explorar os arredores da colônia depositando feromônio até atingirem a fonte de alimento. Inicialmente cada

formiga escolhe aleatoriamente um caminho para seguir, porém, depois de algum tempo, um caminho se torna mais atraente, uma vez que o acúmulo dessa substância nesse trajeto será maior. Essa situação implica na convergência de toda a colônia para esse caminho, preferencialmente o mais curto. No algoritmo, a interação entre os agentes é resultado da comunicação indireta no processo de leitura e escrita de variáveis na trilha de feromônios. Essa trilha atua como um canal de comunicação entre os agentes e, mudanças de valores modificam a forma como o ambiente (a situação do problema) é localmente percebido pelas formigas em função de todo o histórico da colônia. Ao contribuir com o depósito de feromônio na construção de uma solução, cada agente muda a forma como o problema é apresentado a outros agentes, resultando em um processo de aprendizado distribuído.

No ACODS, a simulação do comportamento de forrageio de formigas aliada à capacidade de encontrar o caminho mais curto entre a colônia e a comida é usada para caracterizar o tráfego da rede, classificando-o em grupos a partir dos dados de entrada. Inicialmente, os atributos de fluxos analisados são dispostos na 6-tupla $\mathbf{x}$, denominada de objeto ou elemento no processo de clusterização, pois representa um único ponto no espaço de busca. A Figura 5.4 ilustra os elementos classificados em dois *clusters*. Cada grupo contém seu próprio centroide, o qual é responsável por representar todos os elementos que pertencem ao seu grupo. É importante notar também que, diferentemente do exemplo tridimensional mostrado, durante a caracterização do tráfego a clusterização ocorre em um espaço euclidiano $\mathbb{R}^{|F|}$.

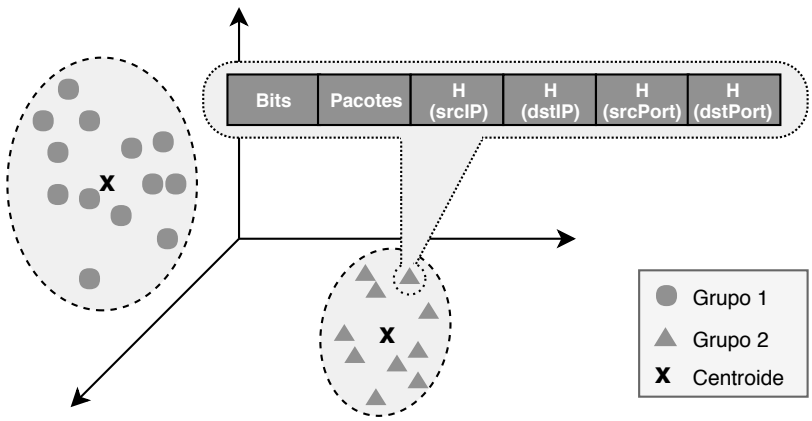

Figura 5.4 – Agrupando elementos durante o processo de clusterização. Cada um deles é composto por uma 6-tupla.

Usando estatísticas e probabilidades, as formigas percorrem o espaço de busca representado por um grafo $\mathcal{G}(\mathcal{V}, \mathcal{E})$, em que $\mathcal{V}$ é o conjunto finito de todos os vértices e $\mathcal{E}$ é o conjunto de arestas. Assume-se que os caminhos são formados entre o centroide de um *cluster* e cada 6-tuplas que será agrupada. As formigas são atraídas por caminhos que levam às soluções de partição mais promissoras. No ACODS, a solução gerada por cada formiga é avaliada por uma função denominada de função objetivo. A função objetivo

busca minimizar a distância intra-*clusters*, ou seja, a soma da distância entre as tuplas e o centroide do *cluster* ao qual pertencem. A distância intra-*cluster* é dada por:

$$J = \sum_{j=1}^{K} \sum_{\mathbf{x}_i \in C_j} dist\left(\mathbf{x}_i, \bar{\mathbf{x}}^{(j)}\right), \tag{5.5}$$

em que K é o número de *clusters*, $\mathbf{x}_i$ é a i-ésima 6-tupla do tráfego histórico, $\bar{\mathbf{x}}^{(j)}$ é o centro do *cluster* C_j, $dist\left(\mathbf{x}_i, \bar{\mathbf{x}}^{(j)}\right)$ é a distância Euclidiana entre $\mathbf{x}_i$ e $\bar{\mathbf{x}}^{(j)}$.

O ACODS é executado de forma iterativa, ou seja, devem haver critérios explícitos de parada durante a execução do programa. As atividades realizadas em cada iteração podem ser divididas em três grupos:

- **Construção de soluções:** Esta etapa consiste no movimento assíncrono e simultâneo das formigas de um vértice para outro vizinho na estrutura do grafo;
- **Busca Local:** Destina-se a avaliar soluções criadas pelas formigas no intuito de aprimorá-las. No modelo apresentado, esta atividade é usada para remover porções não promissoras das soluções;
- **Atualização do feromônio:** Este é o processo em que a trilha de feromônio τ é modificada. Os valores da trilha podem ser incrementados (quando formigas depositam feromônio nas arestas entre os vértices usados) ou pode ser diminuído. O aumento da concentração de feromônio é um fator essencial para a implementação do algoritmo, uma vez que direciona as formigas a buscar novos locais mais propensos a construção de uma boa solução.

De maneira geral, a operação do ACODS para clusterização é dividida em seis passos:

1. A trilha de feromônio τ é iniciada com um valor aleatório positivo pequeno [133];
2. Cada formiga κ seleciona uma tupla de índice i. Para definir a associação da tupla a um *cluster* C_j e construir o vetor de solução R, a formiga realiza a verificação de todas as possibilidades de movimento, ou seja, atribuição de $\mathbf{x}_i$ a cada um dos *clusters*. A tupla é associada ao *cluster* com maior probabilidade, dada pela equação:

$$j = \max_{j \in N_i} \left[\tau\left(i,j\right)\right]^{\alpha} \left[\eta\left(i,j\right)\right]^{\beta}, \tag{5.6}$$

   O termo $\left[\tau\left(\mathbf{x}_i, C_j\right)\right]^{\alpha}$ representa a concentração de feromônio na trilha (aresta) entre $\mathbf{x}_i$ e centro do *cluster* C_j, medindo quão boa é essa ligação para a criação de uma partição que minimiza a distância intra-*cluster*. Já $\left[\eta\left(\mathbf{x}_i, C_j\right)\right]^{\beta}$ indica o valor da

função heurística, calculada a partir do inverso da distância Euclidiana entre $\mathbf{x}_i$ e C_j. N_i é o conjunto de vértices vizinhos a $\mathbf{x}_i$ e que correspondem a um *cluster*.

3. Verificar a solução R da formiga κ. Caso $R(\kappa)$ não esteja completa, a formiga deve selecionar outra tupla e repetir o processo a partir do passo 2.

4. Calcular o quão boa é a solução gerada pela formiga κ usando a equação (5.5).

5. Atualizar a trilha de feromônio. Essa etapa conduz à criação de soluções mais aprimoradas a cada iteração, pois direciona os agentes na busca de novas soluções usando caminhos promissores estabelecidos anteriormente. Os L caminhos que conduzem às melhores soluções são reforçados depositando-se o feromônio como mostrado na equação (5.7), pois espera-se que eles sejam usados na construção de soluções cada vez melhores. Em contraste, os caminhos, ou seja, as ligações tupla-*cluster* de soluções menos promissoras são lentamente esquecidas pela colônia por meio da evaporação do feromônio. Isto proporciona um excelente tempo de convergência e contribui para a construção de soluções não prematuras que podem ficar presas a ótimos locais.

$$\tau_{i,j}\,(t+1) = (1-\rho)\,\tau_{ij}\,(t) + \sum_{l=1}^{L} \Delta\tau(\mathbf{x}_i, C_j) \tag{5.7}$$

   A constante ρ define a taxa de evaporação do feromônio e tem valor $\rho \in (0, 1]$. A variável t indica a iteração corrente. A quantidade de feromônio $\Delta\tau(\mathbf{x}_i, C_j)$ depositada na aresta que liga a tupla ao centro do *cluster* é calculada como o inverso de J, dividido pela quantidade de tuplas.

6. O último passo corresponde à verificação dos critérios de paradas. Para tanto, duas possibilidades podem ser assumidas e quando uma delas é satisfeita, o algoritmo retorna a melhor solução encontrada durante toda sua execução e, é encerrado imediatamente. Na primeira, calcula-se a média da função objetivo das soluções construídas por cada uma das formigas da colônia em duas iterações consecutivas. Caso a diferença da média das duas iterações seja menor que um limiar pequeno, o algoritmo cessa sua execução. Quando essa situação acontece, toda colônia já convergiu para uma única solução e nenhuma outra é possível de ser criada a partir de então. Na segunda abordagem, para evitar que o processamento ocorra indefinidamente, o algoritmo é encerrado quando um limite máximo de iterações é alcançado. Quando nenhum dos dois critérios de parada é satisfeito, uma iteração é completada e o algoritmo volta à execução a partir do passo 2.

Após a clusterização, o tráfego de um intervalo de análise é separado em grupos, onde cada tupla é mais semelhante aos dados de seu grupo do que qualquer outro.

Devido à alta similaridade do tráfego de rede, a maioria das informações representa um comportamento semelhante e algumas regiões densas no espaço de pesquisa são formadas. Como trabalhos anteriores discutiram [21], em uma partição com *clusters* de vários tamanhos, os grupos mais esparsos e com menos elementos podem ser considerados anômalos (embora não maliciosos) e os mais representativos são considerados normais. As instâncias pertencentes a *clusters* com quantidade de elementos, e_j, abaixo de um limite γ são consideradas incomuns e são descartadas do padrão normal de tráfego. Essa etapa compõe a estratégia de busca local, que evita, ou torna mínimo possível, o envolvimento de tráfego incomum na composição da assinatura.

Pseudocódigo 5.1 - ACODS usado para criação da assinatura digital do tráfego

Entrada: Conjunto de atributos de volume e dos campos do cabeçalho do pacote coletados a partir de uma base de dados histórica, número de *cluster* K, número máximo de iterações I

Saída: matriz **D**

```
1: for t ← 1 : n do
2:     for i ← 1 : I do
3:         Criar solução
4:         Avaliar solução utilizando a função objetivo
5:         Atualizar a trilha de feromônio
6:     Calcular o centro de cada cluster da melhor solução encontrada
7:     for j ← 1 : K do
8:         if e_j < γ then
9:             descartar o cluster C_j
10:    d_t ← média ponderada entre os cluster                ▷ eq. (5.8)
11: return D
```

As etapas para construção da assinatura digital do tráfego estão apresentadas no Pseudocódigo 5.1. O resultado desse algoritmo descreve a combinação dos grupos mais representativos do ambiente clusterizado. Para obter a assinatura do tráfego $\mathbf{d}_t$ para cada intervalo de análise t, a média ponderada é calculada entre os grupos tal como na equação (5.8), assumindo que E é a quantidade total de elementos agrupados.

$$\mathbf{d}_t = \frac{\sum_{j=1}^{K} \bar{\mathbf{x}}^{(j)} e_j}{E} \tag{5.8}$$

Após a caracterização do comportamento normal do tráfego, a assinatura digital é comparada ao tráfego recentemente coletado para o reconhecimento de anomalias, como mostra a Figura 5.5. Cada gráfico apresenta informações sobre um atributo de fluxo analisado. A área em verde indica os valores medidos para o atributo ao longo do dia, a cada intervalo de trinta segundos. Já a linha azul apresenta a assinatura digital, gerada a partir do histórico do tráfego e que fornece uma previsão sobre o comportamento esperado do atributo. Caso uma discrepância seja encontrada no tráfego capturado e no previsto,

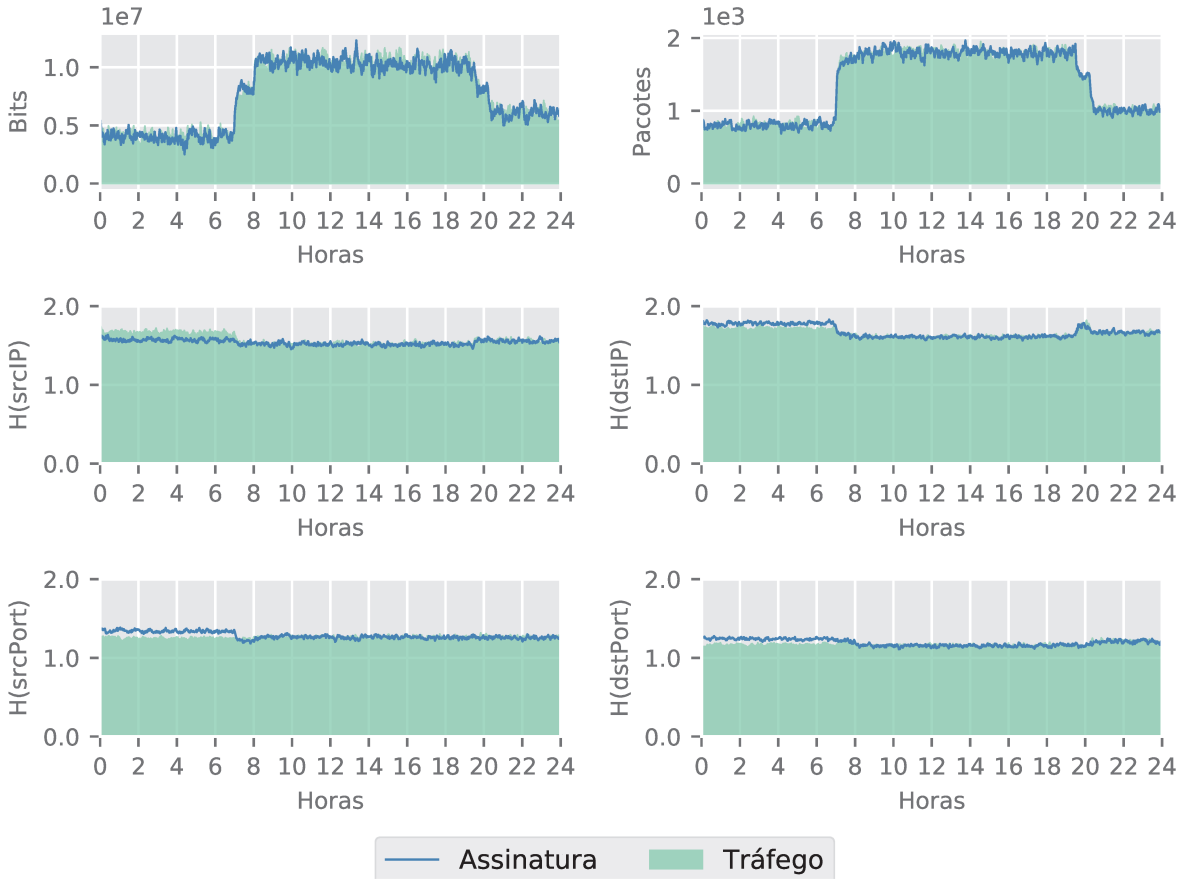

Figura 5.5 – Tráfego e sua respectiva assinatura de comportamento normal gerada pelo ACODS.

ela será detectada pelo módulo de detecção e ocorrerá a instauração de ações de mitigação para evitar a propagação de anomalias de acordo com seu tipo (por exemplo, um dos subespaços de S). É importante observar que o monitoramento da rede e a caracterização do tráfego são processos simultâneos, uma vez que as medições realizadas durante os intervalos de análise recentemente passados são incorporadas automaticamente no tráfego histórico para processamento posterior.

5.3.2.1 Parâmetros do ACODS

A metaheurística ACODS foi proposta para caracterização do comportamento normal do tráfego em redes tradicionais e foi validada em vários cenários [51, 118, 130]. Um dos aspectos mais importantes da sua implementação é a calibração dos valores de seus parâmetros, pois grande parte do sucesso na busca de soluções depende deles. Esses valores foram definidos por meio de testes exaustivos, variando-se os valores entre os limites aceitáveis para obtenção de bons resultados. Os valores desses parâmetros foram mantidos para os resultados desta tese e serão discutidos na próxima seção.

O valor de γ foi definido como 15% do total de elementos a serem clusterizados

em cada iteração. A intenção do uso desse parâmetro é produzir uma caracterização sem a supervisão humana. Como a base histórica analisada durante a criação da assinatura digital pode conter eventos que não correspondam ao comportamento usual da rede, cabe à busca local julgar se eles comporão a assinatura.

O parâmetro ρ tem um impacto importante durante a execução do ACODS. Ele permite ajustar o tempo de convergência das soluções por meio da evaporação do feromônio. Caso seu valor seja muito pequeno, a convergência será imediata e não haverá possibilidade de criação e avaliação de novas soluções, potencialmente promissoras. Por outro lado, valores acentuados para esse parâmetro tornam a convergência lenta, exigindo um número maior de iterações e, consequentemente, aumentando o tempo de geração da assinatura do tráfego normal. Neste trabalho o valor $\rho = 0,1$ foi atribuído à taxa de evaporação de feromônio. Esse valor foi definido por meio da análise da influência desse parâmetro na construção das soluções. O mesmo valor pode ser encontrado em outros trabalhos que utilizam o ACO [134, 135].

Os últimos dois parâmetros estão relacionados aos agentes criadores de soluções. A quantidade de formigas é definida como 50% do total de elementos a serem clusterizados. Em uma iteração do ACODS, cada formiga cria sua própria solução, as quais sofrem ação da evaporação do feromônio. No entanto, as L melhores soluções são reforçadas com o acúmulo de feromônio. Esse último parâmetro foi configurado como 25% das soluções criadas a cada iteração, assegurando que as melhores soluções sejam mais atrativas na próxima iteração. Valores similares são definidos por Shelokar *et al.* [136], que enfatizam a utilização de valores de parâmetros proporcionais aos dados de entrada para a construção de uma metaheurística robusta baseada no ACO.

5.3.2.2 Validação do número de *clusters*

Uma vez finalizado o processo de clusterização, é interessante verificar a disposição dos *clusters* criados, bem como avaliar o número adequado de grupos utilizados para a manipulação do conjunto de dados. Dois métodos são aplicados para esse propósito. O primeiro é o índice Dunn [137], cujo objetivo é identificar o conjunto de *clusters* que estão compactos e melhor separados. Dessa forma, as soluções para o agrupamento apresentam melhores resultados quando a distância entre os *clusters* é grande, enquanto as distâncias dos elementos em relação ao centro do grupo ao qual pertencem são pequenas. Essa medida é calculada através da razão entre a distância mínima intracluster e a distância máxima intercluster como demonstrado pela equação 5.9.

$$\delta = \frac{d_{min}}{d_{max}}, \tag{5.9}$$

em que d_{min} é a menor distância entre dois elementos de diferentes *clusters*, enquanto d_{max} define a maior distância entre dois objetos pertencentes ao mesmo *cluster*. O índice Dunn pode assumir valores no intervalo $0 \leq \delta < \infty$, e quanto maior esse valor, melhor clusterizado está o conjunto de dados.

O segundo método utilizado para a validação e interpretação dos *clusters* neste trabalho é o Silhouette [138]. Esta métrica proporciona uma representação gráfica do agrupamento de dados, permitindo a análise dos elementos dentro dos grupos. Os principais requisitos para a sua construção são: i) a obtenção de um particionamento dos dados, utilizando uma técnica de clusterização e ii) uma medida de proximidade entre os objetos do conjunto de dados. Assim, o gráfico Silhouette é útil quando a métrica de aproximação é apresentada em uma escala (como no caso da distância Euclidiana) e quando grupos compactos e claramente separados são procurados. Os gráficos da Figura 5.6 apresentam testes com diferentes valores de *clusters* utilizando o método Silhouette.

Nos gráficos da primeira coluna da Figura 5.6, o eixo y apresenta o número de grupos usados na clusterização, e o eixo x, o valor da função de Silhouette. Cada elemento i a ser clusterizado recebe o valor $-1 \leq s(i) \leq 1$ correspondente à função Silhouette, calculada em relação ao *cluster* ao qual ele pertence. De acordo com o valor dessa função, três situações são possíveis. Quando $s(i)$ está próximo a 1, a similaridade do elemento i em relação aos demais elementos do mesmo *cluster* é grande. Isso significa que o objeto foi associado corretamente a um conjunto. Quando esse valor tende a zero, o elemento poderia ser atribuído a mais de um grupo. A pior situação acontece quando $s(i)$ se aproxima do valor -1. Nesse caso, o elemento i foi mal classificado e deve ser associado a outro *cluster*.

Como pode ser conferido nos gráficos Silhouette, quando mais de 3 *clusters* foram utilizados, o valor de $s(i)$ de cada elemento tendeu a diminuir. Para facilitar a visualização do decremento desse valor, foi calculada a média da função Silhouette de todos os elementos e apresentada no gráfico com uma linha vertical vermelha. Nota-se também que o aumento no número de *clusters* culminou na criação de grupos formados por pequenas quantidades de elementos. Esses pequenos conjuntos apresentaram objetos com baixo valor da função de avaliação do Silhouette, pois, devido à fragmentação do conjunto de dados, poderiam ser agrupados em um outro *cluster*. Essa tendência pode ser observada nos gráficos da segunda coluna. Neles é apresentada a partição gerada de acordo com o número de *cluster* avaliado. Para melhor visualização, apenas os dois primeiros atributos do tráfego, bits e pacotes, são mostrados. É possível verificar, por exemplo, que para a configuração de 4 *clusters*, os elementos pertencentes ao *cluster* 1 possuem o maior valor de Silhouette enquanto os grupos 0 e 2 poderiam facilmente serem agrupados ao conjunto 3.

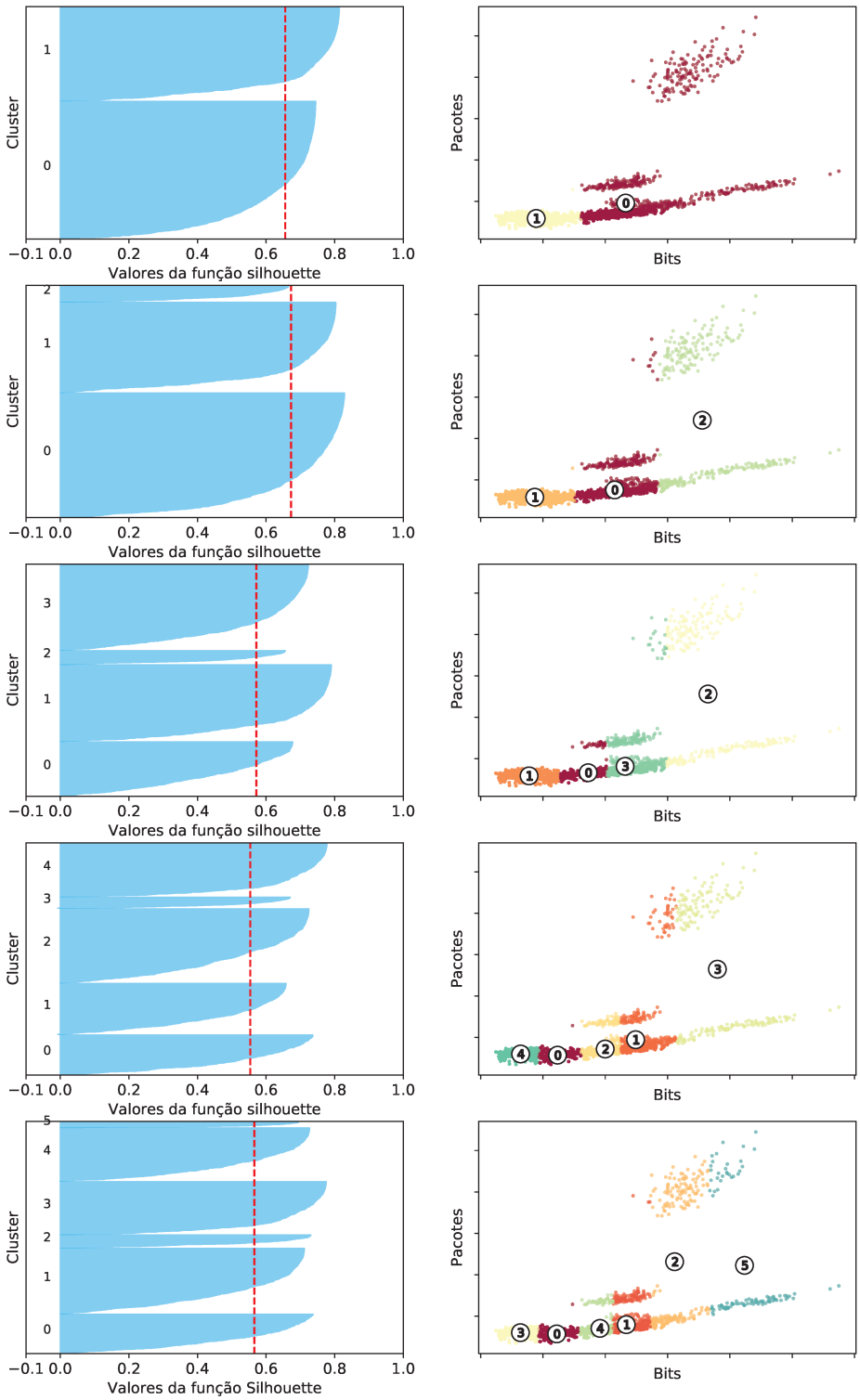

Figura 5.6 – Validação do número de *clusters* usando Silhouette.

O gráfico mostrado na Figura 5.7 apresenta o resultado da validação do número de *clusters* necessários para caracterizar corretamente o comportamento da rede. Para essa finalidade, são expostos os valores dos testes de Índice Dunn e Silhouette para cada configuração de K. Ambos os métodos de avaliação foram aplicados aos conjuntos de dados utilizados para treinamento e avaliação do ACODS.

Figura 5.7 – Avaliação do número de *clusters* utilizado para extração de padrões do comportamento do tráfego.

O resultado do Silhouette é representado como a média dos valores $s(i)$ dos elementos clusterizados. As medidas de avaliação apresentaram resultados com escalas de valores distintas, entretanto, é possível observar que as curvas expressam uma tendência em relação ao número de *clusters* favorável para a criação da partição dos dados. Dessa forma, resultados semelhantes foram encontrados em ambos os testes, indicando que três grupos são necessários para reconhecer e classificar os diferentes comportamentos do tráfego.

5.3.2.3 Base histórica usada para geração da assinatura

O comportamento do tráfego é composto por ciclos diários, distintos e que são resultantes de eventos cotidianos exercidos pelos usuários da rede ao longo dos dias da semana. As diferenças de comportamento se tornam mais evidentes quando são comparadas os tráfegos dos dias úteis com os gerados aos fins de semanas ou feriados. O mecanismo destinado à caracterização normal do tráfego deve ser capaz de reconhecer essas peculiaridades. Nesta tese, o ACODS explora o tráfego histórico da rede para a criação de uma assinatura de comportamento normal para cada dia da semana. Essa abordagem tem o intuito de eliminar os erros provenientes da análise simultânea de dias com diferentes comportamentos, tal como dias úteis e fins de semana.

Para a geração das assinaturas de um dia específico são utilizadas amostras do tráfego desse dia, porém de semanas anteriores, coletadas durante o monitoramento e armazenadas em uma base de dados. Por exemplo, para se estabelecer o comportamento esperado para uma segunda-feira, o tráfego coletado de outras segundas-feiras imediatamente anteriores a ela é utilizado. Após o estabelecimento da assinatura e monitoramento em tempo real da segunda-feira analisada, as amostras coletadas ao longo do dia também passarão a compor a base histórica. Essa característica confere à abordagem de caracteri-

zação uma capacidade de se adaptar a mudanças no ambiente que está sendo monitorado, uma vez que o histórico é constantemente atualizado e, consequentemente, novos comportamentos são inseridos na base.

Para definir o número exato de semanas utilizadas pelo ACODS para a extração e padrões de comportamento do tráfego, foram geradas assinaturas utilizando de duas a dez semanas consecutivas e anteriores ao dia tomado como aquele que a assinatura deveria descrever. O critério de avaliação adotado para esse teste corresponde ao erro da previsão gerado a partir de um número arbitrário de semanas em relação ao movimento do tráfego observado. Para essa tarefa foi utilizada a métrica *Normalized Mean Square Error* (NMSE), amplamente aplicada para mensurar o erro entre séries temporais, em que uma delas contém os valores esperados (por exemplo, tráfego coletado) e a outra exibe o que foi calculado a partir de um procedimento específico (por exemplo, assinatura). O erro NMSE foi calculado para cada um dos atributos do fluxo analisados nesta tese e os valores apresentados na Figura 5.8 correspondem à média desses erros.

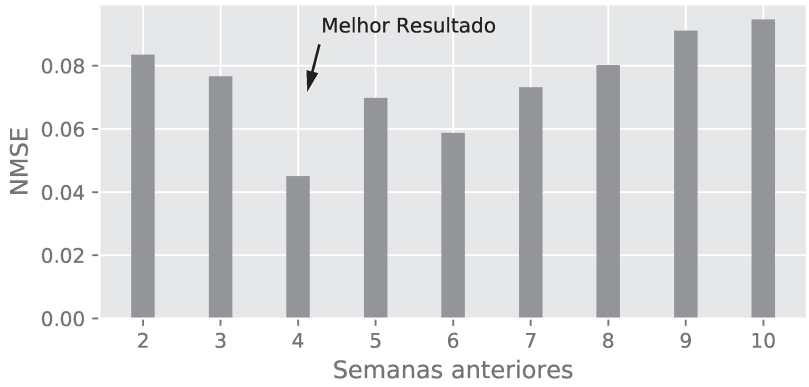

Figura 5.8 – Avaliação do número de semanas utilizadas para a criação da assinatura. Quatro semanas anteriores resultaram na melhor caracterização do tráfego.

O melhor resultado encontrado, isto é, o menor NMSE foi atingido quando quatro semanas anteriores foram usadas para criação da assinatura, resultando em um erro de 0,045 em relação ao tráfego monitorado. Essa quantidade de dias que serviram como base histórica foi o ponto de convergência mostrado pelos resultados. Valores inferiores a esse número resultaram em erros elevados, que foram diminuindo até se encontrar o melhor resultado. Analogamente, quanto maior a base histórica usada, maior o erro observado. A única exceção foi verificada quando cinco semanas anteriores foram usadas para descrever o comportamento normal do dia corrente. Nesse caso, a assinatura apresentou um erro aproximadamente 18% maior do que quando seis semanas foram usadas para caracterização do tráfego. Após analisar esse evento, foi possível constatar que o quinto dia usado como histórico apresentou um comportamento ligeiramente diferente dos demais, culminando no aumento do erro.

Além dos resultados apresentados pelo teste NMSE, a escolha de utilizar quatro semanas é justificada pelo monitoramento automático e contínuo do tráfego. Por não existir um agente externo que valide a assinatura gerada, deve-se levar em consideração o impacto que uma mudança repentina do comportamento do tráfego causará à assinatura. Dessa maneira, a quantidade de semanas utilizadas para geração do comportamento normal não pode ser tão pequena ao ponto de que qualquer alteração em poucos dias analisados seja incluída na assinatura, como, por exemplo, falhas esporádicas, feriados ou recessos. Em contrapartida, esse valor não pode ser demasiadamente elevado, pois eventos legítimos como *backups* realizados pela equipe de gerência poderiam demorar meses para serem incorporados ao perfil normal de comportamento.

Ainda sobre os parâmetros usados para a geração da assinatura digital, o ACODS foi inicialmente idealizado para utilizar as medições de tráfego observadas no intervalo de um minuto nas redes tradicionais. Nesta tese, como as requisições de estatísticas são feitas aos *switches* a cada trinta segundos, o ACODS utiliza duas amostras de cada dia da base histórica para a criação do perfil normal de comportamento. Dessa maneira, para definição da assinatura no instante t, são utilizadas as amostras t e $t-1$ dos dias das semanas anteriores. Mais especificamente, a entrada do ACODS em cada iteração corresponde a:

$$\bigcup_{i=1}^{4} \left\{ \mathbf{x}_{t-1}^{i}, \mathbf{x}_{t}^{i} \right\}, \tag{5.10}$$

em que $\mathbf{x}_t^i$ representa a 6-tupla composta pelas medições do tráfego que foram coletadas no instante t na i-ésima semana anterior. Ao final da execução, o comportamento normal previsto para os atributos do fluxo a cada intervalo de 30 segundos é descrito pela assinatura, disposta em um vetor $\mathbf{d}$ de 2880 elementos.

5.3.3 Identificação de anomalias

A classificação do tipo de evento anômalo ocorrido no tráfego é indispensável para a gerência da rede, pois permite que o problema seja identificado corretamente. A Tabela 5.2 lista algumas das anomalias comumente encontradas em tráfego de *backbone* e seus efeitos sobre as medições de tráfego. Um índice k é atribuído a cada anomalia que a abordagem proposta pode reconhecer. O símbolo "+" indica que a medição do atributo aumenta durante a anomalia, "−" representa um valor decrescente. Os atributos com notação "n.c." não têm seu valor alterado significativamente com a ocorrência da anomalia, enquanto "+−" indica que os valores podem aumentar ou diminuir.

De acordo com a Tabela 5.2, cada evento anômalo exerce um efeito sobre os atributos do tráfego. Portanto, para maximizar a taxa de detecção correta de anomalias,

Tabela 5.2 – Anomalias e seus efeitos nos atributos do tráfego.

k	Anomalia	Atributos afetados					
		Bits	Pacotes	H(srcIP)	H(dstIP)	H(srcPort)	H(dstPort)
1	SYN *flooding*	+−	+	+−	−	+−	−
	UDP *flooding*	+−	+	+−	−	+−	+
	ICMP *flooding*	+−	+	+−	−	+−	+−
2	*Port scan*	n.c.	+	−	−	+	+
3	Flash Crowd	+	+	+	−	−	−

a relevância de cada um desses atributos para a identificação de um ataque deve ser especificada. Modelos de regressão são candidatos adequados para ponderar as mudanças em cada atributo que levam à detecção de um evento.

Em um modelo de regressão linear simples ou múltipla, a variável dependente Y é uma variável aleatória contínua. No entanto, em algumas situações, essa variável é qualitativa ou é expressa por duas ou mais categorias, admitindo dois ou mais valores. Quando isso ocorre, o método dos mínimos quadrados não fornece estimadores viáveis. Essa restrição se aplica ao problema de identificação da anomalia desta tese, em que a variável dependente pode assumir qualquer rótulo de anomalia k apresentado na tabela ou ainda o valor $k = 0$, para classificação de tráfego normal. Uma alternativa ao modelo de regressão linear é a regressão logística, a qual permite o uso de um modelo de regressão para calcular ou prever a probabilidade da ocorrência de um determinado evento.

O modelo de regressão logística multinomial (RLM) é um caso específico de regressão logística usado para dados em que a variável dependente é desordenada ou politômica[1] e as variáveis independentes são preditores contínuos ou numéricos [139]. A RLM induz um classificador capaz de distinguir diferentes classes anômalas usando amostras de treinamento rotuladas. Neste estudo, a RLM calcula a importância das variações nas medições de tráfego para diagnosticar anomalias e, com base nessa variação, atribui uma razão de probabilidade para cada estado no espaço de eventos S.

Dado um conjunto de $r + 1$ variáveis independentes denotadas por $\mathbf{X} = \{X_0, X_1, ..., X_r\}$, assumindo valores $\mathbf{x} = [x_0, x_1, ..., x_r]$ com $x_0 = 1$ e uma variável aleatória Y que pode assumir valores $y \in \{0, 1, ..., q\}$. Ao contrário de um modelo logístico binário, no qual uma variável dependente tem apenas duas opções (por exemplo, normal ou anômala), a variável dependente em uma regressão logística multinomial pode ter mais de duas opções organizadas categoricamente, em que uma dessas categorias representa a categoria de referência. Neste trabalho, $k = 0$ ("normal") é a categoria de referência e, $k = 1$ ("DDoS"), $k = 2$ ("*port scan*") e $k = 3$ ("*flash crowd*") são as categorias restantes.

[1] Variável qualitativa que pode ser descrita por mais de duas categorias.

Dadas as categorias acima, pode-se descrever a função *logit* (ou dependência) comparando $Y = k$ e $Y = 0$ para $k \in \{1, ..., q\}$. A função *logit* é denotada por:

$$\begin{aligned} g_k(\mathbf{x}) &= \ln\left[\frac{P\left(Y_i = k|\mathbf{x}\right)}{P\left(Y_i = 0|\mathbf{x}\right)}\right] \\ &= b_{k0}x_{k0} + b_{k1}x_1 + \cdots + b_{kr}x_r \\ &= \mathbf{x}^\top \mathbf{b}_k, \qquad k \in \{0, \ldots, q\}, \end{aligned} \tag{5.11}$$

em que $\mathbf{b}_k = [b_{k0}, ..., b_{kr}]^\top$ é um vetor de parâmetros com valores desconhecidos, que são os pesos associados a cada atributo do fluxo e $x_{k0} = 1$.

A partir da equação 5.11, pode-se aplicar propriedades dos logaritmos para se obter:

$$\exp\left[g_k(\mathbf{x})\right] = \frac{P(Y_i = k|\mathbf{x})}{P(Y_i = 0|\mathbf{x})}, \qquad k \in \{0, \ldots, q\}$$

$$P(Y_i = k|\mathbf{x}) = \exp\left[g_k(\mathbf{x})\right] P(Y_i = 0|\mathbf{x}). \tag{5.12}$$

Usando a propriedade de que a soma da probabilidade de ocorrência de cada um dos $q + 1$ eventos deve resultar em 1, obtém-se:

$$P(Y_i = 0|\mathbf{x}) = 1 - \sum_{k=1}^{q} P(Y_i = 0|\mathbf{x}) \exp[g_k(\mathbf{x})] = \frac{1}{1 + \sum_{k=1}^{q} \exp[g_k(\mathbf{x})]}. \tag{5.13}$$

Substituindo a equação (5.13) em (5.12), pode-se determinar as demais probabilidades:

$$\pi_k = \frac{\exp[g_k(\mathbf{x})]}{1 + \sum_{k=1}^{q} \exp[g_k(\mathbf{x})]}, \tag{5.14}$$

onde π_k denota $P(Y_i = k|\mathbf{x})$.

Depois de caracterizar o comportamento normal, e como eventos incomuns são identificados, definimos o problema de classificação de anomalias da seguinte forma: dado um intervalo de tempo t, uma medida de tráfego recém-coletada F_t é atribuída à classe k se

$$k = \arg\max_{\ell} \left(\pi_\ell \left(\mathbf{\Theta}_t\right)\right), \tag{5.15}$$

onde $\pi_\ell(.)$ é uma razão de probabilidade atribuída a cada anomalia ℓ, que é devidamente identificada pelo mecanismo de detecção. O vetor $\mathbf{\Theta}_t$ é dado por:

$$\Theta_t = \mathbf{\Lambda}^{-1}|\mathbf{d}_t - F_t|, \tag{5.16}$$

em que $\mathbf{d}_t$ é o perfil esperado, F_t corresponde à medição de tráfego atual e Λ^{-1} é o inverso de uma matriz diagonal cujos elementos são F_t. Cada valor $\mathbf{\Theta}_t = \left[\theta_{t1}, \theta_{t2}, \ldots, \theta_{t|F|}\right]$ denota o erro relativo entre o esperado e a atual medição de tráfego para cada atributo do fluxo.

Embora a abordagem possa detectar as anomalias mostradas na Tabela 5.2, há algumas anormalidades de rede legítimas que afetam os recursos de tráfego da mesma maneira que uma anomalia maliciosa. Para evitar a interrupção de eventos de tráfego benignos, o sistema de detecção mantém uma lista de desbloqueio para armazenar endereços IP e portas cujo comportamento incomum pode, na verdade, ser um tráfego livre de anomalias. Assim, o módulo primeiro verifica se há endereços IP e portas relacionados às anomalias na lista de permissões e, em seguida, as rotinas de mitigação são aplicadas, se necessário.

Quando uma anomalia é reconhecida, é realiza uma análise mais ativa sobre os fluxos suspeitos, em que os endereços IP e as portas que compõem um cenário incomum podem ser inspecionados durante várias janelas de tempo. Endereços identificam os *hosts* envolvidos na comunicação anômala enquanto as portas indicam serviços que podem estar comprometidos. O total de pacotes e bits enviados e recebidos por cada um desses *hosts* é monitorado, bem como o número de conexões estabelecidas por eles. Se esses valores estiverem contribuindo para o desvio do comportamento esperado de um *switch* monitorado, o módulo de mitigação toma medidas para que a rede retorne à sua operação normal.

Pseudocódigo 5.2 - Identificação de Anomalias

Entrada: tráfego atual F_t, medições esperadas do tráfego (assinatura) $\mathbf{d}_t$
Saída: índice da anomalia k

1: Computar $\mathbf{\Theta}_t$ usando F_t e $\mathbf{d}_t$ ▷ eq. (5.16)
2: Calcular a função *logit* usando $\mathbf{\Theta}_t$ ▷ eq. (5.11)
3: Calcular a probabilidade de $\mathbf{\Theta}_t$ ser a classe de referência ▷ eq. (5.13)
4: Medir a probabilidade de $\mathbf{\Theta}_t$ pertencer à classe anômala $k \in \{1, \ldots, q\}$ ▷ eq. (5.14)
5: Atribuir $\mathbf{\Theta}_t$ à classe de anomalias k com a maior probabilidade calculada ▷ eq. (5.15)
6: **return** k

Todos os procedimentos necessários para a identificação de anomalias estão apresentados no Pseudocódigo 5.2. Cada etapa é organizada de modo que o classificador RLM possa identificar anomalias no tráfego de rede SDN após o treinamento com os pesos de cada atributo de fluxo. Depois que o módulo de detecção de anomalias recebe os dados do módulo de coleta do tráfego, o erro relativo entre o tráfego coletado F_t e

o comportamento esperado $\mathbf{d}_t$ é calculado e armazenado em $\mathbf{\Theta}_t$. Então, a função *logit* é calculada usando o erro previamente calculado e os pesos obtidos do treinamento RLM. Na linha 3 do pseudocódigo, obtemos a probabilidade de o tráfego coletado pertencer à classe normal, e a probabilidade de o tráfego ser considerado uma anomalia da classe DDoS, *port scan* ou *flash crowd* é calculada em seguida. Na linha 5, o algoritmo atribui ao tráfego analisado a classe com maior grau de pertinência e, finalmente, o rótulo da classe é retornado na linha 6. O resultado do pseudocódigo para cada intervalo de tempo é mostrado na Figura 6.3(a), no capítulo de resultados e, de acordo com essas saídas, a eficácia do método de detecção pode ser avaliada como mostrado nas Figuras 6.4 e 6.5.

Para treinar o classificador RLM, foram usadas amostras de ataques conduzidos na rede da Universidade Estadual de Londrina, relatadas no trabalho de Hammamoto *et al.* [123] e Carvalho *et al.* [130]. Três tipos de anomalias de diferentes intensidades foram empregados: DDoS, *port scan* e *flash crowd*. Cada observação no conjunto de treinamento passou por todos os estágios do cálculo da regressão logística multinomial, e os pesos apropriados $\mathbf{b}_k = [b_1, b_2, \ldots, b_3]$ foram calculados usando o algoritmo de otimização Broyden–Fletcher–Goldfarb–Shanno (BFGS). A viabilidade de $\mathbf{b_k}$, da equação (5.11) é calculada para cada 6-tupla $\mathbf{x}$ nos dados de treinamento usando uma função de custo. Essa função mede a distância entre a classe prevista e classe real usando os pesos calculados. O ajuste de $\mathbf{b}_k$ é um processo iterativo e o procedimento de otimização termina quando o valor da função de custo é pequeno. Para o propósito dessa tese, o algoritmo de otimização encerra sua execução quando a função custo atinge um valor inferior a 10^{-4}. A saída após o treinamento do classificador de regressão logística multinominal são os pesos ajustados de tal maneira que propiciem a classificação correta das amostras do conjunto de dados de treinamento. Posteriormente, esses pesos serão utilizados para classificação dos eventos anômalos de acordo com os desvios observados em relação à assinatura digital.

5.4 Módulo de Mitigação

Anomalias podem prejudicar a operação da rede, portanto, contramedidas devem ser tomadas para anular qualquer evento que possa afetar a qualidade dos serviços fornecidos aos usuários. Os pontos centrais de uma estratégia de resiliência são o gerenciamento e a reconfiguração de mecanismos de detecção e restauração, que funcionam como componentes autônomos na infraestrutura de rede [140]. Embora os esforços de detecção de anomalias, como monitoramento e classificação de tráfego, forneçam o reconhecimento e a categorização de ataques, os mecanismos de reparo podem ser usados na atenuação dessas ameaças. Dessa forma, o ecossistema proposto implementa rotinas para localização de fluxos e equipamentos que causam ou são afetados por anomalias e políticas de

mitigação são aplicadas a eles.

Políticas são uma coleção de regras que determinam como as decisões atuais e futuras serão tomadas. As políticas possuem ações que são executadas quando eventos específicos ocorrem e certas condições são satisfeitas. Assim, as políticas são definidas como um modelo *Evento-Condição-Ação*, que é adequado para o gerenciamento dinâmico de políticas [141]. Cada *evento* refere-se a um ataque ou incidente específico e está associado a um conjunto de regras. As regras são descritas como um conjunto de *condições* que correspondem ao contexto em que o ataque ou incidente ocorre. Finalmente, *ação* é essencialmente uma resposta a ser aplicada aos fluxos anômalos identificados.

O uso de políticas pode fornecer três benefícios [142]. Primeiro, os administradores predefinem as políticas e as armazenam em um repositório. Quando ocorre um evento, o sistema solicita e acessa essas políticas automaticamente, sem intervenção manual. Em segundo lugar, a descrição formal das políticas permite a análise autônoma e a verificação de sua consistência. Terceiro, usando detalhes técnicos abstratos, as políticas podem ser inspecionadas e alteradas em tempo de execução sem modificar a implementação do sistema.

Uma lista de políticas implementadas é mostrada na Tabela 5.3. Em suma, nosso foco está em aplicar a política para mitigar as anomalias detectadas. Para esse propósito, entradas de fluxo contendo o endereço IP e a porta aprendida no módulo de detecção de anomalias são inseridas nas tabelas de encaminhamento de *switches*, juntamente com ações que devem ser executadas nos novos pacotes que correspondem a esses fluxos, como mostra a Figura 5.9. As principais ações incluem encaminhar, descartar ou modificar campos do cabeçalho do pacote.

A política *block_src_IP* suspende o tráfego de entrada de um *host* malintencionado (por exemplo, o ataque *port scan*) enquanto *block_flow* bloqueia a comunicação de um determinado *host* para um serviço específico no endereço IP de destino. O módulo de mitigação anexa uma ação de encaminhamento em *load_balance* para distribuir a carga do tráfego de rede gerado por um ataque DDoS através de vários servidores com o objetivo de evitar a degradação dos serviços nos servidores monitorados. Para

Tabela 5.3 – Lista de políticas implementadas.

Política	Descrição
Block_src_IP	Bloqueia o tráfego de um endereço IP com mau comportamento
Block_flow	Descarta pacotes que pertencem a um fluxo específico
Load_balance	O tráfego é distribuído para outros servidores durante atividades de rede intensivas

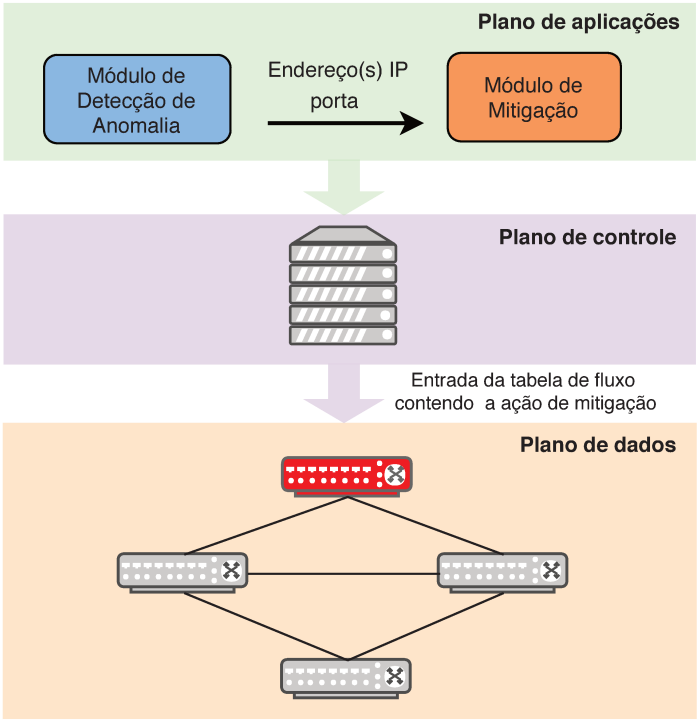

Figura 5.9 – Plano de controle instruindo um *swtich* para lidar com o tráfego anômalo.

promover a recuperação após a falha (*failover*) de servidores de alta disponibilidade, o administrador da rede pode indicar quais outros servidores podem executar a mesma tarefa que um servidor sobrecarregado. Esta política usa um algoritmo Round-Robin para garantir que todos os servidores usados para balancear a carga serão visitados em um *loop*, determinando qual servidor usar de acordo com o último selecionado. Depois que a contramedida é projetada, o módulo de mitigação transfere a diretiva para o *switch* através do controlador de rede usando o protocolo OpenFlow.

As entradas de fluxo correspondentes às regras de atenuação são atribuídas com maior prioridade de execução do que outras entradas ativas na tabela de encaminhamento. Além disso, é atribuído um período de tempo (*idle time*) para que esses fluxos permaneçam ativos no *switch* após a última correspondência de um pacote anômalo. Esse valor é definido como trinta segundos, ou seja, mesmo tamanho do intervalo de análise do tráfego. Essa estratégia permite que a rotina de mitigação permaneça em funcionamento enquanto a anomalia estiver ativa.

5.5 Módulo de relatório

De acordo com Bhuyan *et al.* [24], uma fraqueza da detecção baseada em perfil é que as anomalias podem ser introduzidas lentamente em padrões usuais até se tornarem um evento legítimo. O ecossistema apresentado oferece ao administrador informações relevantes para ajudar a determinar uma solução para esses problemas. Essas informações são

organizadas em relatórios gráficos e demonstram o uso de recursos de rede no momento em que uma anomalia é detectada. Se esses eventos anômalos forem identificados e interrompidos, eles não serão incluídos no conjunto de dados históricos e, consequentemente, não serão incluídos no perfil de tráfego normal. Além disso, esses relatórios ajudarão o administrador da rede a desenvolver novas políticas para reduzir o impacto nos serviços oferecidos aos usuários finais.

5.6 Considerações sobre o capítulo

Criada para atuar no ambiente SDN, a proposta descrita neste capítulo tem como finalidade a identificação e solução das anomalias do tráfego. O ecossistema exposto posiciona-se como uma alternativa frente aos problemas ainda abertos na literatura, como a intervenção humana no processo de detecção de anomalias, o reconhecimento de padrões não usuais do tráfego em tempo real e a mitigação das consequências desses eventos.

Para atingir os objetivos pretendidos, o ecossistema realiza o monitoramento constante da rede e detecta eventos anômalos de acordo com diferenças comportamentais do tráfego, baseado no perfil normal previamente estabelecido. Isso permite que o processo de detecção se ajuste automaticamente ao comportamento da rede e não necessite da interposição do gerente. Além de retirar da gerência a complexidade decorrente do monitoramento, detecção e mitigação dos problemas de rede, torna a execução dessas tarefas mais rápida e permite a exclusão do erro proveniente da intervenção humana.

O ecossistema é formado por um conjunto de módulos que atua diretamente com o plano de controle da rede, instruindo-o quanto ao encaminhamento das informações no plano de dados. A atuação conjunta dos módulos fornece mecanismos necessários para se manter uma visão consistente sobre o estado da rede. Além disso, essa estrutura modular permite que módulos inteiros do ecossistema possam ser alterados sem afetar as demais funcionalidades. Essa propriedade confere ao ecossistema flexibilidade e facilidade de manutenção.

O próximo capítulo aborda a avaliação do ecossistema durante a detecção e mitigação de anomalias. A modularidade também é melhor explorada, uma vez que diversos métodos de detecção são confrontados, além da alteração do módulo responsável pela coleta do tráfego.

6 Resultados

Para validar a solução proposta, quatro cenários de teste foram criados. Cada um deles possui um objetivo específico e distingue-se dos demais pelas diferentes características avaliadas. No primeiro cenário é verificada a eficiência das abordagens tanto para detecção quanto para mitigação de anomalias. Para que isso fosse realizado, os módulos desenvolvidos foram incorporados ao controlador POX. Ainda nesse cenário, a abordagem de detecção é comparada com outras técnicas de mesmo propósito. O segundo cenário analisa o uso de recursos de *hardware* durante a execução dos módulos que compõem a proposta. O cenário seguinte realiza uma avaliação semelhante ao primeiro cenário, entretanto é utilizado o controlador Floodlight. Finalmente, o quarto cenário avalia a solução proposta em um ambiente de rede em que o monitoramento do tráfego é realizado em intervalos de 15 segundos. Os cenários estão descritos na Tabela 6.1.

Tabela 6.1 – Descrição dos objetivos em cada cenários de testes.

Cenário	Controlador	Objetivo
1	POX	Detectar e mitigar anomalias no tráfego de rede
2		Avaliar o uso do *hardware* durante a execução da solução proposta
3	Floodlight	Avaliar o sistema de detecção em ambiente com controlador Floodlight
4		Estudo da proposta em um ambiente com monitoramento do tráfego em intervalos de 15 segundos

Os experimentos conduzidos nos cenários de testes foram realizados no emulador de redes Mininet[1], o qual permite a criação de redes compostas por *hosts* virtuais, *switches*, controladores e enlaces. A escolha por esse emulador foi decidida por alguns fatores. O primeiro é o de que é um *software* de código aberto que conta com uma comunidade ativa e atualizações constantes. Outro fator que contribuiu para sua utilização nos testes é que, desde sua criação, o Mininet recebeu significativo esforço para suportar as tecnologias, como o protocolo OpenFlow e diversos controladores. Aplicações desenvolvidas no Mininet podem ser implantadas em um ambiente real com pouca ou nenhuma alteração. Isso só é conseguido porque ele permite que diversos aspectos da rede possam ser configurados e monitorados em tempo real. Por fim, ele já é considerado um padrão *de facto* para o ensino, pesquisa e desenvolvimento da SDN.

A metodologia assumida neste trabalho cria um perfil normal com base no

[1] http://mininet.org

tráfego histórico. Dessa maneira, a ferramenta Scapy[2] foi utilizada para a criação de todo o tráfego legítimo usado nos testes, incluindo o tráfego histórico. Scapy propicia a geração, envio e decodificação de pacotes de vários tipos de protocolos. É possível forjar pacotes incluindo informações das camadas de enlace, rede, transporte e o próprio conteúdo do pacote.

Com o intuito de tornar os cenários mais realistas e confiáveis, o tráfego gerado pela ferramenta Scapy foi criado considerando as sugestões propostas por [14]. O tráfego normal criado e transmitido nos testes foi composto por aproximadamente 85% de fluxos TCP, 10% UDP e 5% ICMP (*Internet Control Message Protocol*). A taxa de transmissão de pacotes seguiu uma distribuição de Poisson, enquanto o tamanho do pacote e a taxa de tráfego gerado obedeceram a uma distribuição uniforme. A intensidade do tráfego foi ajustada para que seu comportamento se assemelhasse ao tráfego de uma rede de um campus universitário. Assim, em intervalos de maior atividade como das 7 às 22 horas, o volume do tráfego foi acentuado, como mostrado na Figura 5.5.

A próxima seção apresenta os conceitos e métricas utilizadas para mensurar a eficiência da proposta. Posteriormente, é detalhado como a avaliação foi realizada em cada um dos cenários, seguida dos respectivos resultados.

6.1 Métricas de avaliação

A avaliação da proposta nesta seção tem como objetivo aferir se os intervalos de análise onde anomalias ocorreram foram devidamente reconhecidos. Além disso, avalia-se também se a identificação das anomalias propiciou uma detecção consistente, sem erros de classificação. Antes da discussão dos resultados, é necessário apresentar as métricas adotadas durante a avaliação. Todas as métricas utilizadas compartilham dos seguintes conceitos:

- Verdadeiro positivo (VP): caso a instância seja anômala e ela for classificada como tal;
- Verdadeiro negativo (VN): caso a instância seja normal e for classificada como tal;
- Falso positivo (FP): caso a instância seja normal e foi erroneamente classificada como anômala;
- Falso negativo (FN): caso a instância seja anômala e foi erroneamente classificada como normal.

[2] https://scapy.net

A nomenclatura dessas métricas deriva da classificação binária, em que apenas dois rótulos são possíveis de serem atribuídos a cada amostra da base de dados. Normalmente, adota-se a convenção de rotular os exemplos da classe de maior interesse como positivos. No caso da detecção de anomalias, o maior interesse está no reconhecimento dos intervalos em que as anomalias ocorreram. O resultado ideal de um classificador ocorre na ausência de erros de classificação, dados pelos valores de falso positivo e falso negativo.

Por meio dos conceitos citados anteriormente, pode-se derivar algumas métricas, tal como a sensibilidade e especificidade do classificador, mostrada nas equações (6.1) e (6.2), respectivamente.

$$\text{Taxa de verdadeiro positivo (TVP)} = \frac{VP}{VP + FN} \tag{6.1}$$

$$\text{Taxa de falso positivo (TFP)} = \frac{FP}{FP + TN} \tag{6.2}$$

A taxa de verdadeiro positivo (TVP), também chamada de sensibilidade, revocação ou *recall*, é a medida que indica qual a proporção de intervalos anômalos foi classificada corretamente. Em termos práticos, essa medida pode ser entendida como a capacidade do classificador de identificar os casos positivos. Já a taxa de falso positivo (TFP) indica qual a proporção de classificação erradas foi feita sob os intervalos que não apresentavam anomalias. As taxas TVP e TFP podem ainda ser combinadas para a construção do gráfico *Receiver Operating Characteristics* (ROC) [143], que possibilita uma análise visual da aptidão do sistema de detecção no reconhecimento de comportamentos anômalos. Entretanto, quando deseja-se comparar o resultado de vários classificadores é preferível reduzir a curva ROC a um simples escalar. Para isso, calcula-se a área sob a curva (do inglês, *Area Under the Curve* - AUC). O classificador com maior valor para essa medida é o mais apto a realizar classificações corretas.

Uma medida geral para avaliar o resultado da detecção é a acurácia, apresentada na equação (6.3). No numerador tem-se a soma de todos os acertos realizados durante a classificação, tanto dos intervalos anômalos quanto dos não anômalos. Por levar em consideração apenas os acertos, essa medida deve ser usada com moderação quando o número de instâncias anômalas e normais está desbalanceado. Nesse caso, o resultado da acurácia poderia ser tendencioso e mostrar um bom resultado caso o sistema de detecção classificasse corretamente todas as instâncias normais e errasse todas as anômalas. Uma maneira de resolver esse problema é dar ênfase à detecção de intervalos anômalos e penalizar o sistema por classificar erroneamente os intervalos normais. A métrica precisão expressa essa situação, como apresentado na equação (6.4). Uma outra alternativa é utilizar a métrica F-*measure*, que consiste na média harmônica entre a precisão e a revocação.

O valor retornado por essa medida estará mais próximo do menor valor entre precisão e revocação, fornecendo uma pontuação apropriada ao classificador.

$$\text{Acurácia} = \frac{VP + VN}{VP + FP + FN + VN} \tag{6.3}$$

$$\text{Precisão} = \frac{VP}{VP + FP} \tag{6.4}$$

$$\text{F–}measure = \frac{2\,(\text{revocação} \times \text{precisão})}{(\text{revocação} + \text{precisão})} \tag{6.5}$$

Todas as métricas derivadas dos conceitos VP, FP, VN e FN, citadas anteriormente, têm seus valores contidos no intervalo de zero a um. Com exceção da TFP e FN, cujo melhor resultado é zero, todas as outras têm seu melhor resultado quando o valor 1 é retornado. Esse é o objetivo de qualquer sistema de detecção, aumentar a proporção de classificação correta enquanto mantém-se a taxa de erros a mais baixa possível.

6.2 Cenário 1: Avaliação da detecção e mitigação de anomalias

No primeiro cenário, duas máquinas foram usadas para a avaliação do sistema de detecção e mitigação de anomalias. A primeira abriga o controlador SDN, enquanto a segunda é onde se encontra o Mininet. Embora tanto o controlador quanto a rede pudessem ser executados sob o mesmo *hardware*, a configuração utilizada foi escolhida no intuito de que o desempenho do controlador não fosse prejudicado pela geração do tráfego e a emulação da rede. Ambas as máquinas utilizam Linux Ubuntu 16.04 como sistema operacional, possuem 8 GB de memória RAM, 4 núcleos de processamento cujas frequências são de 3,2 e 2,8 GHz, para a máquina do Mininet e do controlador, respectivamente.

O controlador usado neste cenário foi o POX, que já possui funcionalidades básicas para manutenção da operação da rede, como, por exemplo, encaminhamento de pacotes, aprendizado de endereço MAC, integração com o protocolo OpenFlow, entre outros. Os desenvolvedores podem estender essas funcionalidades, criando um controlador mais complexo, implementando novos componentes por meio da API POX disponível. Os módulos do ecossistema apresentado (coleta do tráfego, detecção de anomalias, mitigação e relatórios) foram desenvolvidos em Python e integrados ao POX. Todos os módulos, com exceção do módulo de relatórios usado apenas para armazenamento e visualização de *logs*, foram projetados para lidar diretamente com as propriedades do OpenFlow e do controlador POX. Dessa maneira, para maior controle do ecossistema, os eventos ligados à coleta, à detecção e à mitigação foram realizados sem o auxílio de qualquer API da interface *northbound*.

O ambiente de rede emulado configura uma rede em forma de árvore com profundidade dois, como mostrado na Figura 6.1. Para o encaminhamento do tráfego pela rede foram utilizados *switches* do tipo Open vSwitch, já que eles podem manipular o tráfego entre os *hosts* virtuais, além de oferecerem suporte para o protocolo OpenFlow. Todos os *switches* são interconectados por um *switch* raiz e cada sub-rede (N1, N2, N3 e N4) é composta por vinte *hosts*. Todos os enlaces foram definidos com uma taxa máxima de transmissão de 1 Gbps.

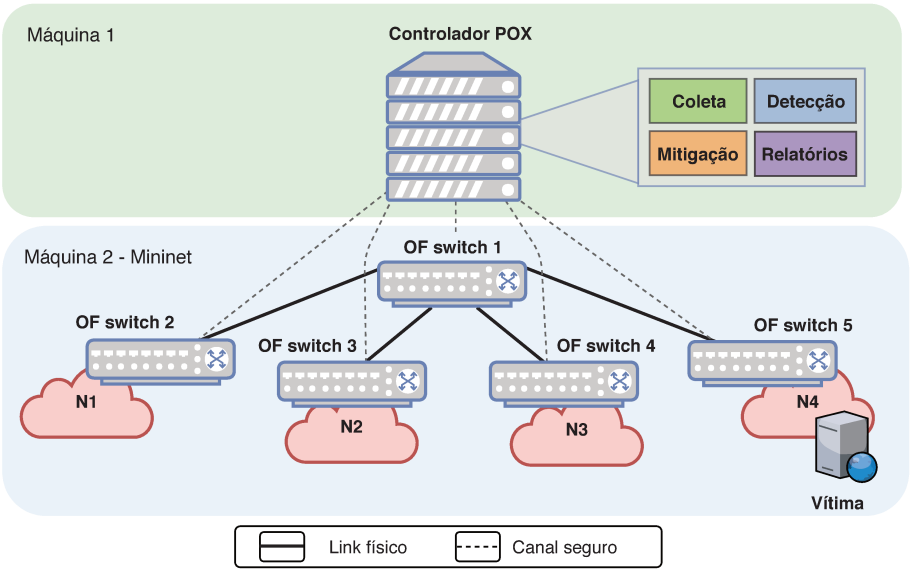

Figura 6.1 – Ambiente de rede usado no primeiro cenário.

Para a efetivação do teste, dois tipos de anomalias foram gerados para reproduzir certos comportamentos de ataques. Esses eventos foram produzidos utilizando Scapy e hping3[3], um gerador de tráfego usado principalmente com uma ferramenta para avaliação da segurança de redes. Para geração de um ataque SYN *flooding*, o scapy instruiu 10 *hosts* zumbis a enviar pacotes com a *flag* SYN ativada para um *host* alvo a partir de um conjunto aleatório e constantemente alterado de endereços IP e porta de origem. Esse ataque foi constituído então por um DDoS que utiliza IP *spoofing*. A outra anomalia foi gerada pelo hping3 e corresponde ao ataque *port scan*. Nesse ataque, uma fonte envia pacotes para diferentes portas do *host* de destino, a fim de confirmar quais delas estão em funcionamento. Ambas as anomalias foram incluídas no tráfego legítimo e tiveram como alvo um *host* situado na sub-rede N4, cujo endereço IP é 10.0.0.78.

A Figura 6.2 mostra os dados coletados a partir do *switch* 5 durante a avaliação da estratégia de detecção de anomalias. A área em verde representa o tráfego observado no período analisado e a assinatura digital do tráfego para cada atributo é mostrada pela linha azul. Dois ataques DDoS e *port scan* foram gerados, cada qual com diferente intensidade. O primeiro ataque DDoS, presente nos primeiros intervalos de análise, causou

[3] https://www.hping.org

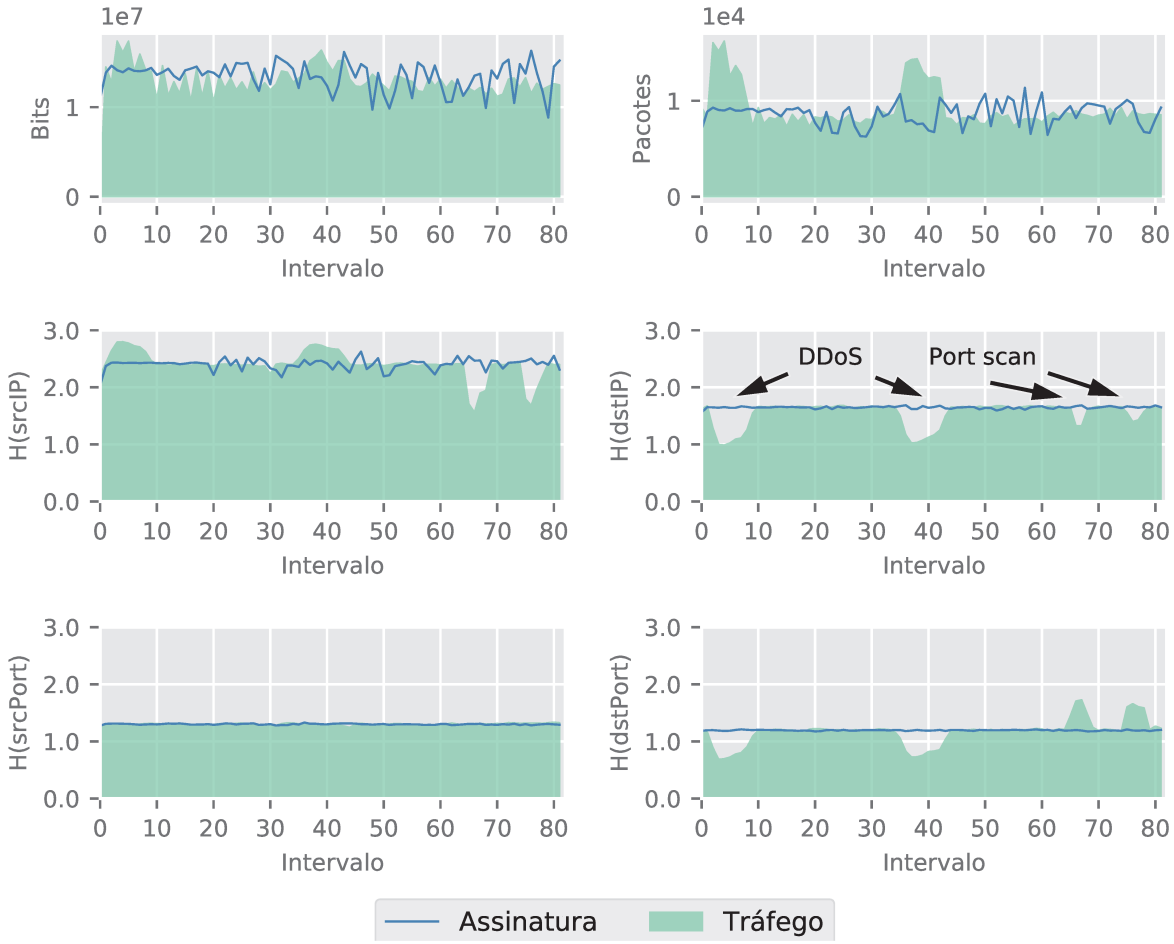

Figura 6.2 – Tráfego analisado contendo ataques DDoS e *port scan*. A linha azul indica a assinatura do comportamento normal esperado.

um aumento incomum de pacotes recebidos, em que pôde ser observada uma diferença de 70% em relação ao esperado. O segundo ataque DDoS, localizado próximo ao intervalo 40, apresentou um acréscimo de pacotes de aproximadamente 55%. Para atingir esses valores, 300 e 250 endereços IP atacantes foram usados no primeiro e no segundo evento, respectivamente. Com relação às anomalias de *port scan* situadas entre os intervalos 70 e 80, apenas um atacante foi usado para acessar as portas de 1 a 1024 do *host* vítima. A diferença entre esses ataques foi o intervalo de chegada de pacote. Para o primeiro ataque *port scan* disparado, uma pausa de 0,03 segundos entre o envio de um pacote e outro, enquanto no segundo esse intervalo passou a ser de 0,05 segundos.

6.2.1 Comparação com outros métodos de detecção

Para uma avaliação mais detalhada da detecção, os resultados da proposta foram comparados com outros métodos encontrados na literatura e que também foram usados no ambiente SDN. O primeiro é o método de classificação do tráfego *k-nearest neighbor* (kNN) com análise de correlação (CkNN) [110], proposto para detectar ataque

com grande eficiência e baixo custo em redes de *data centers*. O CkNN explora informações de correlação dos dados de treinamento para melhorar a classificação de anomalias do tráfego e reduzir a sobrecarga causada pela densidade dos dados de treinamento. O algoritmo agrupa os fluxos de acordo com as características que os distinguem entre anômalos ou legítimos. O segundo método é baseado na técnica de classificação máquina de vetores suporte (*Support Vector Machine*, SVM) [144] projetada para a detecção de ataques de inundação. Uma função *kernel* mapeia implicitamente o espaço de entrada para um espaço dimensional mais elevado, para criar uma separação mais clara entre dados normais e anômalos.

A Figura 6.3(a) apresenta a probabilidade de cada método comparado classificar cada intervalo de acordo com uma das classes definidas: $k = 0$ ("normal") é a classe de referência, $k = 1$ ("DDoS"), $k = 2$ ("*port scan*") e $k = 3$ ("*flash crowd*"). Os métodos atribuem aos intervalos analisados o rótulo da classe cuja maior confiança de classificação é calculada. Nota-se que a regressão logística multinomial (RLM) foi a única abordagem que apresentou alta probabilidade de classificação correta nos intervalos em que os ataques de *port scan* foram realizados. Em contraste, mesmo quando os ataques DDoS alcançaram valores destoantes em até 50% do esperado para alguns atributos do fluxo, a certeza da identificação desses ataques foi menor para a RLM e SVM quando comparada com a probabilidade calculada pelo CkNN. Embora a anomalia *flash crowd* esteja presente apenas no conjunto de treinamento, é interessante verificar a probabilidade de classificação desse evento. A abordagem baseada em SVM atribuiu a probabilidade de 10% e 6% dos ataques DDoS serem considerados *flash crowd*. De maneira análoga, a RLM obteve uma probabilidade menor de realizar essa classificação errada enquanto o CkNN apontou nenhuma chance de ocorrência de *flash crowd* para esses intervalos analisados.

A Figura 6.3(b) mostra os intervalos de tempo em que cada abordagem de classificação detectou anomalias. A RLM reconheceu todos os ataques disparados, enquanto a SVM e CkNN identificaram apenas os ataques DDoS. Uma vez desenvolvido um detector robusto, é necessário decidir se ele é suficiente para identificar as anomalias investigadas. A Figura 6.4 apresenta as métricas de desempenho, comparando as abordagens de classificação. Como pode ser visto, a RLM alcançou melhor desempenho geral em comparação aos outros métodos. O CkNN apresentou índices piores do que a SVM nas métricas de precisão e taxa de falso positivo. Com uma averiguação mais profunda, foi possível verificar que esse resultado está diretamente ligado à má interpretação do CkNN em relação aos intervalos de análises que compreendem ao final dos ataques DDoS. Enquanto os outros dois métodos classificaram apenas os intervalos em que tais anomalias estavam em vigor, o CkNN continuou classificando como anômalos os intervalos em que o tráfego já havia sido normalizado.

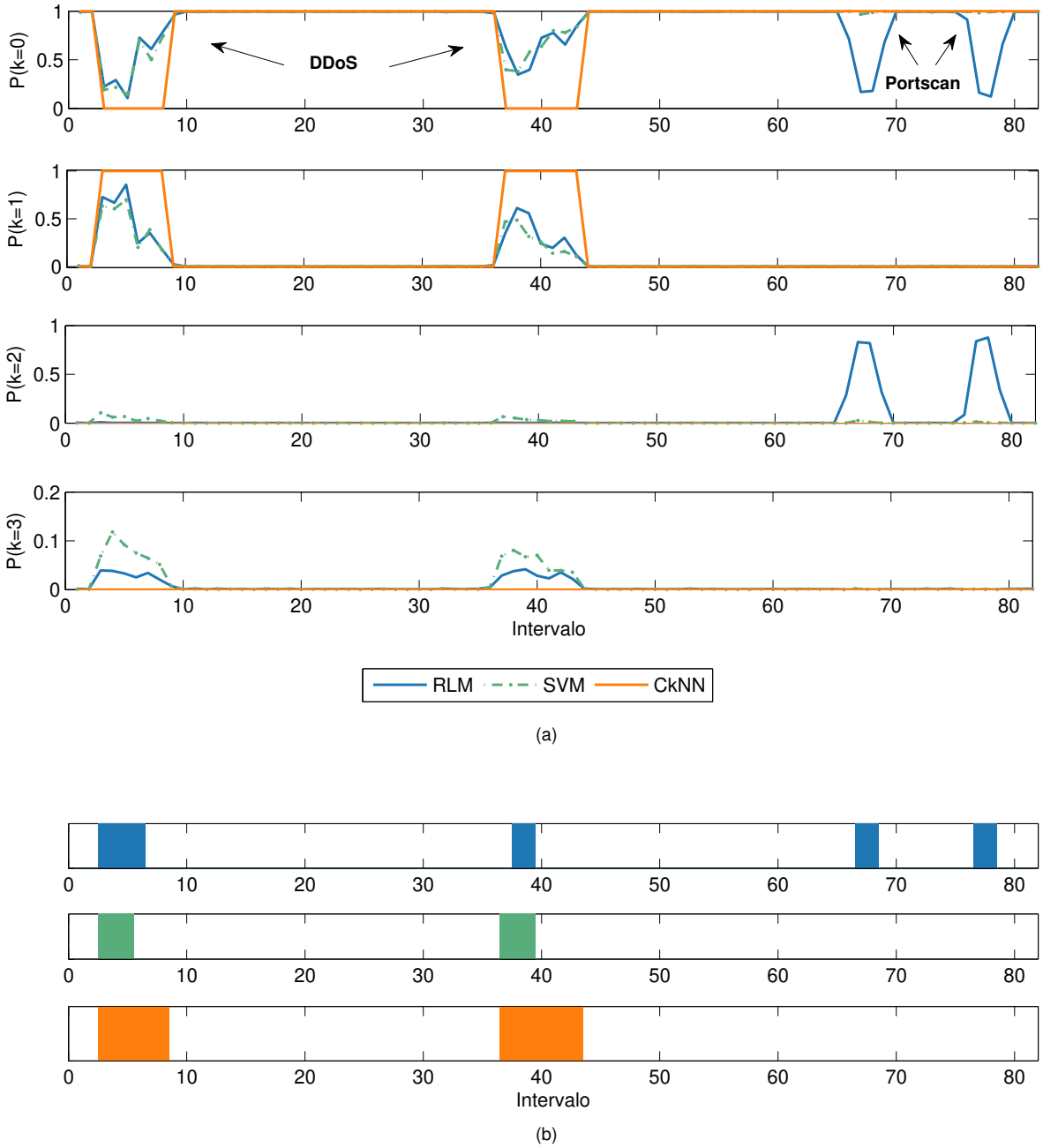

Figura 6.3 – (a) Probabilidade de cada intervalo ser classificado como classe k. (b) Alarmes disparados por cada esquema de detecção de anomalias.

Como esperado, a revocação foi a medida com os valores mais baixos dentre as demais, pois tanto o CkNN quanto a SVM não foram capazes de reconhecer os intervalos em que as anomalias *port scan* estavam ocorrendo. Nota-se também que, embora incapazes de identificar os ataques durante toda a sua duração, a RLM e a SVM não classificaram incorretamente nenhum intervalo não anômalo.

A Figura 6.5 mostra as curvas ROC para as técnicas de detecção comparadas. Curvas ROC são tipicamente usadas para avaliar os resultados em uma classificação binária, em que apenas as classes anômalo e normal são admitidas. A macro-média para as

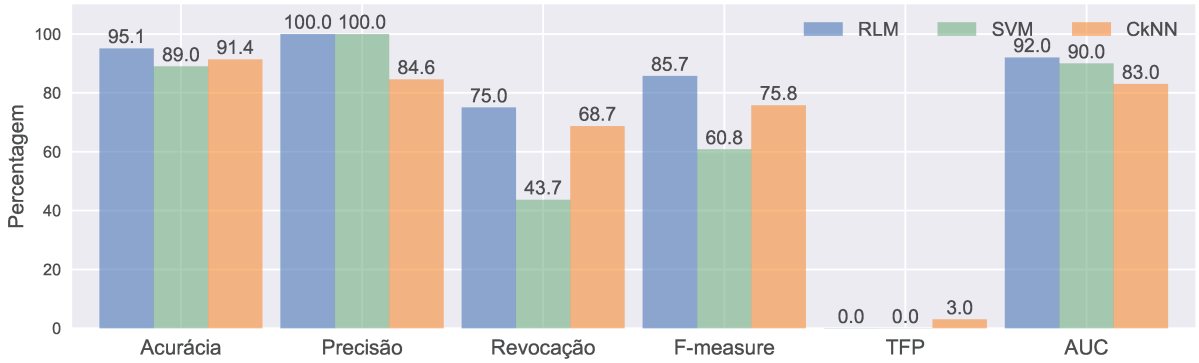

Figura 6.4 – Comparação de detecção de anomalia entre RLM e dois outros métodos.

medidas verdadeiro positivo e falso positivo foi usada para estender as curvas ROC para o nosso cenário multi-classe. Essa abordagem pondera igualmente a classificação de cada classe em um classificador um-contra-todos. Um classificador que utiliza a estratégia um-contra-todos particiona as classes em dois grupos. O primeiro grupo é formado por uma classe e o segundo é constituído pelas classes restantes. A partir de então, um classificador binário é treinado para esses dois grupos e esse procedimento é repetido para cada uma das classes da base de dados. Quando empregada nesse contexto, a macro-média informa o valor médio obtido da classificação utilizando cada classificador binário. A RLM teve melhor desempenho pois alcançou uma taxa maior de verdadeiros positivos com menos falsos positivos do que as outras técnicas. De acordo com a sua curva ROC, a RLM pôde atingir 100% de precisão com uma taxa de falso positivo de 31,2%.

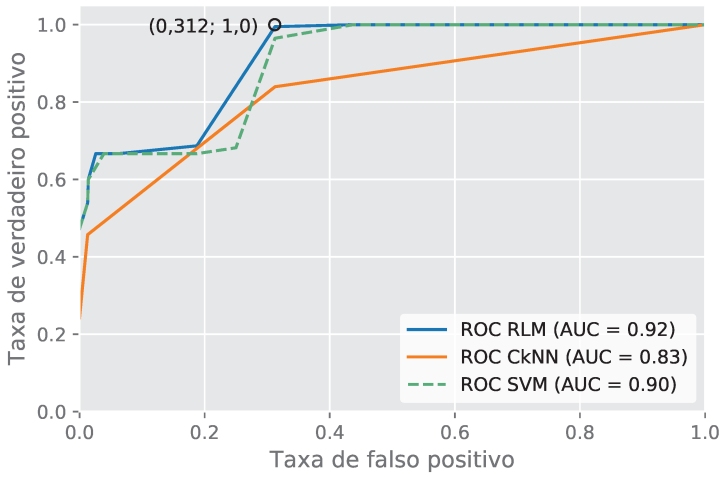

Figura 6.5 – Curvas ROC das abordagens comparadas para detecção de anomalias.

6.2.2 Avaliação da eficiência e da mitigação

Depois que o algoritmo de detecção de anomalias detecta e identifica uma atividade suspeita como um ataque DDoS ou um ataque *port scan*, o módulo de mitigação

pode manipular adequadamente o tráfego anômalo correspondente. A Figura 6.6 mostra o comportamento do tráfego da rede quando as políticas de mitigação são aplicadas, em contraste com a sua ausência. A área verde compreende o tráfego gerado seguindo a metodologia descrita na seção 6.2 junto com os ataques DDoS e *port scan* usados anteriormente para avaliação da detecção. Quando o ataque é mitigado, os valores dos atributos retomam seus intervalos esperados, isto é, eles se aproximam da assinatura digital do tráfego. A linha laranja corresponde à política *block_flow* para atenuar o ataque DDoS. Também foram realizados testes usando a política *load_balance*, mas ela não apresentou alterações significativas nos valores dos recursos porque não altera a taxa de solicitações que chegam a um *host* atacado. Portanto, o resultado obtido por essa política é semelhante ao tráfego sem mitigação. No entanto, no que diz respeito à resiliência da rede, os efeitos *load_balance* podem ser decisivos para adoção dessa política.

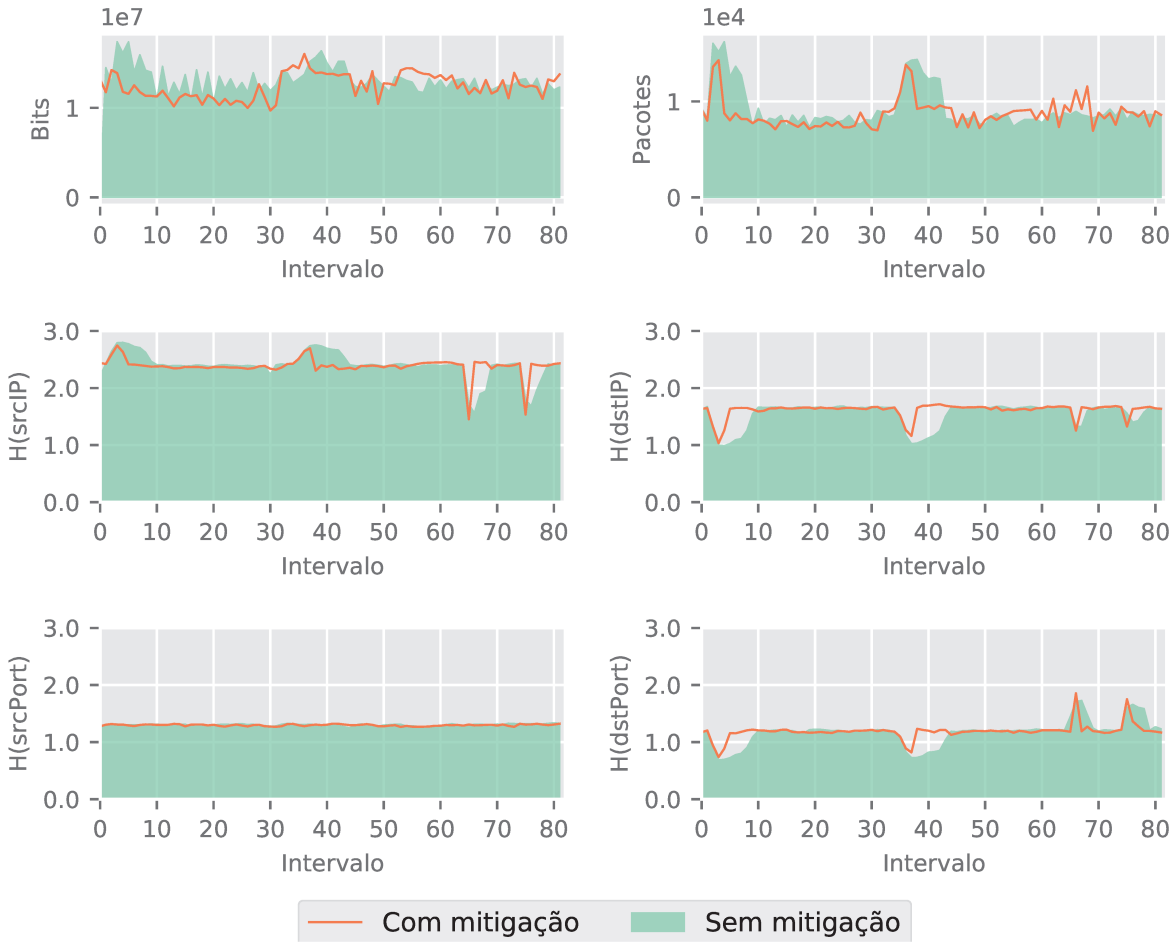

Figura 6.6 – Valores de atributos do tráfego durante ataques DDoS e *port scan* com e sem o mecanismo de mitigação.

Com relação aos ataques de *port scan*, foi aplicada a política *block_src_IP*. Como pode ser observado, sua atenuação exigiu menos intervalos de tempo porque o reconhecimento do IP de origem mal-intencionado que afeta o *host* atacado é mais simples

do que identificar *hosts* com comportamentos inadequados durante um ataque distribuído, como o DDoS.

6.3 Cenário 2: Uso do *hardware* durante a execução

Um fator importante que está atrelado à detecção de anomalias e, consequentemente, à segurança da rede é o desempenho. A Figura 6.7 mostra a quantidade de tempo gasto para executar as rotinas do ecossistema de acordo com as políticas de mitigação e anomalia detectadas. Os valores apresentados foram tomados a partir da detecção de eventos anômalos realizada no primeiro cenário. Na figura, o pré-processamento agrega as tarefas de coleta do tráfego e cálculo da entropia. O tempo identificado como mitigação é dado pelo cálculo da duração de duas tarefas. A primeira é a identificação da(s) origem(ns) e destino de uma anomalia identificada. Na segunda ocorre a formulação das entradas da tabela de fluxo que serão enviadas pelo controlador aos *switches* afetados com o propósito de conter a anomalia. É importante notar que o tempo de comunicação entre os *switches* e o controlador não é levado em consideração.

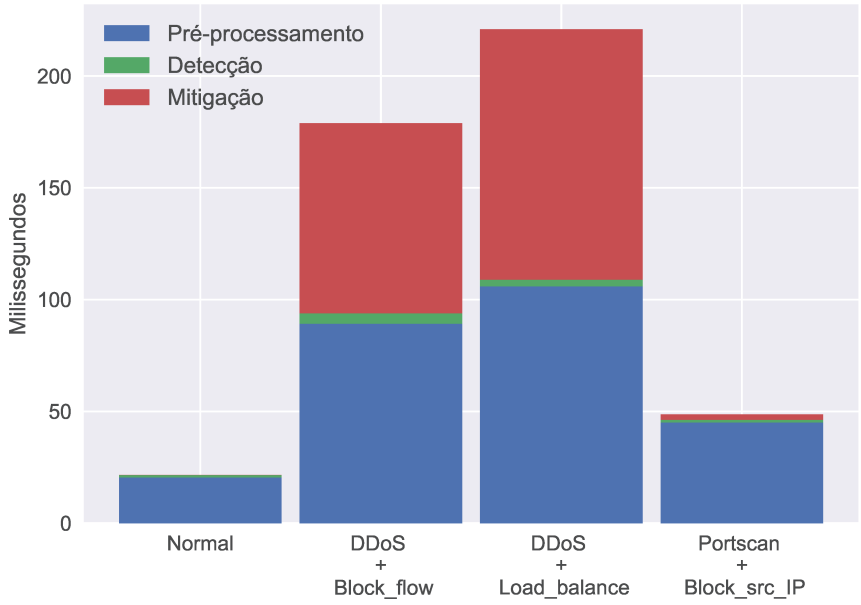

Figura 6.7 – Tempo de processamento em cada etapa.

O módulo de detecção atingiu um tempo de execução semelhante em todas as situações analisadas. Uma vez que a RLM é treinada, a avaliação e a classificação do tráfego são diretas. Com relação ao módulo de mitigação, um fator de tempo determinante é o número de fluxos que devem ser inseridos na tabela de fluxo de *switches* para efetivação da política de mitigação, bloqueando a comunicação ou aplicando o balanceamento de carga. A política de balanceamento de carga demanda mais tempo para ser executada devido à necessidade de determinar quais requisições destinadas ao servidor sobrecarregado

serão encaminhadas aos servidores alternativos. A fase de pré-processamento consumiu muito do tempo de processamento do sistema em todas as situações analisadas, e esse valor é altamente dependente da intensidade de um ataque.

O funcionamento dos módulos de detecção e mitigação também foram investigados quanto ao uso de CPU (*Central Processing Unit*) do controlador, bem como o número de entradas de fluxos presentes nos *switches*. Ambos os valores, que podem ser usados para medir a operação do ecossistema, utilizam os recursos de *hardware*. O uso da CPU indica o desempenho do plano de controle, enquanto o número de entradas de fluxo na tabela de um *switch* pode afetar a capacidade de encaminhamento, porque todos os pacotes recém-chegados são confrontados com entradas da tabela de fluxos [145]. Embora a tabela de fluxos desempenhe um papel importante dentro do ambiente SDN, o seu tamanho é limitado pelo design do *switch* e pela quantidade de TCAM (*Ternary Content-Addressable Memory*) que ele dispõe. Quando já não há mais memória suficiente para armazenar novas regras de encaminhamento, os pacotes são descartados e a visão global da rede obtida pelo controlador pode ser comprometida.

Os três gráficos no topo da Figura 6.8 representam o uso da CPU do controlador. Essas medidas foram tomadas a cada segundo por um período de cinco minutos. Cada gráfico refere-se a uma política aplicada para conter um ataque, e os círculos vermelhos indicam o momento exato em que tais políticas foram enviadas do controlador para o plano de dados. Os três gráficos na parte inferior da figura indicam o número de entradas de fluxo contidas no *switch* 5 durante os ataques. Essa última medida foi realizada a cada intervalo de análise de 30 segundos.

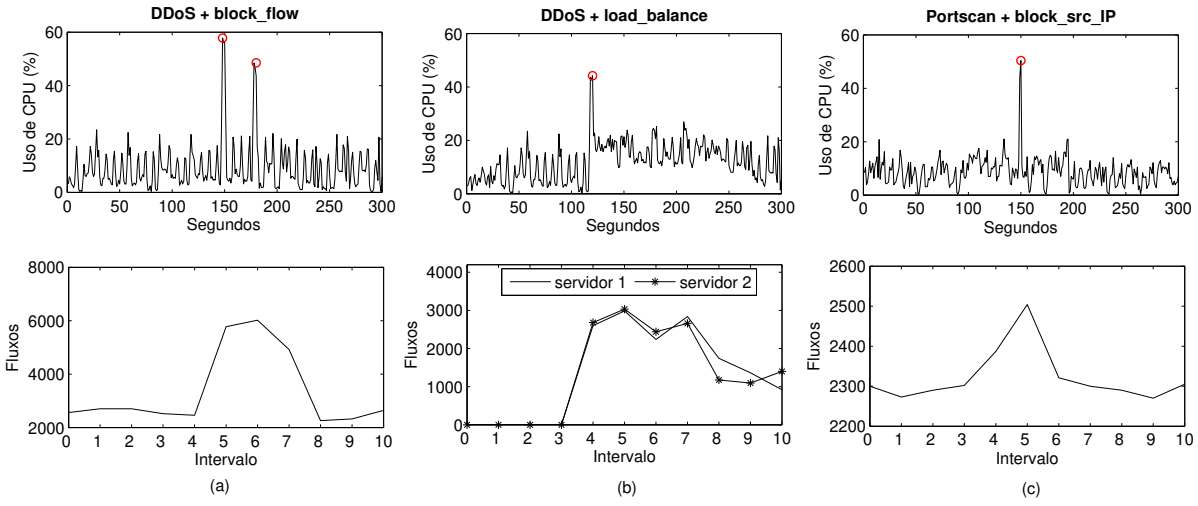

Figura 6.8 – Desempenho avaliado de acordo com o tipo de ameaça e a política de mitigação. Cada intervalo durante o monitoramento de fluxos nos *switch* corresponde a 30 segundos.

A Figura 6.8(a) revela que a política *block_flow* precisou de três intervalos para

restringir o ataque DDoS. Depois de inserir entradas de fluxo com a ação de descartar pacotes maliciosos (intervalo 5), foi necessário adicionar novas entradas de mitigação com os endereços IP dos novos atacantes (intervalo 6). Em torno do intervalo 8, os fluxos começaram a diminuir, por conta do bloqueio, e o tráfego voltou ao normal como esperado.

Na Figura 6.8(b) é possível observar um pico no uso da CPU por volta dos 120 segundos após o início do monitoramento. Esse pico é causado pela atividade do módulo de detecção e mitigação para identificação de DDoS e implementação da política *load_balance*. A partir de então, o uso da CPU permanece mais acentuado do que o habitual enquanto é realizado o balanceamento da carga. Finalmente, o uso da CPU na Figura 6.8(c) apresenta um pico quando o ataque *port scan* é identificado. Como o módulo de detecção precisa descobrir um único IP de origem mal-intencionado, a política *block_src_IP* interrompe a anomalia após a inserção de uma única entrada de fluxo no *switch* afetado. Essa entrada contém, nos campos de correspondência, o endereço de origem do *host* atacante e nenhuma ação é definida em seu *action set*. Isso faz com que todos os pacotes provenientes do atacante sejam descartados. Observa-se também que, por ser um processo mais simples, a mitigação do evento *port scan* causou um pico de menor duração no uso da CPU do que aqueles observados na deteção de ataques DDoS.

Na figura também é possível observar a quantidade de entradas de fluxos no *switch* durante o período monitorado. Nota-se que após o acionamento das medidas de mitigação, as anomalias foram gradativamente sendo controladas. O exemplo mais proeminente é o do *port scan*, em que após o bloqueio do tráfego do atacante, a quantidade de fluxos retomou aos valores semelhantes que eram medidos anteriormente ao evento anômalo. Quando a política de balanceamento de carga é usada, diferentemente da Figura 6.8(a), o tráfego anômalo não é contido, ele só é distribuído entre os servidores previamente definidos.

6.4 Cenário 3: Avaliação usando o controlador Floodlight

Neste cenário, três anomalias: DoS, *port scan* e DDoS foram usadas para verificar a sensibilidade do detector proposto. Três ataques de cada uma das anomalias foram gerados com diferentes intensidades. Os ataques DoS foram disparados de tal forma que uma única fonte exaustivamente transmitisse pacotes para uma porta específica no endereço IP de destino. O módulo de detecção proposto considera essa anomalia como uma particularidade do ataque DDoS, em que, a única diferença, é a diminuição do valor da entropia de IP de origem, uma vez que um único atacante tende a dominar boa parte da comunicação observada no decorrer do ataque. De forma análoga, durante o *port scan* apenas um atacante foi utilizado para disparar uma série de requisições à vítima, entre-

tanto tendo como destino as portas de 1 a 65535. Por fim, cinco *hosts* atacantes utilizaram a técnica de IP *spoofing* para forjar os endereços IP de destino e realizar o DDoS.

A Tabela 6.2 apresenta os detalhes sobre as anomalias geradas nesse cenário. Também é mostrada a quantidade de bits e pacotes que correspondem ao tráfego anômalo. Esses valores são expressos em porcentagem em relação ao total de tráfego analisado durante o monitoramento da rede. Como pode ser observado, as anomalias intituladas com #1 são as que apresentam menor intensidade, enquanto aquelas sucedidas de #3 são as mais acentuadas. Todos os ataques tiveram como alvo o endereço IP 10.0.0.78, situado na sub-rede N4. Para os ataques de inundação do tipo SYN *flooding*, DoS e DDoS, a porta de destino foi fixada como 80, enquanto a porta de origem, ou seja, das fontes que geraram os ataques, foi constantemente alterada.

Tabela 6.2 – Descrição das anomalias utilizadas no cenário.

Anomalia	Tráfego anômalo	
	Bits [%]	Pacotes [%]
DDoS #1	1,3	17,9
DDoS #2	3,4	36,4
DDoS #3	5,0	45,5
Port scan #1	0,08	2,7
Port scan #2	0,11	12,3
Port scan#3	0,15	15,1
DoS #1	1,0	14,9
DoS #2	3,0	34,1
DoS #3	4,6	43,5

A topologia usada para análise é a mesma utilizada no cenário 1, porém agora os módulos do sistema de detecção proposto operam sobre o controlador Floodlight[4]. Tal controlador é um *software* de código aberto implementado na linguagem Java e funciona com *switches* OpenFlow físicos e virtuais. Atualmente o seu desenvolvimento é mantido pela comunidade de desenvolvedores, incluindo um grupo de engenheiros da companhia Big Switch Networks, uma das pioneiras no desenvolvimento de soluções voltadas à SDN. Os resultados apresentados nesta seção utilizaram a versão 1.2 do Floodlight, publicada em 2016.

O sistema voltado à detecção de anomalias novamente foi comparado aos métodos CkNN e SVM para identificação das anomalias inseridas no tráfego da rede. Também foi adicionado à análise o método denominado Random Forest (RF), um algoritmo de aprendizado de máquina destinado a classificação e regressão. A tarefa de detectar e identificar anomalias pode ser compreendida como um problema de classificação de trá-

[4] http://www.projectfloodlight.org/

fego. Dessa maneira, pode-se induzir um modelo a reconhecer padrões que diferem do comportamento esperado do tráfego. Essa tarefa pode ser executada por classificação supervisionada, que parte de um conjunto de dados sobre observações ou casos descritos por atributos ou recursos. Cada observação tem um valor atribuído (alvo) de uma variável em estudo e o objetivo é construir o conhecimento que prevê o rótulo quando surgir uma nova observação. Para construir um classificador a partir de um conjunto de dados, diferentes abordagens podem ser usadas, como métodos estatísticos clássicos, árvores de decisão ou redes neurais artificiais. Especificamente, as árvores de decisão são adequadas para classificação porque obtêm resultados satisfatórios e apresentam uma estrutura em que a representação do conhecimento é simples de interpretar [146].

Uma árvore de decisão é um grafo acíclico direcionado em que cada nó pode ser um nó de divisão, com dois ou mais sucessores, ou um nó folha. Um nó de divisão contém um teste condicional, baseado nos valores dos atributos. Já um nó folha armazena o rótulo (classe) das instâncias de treinamento que, passando pelos testes condicionais, chegam até esse nó. A Figura 6.9 apresenta o exemplo de uma árvore de decisão em que os dois atributos, x_1 e x_2 são utilizados. Em cada nó de divisão, os valores desses atributos passam por um teste lógico, resultando em verdadeiro e falso. Por fim, os nós folhas representam cinco classes mutualmente excludentes.

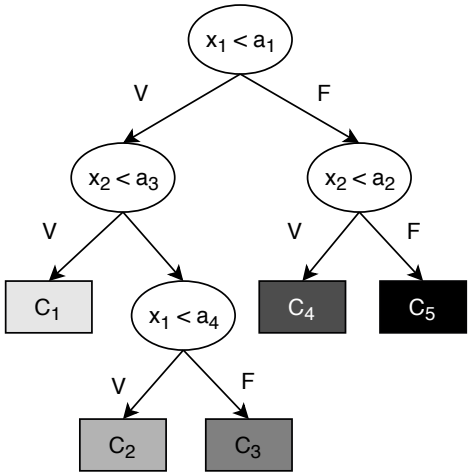

Figura 6.9 – Exemplo de uma árvore de decisão.

A Random Forest é um dos algoritmos mais representativos de indução de um modelo de classificação baseada em árvores. Esse algoritmo cria uma floresta aleatória, ou seja, um grupo de árvores de decisão independentes umas das outras. Os resultados significativos que esse algoritmo produz são provenientes da combinação dos resultados de todas as árvores geradas. Esse paradigma de combinar as saídas de vários componentes para obtenção de uma solução final é conhecido no campo da inteligência computacional como *ensemble*. Intuitivamente, a combinação dos resultados de várias árvores é promis-

sora, pois cada uma delas pode implicitamente representar aspectos distintos para solução de um dado problema. No entanto, a melhoria do resultado final depende da diversidade dos erros obtidos por cada árvore. Dessa forma, cada uma das árvores deve apresentar um bom desempenho quando aplicada ao problema e deve cometer erros de classificação diferentes das demais árvores. Erros iguais para um mesmo conjunto de entrada, também podem implicar em acertos iguais, logo, não haverá aumento do desempenho do *ensemble*.

Para garantir que árvores distintas sejam geradas na floresta, a RF introduz aleatoriedade ao processo de indução. A primeira estratégia de aleatoriedade é o uso do modelo *bagging* (*Bootstrap AgreGGatING*) para selecionar as instâncias dos dados de treinamento para construção de cada uma das árvores de decisão da floresta. Nessa estratégia, b conjuntos de *bootstrap*, ou amostras são criadas. Cada conjunto é composto de n observações dos dados de treinamento, escolhidas por amostragem e com reposição. Apenas 2/3 de cada conjunto de *bootstrap* são usados para construir cada árvore. O 1/3 restante, chamado de amostras *Out-of-bag* (Oob), fornece uma avaliação do erro de classificação da árvore. A segunda estratégia é que apenas um subconjunto aleatório dos atributos de cada amostra é considerado para construção do teste no nó divisor.

O número ideal de árvores atribuído ao *ensemble* foi determinado pelo estudo da taxa de erro Oob nos dados de treinamento, conforme mostrado na Figura 6.10. Foram testados valores a partir de um conjunto formado por 10 árvores até um total de 300 e a combinação de 75 classificadores foi a que produziu o menor erro de avaliação. De acordo com o critério de avaliação utilizado, o aumento da quantidade de árvores apenas exigiria mais poder computacional sem nenhum ganho significativo de desempenho. Essa foi a configuração adotada para a RF nos resultados apresentados neste cenário.

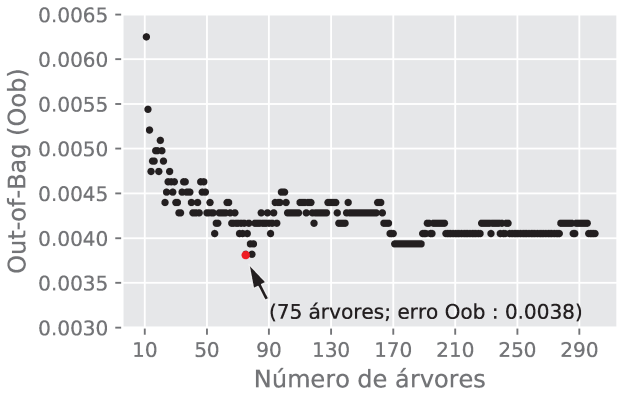

Figura 6.10 – Taxa de erro Oob produzido pela RF a partir da indução das árvores utilizando o conjunto de dados de treinamento.

A Figura 6.11 apresenta o resultado da detecção realizada para cada um dos métodos comparados. Para isso, é mostrada uma matriz de confusão para cada detector. Essas matrizes permitem visualizar facilmente se o método de detecção de anomalias foi

capaz de identificar corretamente cada um dos ataques e, assim, entender quais são os acertos e quais os tipos de erros o classificador está cometendo. As linhas representam as classes analisadas, ao passo que as colunas apresentam o resultado da classificação para cada uma das anomalias. Os elementos da diagonal principal da matriz de confusão mostram a quantidade de classificações realizadas corretamente, enquanto os demais valores indicam os erros cometidos pelo classificador.

Nos resultados mostrados pelas matrizes de confusão foi considerada apenas a identificação dos ataques como critério de avaliação. Entretanto, nas Figuras 6.11(a)-(c) também pode ser vista a classe normal, a qual foi acrescentada apenas para demonstrar os falsos negativos produzidos. Na Figura 6.11(d) essa classe é ausente, pois a RF não classificou incorretamente nenhum intervalo livre de anomalias.

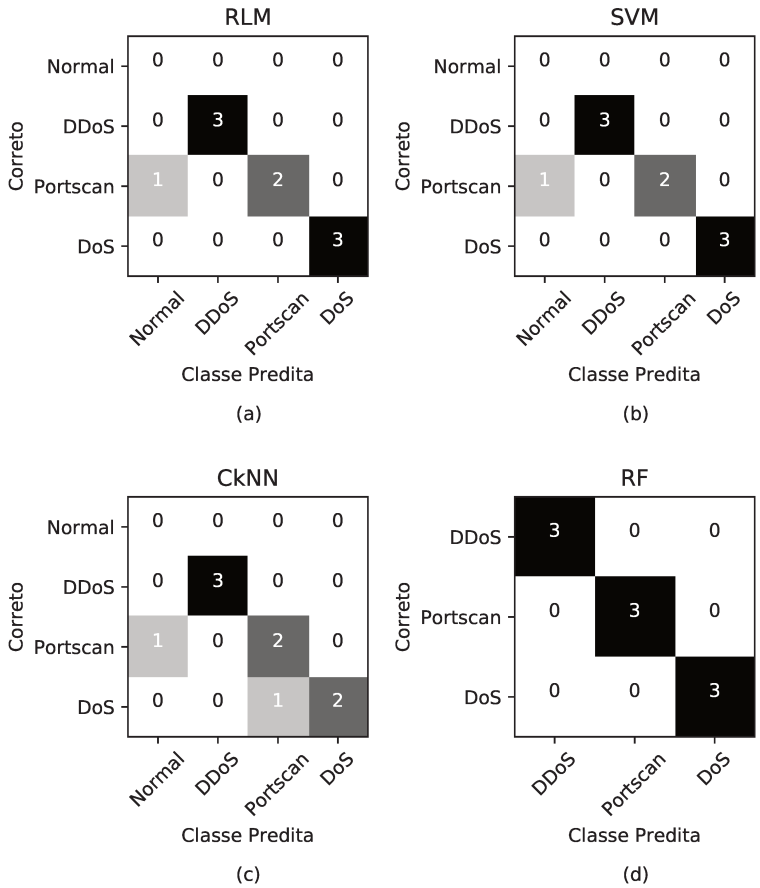

Figura 6.11 – Matriz de confusão de cada método de detecção comparado.

A partir da análise da diagonal principal da matriz apresentada na Figura 6.11(a), observa-se que a RLM reconheceu corretamente oito das nove anomalias. Também é possível verificar que o erro obtido pelo classificador se deu em relação a uma anomalia do tipo *port scan*. Como demonstrado na terceira linha, primeira coluna, esse evento foi equivocadamente caracterizado como normal. O resultado é o mesmo conseguido pela SVM, como mostra a Figura 6.11(b). Diferente dos dois métodos anteriores, o

CkNN apresentou mais uma ocorrência de classificação errada, como pode ser verificado na Figura 6.11(c). A quarta linha, terceira coluna, aponta que o CkNN atribuiu o rótulo *port scan* ao evento quando deveria tê-lo classificado como DoS.

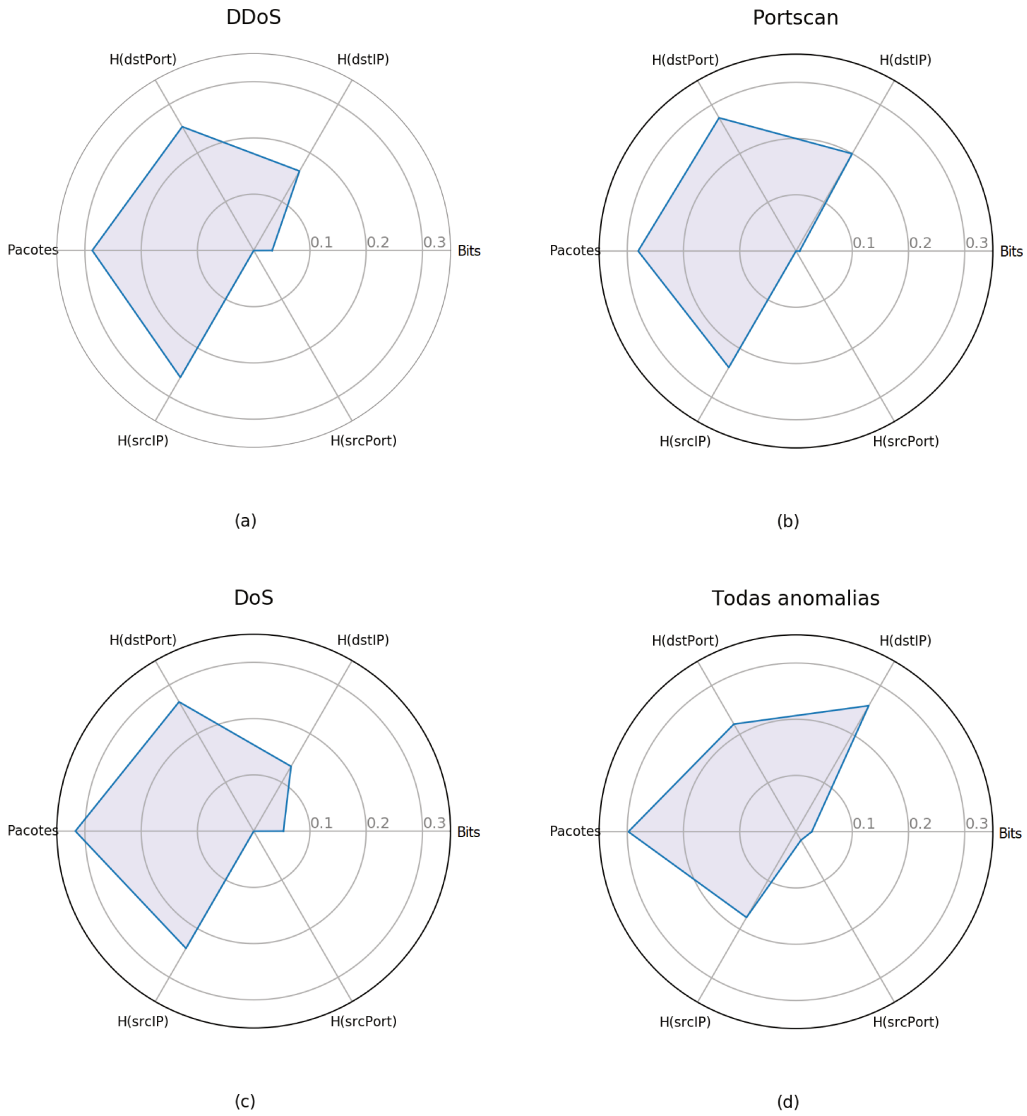

Figura 6.12 – Importância dos atributos do fluxo para a identificação de cada anomalia.

Após a verificação dos alarmes disparados durante o teste, constatou-se que os erros cometidos pelos métodos RLM, SVM e CkNN estão ligados às anomalias de menor intensidade. A partir disso, o *ensemble* de árvores de decisões geradas pela RF foi usado para entender a relevância de cada atributo do fluxo para o reconhecimento das anomalias. Durante o treinamento a RF utiliza das amostras Oob para construir uma medida de importância de cada atributo, ou seja, a força de predição de cada atributo. Quando uma árvore é criada, as amostras de seu conjunto Oob passam pela árvore e o resultado da classificação é armazenado. Em seguida, os valores do atributo j são permutados aleatoriamente e a taxa de classificação novamente é computada. A diminuição na taxa de acerto como resultado dessa permutação é calculada para cada árvore da floresta e toma-

se a média dessa diminuição como o fator de importância do atributo j. Se a diminuição da taxa de acerto é pequena, o atributo tem pouca importância.

A Figura 6.12 mostra a importância de cada atributo do fluxo para o conjunto de dados usado neste cenário. Quanto maior o valor associado a um atributo, mais ele contribui para o reconhecimento do evento anômalo. Por meio do gráfico de radar mostrado na Figura 6.12(a) pode-se notar que, durante os ataques DDoS, a quantidade de pacotes transmitidos bem como as entropias calculadas para os endereços IP de origem, endereço IP de destino e porta de destino são as características que mais influenciam na hora da diferenciação do ataque em relação ao comportamento normal esperado do tráfego. A Figura 6.12(b) indica que os mesmos atributos afetados pelo DDoS têm grande impacto na hora da detecção de um evento *port scan*. Entretanto, a diferença entre os ataques está na intensidade, que se traduz no montante de tráfego gerado. Enquanto o DDoS tenta inundar um serviço e torná-lo indisponível para os demais usuários, o *port scan* gera um tráfego mínimo necessário apenas para realizar a sondagem de portas ativas no alvo. Desse modo, mesmo que a importância dos atributos afetados seja semelhante nesses ataques, a variação observada nos valores dos atributos dos fluxo em relação ao padrão normal é menor durante um *port scan*. Essa característica pode levar o sistema de detecção a interpretar um ataque de pequena intensidade como variações do tráfego legítimo, resultando em erros de classificação como demonstrado na Figura 6.11(a).

Analisando o gráfico da Figura 6.12(c) é possível inferir que, analogamente ao ataque DDoS, a entropia calculada para os endereços IP de origem tem grande importância para detecção de DoS. Durante o DDoS, isso ocorre porque a anomalia é gerada por diversos atacantes e, consequentemente, o valor do atributo H(srcIP) pode apresentar um aumento ou uma diminuição por conta da dispersão dos IP de origem observados durante o ataque. Já no DoS, a mudança no valor de tal atributo é notória, pois um único IP de origem representa grande parte da comunicação observada no instante em que o ataque ocorre. Nesse caso, o valor de H(srcIP) tende a diminuir por conta da concentração de endereços IP de origem observados durante o monitoramento.

A Figura 6.12(d) apresenta a importância conferida a cada atributo para classificação correta dos três ataques analisados. Como pode ser visto, a propriedade do tráfego considerada mais relevante foi a quantidade de pacotes transmitidos, seguida pelas entropias de IP de origem (H(srcIP)), endereço de destino (H(dstIP)) e porta de destino (H(dstPort)). A atribuição de atributo mais significativo à quantidade de pacotes é justificada pelo fato de que todas as anomalias analisadas promovem um aumento no valor desse atributo, em menor ou maior intensidade. Quanto às entropias, elas sofrem uma redução em seus valores. Durante a ocorrência das anomalias, o tráfego produzido pelos atacantes é direcionado a uma porta específica em um alvo. Essa concentração faz

com que a distribuição de probabilidades dos endereços IP e porta de destino se tornem concentradas.

6.5 Cenário 4: Redução do intervalo de monitoramento

Neste cenário é apresentado um estudo de caso em que o intervalo de análise do tráfego é reduzido para 15 segundos. Dois objetivos são pretendidos com essa redução. O primeiro é evitar que os recursos, como as entradas das tabelas de encaminhamento dos *switches* e o canal de comunicação entre o plano de controle e dados, sejam saturados. O segundo é diminuir o *overhead* na criação da assinatura digital do tráfego, analisando invés de dias anteriores, minutos anteriores. Espera-se com isso ter um tempo de reação menor frente a possíveis anomalias que possam ocorrer, minimizando os problemas que possam afetar os usuário da rede.

Juntamente com a redução do intervalo de detecção também é realizada uma análise sobre a geração da assinatura digital do comportamento normal do tráfego. A finalidade é verificar se é possível criar uma assinatura adequada para a detecção baseada no histórico do próprio dia analisado. Para isso, medições F das propriedades do tráfego dos últimos w intervalos são usadas para a caracterização, como mostra a equação 6.6. O conjunto $\mathcal{C}_t$ é passado ao ACODS e como resultado é obtida a assinatura do comportamento normal $\mathbf{d}_t$ para o intervalo de tempo corrente t. No intervalo de análise seguinte, F_{t-w} é retirado do conjunto para a entrada da nova 6-tupla F_t. Essa característica faz com que o conjunto $\mathcal{C}_t$ seja formado por uma janela deslizante, atualizada a cada intervalo de 15 segundos.

$$\mathcal{C}_t = \{F_{t-1}, F_{t-2}, ..., F_{t-w}\} \tag{6.6}$$

Os dias que eram usados tradicionalmente como base histórica para a criação de uma única assinatura foram usados para avaliação da nova abordagem. A assinatura para esse conjunto de dados, baseada na janela deslizante, leva em conta os w intervalos anteriores. Essa nova assinatura foi submetida ao teste de *Normalized Mean Square Error* (NMSE). O NMSE avalia a diferença entre os valores esperados e os valores que realmente foram obtidos. No ambiente de teste voltado à verificação da caracterização do tráfego, essa métrica tem como objetivo mensurar a diferença entre o que foi previsto pela assinatura e o que realmente foi verificado na coleta de estatísticas do tráfego. Para se calcular o NMSE entre duas séries temporais, no caso a assinatura e o tráfego observado, utiliza-se

a seguinte equação:

$$\text{NMSE} = \frac{1}{N} \sum_{i} \frac{(P_i - M_i)^2}{\bar{P}\bar{M}}, \tag{6.7}$$

em que P é a série predita e M é a série criada pelas sucessivas estatísticas do tráfego coletadas. A variável N indica o tamanho das séries ao passo que $\bar{P}$ e $\bar{M}$ representam as médias dos valores previstos e observados, respectivamente.

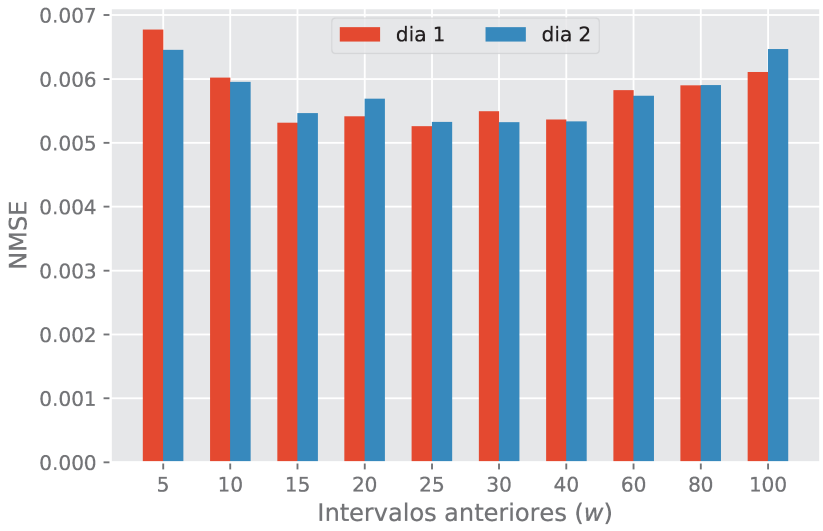

Figura 6.13 – NMSE avaliado para cada valor de w.

Para encontrar o valor de w que minimiza o NMSE, um conjunto de valores foi avaliado. O gráfico da Figura 6.13 compara os resultados para dois dias em que a assinatura foi gerada. Os valores de NMSE indicam o erro médio calculado entre os seis atributos do fluxo. Resultados próximos a zero indicam excelente caracterização do tráfego enquanto valores elevados indicam divergências entre a assinatura e o tráfego. Como pode ser observado, os menores erros registrados em ambos os dias analisados foram encontrados quando o valor de w foi atribuído como 25. Tanto valores superiores quanto inferiores produziram assinaturas digitais que se distanciaram do tráfego observado.

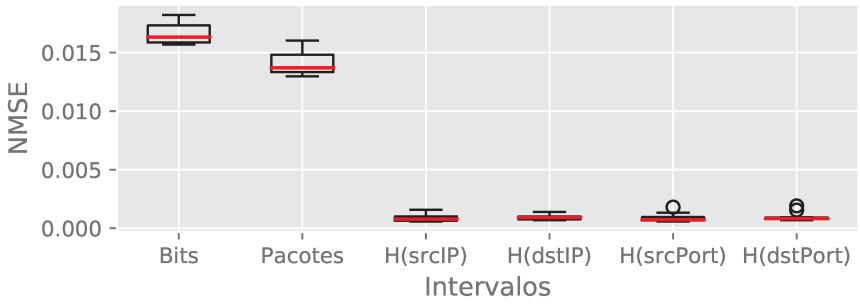

Figura 6.14 – *Boxplot* do erro NMSE de cada atributo.

A Figura 6.14 traz outra visão sobre os resultados da caracterização do tráfego. Os valores apresentados, por meio do gráfico *boxplot*, referem-se ao erro médio calculado para cada atributo do tráfego, variando-se o valor de w nos limites previamente fixados. Nota-se que os atributos de volume, bytes e pacotes, apresentaram maiores erros, demonstrando a dificuldade de sua caracterização. Tal característica é influenciada diretamente pelo tamanho da rede analisada, em que poucos *hosts* utilizando excessivamente a rede podem gerar grandes picos de tráfego, fazendo com que o comportamento analisado seja diferente do especificado pela assinatura.

A detecção dos eventos anômalos foi realizada utilizando as assinaturas geradas para cada configuração de w, como mostra a Figura 6.15. Para a avaliação, foram usadas as mesmas métricas do primeiro cenário. Quanto maior o valor fornecido pelas métricas, melhor a detecção realizada. Também foi acrescentada à analise a média aritmética calculada sobre os valores obtidos pelas métricas para cada assinatura. É possível observar que revocação, AUC e acurácia tiveram seus melhores valores quando $15 \leq w \leq 25$, como destacado na figura, demonstrando que a maioria dos intervalos com anomalias foi reconhecida corretamente. Entretanto, os valores de precisão e F-*measure* não foram condizentes com esse resultado. A partir de $w = 10$ notou-se uma tendência de decrescimento dessas métricas, que foi acompanhada pelas demais quando $w \geq 40$. A média também apresentou quedas nos valores das métricas de avaliação quando um número maior de intervalos anteriores foi usado para composição da assinatura.

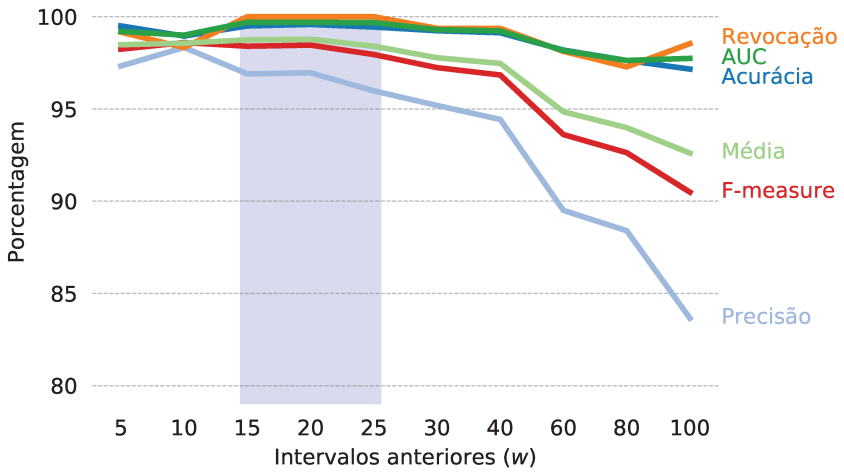

Figura 6.15 – Métricas de detecção para avaliação dos valores de w.

Para compreender o comportamento das métricas precisão e F-*measure* conforme a alteração no valor de w, foi verificada a taxa de falso positivo produzida pela detecção de acordo com cada assinatura. O gráfico da Figura 6.16 indica que o aumento dessa taxa é acompanhado pelo aumento do valor de w. Além disso, esse aumento causou a diminuição da precisão durante a detecção e, consequentemente, a redução do valor ob-

servado pela métrica F-*measure*. É importante notar que, mesmo obtendo a menor taxa de falso positivo, o número de intervalos anômalos reconhecidos resultante de $w = 10$ se mantém inferior aos valores decorrentes da construção da assinatura utilizando até 40 intervalos anteriores.

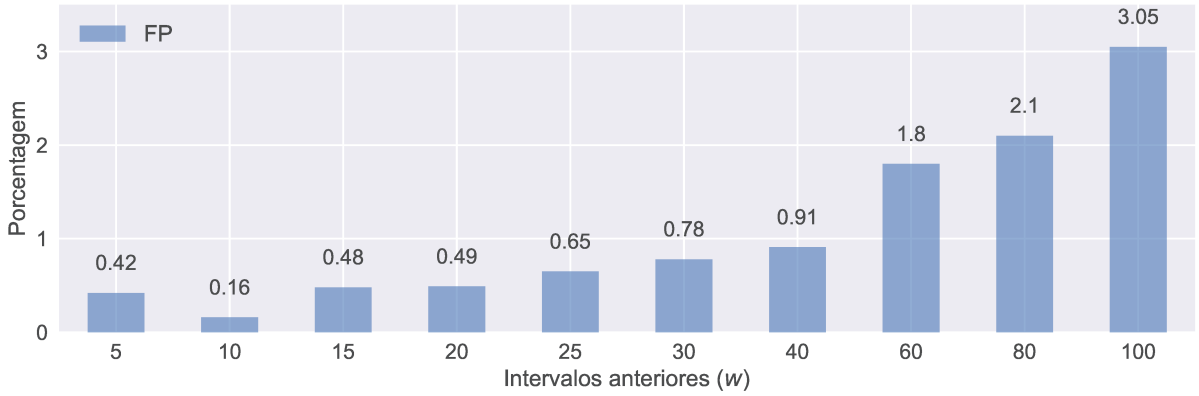

Figura 6.16 – Taxa de falso positivo para cada valor de w.

A Figura 6.17 apresenta o tráfego durante um ataque de DDoS do tipo TCP *flooding*, em que é possível verificar o seu início e o instante em que ele é encerrado. Observa-se que os valores de todas as entropias se tornaram concentrados durante o ataque. Para os endereços IP de origem (srcIP) e destino (dstIP) esse efeito foi consequência do uso de cinco atacantes transmitindo uma quantidade considerável de pacotes e fluxos para um único destino. Os envolvidos nesse evento malicioso foram responsáveis pela maior parte do tráfego gerado na rede e, por conta disso, dominaram a distribuição de probabilidade desses atributos. O mesmo ocorreu com as portas de origem (srcPort) e destino (dstPort), uma vez que os atacantes se utilizaram de um pequeno número de portas para o envio do tráfego à porta 80 do alvo.

Nenhuma política de mitigação foi ativada para que se pudesse comparar o resultado da detecção utilizando os casos extremos analisados neste cenário, isto é, a assinatura gerada pela análise de cinco e por cem intervalos anteriores. A figura demonstra ainda os alarmes acionados para cada intervalo anômalo classificado de acordo com a assinatura gerada. Quando utilizada a assinatura criada a partir da análise de cem intervalos anteriores, os alarmes são deslocados em relação ao período que a anomalia estava ocorrendo. Logo, os primeiros alarmes foram disparados vários intervalos de análise depois que o ataque estava em vigor, ocasionando diversos falsos negativos. Esse atraso de detecção também ocorreu na identificação do fim do DDoS. A mudança para o estado normal do tráfego só foi detectada após a ocorrência de diversos intervalos livres de anomalias, originando os falsos positivos. Esses erros de classificação também ocorreram quando a assinatura utilizando cinco intervalos anteriores foi aplicada na detecção, porém em uma menor proporção. O período em que os alarmes estão concentrados é mais ajustado ao

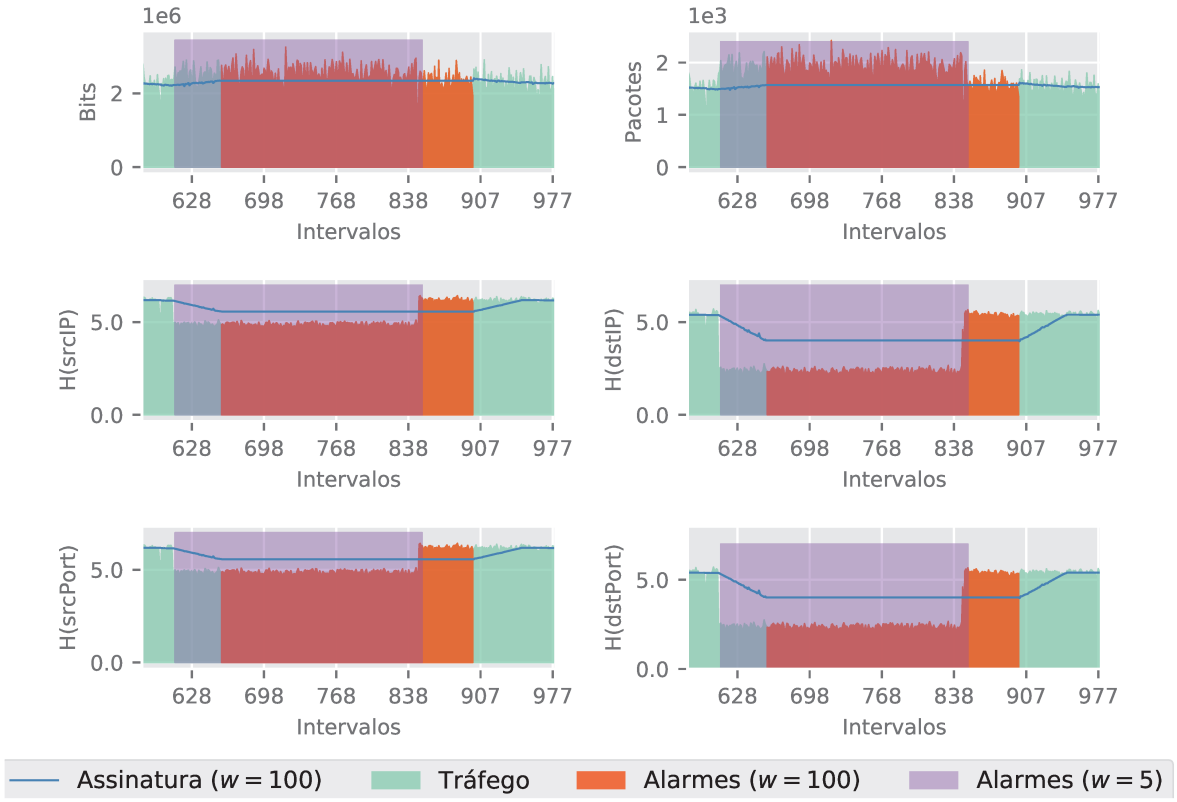

Figura 6.17 – Comparação dos alarmes disparados durante a detecção usando as assinaturas $w = 5$ e $w = 100$.

próprio período em que o DDoS estava ocorrendo.

Esses resultados apontam que a mudança de comportamento, seja anômalo para normal ou vice-versa, causa pouca influência na assinatura gerada quando $w = 100$. Dessa maneira, diversos intervalos devem ser analisados para que a assinatura comece a incorporar a alteração exibida pelo tráfego da rede. Por outro lado, quando $w = 5$ é utilizado, a assinatura gerada tende a se adaptar rapidamente ao movimento do tráfego recém-coletado. Esse efeito pode levar a classificações erradas, caso mudanças pontuais do uso legítimo da rede ocorram de forma abrupta. Portanto, a quantidade de intervalos usados como histórico do tráfego não pode ser grande a ponto de produzir uma detecção de anomalias tardia, tampouco ser pequeno para permitir a incorporação na assinatura de flutuações que não correspondam ao padrão normal de comportamento.

Com base nos resultados apresentados, pode-se inferir que o equilíbrio entre os maiores valores de acerto (AUC, revocação, acurácia e precisão) e a redução da taxa de falsos positivos (TFP) foi atingida quando $w = 15$ intervalos anteriores de 15 segundos foram utilizados para geração da assinatura digital. Em comparação com a abordagem tradicional, apresentada nas seções anteriores, pôde-se verificar que a redução do intervalo introduziu um erro menor de previsão do comportamento normal, como mostra a Tabela 6.3.

Tabela 6.3 – Comparação dos resultados entre abordagem tradicional e intervalo de análise reduzido.

	Abordagem tradicional (4 dias, janela de 30 segundos)	Intervalo reduzido (Mesmo dia, janela de 15 segundos)
NMSE	0,0055	0,0051
Acurácia	0,9955	0,995
Precisão	0,9127	0,969
Revocação	0,9829	1
F-Measure	0,9465	0,984
AUC	0,9895	0,997
TFP	0,0026	0,0048

Na abordagem que utiliza o mesmo dia como tráfego histórico, a redução no erro NMSE também foi acompanhada pelo aumento das demais métricas, incluindo a taxa de falsos positivos. Essa análise permitiu verificar que, nos testes realizados, o tráfego corrente é melhor descrito pelo comportamento desempenhado pelos intervalos de análise anteriores ao invés de uma base histórica que leva em consideração os comportamentos observados sucessivamente ao longo de dias. No entanto, quando o intervalo de análise é reduzido, uma variação normal, porém brusca no tráfego corrente, pode erroneamente ser considerada uma anomalia e a detecção poderá ser prejudicada por conta do erro de classificação nessa situação. Ao se utilizar uma base histórica maior, como no caso de dias, essas flutuações do tráfego terão menor influência na criação da assinatura digital e tais eventos poderão ser classificados corretamente.

Além dos melhores resultados, a redução do intervalo de análise permite que os eventos anômalos sejam identificados em um período menor de tempo. Como consequência, as rotinas voltadas à mitigação das anomalias poderão ser acionadas mais rapidamente após a detecção. Analogamente, a diminuição da base histórica do tráfego de quatro dias anteriores para w intervalos passados do mesmo dia de análise evita o problema de se manter e processar um grande volume de dados para a criação da assinatura digital.

6.6 Considerações sobre o capítulo

A avaliação do ecossistema proposto foi dividida em cenários. Utilizando o controlador POX, o primeiro cenário comprovou a eficiência dos mecanismos de detecção e mitigação adotados. Dada a implementação em módulos do ecossistema, outros métodos de detecção puderam ser implementados e comparados durante os testes. No segundo cenário, a análise se estendeu para verificar o tempo consumido por cada atividade do ecossistema, bem como o uso de recursos computacionais.

O controlador Floodlight foi utilizado nos dois últimos cenários. O terceiro cenário evidenciou a importância de cada atributo de fluxo para o reconhecimento das anomalias. Cada ataque contribui de maneira diferente para o desvio do tráfego em relação ao padrão normal estabelecido. Caso atributos relevantes para a identificação do comportamento anômalo sejam minimamente afetados, o método de detecção pode sofrer com erros de classificação, ocasionando falsos negativos.

O quarto cenário difere dos demais por alterar o funcionamento das atividades de coleta do tráfego, detecção e mitigação. Nele foi verificada a viabilidade de se criar uma assinatura digital tendo como base histórica o tráfego do dia corrente. Os resultados mostraram que, dado uma quantidade de intervalos anteriores, a previsão do tráfego normal pode derivar em melhores taxas de detecção do que aquelas obtidas quando uma assinatura é gerada por uma base histórica mais extensa. A redução do intervalo de análise também permitiu constatar que a taxa de reconhecimento de anomalias se mantém e o módulo de mitigação pode se aproveitar esse resultado para reagir mais rapidamente à incidência de anomalias.

Em síntese, objetivo dos cenários propostos foi avaliar o sistema de detecção e mitigação de anomalias. A utilização dos controladores POX e Floodlight revelou que o ecossistema pode ser implantado em diferentes plataformas. Além disso, o uso dos controladores nos cenários não demonstrou diferenças que pudessem impactar na eficiência do ecossistema proposto.

7 Conclusão

A detecção e o tratamento de anomalias presentes no tráfego são tarefas indispensáveis para assegurar a efetividade da gerência de redes. Entretanto, a realização dessas atividades se torna cada vez mais complexa nas redes atuais. Maior parte dessa dificuldade é resultante da diversidade de dispositivos e *software* com diferentes tecnologias que devem ser configurados separadamente, de acordo com as interfaces fornecidas pelos fabricantes. Com a finalidade de desvincular a inteligência de controle dos dispositivos de encaminhamento e promover soluções de comunicação padronizadas, as redes definidas por software foram criadas. As facilidades oferecidas por essa tecnologia fizeram com que as tarefas que exigissem o controle dos dispositivos de encaminhamento fossem simplificadas. Utilizando-se dessas facilidades, nesta tese foi aprestada uma solução para a detecção e mitigação de anomalias de redes. A proposta foi desenvolvida no intuito de cooperar pró-ativamente com gerenciamento de redes por meio da automatização das tarefas de monitoramento, detecção de ameaças e autorreparo. As principais contribuições obtidas com esta proposta foram:

- **Detecção de anomalias em ambientes SDN:** proposição de um ecossistema que integra o plano de aplicações, controle e dados a fim de reconhecer anomalias de rede. O funcionamento da proposta se dá pela operação de módulos anexados ao controlador SDN que podem ser individualmente alterados;
- **Mitigação de anomalias em tempo real:** políticas destinadas à manutenção do funcionamento normal da rede instruem o plano de dados para atenuação de atividades maliciosas detectadas. Isso ocorre pela inserção de entradas de fluxo nos *switches* em resposta aos desvios do tráfego monitorado em relação ao padrão normal previamente definido;
- **Aplicação da proposta em ambientes SDN com diferentes controladores:** o ecossistema se mostrou robusto e versátil quando avaliado em dois ambientes de rede emulados, em que um deles foi utilizado o controlador POX e em outro o controlador Floodlight;
- **Redução do intervalo de análise para detecção de anomalias:** um estudo sobre a viabilidade da diminuição do histórico do tráfego necessário para a criação da assinatura do tráfego normal foi realizado. Também foi reduzido o período entre coletas de estatísticas do tráfego a fim de diminuir o tempo de resposta aos ataques detectados.

Para a validação da solução proposta, quatro cenários de testes foram utilizados. Os cenários foram construídos com a intenção de avaliar características importantes do sistema proposto. Cada um deles diferenciou-se pelo controlador SDN utilizado e pelas anomalias geradas. No primeiro cenário, a taxa de detecção obtida pela proposta foi comparada com outros métodos destinados ao mesmo propósito. Os resultados mostraram que a abordagem conseguiu se sobressair em relação aos demais métodos de detecção, sendo a única a reconhecer todos os eventos anômalos presentes no tráfego. Não se restringindo à taxa de acerto, foram realizados também diversos testes para se verificar com maior clareza o desempenho da detecção realizada. A acurácia, precisão, revocação e AUC, comprovaram que a solução foi mais sensível às anomalias e conseguiu reconhecer a grande maioria dos intervalos analisados em que elas ocorreram. Outro ponto a se destacar é o de que nenhum falso positivo foi observado, ou seja, nenhum intervalo cujo o tráfego era legítimo foi considerado anômalo.

O segundo cenário utilizou-se do ambiente de teste do cenário anterior para avaliação da mitigação e dos recursos computacionais utilizados. Durante o monitoramento, quando o primeiro intervalo anômalo de um ataque era reconhecido, uma rotina de mitigação era acionada para conter tal evento. As estratégias de mitigação elaboradas se mostraram efetivas para a contenção dos ataques DDoS e *port scan*. Quanto ao desempenho, uma das preocupações preponderantes em redes definidas por *software*, o uso da CPU do controlador foi verificado. Foram constatados pequenos picos por conta da atividade do módulo de coleta do tráfego e outros ligeiramente acentuados devido à atividade do módulo de mitigação. Entretanto, respostas aos ataques foram enviadas aos dispositivos do plano de dados na ordem de milissegundos. Com relação ao uso das entradas de fluxos das tabelas dos *switches*, constatou-se que as políticas aplicadas a cada uma das anomalias contribuíram eficientemente para evitar o uso indevido desse recurso.

A sensibilidade do detector proposto nesta tese foi testada no terceiro cenário e comparada com outros métodos de detecção e anomalia. Dentre os ataques analisados, apenas um não foi possível de ser identificado por conta da sua intensidade. Tal anomalia alterou minimamente os atributos de fluxos analisados, fazendo com que a diferença entre o tráfego verificado e a assinatura digital não fosse suficiente para o método de detecção caracterizá-la como anomalia. Por conta disso, também foi realizada uma análise sobre a importância associada a cada atributo do fluxo para a identificação correta dos tipos de anomalias testados.

A preocupação com o tempo de reação a uma anomalia levou aos resultados apresentados no último cenário. Um caso de estudo foi conduzido para verificar a aplicabilidade da abordagem em um ambiente em que a janela de análise do tráfego passou de 30 para 15 segundos. O estudo ainda contou com uma metodologia alternativa para a

criação da assinatura digital. A base histórica utilizada para a criação do padrão normal de comportamento foi reduzida de quatro dias para alguns intervalos do tráfego corrente, no modelo de janela deslizante. Os resultados apontaram que a utilização de um número pequeno de intervalos para a previsão do comportamento normal pode ser susceptível aos eventos esporádicos e às mudanças súbitas do tráfego, podendo incorporar à assinatura um evento anômalo. Porém, quando uma janela de análise grande é utilizada, a assinatura pode demorar a capturar um mudança legítima no tráfego e, consequentemente, reconhecer essas alterações como anomalias.

Os resultados obtidos demonstram que o ecossistema é eficiente tanto para a detecção quanto para a mitigação de anomalias. Por meio da automatização do processo de identificação de eventos prejudiciais às redes e o acionamento de contramedidas, a responsabilidade atribuída ao administrador de monitorar o tráfego e controlar vários segmentos de rede ao mesmo tempo é facilitada. As vantagens decorrentes dessa praticidade são a exclusão do erro proveniente de operadores humanos, além da possibilidade de concentrar esforços em outros problemas que requerem atenção. Dessa maneira, a solução apresentada nesta tese se mostra adequada para auxiliar o gerenciamento de redes, contribuindo para a manutenção da disponibilidade e resiliência da rede e dos serviços prestados.

O ecossistema apresentado possui um design modular, pois as tarefas de coleta do tráfego, detecção de anomalias, mitigação e relatório são desacopladas para que possam ser convenientemente adaptadas para tratar de novos problemas de segurança. Isso permitiu que outros métodos de detecção pudessem ser inseridos ao ecossistema e avaliados nesta tese. Utilizando essa característica, em trabalhos futuros, outras técnicas voltadas à detecção de anomalias podem ser incorporadas, bem como novas políticas de mitigação. Nesse âmbito, o reconhecimento de ataques do tipo *zero day* e de múltiplas anomalias que ocorrem no mesmo intervalo de análise compõem campos promissores de pesquisa. O trabalho também pode ser estendido para considerar o uso de múltiplos controladores para conferir ao plano de controle maior desempenho e tolerância a falhas.

Referências

[1] M. Karakus and A. Durresi, "A survey: Control plane scalability issues and approaches in Software-Defined Networking (SDN)," *Computer Networks*, vol. 112, pp. 279 – 293, 2017.

[2] N. Bizanis and F. A. Kuipers, "SDN and Virtualization Solutions for the Internet of Things: A Survey," *IEEE Access*, vol. 4, pp. 5591–5606, 2016.

[3] Cisco, "Cisco visual networking index: Forecast and methodology, 2016–2021," tech. rep., Cisco, 2017. `https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.pdf`.

[4] M. Jammal, T. Singh, A. Shami, R. Asal, and Y. Li, "Software defined networking: State of the art and research challenges," *Computer Networks*, vol. 72, pp. 74 – 98, 2014.

[5] W. Zhang, X. Zhang, H. Shi, and L. Zhou, "An efficient latency monitoring scheme in software defined networks," *Future Generation Computer Systems*, vol. 83, pp. 303 – 309, 2018.

[6] B. Carpenter and S. Brim, "Middleboxes: Taxonomy and issues," RFC 3234, RFC Editor, February 2002. `http://www.rfc-editor.org/rfc/rfc3234.txt`.

[7] D. Kreutz, F. M. Ramos, and P. Verissimo, "Towards Secure and Dependable Software-defined Networks," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, HotSDN '13, (New York, NY, USA), pp. 55–60, ACM, 2013.

[8] M. Karakus and A. Durresi, "Economic Viability of Software Defined Networking (SDN)," *Computer Networks*, vol. 135, pp. 81 – 95, 2018.

[9] B. Wang, Y. Zheng, W. Lou, and Y. T. Hou, "DDoS attack protection in the era of cloud computing and Software-Defined Networking," *Computer Networks*, vol. 81, pp. 308 – 319, 2015.

[10] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "Avant-guard: Scalable and vigilant switch flow management in software-defined networks," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, (New York, NY, USA), pp. 413–424, ACM, 2013.

[11] A. Aleroud and I. Alsmadi, "Identifying DoS attacks on software defined networks: A relation context approach," in *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, pp. 853–857, April 2016.

[12] T. Ha, S. Kim, N. An, J. Narantuya, C. Jeong, J. Kim, and H. Lim, "Suspicious traffic sampling for intrusion detection in software-defined networks," *Computer Networks*, vol. 109, Part 2, pp. 172 – 182, 2016.

[13] L. Wei and C. Fung, "FlowRanger: A request prioritizing algorithm for controller DoS attacks in Software Defined Networks," in *2015 IEEE International Conference on Communications (ICC)*, pp. 5254–5259, June 2015.

[14] Y. Cui, L. Yan, S. Li, H. Xing, W. Pan, J. Zhu, and X. Zheng, "SD-Anti-DDoS: Fast and efficient DDoS defense in software-defined networks," *Journal of Network and Computer Applications*, vol. 68, pp. 65 – 79, 2016.

[15] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, "Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments," *Computer Networks*, vol. 62, pp. 122 – 136, 2014.

[16] O. Joldzic, Z. Djuric, and P. Vuletic, "A transparent and scalable anomaly-based DoS detection method," *Computer Networks*, vol. 104, pp. 27 – 42, 2016.

[17] N. Zahadat, P. Blessner, T. Blackburn, and B. A. Olson, "BYOD security engineering: A framework and its analysis," *Computers & Security*, vol. 55, pp. 81 – 99, 2015.

[18] A. Akhunzada, A. Gani, N. B. Anuar, A. Abdelaziz, M. K. Khan, A. Hayat, and S. U. Khan, "Secure and dependable software defined networks," *Journal of Network and Computer Applications*, vol. 61, pp. 199 – 221, 2016.

[19] H. Farhady, H. Lee, and A. Nakao, "Software-Defined Networking: A survey," *Computer Networks*, vol. 81, pp. 79 – 95, 2015.

[20] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computer Surveys*, vol. 41, pp. 15:1–15:58, July 2009.

[21] M. Ahmed, A. N. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19 – 31, 2016.

[22] A. Patcha and J.-M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Computer Networks*, vol. 51, no. 12, pp. 3448 – 3470, 2007.

[23] G. Fernandes, J. J. P. C. Rodrigues, L. F. Carvalho, J. F. Al-Muhtadi, and M. L. Proença, "A comprehensive survey on network anomaly detection," *Telecommunication Systems*, Jul 2018.

[24] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: Methods, systems and tools," *IEEE Communications Surveys Tutorials*, vol. 16, pp. 303–336, First 2014.

[25] S. W. A.-H. Baddar, A. Merlo, and M. Migliardi, "Anomaly Detection in Computer Networks: A State-of-the-Art Review," *Journal of Wireless Networks, Ubiquitous Computing, and Dependable Applications*, vol. 5, pp. 29–64, 2014.

[26] X. Song, M. Wu, C. Jermaine, and S. Ranka, "Conditional anomaly detection," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, pp. 631–645, May 2007.

[27] P. Barford, J. Kline, D. Plonka, and A. Ron, "A signal analysis of network traffic anomalies," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment*, IMW' 02, (New York, NY, USA), pp. 71–82, ACM, 2002.

[28] A. Marnerides, A. Schaeffer-Filho, and A. Mauthe, "Traffic anomaly diagnosis in internet backbone networks: A survey," *Computer Networks*, vol. 73, pp. 224 – 243, 2014.

[29] A. Löf and R. Nelson, "Annotating network trace data for anomaly detection research," in *39th Annual IEEE Conference on Local Computer Networks Workshops*, pp. 679–684, Sept 2014.

[30] P. Barford and D. Plonka, "Characteristics of network traffic flow anomalies," in *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, IMW' 01, (New York, NY, USA), pp. 69–73, ACM, 2001.

[31] S. Behal, K. Kumar, and M. Sachdeva, "Characterizing DDoS attacks and flash events: Review, research gaps and future directions," *Computer Science Review*, vol. 25, pp. 101 – 114, 2017.

[32] M. N., A. Parmar, and M. Kumar, "A flow based anomaly detection system using chi-square technique," in *2010 IEEE 2nd International Advance Computing Conference (IACC)*, pp. 285–289, Feb 2010.

[33] N. Weaver, V. Paxson, S. Staniford, and R. Cunningham, "A taxonomy of computer worms," in *Proceedings of the 2003 ACM Workshop on Rapid Malcode*, WORM' 03, (New York, NY, USA), pp. 11–18, ACM, 2003.

[34] I. Dubrawsky, "Chapter 2 - general security concepts: Attacks," in *Security+ (Second Edition)* (I. Dubrawsky, ed.), pp. 55 – 100, Burlington: Syngress, second ed., 2007.

[35] R. Russell, T. Bidwell, O. Steudler, R. Walshaw, and L. B. Huston, "Chapter 2 - DDoS Attacks: Intent, Tools, and Defense," in *Hack Proofing Your E-Commerce Site*, The Only Way to Stop a Hacker is to Think Like One, pp. 45 – 118, Rockland: Syngress, 2001.

[36] R. Hunt, "SNMP, SNMPv2 and CMIP — the technologies for multivendor network management," *Computer Communications*, vol. 20, no. 2, pp. 73 – 88, 1997.

[37] M. Thottan and C. Ji, "Anomaly detection in IP networks," *IEEE Transactions on Signal Processing*, vol. 51, pp. 2191–2204, Aug 2003.

[38] J. Case, R. Mundy, D. Partain, and B. Stewart, "Introduction and applicability statements for internet-standard management framework," RFC 3410, RFC Editor, December 2002. `http://www.rfc-editor.org/rfc/rfc3410.txt`.

[39] P. Phaal, S. Panchen, and N. McKee, "Inmon corporation's sflow: A method for monitoring traffic in switched and routed networks," tech. rep., RFC Editor, United States, 2001. `https://www.ietf.org/rfc/rfc3176.txt`.

[40] B. Claise, "Cisco systems netflow services export version 9," RFC 3954, RFC Editor, October 2004. `http://www.rfc-editor.org/rfc/rfc3954.txt`.

[41] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 69–74, Mar. 2008.

[42] Z. Su, T. Wang, Y. Xia, and M. Hamdi, "Cemon: A cost-effective flow monitoring system in software defined networks," *Computer Networks*, vol. 92, pp. 101 – 115, 2015.

[43] A. Tootoonchian, M. Ghobadi, and Y. Ganjali, "OpenTM: Traffic Matrix Estimator for OpenFlow Networks," in *Passive and Active Measurement* (A. Krishnamurthy and B. Plattner, eds.), (Berlin, Heidelberg), pp. 201–210, Springer Berlin Heidelberg, 2010.

[44] C. Yu, C. Lumezanu, Y. Zhang, V. Singh, G. Jiang, and H. V. Madhyastha, "Flowsense: Monitoring network utilization with zero measurement cost," in *Proceedings*

of the 14th International Conference on Passive and Active Measurement, PAM'13, (Berlin, Heidelberg), pp. 31–41, Springer-Verlag, 2013.

[45] L. Jose, M. Yu, and J. Rexford, "Online measurement of large traffic aggregates on commodity switches," in *Proceedings of the 11th USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*, Hot-ICE' 11, (Berkeley, CA, USA), pp. 13–13, USENIX Association, 2011.

[46] R. A. Kemmerer and G. Vigna, "Intrusion detection: a brief history and overview," *Computer*, vol. 35, pp. 27–30, Apr 2002.

[47] P. L. Campbell, "The denial-of-service dance," *IEEE Security Privacy*, vol. 3, pp. 34–40, Nov 2005.

[48] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16 – 24, 2013.

[49] R. Gwadera, M. Atallah, and W. Szpankowski, "Reliable detection of episodes in event sequences," in *Third IEEE International Conference on Data Mining*, pp. 67–74, Nov 2003.

[50] G. Fernandes, J. J. Rodrigues, and M. L. Proença, "Autonomous profile-based anomaly detection system using principal component analysis and flow analysis," *Applied Soft Computing*, vol. 34, pp. 513 – 525, 2015.

[51] E. H. Pena, L. F. Carvalho, S. Barbon, J. J. Rodrigues, and M. L. Proença, "Anomaly detection using the correlational paraconsistent machine with digital signatures of network segment," *Information Sciences*, vol. 420, pp. 313 – 328, 2017.

[52] R. Mitchell and I.-R. Chen, "A survey of intrusion detection in wireless network applications," *Computer Communications*, vol. 42, pp. 1 – 23, 2014.

[53] J. R. Vacca, *Computer and Information Security Handbook*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009.

[54] G. Fernandes, E. H. M. Pena, L. F. Carvalho, J. J. P. C. Rodrigues, and M. L. Proença, "Statistical, forecasting and metaheuristic techniques for network anomaly detection," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, SAC '15, (New York, NY, USA), pp. 701–707, ACM, 2015.

[55] R. Masoudi and A. Ghaffari, "Software defined networks: A survey," *Journal of Network and Computer Applications*, vol. 67, pp. 1 – 25, 2016.

[56] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE*, vol. 103, pp. 14–76, Jan 2015.

[57] B. Raghavan, M. Casado, T. Koponen, S. Ratnasamy, A. Ghodsi, and S. Shenker, "Software-defined internet architecture: Decoupling architecture from infrastructure," in *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, HotNets-XI, (New York, NY, USA), pp. 43–48, ACM, 2012.

[58] N. Feamster, J. Rexford, and E. Zegura, "The road to sdn: An intellectual history of programmable networks," *SIGCOMM Computer Communication Review*, vol. 44, pp. 87–98, Apr. 2014.

[59] R. M. da Silva Bezerra and J. S. B. Martins, "Network autonomic management: A tutorial with conceptual, functional and practical issues," *IEEE Latin America Transactions*, vol. 12, pp. 306–314, March 2014.

[60] X. Liu, P. Juluri, and D. Medhi, "An experimental study on dynamic network reconfiguration in a virtualized network environment using autonomic management," in *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pp. 616–622, May 2013.

[61] Y. Zhang, L. Cui, W. Wang, and Y. Zhang, "A survey on software defined networking with multiple controllers," *Journal of Network and Computer Applications*, vol. 103, pp. 101 – 118, 2018.

[62] ONF, "The SDN architecture," tech. rep., Open Network Foundation, June 2014. `https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR_SDN_ARCH_1.0_06062014.pdf`.

[63] ONF, "The SDN architecture," tech. rep., Open Network Foundation, February 2016. `https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR-521_SDN_Architecture_issue_1.1.pdf`.

[64] ONF, "Openflow switch specification 1.5.1," tech. rep., Open Network Foundation, 2015. `https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf`.

[65] ONF, "Openflow switch specification 1.4," tech. rep., Open Network Foundation, 2013. `https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2014/10/openflow-spec-v1.4.0.pdf`.

[66] ONF, "Openflow switch specification 1.0," tech. rep., Open Network Foundation, 2009. https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2013/04/openflow-spec-v1.0.0.pdf.

[67] ONF, "Openflow switch specification 1.1," tech. rep., Open Network Foundation, 2011. https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2014/10/openflow-spec-v1.1.0.pdf.

[68] ONF, "Openflow switch specification 1.2," tech. rep., Open Network Foundation, 2011. https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2014/10/openflow-spec-v1.2.pdf.

[69] ONF, "Openflow switch specification 1.3," tech. rep., Open Network Foundation, 2012. https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf.

[70] ONF, "Openflow switch specification 1.5," tech. rep., Open Network Foundation, 2014. https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2014/10/openflow-switch-v1.5.0.pdf.

[71] T. Dierks and C. Allen, "The TLS Protocol Version 1.0," RFC 2246, RFC Editor, January 1999. http://www.rfc-editor.org/rfc/rfc2246.txt.

[72] L. Yang, R. Dantu, T. Anderson, and R. Gopal, "Forwarding and control element separation (forces) framework," RFC 3746, RFC Editor, April 2004.

[73] B. Pfaff and B. Davie, "The open vswitch database management protocol," RFC 7047, RFC Editor, December 2013. http://www.rfc-editor.org/rfc/rfc7047.txt.

[74] H. Song, "Protocol-oblivious Forwarding: Unleash the Power of SDN Through a Future-proof Forwarding Plane," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, HotSDN' 13, (New York, NY, USA), pp. 127–132, ACM, 2013.

[75] M. Smith, M. Dvorkin, Y. Laribi, V. Pandey, P. Garg, and N. Weidenbacher, "Opflex control protocol," Internet-Draft draft-smith-opflex-00, IETF Secretariat, April 2014. http://www.ietf.org/internet-drafts/draft-smith-opflex-00.txt.

[76] S. Li, D. Hu, W. Fang, S. Ma, C. Chen, H. Huang, and Z. Zhu, "Protocol Oblivious Forwarding (POF): Software-Defined Networking with Enhanced Programmability," *IEEE Network*, vol. 31, pp. 58–66, March 2017.

[77] Y. E. Oktian, S. Lee, H. Lee, and J. Lam, "Distributed SDN Controller System," *Computer Networks*, vol. 121, pp. 100–111, July 2017.

[78] D. Erickson, "The beacon openflow controller," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, HotSDN' 13, (New York, NY, USA), pp. 13–18, ACM, 2013.

[79] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks," in *Proceedings of the 2Nd USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*, Hot-ICE' 12, (Berkeley, CA, USA), pp. 10–10, USENIX Association, 2012.

[80] B. A. A. Nunes, M. Mendonca, X. N. Nguyen, K. Obraczka, and T. Turletti, "A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks," *IEEE Communications Surveys Tutorials*, vol. 16, pp. 1617–1634, Third 2014.

[81] C. Trois, M. D. D. Fabro, L. C. E. de Bona, and M. Martinello, "A Survey on SDN Programming Languages: Toward a Taxonomy," *IEEE Communications Surveys Tutorials*, vol. 18, pp. 2687–2712, Fourthquarter 2016.

[82] A. Hakiri, A. Gokhale, P. Berthou, D. C. Schmidt, and T. Gayraud, "Software-Defined Networking: Challenges and research opportunities for Future Internet," *Computer Networks*, vol. 75, pp. 453 – 471, 2014.

[83] N. Foster, M. J. Freedman, R. Harrison, J. Rexford, M. L. Meola, and D. Walker, "Frenetic: A high-level language for openflow networks," in *Proceedings of the Workshop on Programmable Routers for Extensible Services of Tomorrow*, PRESTO '10, (New York, NY, USA), pp. 6:1–6:6, ACM, 2010.

[84] C. Monsanto, J. Reich, N. Foster, J. Rexford, and D. Walker, "Composing Software Defined Networks," in *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, (Lombard, IL), pp. 1–13, USENIX Association, 2013.

[85] A. Voellmy, H. Kim, and N. Feamster, "Procera: A Language for High-level Reactive Network Control," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN' 12, (New York, NY, USA), pp. 43–48, ACM, 2012.

[86] Z. Wan and P. Hudak, "Functional Reactive Programming from First Principles," in *Proceedings of the ACM SIGPLAN 2000 Conference on Programming Language Design and Implementation*, PLDI' 00, (New York, NY, USA), pp. 242–252, ACM, 2000.

[87] J. Galán-Jiménez, "Legacy IP-upgraded SDN nodes tradeoff in energy-efficient hybrid IP/SDN networks," *Computer Communications*, vol. 114, pp. 106 – 123, 2017.

[88] X. Jia, Y. Jiang, Z. Guo, G. Shen, and L. Wang, "Intelligent path control for energy-saving in hybrid SDN networks," *Computer Networks*, vol. 131, pp. 65 – 76, 2018.

[89] M. Karakus and A. Durresi, "Quality of Service (QoS) in Software Defined Networking (SDN): A survey," *Journal of Network and Computer Applications*, vol. 80, pp. 200 – 218, 2017.

[90] H. T. Nguyen, A. V. Vu, D. L. Nguyen, V. H. Nguyen, M. N. Tran, Q. T. Ngo, T.-H. Truong, T. H. Nguyen, and T. Magedanz, "A generalized resource allocation framework in support of multi-layer virtual network embedding based on SDN," *Computer Networks*, vol. 92, pp. 251 – 269, 2015. Software Defined Networks and Virtualization.

[91] Y. Yu, D. Li, and Y. Huang, "SVirt: A Substrate-agnostic SDN Virtualization Architecture for Multi-tenant Cloud," in *2015 IEEE 23rd International Conference on Network Protocols (ICNP)*, pp. 313–322, Nov 2015.

[92] I. F. Akyildiz, S.-C. Lin, and P. Wang, "Wireless software-defined networks (W-SDNs) and network function virtualization (NFV) for 5G cellular systems: An overview and qualitative evaluation," *Computer Networks*, vol. 93, pp. 66 – 79, 2015.

[93] M. L. Proença, G. Fernandes, L. F. Carvalho, M. V. O. Assis, and J. J. P. C. Rodrigues, "Digital signature to help network management using flow analysis," *International Journal of Network Management*, vol. 26, no. 2, pp. 76–94, 2015.

[94] E. H. M. Pena, L. F. Carvalho, S. Barbon, J. J. P. C. Rodrigues, and M. L. Proença, "Correlational paraconsistent machine for anomaly detection," in *2014 IEEE Global Communications Conference*, pp. 551–556, Dec 2014.

[95] M. V. O. de Assis, J. J. P. C. Rodrigues, and M. L. Proença, "A novel anomaly detection system based on seven-dimensional flow analysis," in *2013 IEEE Global Communications Conference (GLOBECOM)*, pp. 735–740, Dec 2013.

[96] L. F. Carvalho, T. Abrão, L. de Souza Mendes, and M. L. Proença, "An ecosystem for anomaly detection and mitigation in software-defined networking," *Expert Systems with Applications*, vol. 104, pp. 121 – 133, 2018.

[97] A. Lakhina, M. Crovella, and C. Diot, "Diagnosing network-wide traffic anomalies," in *Proceedings of the 2004 Conference on Applications, Technologies, Architectures,*

and Protocols for Computer Communications, SIGCOMM' 04, (New York, NY, USA), pp. 219–230, ACM, 2004.

[98] T. Huang, H. Sethu, and N. Kandasamy, "A New Approach to Dimensionality Reduction for Anomaly Detection in Data Traffic," *IEEE Transactions on Network and Service Management*, vol. 13, pp. 651–665, Sept 2016.

[99] M. V. de Assis, J. J. Rodrigues, and M. L. Proença, "A Seven-dimensional flow analysis to help autonomous network management," *Information Sciences*, vol. 278, pp. 900 – 913, 2014.

[100] J. Jiang and S. Papavassiliou, "Enhancing network traffic prediction and anomaly detection via statistical network traffic separation and combination strategies," *Computer Communications*, vol. 29, no. 10, pp. 1627 – 1638, 2006. Monitoring and Measurements of IP Networks.

[101] D. Jiang, Z. Xu, P. Zhang, and T. Zhu, "A transform domain-based anomaly detection approach to network-wide traffic," *Journal of Network and Computer Applications*, vol. 40, pp. 292 – 306, 2014.

[102] P. Barford, N. Duffield, A. Ron, and J. Sommers, "Network performance anomaly detection and localization," in *IEEE INFOCOM 2009*, pp. 1377–1385, April 2009.

[103] D. Brauckhoff, X. Dimitropoulos, A. Wagner, and K. Salamatian, "Anomaly extraction in backbone networks using association rules," *IEEE/ACM Transactions on Networking*, vol. 20, pp. 1788–1799, Dec 2012.

[104] M. H. Bhuyan, D. Bhattacharyya, and J. Kalita, "A multi-step outlier-based anomaly detection approach to network-wide traffic," *Information Sciences*, vol. 348, pp. 243 – 271, 2016.

[105] P. Bereziński, B. Jasiul, and M. Szpyrka, "An entropy-based network anomaly detection method," *Entropy*, vol. 17, no. 4, pp. 2367–2408, 2015.

[106] J. David and C. Thomas, "DDoS Attack Detection Using Fast Entropy Approach on Flow- Based Network Traffic," *Procedia Computer Science*, vol. 50, pp. 30 – 36, 2015. Big Data, Cloud and Computing Challenges.

[107] A. A. Amaral, L. de Souza Mendes, B. B. Zarpelão, and M. L. Proença, "Deep IP flow inspection to detect beyond network anomalies," *Computer Communications*, vol. 98, pp. 80 – 96, 2017.

[108] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in *2010 IEEE 35th Conference on Local Computer Networks (LCN)*, pp. 408–415, Oct 2010.

[109] S. M. Mousavi and M. St-Hilaire, "Early detection of DDoS attacks against SDN controllers," in *2015 International Conference on Computing, Networking and Communications (ICNC)*, pp. 77–81, Feb 2015.

[110] P. Xiao, W. Qu, H. Qi, and Z. Li, "Detecting DDoS attacks against data center with correlation analysis," *Computer Communications*, vol. 67, pp. 66 – 74, 2015.

[111] J. Francois and O. Festor, "Anomaly traceback using software defined networking," in *2014 IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 203–208, Dec 2014.

[112] Y. Wang, Y. Zhang, V. Singh, C. Lumezanu, and G. Jiang, "Netfuse: Short-circuiting traffic surges in the cloud," in *2013 IEEE International Conference on Communications (ICC)*, pp. 3514–3518, June 2013.

[113] K. y. Chen, A. R. Junuthula, I. K. Siddhrau, Y. Xu, and H. J. Chao, "SDNShield: Towards more comprehensive defense against DDoS attacks on SDN control plane," in *2016 IEEE Conference on Communications and Network Security (CNS)*, pp. 28–36, Oct 2016.

[114] R. Sahay, G. Blanc, Z. Zhang, and H. Debar, "ArOMA: An SDN based autonomic DDoS mitigation framework," *Computers & Security*, vol. 70, pp. 482 – 499, 2017.

[115] Y. Cho and Y. Kim, "Case study of an anomalous traffic detection on the aggregation points of enterprise network," in *13th International Conference on Advanced Communication Technology (ICACT2011)*, pp. 1245–1248, Feb 2011.

[116] D. Liu, C. H. Lung, N. Seddigh, and B. Nandy, "Network traffic anomaly detection using adaptive density-based fuzzy clustering," in *2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications*, pp. 823–830, Sept 2014.

[117] M. L. Proença, F. Sakuray, C. Coppelmans, M. Bottoli, A. M. Alberti, and L. de S. Mendes, "A Pratical Approach for Automatic Generation of Network Segment Traffic Baselines," *Journal of Communication and Information Systems*, vol. 20, no. 1, pp. 15–27, 2005.

[118] G. Fernandes, L. F. Carvalho, J. J. Rodrigues, and M. L. Proença, "Network anomaly detection using IP flows with Principal Component Analysis and Ant Colony

Optimization," *Journal of Network and Computer Applications*, vol. 64, pp. 1 – 11, 2016.

[119] M. L. Proença, F. Sakuray, C. Coppelmans, M. Bottoli, A. M. Alberti, and L. de S. Mendes, "Anomaly Detection Using Digital Signature of Network Segment Aiming to Help Network Management," *Journal of Communication and Information Systems*, vol. 23, no. 1, pp. 1–11, 2008.

[120] M. H. A. C. Adaniya, T. Abrão, and M. L. Proença, "Anomaly detection using metaheuristic firefly harmonic clustering," *Journal of Networks*, vol. 8, pp. 82–91, 2013.

[121] C. E. Shannon, "A Mathematical Theory of Communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[122] E. H. M. Pena, S. Barbon, J. J. P. C. Rodrigues, and M. L. Proença, "Anomaly detection using digital signature of network segment with adaptive arima model and paraconsistent logic," in *2014 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1–6, June 2014.

[123] A. H. Hamamoto, L. F. Carvalho, L. D. H. Sampaio, T. Abrão, and M. L. Proença, "Network Anomaly Detection System using Genetic Algorithm and Fuzzy Logic," *Expert Systems with Applications*, vol. 92, pp. 390 – 402, 2018.

[124] M. L. Proença, C. Coppelmans, M. Bottoli, A. Alberti, and L. S. Mendes, "The Hurst Parameter for Digital Signature of Network Segment," in *Telecommunications and Networking - ICT 2004* (J. N. de Souza, P. Dini, and P. Lorenz, eds.), (Berlin, Heidelberg), pp. 772–781, Springer Berlin Heidelberg, 2004.

[125] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on Neural Networks*, vol. 16, pp. 645–678, May 2005.

[126] S. Agrawal and J. Agrawal, "Survey on Anomaly Detection using Data Mining Techniques," *Procedia Computer Science*, vol. 60, pp. 708 – 713, 2015. Knowledge-Based and Intelligent Information & Engineering Systems 19th Annual Conference, KES-2015, Singapore, September 2015 Proceedings.

[127] M. F. Lima, L. D. H. Sampaio, B. B. Zarpelão, J. J. P. C. Rodrigues, T. Abrão, and M. L. P. Jr., "Networking Anomaly Detection Using DSNs and Particle Swarm Optimization with Re-Clustering," in *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, pp. 1–6, Dec 2010.

[128] S. Roshan, Y. Miche, A. Akusok, and A. Lendasse, "Adaptive and online network intrusion detection system using clustering and extreme learning machines," *Journal of the Franklin Institute*, vol. 355, no. 4, pp. 1752 – 1779, 2018. Special Issue on Recent advances in machine learning for signal analysis and processing.

[129] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Computational Intelligence Magazine*, vol. 1, pp. 28–39, Nov 2006.

[130] L. F. Carvalho, S. Barbon, L. de Souza Mendes, and M. L. Proença, "Unsupervised learning clustering and self-organized agents applied to help network management," *Expert Systems with Applications*, vol. 54, pp. 29 – 47, 2016.

[131] J. L. Deneubourg, S. Goss, N. Franks, A. Sendova-Franks, C. Detrain, and L. Chrétien, "The Dynamics of Collective Sorting Robot-like Ants and Ant-like Robots," in *Proceedings of the First International Conference on Simulation of Adaptive Behavior on From Animals to Animats*, (Cambridge, MA, USA), pp. 356–363, MIT Press, 1990.

[132] C. Kolias, G. Kambourakis, and M. Maragoudakis, "Swarm intelligence in intrusion detection: A survey," *Computers & Security*, vol. 30, no. 8, pp. 625 – 642, 2011.

[133] T. Stützle, M. López-Ibáñez, P. Pellegrini, M. Maur, M. Montes de Oca, M. Birattari, and M. Dorigo, *Parameter Adaptation in Ant Colony Optimization*, pp. 191–215. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[134] B. J. Zhao, "An ant colony clustering algorithm," in *2007 International Conference on Machine Learning and Cybernetics*, vol. 7, pp. 3933–3938, Aug 2007.

[135] H. Fu, "A novel clustering algorithm with ant colony optimization," in *2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application*, vol. 2, pp. 66–69, Dec 2008.

[136] P. Shelokar, V. Jayaraman, and B. Kulkarni, "An ant colony approach for clustering," *Analytica Chimica Acta*, vol. 509, no. 2, pp. 187 – 195, 2004.

[137] J. C. Dunn, "Well-separated clusters and optimal fuzzy partitions," *Journal of Cybernetics*, vol. 4, no. 1, pp. 95–104, 1974.

[138] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53 – 65, 1987.

[139] Y. Wang, "A multinomial logistic regression modeling approach for anomaly intrusion detection," *Computers & Security*, vol. 24, no. 8, pp. 662 – 674, 2005.

[140] I. Nunes, F. Schardong, and A. Schaeffer-Filho, "BDI2DoS: An application using collaborating BDI agents to combat DDoS attacks," *Journal of Network and Computer Applications*, vol. 84, pp. 14 – 24, 2017.

[141] R. Sahay, G. Blanc, Z. Zhang, K. Toumi, and H. Debar, "Adaptive Policy-driven Attack Mitigation in SDN," in *Proceedings of the 1st International Workshop on Security and Dependability of Multi-Domain Infrastructures*, XDOMO' 17, (New York, NY, USA), pp. 4:1–4:6, ACM, 2017.

[142] W. Han and C. Lei, "A survey on policy languages in network and security management," *Computer Networks*, vol. 56, no. 1, pp. 477 – 489, 2012.

[143] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861 – 874, 2006. ROC Analysis in Pattern Recognition.

[144] T. V. Phan, T. V. Toan, D. V. Tuyen, T. T. Huong, and N. H. Thanh, "OpenFlow-SIA: An optimized protection scheme for software-defined networks from flooding attacks," in *2016 IEEE Sixth International Conference on Communications and Electronics (ICCE)*, pp. 13–18, July 2016.

[145] B. Yuan, D. Zou, S. Yu, H. Jin, W. Qiang, and J. Shen, "Defending against Flow Table Overloading Attack in Software-Defined Networks," *IEEE Transactions on Services Computing*, pp. 1–1, 2017.

[146] J. Abellán, C. J. Mantas, and J. G. Castellano, "A Random Forest approach using imprecise probabilities," *Knowledge-Based Systems*, vol. 134, pp. 72 – 84, 2017.