A Computational Intelligence-Driven Reference Architecture for Anomaly Detection in Software-Defined Networking (SDN)

Rivaldo Fernandes[1], Bruno L. Dalmazo[2], André Riker[3], Geraldo Pereira Rocha Filho[4], Rodolfo Meneguette[5], Lourenço Pereira Júnior[6], Roger Immich[1]

[1] Universidade Federal do Rio Grande do Norte (UFRN)
[2]Universidade Federal do Rio Grande (FURG)
[3]Universidade Federal do Pará (UFPA)
[4]Universidade Estadual do Sudoeste da Bahia (UESB)
[5]Universidade de São Paulo (USP)
[6]Instituto Tecnológico de Aeronáutica (ITA)

rivaldofernandes@gmail.com, dalmazo@furg.br, ariker@ufpa.br
geraldoprfilho@gmail.com, meneguette@icmc.usp.br
lourenco.junior@gp.ita.br, roger@imd.ufrn.br

Abstract. ***Research Context:*** *The rise of Cloud, 5G, and IoT demands the continuous control of millions of networked devices. While Software-Defined Networking (SDN) simplifies network management, it struggles with security and the fine-grained analysis necessary for reliable operational problem detection.* ***Scientific Problem:*** *A critical architectural gap exists as there is no standardized, flexible framework to catalog diverse network scenarios and automatically associate them with the most effective Computational Intelligence (CI) techniques. This lack of a self-learning structure severely hinders the intelligence of the SDN control layer.* ***Proposed Solution:*** *We present a novel, technology-agnostic Reference Architecture for anomaly detection, applicable to both SDN and traditional networks. This highly resilient design leverages hexagonal microservices and rigorously adopts the TM Forum's Open Digital Architecture (ODA) model (TAM and eTOM) to achieve crucial industry standardization and interoperability.* ***Related IS Theory:*** *Work Systems Theory (proposing a systemic framework for an organizational process) and Institutional Theory, highlighted by the strategic adoption of ODA/TM Forum standards to ensure industry legitimacy.* ***Research Method:*** *An experimental methodology utilizing a Proof-of-Concept (PoC) prototype validated the architecture. We trained and assessed seven distinct Machine Learning (ML) algorithms against two different public datasets, proving the architecture's inherent versatility.* ***Summary of Results:*** *Results confirm the absolute need for a flexible architecture, as the solution model varies significantly across scenarios. The analysis showed that the choice of the most effective algorithm is strictly contingent upon the specific anomaly type and the chosen evaluation metric.* ***Contributions and Impact to IS Area:*** *The main contribution is a standardized Reference Architecture that standardizes the evaluation, promotion, and application of diverse computational intelligence techniques according to the network context. This provides a blueprint for next-generation, self-learning, and standards-compliant network operations, marking a significant step towards truly autonomous networks.*

1. Introdução

O paradigma de Redes Definidas por Software (SDN) simplifica a gestão de redes complexas ao separar as camadas de controle e dados [Neto et al. 2021]. Contudo, para lidar com ambientes cada vez mais dinâmicos, os controladores precisam evoluir de simples gestores de regras para mecanismos capazes de identificar problemas e adaptar-se automaticamente. Uma abordagem promissora é o uso de técnicas de autoaprendizado, que analisam padrões de tráfego e métricas de rede em tempo real [Hu et al. 2014]. Entretanto, a especificação da SDN não contempla esse tipo de inteligência, e ainda faltam ferramentas que permitam medições granulares do tráfego para apoiar decisões mais avançadas, como detecção de padrões, seleção de rotas e previsão de utilização [Queiroz et al. 2019].

Nesse contexto, ganha relevância o estudo de anomalias de rede, caracterizadas por desvios em relação ao comportamento esperado das métricas e operações [Silva et al. 2023]. Essas anomalias podem ter origem em falhas de dispositivos, sobrecarga, condições externas adversas ou ataques cibernéticos, como DoS e inserção de tráfego malicioso. Pesquisas voltadas a aumentar a robustez de transmissões sobre redes, mostram como padrões distintos de falhas emergem em diferentes ambientes de rede [Immich et al. 2015], motivando a criação de uma arquitetura padronizada e orientada por inteligência. O desafio está no fato de que tais ocorrências muitas vezes se assemelham ao tráfego legítimo, dificultando sua detecção. Quando não identificadas, podem causar perdas financeiras e danos à reputação das organizações, o que reforça a necessidade de mecanismos inteligentes de monitoramento e resposta em arquiteturas SDN.

Este trabalho identificou uma lacuna nas arquiteturas existentes para catalogação de tipos de rede, perfis de usuário e anomalias, bem como na associação desses elementos às técnicas de inteligência computacional (IC) mais eficazes para detecção em diferentes cenários. Para endereçar essa limitação, propomos uma arquitetura flexível, inicialmente voltada para redes SDN, que permite integrar e substituir modelos de IC adaptados a distintos contextos e classes de anomalias.

Partindo do pressuposto já consolidado de que a IC é eficaz na detecção de anomalias, o objetivo não é revalidar essa premissa, mas desenvolver uma arquitetura de referência que facilite o treinamento, avaliação e seleção dos métodos mais adequados a cada situação. Além de SDN, a proposta também se aplica a redes tradicionais, oferecendo uma solução versátil e abrangente para provedores de telecomunicações e demais setores que demandam monitoramento eficiente.

O principal objetivo deste trabalho é propor uma arquitetura de referência que apoie a implementação de sistemas baseados em inteligência computacional (IC) para detecção de anomalias, ampliando sua aplicabilidade em diferentes tipos de empresas. Trabalhos que endereçam degradações complexas em cenários altamente variáveis demonstram que a detecção eficiente de anomalias depende de abordagens sensíveis ao contexto [Immich et al. 2014], essa constatação evidencia a lacuna por arquiteturas padronizadas e interoperáveis. Sendo assim, esta proposta contempla uma arquitetura tecnológica agnóstica, capaz de avaliar e integrar diversas técnicas de IA, além de adotar padrões que aumentem a resiliência e escalabilidade. Também são introduzidas janelas deslizantes dinâmicas para separar treinamentos e análises em tempo de execução, bem como técnicas que melhorem a interpretabilidade das predições. Por fim, um protótipo

foi desenvolvido e avaliado como prova de conceito em cenários de SDN com diferentes tipos de tráfego.

A arquitetura proposta baseia-se em SDN e permite coletar métricas da rede, armazenar dados de utilização e identificar desvios em relação a padrões históricos. Esses desvios disparam processos de IC que não apenas detectam anomalias, mas também acionam fluxos automáticos de correção. Além disso, os dados armazenados servem para identificar tendências e padrões de comportamento, possibilitando que a SDN realize autoaprendizado, melhore continuamente sua eficiência e emita alarmes diante de situações críticas.

O restante deste trabalho é dividido da seguinte forma. A Seção 2 apresenta os trabalhos relacionados, seguida pela proposta da arquitetura de referência na Seção 3. A descrição da prova de conceito, bem como a avaliação dos experimentos e discussão dos resultados estão descritos na Seção 4. A Seção 5 apresenta a conclusão e trabalhos futuros.

2. Trabalhos relacionados

Diversos estudos têm explorado arquiteturas e algoritmos de aprendizado de máquina para detecção de anomalias em redes definidas por software (SDN) e ambientes relacionados. [Pan et al. 2022] propõem uma arquitetura para redes ópticas que coleta dados por agentes na camada de pacotes e utiliza inteligência computacional para apoiar o controlador SDN em ações de correção. De forma semelhante, [Zhao et al. 2018] exploram arquiteturas para SDON, integrando coleta de indicadores ópticos com modelos de aprendizado de máquina, reforçando a importância da análise em tempo real em camadas de rede de alta performance.

Existem propostas de arquitetura de cibersegurança para SDN [Oliveira et al. 2021] e para IoT, composta por módulos de detecção de anomalias, mitigação e validação de tráfego, destacando o uso de GNNs para detecção de ataques DoS com desempenho superior a SVM, DT e RF [Protogerou et al. 2022]. Estudos ampliam esse enfoque com um framework que combina SDN, NFV e aprendizado supervisionado, alcançando precisão de até 99,71% na detecção de anomalias em IoT [Bagaa et al. 2020]. Ambos os trabalhos mostram o papel do SDN como integrador de múltiplos módulos inteligentes para segurança.

[Qi et al. 2021] utilizam Group Anomaly Detection (GAD) com redes convolucionais gráficas para analisar séries espaço-temporais, obtendo melhor desempenho em comparação a LSTM e Autoencoder. De forma próxima, [Le et al. 2021] investigam RNNs e suas variantes (LSTM, BiLSTM e GRU) para previsão de matrizes de tráfego, destacando o GRU como mais eficiente. Essas abordagens evidenciam a relevância de modelos temporais e espaciais para prever e detectar padrões anômalos em SDN.

No campo de arquiteturas baseadas em aprendizado profundo, [Phan et al. 2020] propõem o DEEP GUARD, framework que aplica Double Deep Q-Network (DDQN) para melhorar a granularidade de fluxo em SDN e otimizar a detecção de ciberataques. Essa linha se relaciona com [Tsogbaatar et al. 2020], que estruturam um framework modular com coleta, detecção de tráfego malicioso e gerenciamento de regras em OpenFlow, ambos destacando o papel de arquiteturas reforçadas com IA em planos de dados SDN.

[Das et al. 2021] avançam na direção da Inteligência Artificial Explicável (XAI), propondo o CEAD, um pipeline que aumenta a interpretabilidade dos modelos de detecção em SDN. Esse esforço responde a limitações também apontadas por [Qi et al. 2021], que identificam os desafios da "caixa preta" nos modelos de IA aplicados a redes.

Outros estudos exploram alternativas a classificadores tradicionais. [Mathas et al. 2018] utilizam Latent Dirichlet Allocation (LDA) em Apache Spot para identificar padrões de tráfego e detectar ataques como Slowloris, diferenciando-se por empregar técnicas de NLP. [Nobakht et al. 2016], por sua vez, introduzem o framework IoT-IDM, que aplica regressão logística linear e não linear para detectar acessos não autorizados em IoT, alcançando até 98,53% de precisão.

Em termos de desafios, vários trabalhos convergem na identificação de problemas de escalabilidade, latência e interpretabilidade. Enquanto [Pan et al. 2022] alertam para riscos de envenenamento de dados em sistemas automatizados, [Qi et al. 2021] e [Das et al. 2021] enfatizam a necessidade de explicabilidade nos modelos. Já [Dinh and Park 2021] exploram cenários específicos como os ataques de Economic Denial of Sustainability (EDoS), destacando lacunas na proteção contra ameaças financeiras em ambientes de nuvem baseados em SDN.

Em síntese, os trabalhos analisados mostram que as arquiteturas para detecção de anomalias em SDN costumam incluir camadas para coleta de dados, análise com algoritmos de aprendizado de máquina e integração com o controlador SDN. No entanto, não há padrões consolidados para a organização interna dessas camadas. As contribuições variam desde frameworks de alta precisão para IoT e SDN ([Protogerou et al. 2022], [Bagaa et al. 2020], [Nobakht et al. 2016]) até arquiteturas distribuídas para resiliência ([Starke et al. 2018]), abordagens baseadas em aprendizado profundo ([Phan et al. 2020]) e mecanismos de interpretabilidade ([Das et al. 2021]). Em conjunto, eles apontam oportunidades para unificação de arquiteturas e avanços em modelos explicáveis, robustos e escaláveis para segurança em redes definidas por software.

A nossa proposta aproxima-se dos trabalhos existentes ao integrar aprendizado de máquina à detecção de anomalias em SDN/IoT [Pan et al. 2022, Zhao et al. 2018, Protogerou et al. 2022, Bagaa et al. 2020] e ao reconhecer a importância da resiliência e da explicabilidade [Das et al. 2021, Starke et al. 2018]. Contudo, diferencia-se ao apresentar uma Referência Arquitetural padronizada, baseada no TM Forum ODA, que é agnóstica a tecnologias e algoritmos. O uso de hexagonal microservices e a integração de teorias de Sistemas de Informação ampliam sua relevância, enquanto a validação com múltiplos algoritmos e datasets públicos comprova a versatilidade e a aplicabilidade da solução.

3. Arquitetura de referência para detecção de anomalias

Arquitetura de referência proposta não tem dependência de tecnologias, plataformas ou linguagem de programação. A sua principal premissa é a utilização de padrões de arquitetura agnósticos. Para os padrões técnicos de criação de aplicação, propõe-se utilizar arquitetura hexagonal com microsserviços e definição de domínios de aplicações e processos com TAM e eTOM do ODA, TMForum [ODA 2023], assim cada microsserviço será responsável por uma preocupação separada, uma área de negócio separada, minimizando

a sobreposição de responsabilidades tanto quanto possível.

A Figura 1 exibe a arquitetura geral proposta. Dentre os quais, grupos de componentes de arquitetura responsáveis por interagir com a rede (01 e 02), componente de ingestão e processamento de dados (04), módulos de aprendizagem de máquina (05), de remediação e alarmística (06, 07 08 e 09), uma base de dados compartilhada entre as instâncias dos controladores SDN e que também armazena um inventário de recursos em grafo (03). Todos os detalhes sobre estes componentes serão detalhados a seguir.

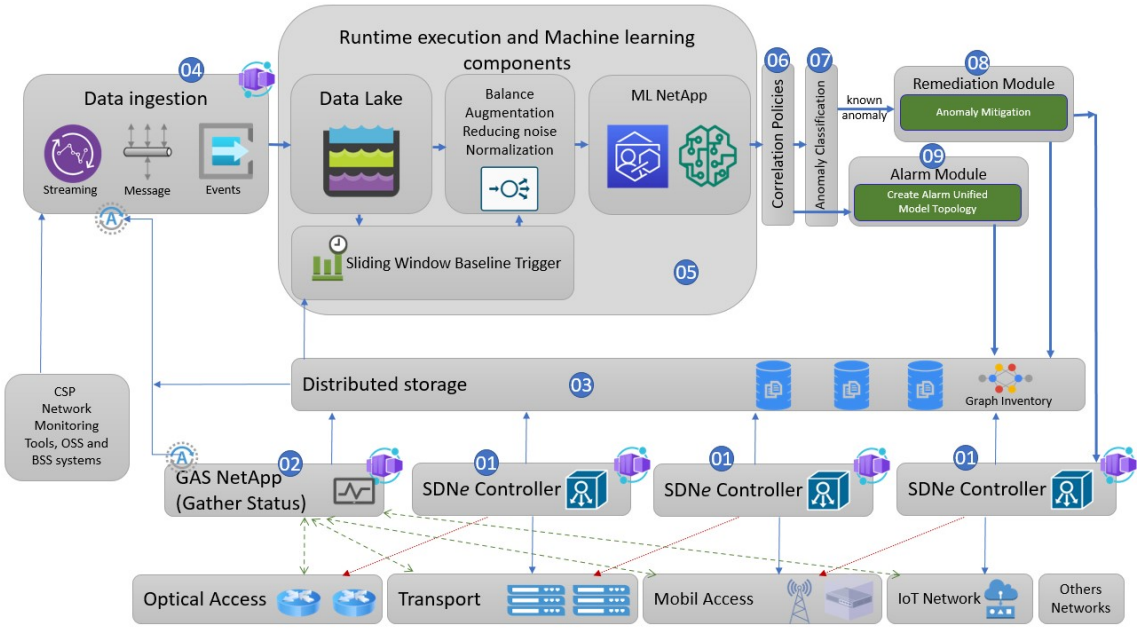

Figura 1. Componentes de arquitetura proposta

Embora este trabalho seja redigido em Português, os nomes dos componentes foram definidos em Inglês para garantir padronização com a literatura científica internacional e facilitar a compreensão por parte da comunidade acadêmica e profissional da área. Termos técnicos em Inglês são amplamente utilizados como convenções globais e, em muitos casos, não possuem tradução precisa ou consolidada para o Português. Além disso, a definição em Inglês facilita a relação direta com padrões já definidos em documentos técnicos, especificações e normas internacionais, preservando a clareza conceitual e assegurando maior aderência às práticas de pesquisa e desenvolvimento utilizadas mundialmente.

3.1. SDN Controller

O controlador SDN, componente 01, é responsável por aplicar as regras de roteamento do tráfego de dados a cada equipamento de rede utilizando protocolos SDN. Uma das características de um controlador SDN é centralizar a gestão da rede. Esta centralização leva a desafios como ter um único ponto de falha. Com a intenção de aliviar este desafio de vulnerabilidade, a arquitetura proposta suporta controladores distribuídos. Desta maneira, é possível definir diferentes *slices* de rede, que poderiam ser um conjunto de equipamentos, um tipo de rede específica, ou até mesmo uma ou mais camadas do modelo de Interconexão de Sistemas Abertos (*Open Systems Interconnection* - OSI) para um

conjunto de equipamentos. Cada *slice* poderá ser gerenciado por um controlador SDN distinto aumentando assim a tolerância a falhas em grande escala.

Por ser um microsserviço, os controladores poderão ser escalados horizontalmente executando novas instâncias do controlador. Além de ter flexibilidade de criar diferentes controladores com diferentes plataformas de programação, desta maneira permitindo utilizar a plataforma mais adequada para cada diferente tipo de protocolo que cada controlador necessite utilizar.

3.2. GAS NetApp

O componente GAS (*Gather Status*) NetApp (02) é composto por microsserviços responsáveis por coletar informações da rede, seja informações de performance, alarmes, contadores/indicadores ou para descobrir recursos físicos ou lógicos que serão reconciliados para o inventário de topologia e estado de cada dispositivo que estão no *Distributed Storage*. Além de ter a capacidade de enviar mensagens para o componente de *Data Ingestion* que será detalhado mais abaixo.

Neste componente e nos controladores SDN a escalabilidade de uma arquitetura de microsserviços tem um papel fundamental, pois à medida que a rede se expande, o controlador e o GAS precisam controlar e monitorar mais nós e quando as informações são trocadas com frequência, uma grande quantidade de fluxo de dados será gerada a cada segundo afetando o monitoramento em tempo real conforme aponta [Qi et al. 2021].

3.3. Distributed Storage

O *Distributed Storage* (03), é a base de dados distribuída que conterá toda a topologia física e lógica da rede, também terá o modelo de regras de roteamento para cada equipamento associado a seu *slice* e o respectivo controlador responsável por se comunicar com os equipamentos deste *slice* em questão.

Cada regra de roteamento terá um estado associado, assim em casos de falha na aplicação da regra o controlador responsável atualizará o estado de implantação da regra como falha, assim o controlador poderá tentar reexecutar esta regra posteriormente. E em caso de uma falha geral do controlador, as regras ainda não aplicadas estarão persistidas na base de dados e poderão ser aplicadas quando o controlador voltar a funcionar depois de uma falha geral.

3.4. CSP - Network Monitoring Tools, OSS and BSS systems

Por não ser um componente da arquitetura, o CSP *Network Monitoring Tools, OSS and BSS systems* não está enumerado, mas está presente no diagrama da arquitetura pois tem um papel importante para que os módulos de IC sejam capazes de correlacionar dados de negócio, como informações de um CRM ou qualquer outro sistema que contenha dados que possam ajudar os módulos de IC a classificar mais adequadamente. Assim, estes componentes são os que enviam informações de sistemas que não fazem parte dos domínios desta arquitetura.

Além de informações de ferramentas de monitoramento já existentes nas empresas será possível criar *pipelines* de ingestão de dados de Sistemas de Suporte a Operação (*Operations Support Systems* - OSS) e(ou) dos Sistemas de Suporte ao Negócio (*Business Support Systems* - BSS) como informações de topologias, serviços, plataformas de

serviços, perfis dos clientes que utilizam serviços e recursos, plataforma de reclamações dos clientes, sistema de gestão de manutenção na rede, ou qualquer outro sistema da CSP. Desta maneira os classificadores poderão buscar por padrões de informações que não fazem parte diretamente da rede e correlacionar com os KPIs do monitoramento da rede.

3.5. Data ingestion

O *Data Ingestion* (04) é o responsável por toda ingestão de dados. Poderá receber informações como *streaming*, mensagens, eventos, expor um *endpoint* HTTP e pode utilizar uma interface não canonizada com SID, ele também irá receber os dados enviados pelo GAS ou de qualquer outro sistema da CSP.

Este componente recebe o dado bruto, não canonizado para o padrão do SID e é responsável por transformar o dado para o modelo de dados único baseado no SID do TMForum. Além disso, caso receba um alarme crítico, este componente pode solicitar a execução de processos automatizados para detecção de anomalias invocando o componente *Sliding Windown Baseline Trigger* (05) ou o componente *Balance* (05).

3.6. Runtime execution and Machine learning components

O *Runtime execution and Machine learning components* (05) é composto pelos módulos que tem relação com o processo para detecção de anomalias em tempo de execução pelos módulos de IC. São eles Data Lake, Sliding Window Baseline Trigger, Balance, e ML NetApp.

O componente Data Lake é o responsável pelo armazenamento de todos os alarmes, indicadores, contadores, eventos, informações sobre tipo de rede, serviços, plataforma de serviços, perfil de cliente, recursos de rede, ou qualquer outra informação que se necessite correlacionar. Ele também servirá como repositório de dados para armazenar métricas que serão utilizadas para gerar *baselines* que por sua vez serão avaliados em tempo de execução para confirmar se algum limite de algum indicador chegou a algum nível crítico.

O componente *Sliding Window Baseline Trigger (SWBT)* é o responsável por definir o tamanho das janelas dinâmicas deslizantes que são utilizadas. Para esta finalidade ele filtra os dados utilizados no treinamento dos modelos de IC e também para analisar se os dados em tempo de execução ultrapassaram algum *baseline* definido. Além da definição das janelas deslizantes dinâmicas, o SWBT também é o responsável por disparar o processo de treinamento informando a janela deslizante dinâmica específica para coleta dos dados do *Data Lake* que serão utilizados como conjunto de dados de treinamento. Este disparo para o processo de treinamento começa com a execução do componente *Balance* que poderá balancear os dados referente à janela deslizante dinâmica em questão antes de executar o treinamento.

Estas janelas deslizantes têm um papel de muita relevância nesta arquitetura pois serão responsáveis por ajudar a definir o tamanho do conjunto de dados de treinamento e também para sugerir limites para cada tipo de indicador analisado na janela. Estes *threshold* serão utilizados como *baselines* para disparar eventos de análises de dados a serem processados pelos modelos já treinados com a intenção de identificar anomalias e possivelmente disparar um fluxo de remediação da anomalia ou gerar alarmes. É im-

portante destacar que esta capacidade não limita a execução dos modelos de IC a estes cenários de validação de limites.

O *Balance* é responsável pelo balanceamento de dados antes do envio para o processo de treinamento. Seu papel é balancear os dados no conjunto de treinamento para evitar que o modelo se ajuste às classes majoritárias. Quando há um desequilíbrio drástico entre as classes de dados, pode ocorrer uma degradação no desempenho da detecção de anomalias, por isso é importante balancear os dados [Das et al. 2021].

Para evitar esse desequilíbrio, pode utilizar métodos como o de *oversampling* aleatório, que consiste em aumentar o conjunto de dados com cópias das classes minoritárias para fornecer amostras mais representativas e proporcionalmente distribuídas. Dessa forma, as classes minoritárias e majoritárias possuem uma proporção equilibrada no conjunto de dados a ser processado. Se o volume de dados for alto pode ser executado *undersampling* aleatório com a mesma intenção de equilibrar as classes majoritárias e minoritárias do conjunto de dados e com isto reduzir o volume de dados a serem processados e o tempo de processamento. Além disso, este componente é responsável por reduzir o ruído de informações retirando da amostra eventos/alarmes duplicados que podem ser enviados pela rede referente ao mesmo problema.

Este componente também é responsável por implementar técnicas como o *k-Cross Validation*, que divide o conjunto de dados em *k* vezes, mantendo a quantidade de cada tipo de classe em cada pedaço proporcionalmente. Depois, realizando o treinamento com *k-1* partes dos dados e a parte restante utilizada para testes. Este processo é repetido até que todas as partes tenham sido usadas nove vezes para treinamento e uma como dados de teste. Diminuindo assim a variação no resultado.

E por fim o módulo *ML NetApp*, que é o responsável por executar os modelos promovidos por terem sido mais bem avaliados. Este componente receberá em tempos de execução dados do SWBT associados a uma janela com a intenção de detectar anomalias e posteriormente enviar eventos para correlacionar alarmes e posteriormente executar o fluxo de remediação automática de anomalias. Este componente está associado ao domínio de *Anomaly Management* da ODA do TMForum pois é o módulo responsável por identificar uma anomalia.

3.7. Correlation Policies

O componente *Correlation Policies* (06) é responsável pelas políticas de correlacionar as anomalias enviadas pelos modelos de IC em tempo de execução e correlacionar com os eventos, alarmes ou falhas já ativos, ou seja, que possam já estar associados a topologia dos recursos/equipamentos de rede em questão. Por exemplo, se a detecção da anomalia estiver associada para um nó da topologia de rede, por exemplo, uma porta física de um dispositivo de rede, em que o nó pai desta porta física (o próprio dispositivo) já tenha uma anomalia identificada anteriormente, este módulo poderá correlacionar a anomalia detectada ao evento da porta física a anomalia do seu nó pai (dispositivo) sem necessidade de disparar um processo de alarme ou fluxo de remediação duplicado.

3.8. Anomaly Classification

O componente é o *Anomaly Classification* (07), responsável por receber a anomalia identificada e classificar com algum tipo de anomalia catalogada anteriormente manualmente

e que pode estar associada a um fluxo de remediação automática da anomalia.

Caso exista algum fluxo de remediação associado ao tipo de anomalia identificado, o *Anomaly Classification* é responsável por disparar o evento necessário para iniciar o fluxo de remediação e também executar o processo que associa a anomalia a topologia de recursos de rede. Caso não exista um fluxo de remediação associado ao tipo de anomalia, o alarme é associado a topologia e com isto pode ser visualizado em uma interface de usuário pelos administradores de rede. Também é de responsabilidade deste componente enviar evento ou mensagem para um ou mais serviços externos, que podem por exemplo enviar um e-mail ou criar um *ticket* num sistema de gestão de falhas.

3.9. Remediation Module

O componente *Remediation Module* (08) é responsável por enviar as ações de reconfiguração de rotas necessárias aos controladores SDN, com a intenção de corrigir a anomalia identificada. Para isto, este componente utiliza a API *northbound* do controlador SDN *Principal*.

3.10. Alarm Module

O componente *Alarm Module* (09) é o responsável por criar um alarme e associá-lo a topologia de rede juntamente com a referência a anomalia identificada. Estes alarmes serão utilizados para identificar a localização do(s) dispositivo(s) afetado(s) na topologia de rede e desta maneira será possível fazer o acompanhamento do estado do processo de remediação automática e assim validar se o problema foi resolvido.

4. Resultados e avaliação dos experimentos

A proposta apresentada é ampla e abrangente, oferecendo uma arquitetura completa para detecção de anomalias em redes. No entanto, para a etapa de prova de conceito, foram implementados os seguintes componentes: Componente 01(*SDN Controller*), Componente 05 (*Data lake*, *Balance*, *SWBT*, *ML NetApp*), Componente 07 (*Anomaly Classification*), e Componente 08(*Remediation Module*). Essa escolha não comprometeu a avaliação, pois os módulos selecionados são representativos e suficientes para demonstrar a viabilidade da solução. Assim, mesmo sem a implementação integral da arquitetura, foi possível validar o funcionamento do modelo e comprovar seu potencial, sem prejuízo para a análise de aplicabilidade e eficiência da proposta.

Para realização das ações de IC foi utilizada a ferramenta de mineração de dados e de KDD chamada de *Waikato Environment for Knowledge Analysis* (WEKA). O WEKA permite comparar o desempenho de vários métodos simultaneamente [Devi et al. 2023]. Foi utilizado um servidor Linux Ubuntu com um processador 64-Bits Intel Core i5 i5-10500H CPU de 2.50GHz, com 16GB de memória RAM, placa de vídeo GEFORCE GTX de 4GB e SSD de 500GB.

Nesta análise, foram implantadas diferentes metodologias dos modelos de previsão no conjunto de dados para explorar a precisão de cada modelo em cada metodologia de IC. Se demonstra a avaliação dos diferentes algoritmos usando quatro tipos de métricas: porcentagem de falsos positivos (FP), e verdadeiros positivos (VP), *Precision* e Revocação (*Recall*). Além de detalhar o processo de *Stratified K-Fold Cross-Validation*.

Os modelos de IC foram treinados com 7 algoritmos distintos, dos quais quatro são modelos supervisionados e três modelos não-supervisionados. Os modelos supervisionados utilizados foram o *Random Forest*, *K-Nearest Neighbors* , *J48* e *AdaBoost*. Já os modelos não-supervisionados utilizados foram o *K-Means*, *X-Means* e *Canopy*.

Dois *dataset* públicos foram utilizados, o *IoT network intrusion dataset* [Ullah and Mahmoud 2020] e o *KDD99 dataset* [Kang et al. 2019]. Ambos foram armazenados no *data lake* e foram a fonte de dados para o treinamento. O primeiro é um conjunto de dados relacionado a rede IoT e o segundo para redes mais tradicionais que têm tipos de usuários e informações qualitativamente diferentes por se tratar de duas redes completamente diferentes. O *dataset IoT network intrusion* é composto por dados rotulados como um dos seguintes tipos de classes, onde cada classe de dados identifica um tipo de anomalia ou comportamento normal da rede. Ao total, foram utilizadas todas as 625.783 instâncias de informações. Para o *dataset* KDD foram processados os dados que foram identificados como um dos seguintes tipos de classe de dados, onde cada classe identifica um tipo de anomalia ou comportamento normal. que estão distribuídas em 125.973 instancias

Para esta prova de conceito foi definida a janela no SWBT com um valor fixo, pois a intenção foi utilizar 100% dos dados de cada *dataset*. Ao final, não foi necessário utilizar técnicas de *augmentation*, mas os dados foram divididos em 10 partes para utilização do *Stratified K-Fold Cross-Validation* que será explicado a seguir.

4.1. Resultados do processo de aprendizagem de máquina

Foram realizados experimentos com base em uma revisão cuidadosa da literatura. Essa abordagem permite uma avaliação inicial da capacidade da arquitetura de identificar anomalias, ainda que os algoritmos possam não estar otimizados para os conjuntos de dados específicos utilizados no estudo. Todos os indicadores detalhados foram avaliados utilizando *Stratified K-Fold Cross-Validation* com *k* igual a 10.

A Figura 2 apresenta o resultado da predição de cada algoritmo para cada conjunto de dados. Para o *dataset* de IoT o algoritmo de IC que teve a maior taxa que corretamente classificou as instâncias de dados foi o J48 com 76.65%. Enquanto para o dataset *KDD99*, o melhor algoritmo de IC foi o *Random Forest* com 99.82%, é importante destacar que o J48 e o *K-Nearest Neighbors* tiveram praticamente a mesma performance que o *Random Forest*.

É possível observar que os algoritmos têm resultados distintos conforme os conjuntos de dados. Por exemplo, o *AdaBoost* que variou de 83.20% no KDD para 43.31% no *dataset* de IoT, bem como o *Canopy* que variou sua média de 73.27% no KDD para 34.82% do IoT, em ambos os casos uma diferença de quase 40%. As diferenças encontradas na análise dos resultados corroboram a necessidade de ter um processo de avaliação dos modelos e uma arquitetura flexível, onde diferentes modelos podem ser utilizados para detectar diferentes cenários de anomalias.

4.2. Resultado do *dataset* de IoT

A Tabela 1 apresenta as taxas de classificação das anomalias baseadas no *Precision* de cada modelo, em verde estão os algoritmos com maior *Precision* para cada tipo de anomalia. Apesar do J48 ter se destacado com a melhor média geral para o *dataset* de IoT

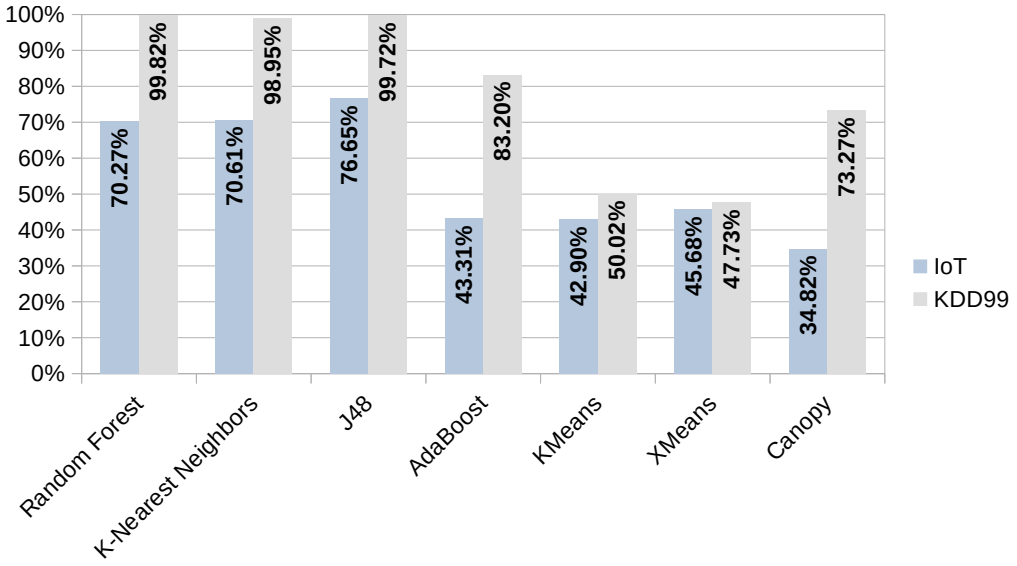

Figura 2. Detecção de anomalias por método de IC.

(0.764), ele não foi o melhor para identificar anomalias do tipo *Mirai Ackflooding* e *Mirai UDP Flooding*, especificamente para estes dois tipos de anomalia o *X-Means* foi o melhor, porém, com uma diferença pequena. Na média o *Random Forest* e o *K-Nearest Neighbors* tiveram praticamente a mesma performance que o J48 para quase todos os tipos de anomalias com exceção da *Mirai Ackflooding* e *Mirai HTTP Flooding*, por isto é possível afirmar que com exceção destes dois tipos de anomalia, o *Random Forest* é tão adequado quanto o J48 para identificar as anomalias segundo o *Precision* do modelo.

Tabela 1. Análise da *Precision* dos modelos para o dataset de IoT.

Classe / Anomalia	AdaBoost	Canopy	J48	k-Means	K-Nearest Neighbors	Random Forest	X-Means
DoS-Synflooding	0.001	1	1	0.668	0.999	1	0.548
Mirai-Ackflooding	0.001	0.216	0.297	0.217	0.092	0.102	0.311
Mirai-Hostbruteforceg	0.288	0.372	0.872	0.394	0.871	0.854	0.394
Mirai-HTTP-Flooding	0.001	0.209	0.319	0.2	0.156	0.114	0.214
Mirai-UDP-Flooding	0.674	0.441	0.785	0.792	0.741	0.737	0.804
MITM-ARP-Spoofing	0.001	0.001	0.967	0.144	0.945	0.913	0.117
Normal	0.001	0.001	0.941	0.177	0.936	0.937	0.315
Scan-Hostport	0.001	0.001	0.917	0.001	0.801	0.794	0.021
Scan-Port-OS	0.001	0.215	0.774	0.319	0.746	0.753	0.319
Média	**0.108**	**0.273**	**0.764**	**0.324**	**0.699**	**0.689**	**0.338**

A Tabela 2 exibe as taxas de *Recall* detalhadas para cada anomalia do *dataset* de IoT. Assim como na avaliação por *Precision* o J48 obteve o maior média com relação a todas as anomalias. Já com relação a cada anomalia o J48 só não teve a maior taxa para *Mirai-Ackflooding*, *Mirai HTTP Flooding* e *Mirai UDP Flooding* onde respectivamente o *Canopy*, *X-Means* e *AdaBoost* tiveram o maior *Recall*. Aqui foi possível comprovar que mudando o método de avaliação, diferentes algoritmos seguem obtendo diferentes resultados sobre para os mesmos tipos de anomalias com os mesmos dados. Isto reforça a necessidade de ter uma arquitetura onde possam adicionar tantos métodos de avaliação dos modelos quanto sejam necessários conforme proposto por esta pesquisa.

A Tabela 3 destaca os resultados para o tipo de avaliação FP e os algoritmos com a maior taxa estão destacados em vermelho. O *AdaBoost* teve o maior valor para a anomalia

Tabela 2. Análise do *Recall* dos modelos para o *dataset* de IoT.

Classe / Anomalia	AdaBoost	Canopy	J48	k-Means	K-Nearest Neighbors	Random Forest	X-Means
DoS-Synflooding	0	0.009	0.999	0.733	0.997	0.999	0.98
Mirai-Ackflooding	0	0.574	0.272	0.567	0.136	0.099	0.02
Mirai-Hostbruteforceg	0.926	0.194	0.942	0.252	0.923	0.921	0.252
Mirai-HTTP-Flooding	0	0.029	0.299	0.03	0.081	0.111	0.523
Mirai-UDP-Flooding	0.866	0.777	0.796	0.733	0.726	0.729	0.746
MITM-ARP-Spoofing	0	0	0.968	0.168	0.938	0.895	0.17
Normal	0	0	0.921	0.319	0.912	0.914	0.218
Scan-Hostport	0	0	0.438	0	0.438	0.43	0.005
Scan-Port-OS	0	0.339	0.894	0.283	0.87	0.858	0.283
Média	**0.199**	**0.214**	**0.725**	**0.343**	**0.669**	**0.662**	**0.355**

Mirai Hostbruteforceg, já o Canopy teve a maior taxa para *Mirai Ackflooding*, *Mirai UDP Flooding* e *Scan-Port-OS*. E o *X-Means* teve o maior índice para as anomalias *DoS Synflooding*, *Mirai HTTP Flooding*, *MITM ARP Spoofing* e *Scan Hostport*. Aqui é possível observar que os três algoritmos J48, *Random Forest* e *K-Nearest Neighbor* que tiveram as maiores taxas médias de *Recall* e *Precision*, tiveram a menor taxa, atestando sua eficiência em detectar as anomalias sem gerar muito FP.

Tabela 3. Análise dos FP dos modelos para o dataset de IoT.

Classe / Anomalia	AdaBoost	Canopy	J48	k-Means	K-Nearest Neighbors	Random Forest	X-Means
DoS-Synflooding	0	0	0	0.039	0	0	0.085
Mirai-Ackflooding	0	0.202	0.062	0.201	0.129	0.084	0.004
Mirai-Hostbruteforceg	0.551	0.079	0.033	0.095	0.033	0.038	0.093
Mirai-HTTP-Flooding	0	0.011	0.063	0.012	0.043	0.085	0.188
Mirai-UDP-Flooding	0.174	0.409	0.091	0.076	0.105	0.108	0.076
MITM-ARP-Spoofing	0	0	0.002	0.06	0.003	0.005	0.077
Normal	0	0	0.004	0.103	0.004	0.004	0.032
Scan-Hostport	0	0	0.001	0	0.004	0.004	0.008
Scan-Port-OS	0	0.115	0.024	0.057	0.028	0.026	0.056
Média	**0.081**	**0.091**	**0.031**	**0.071**	**0.039**	**0.039**	**0.069**

Os resultados da avaliação por VP estão detalhados na Tabela 4. Assim como para os métodos de avaliação *Precision* e *Recall* o J48 obteve a melhor média geral para VP e as exceções foram para *Mirai Ackflooding*, onde o *Canopy* teve o maior valor, e a *Mirai HTTP Flooding*, onde o *X-Means* teve a maior taxa e a anomalia *Mirai UDP Flooding* que o *AdaBoost* obteve 0.866.

Tabela 4. Análise dos VP dos modelos para o dataset de IoT.

Classe / Anomalia	AdaBoost	Canopy	J48	k-Means	K-Nearest Neighbors	Random Forest	X-Means
DoS-Synflooding	0	0.009	0.999	0.733	0.997	0.999	0.98
Mirai-Ackflooding	0	0.574	0.272	0.567	0.136	0.099	0.02
Mirai-Hostbruteforceg	0.926	0.194	0.942	0.252	0.923	0.921	0.252
Mirai-HTTP-Flooding	0	0.029	0.299	0.03	0.081	0.111	0.523
Mirai-UDP-Flooding	0.866	0.777	0.796	0.733	0.726	0.729	0.746
MITM-ARP-Spoofing	0	0	0.968	0.168	0.938	0.895	0.17
Normal	0	0	0.921	0.319	0.912	0.914	0.218
Scan-Hostport	0	0	0.438	0	0.438	0.43	0.005
Scan-Port-OS	0	0.339	0.894	0.283	0.87	0.858	0.283
Média	**0.199**	**0.214**	**0.725**	**0.343**	**0.669**	**0.662**	**0.355**

Com estes resultados, é possível observar que dentro de um mesmo conjunto de dados, ou perfil de dados que tem relação com o tipo de rede específico, diferentes algoritmos podem ser mais adequados para detectar diferentes anomalias. Ainda, que o método

de avaliação também pode mudar o resultado de qual algoritmo é mais adequado para cada anomalia. Desta maneira, a arquitetura de referência também deve permitir que se utilize um ou uma combinação de métodos de avaliação para identificar o algoritmo mais pertinente para cada cenário.

4.3. Resultado do *dataset* KDD

A Tabela 5 mostra os resultados do método de avaliação *Precision* para o *dataset* de KDD99. Neste cenário, a maior média geral ficou com o *Random Forest*, que obteve o maior valor para 19 tipos de anomalias. Destaca-se também que nenhum dos algoritmos foi adequado para detectar anomalias do tipo *perl*, *rootkit* e *spy*. Isto reforça a proposta deste trabalho em afirmar que é necessário ter uma arquitetura flexível correlacionada com um processo de treinamento e avaliação dos modelos de IC em que permita eleger um ou mais modelos de IC, onde cada modelo será mais adequado a identificar um ou mais tipos de anomalia.

Tabela 5. Análise do *Precision* dos modelos para o dataset de KDD.

Classe / Anomalia	AdaBoost	Canopy	J48	k-Means	K-Nearest Neighbors	Random Forest	X-Means
back	0.001	0	0.994	0.022	0.957	1	0.023
buffer_overflow	0.001	0.001	0.864	0.25	0.75	0.913	0.001
ftp_write	0.001	0.001	0	0.001	0.5	1	0.001
guess_passwd	0.001	0.017	0.962	0.018	0.923	1	0.016
imap	0.001	0.001	0.75	0.001	1	1	0.001
ipsweep	0.001	0.292	0.991	0.308	0.975	0.996	0.23
land	0.001	0.008	0.563	0	0.722	0.526	0.001
loadmodule	0.001	0.001	0	0.001	0.333	0.5	0.001
multihop	0.001	0.001	0.2	0.001	0.25	0.6	0.001
neptune	0.739	0.987	1	0.992	1	1	0.995
nmap	0.001	0.04	0.987	0.104	0.943	0.992	0.067
normal	0.906	0.948	0.997	0.968	0.993	0.998	0.973
perl	0.001	0.001	0	0.001	0	0.001	0.001
phf	0.001	0.001	0.6	0	1	1	0.001
pod	0.001	0.001	0.975	0.001	0.792	0.995	0.001
portsweep	0.001	0.281	0.994	0.209	0.995	0.998	0.421
rootkit	0.001	0.001	0.001	0.001	0	0	0.001
satan	0.001	0.264	0.993	0.141	0.983	0.998	0.251
smurf	0.001	0.339	0.998	0.21	0.88	0.998	0.141
spy	0.001	0.001	0.001	0.001	0.001	0.001	0.001
teardrop	0.001	0.074	0.999	0.062	0.899	1	0.001
warezclient	0.001	0.06	0.977	0.049	0.933	0.994	0.07
warezmaster	0.001	0.001	0.737	0.001	0	0.882	0.001
Média	**0.072**	**0.144**	**0.678**	**0.145**	**0.688**	**0.800**	**0.139**

A Tabela 6 destaca, em verde, as maiores taxas de *Recall*. A maior média geral ficou com o *Random Forest* (0.700), que teve a maior taxa em 14 das 23 classes. Já o *K-Nearest Neighbors* e o J48 tiveram maior valor em 8 tipos de classe e o *AdaBoost* para um tipo de anomalia. Mais uma vez nenhum dos algoritmos foi apto para detectar as anomalias do tipo *perl*, *rootkit* e *spy*. Esta situação pode estar relacionada a algumas limitações inerentes ao conjunto de dados utilizados, como número reduzido de instâncias em comparação a outras. Além disso, é possível que as características extraídas para representar esses ataques não sejam suficientemente discriminativas, fazendo com que os modelos não consigam distinguir seus padrões em relação ao tráfego normal ou a outros tipos de anomalias. Ainda é possível que essas anomalias tenham comportamentos mais sutis ou complexos, exigindo técnicas avançadas de detecção, como modelos especializados em classes raras ou algoritmos mais robustos a dados desbalanceados.

Tabela 6. Análise do *Recall* dos modelos para o dataset de KDD.

Classe / Anomalia	AdaBoost	Canopy	J48	k-Means	K-Nearest Neighbors	Random Forest	X-Means
back	0	0	0.993	0.383	0.977	0.997	0.349
buffer_overflow	0	0	0.633	0.067	0.7	0.7	0
ftp_write	0	0	0	0	0.375	0.25	0
guess_passwd	0	0.83	0.943	0.189	0.906	0.943	0.17
imap	0	0	0.273	0	0.727	0.727	0
ipsweep	0	0.723	0.996	0.848	0.98	0.996	0.871
land	0	0.059	0.5	0	0.722	0.556	0
loadmodule	0	0	0	0	0.222	0.111	0
multihop	0	0	0.143	0	0.286	0.429	0
neptune	1	0.832	1	0.762	1	1	0.696
nmap	0	0.001	0.981	0.142	0.948	0.983	0.177
normal	0.944	0.729	0.998	0.391	0.993	0.999	0.438
perl	0	0	0	0	0	0	0
phf	0	0	0.75	0	0.75	0.75	0
pod	0	0	0.98	0	0.796	0.945	0
portsweep	0	0.813	0.989	0.876	0.994	0.994	0.728
rootkit	0	0	0	0	0	0	0
satan	0	0.462	0.989	0.434	0.985	0.991	0.569
smurf	0	0.661	0.998	0.589	0.884	0.997	0.676
spy	0	0	0	0	0	0	0
teardrop	0	0.358	0.999	0.098	0.888	0.996	0
warezclient	0	0.23	0.982	0.187	0.931	0.982	0.389
warezmaster	0	0	0.7	0	0	0.75	0
Média	**0.085**	**0.248**	**0.646**	**0.216**	**0.655**	**0.700**	**0.220**

A Tabela 7 exibe os dados referente aos FP para o *dataset* de KDD. Na média, nenhum indicador ficou acima de 0.021, mas o *AdaBoost* obteve 0.112 para detectar a classe normal e foi ainda pior para a anomalia *neptuno* com 0.172. As anomalias *perl*, *rootkit* e *spy* que tiveram *Precision* e *Recall* zerados pelo menos não receberam nenhum FP. O *Random Forest* e o J48 obtiveram 0 na média, praticamente não tendo casos.

Tabela 7. Análise do *FP* dos modelos para o dataset de KDD.

Classe / Anomalia	AdaBoost	Canopy	J48	k-Means	K-Nearest Neighbors	Random Forest	X-Means
back	0	0.001	0	0.136	0	0	0.123
buffer_overflow	0	0	0	0	0	0	0
ftp_write	0	0	0	0	0	0	0
guess_passwd	0	0.02	0	0.005	0	0	0.005
imap	0	0	0	0	0	0	0
ipsweep	0	0.051	0	0.059	0.001	0	0.095
land	0	0.001	0	0.001	0	0	0
loadmodule	0	0	0	0	0	0	0
multihop	0	0	0	0	0	0	0
neptune	0.172	0.005	0	0.003	0	0	0.001
nmap	0	0	0	0.015	0.001	0	0.033
normal	0.112	0.046	0.003	0.016	0.008	0.003	0.017
perl	0	0	0	0	0	0	0
phf	0	0	0	0.003	0	0	0
pod	0	0	0	0	0	0	0
portsweep	0	0.05	0	0.083	0	0	0.026
rootkit	0	0	0	0	0	0	0
satan	0	0.038	0	0.083	0.001	0	0.056
smurf	0	0.028	0	0.05	0.003	0	0.098
spy	0	0	0	0	0	0	0
teardrop	0	0.032	0	0.011	0.001	0	0
warezclient	0	0.026	0	0.027	0	0	0.04
warezmaster	0	0	0	0	0	0	0
Média	**0.012**	**0.013**	**0.000**	**0.021**	**0.001**	**0.000**	**0.021**

A Tabela 8 destaca em verde os verdadeiros positivos para o *dataset* de KDD. Mais uma vez o *Random Forest* obteve a maior média (0.700) sendo o maior VP para 14 tipos de classes. O *K-Nearest Neighbors* e o J48 tiveram a maior taxa de VP em 8 classes, e o *AdaBoost* em uma. As mesmas anomalias, *perl*, *rootkit* e *spy*, que não foram adequadamente classificadas segundo o *Precision* e *Recall* também não foram classificadas segundo avaliação da taxa de VP.

Tabela 8. Análise do VP dos modelos para o dataset de KDD.

Classe / Anomalia	AdaBoost	Canopy	J48	k-Means	K-Nearest Neighbors	Random Forest	X-Means
back	0	0	0.993	0.383	0.977	0.997	0.349
buffer_overflow	0	0	0.633	0.067	0.7	0.7	0
ftp_write	0	0	0	0	0.375	0.25	0
guess_passwd	0	0.83	0.943	0.189	0.906	0.943	0.17
imap	0	0	0.273	0	0.727	0.727	0
ipsweep	0	0.723	0.996	0.848	0.98	0.996	0.871
land	0	0.059	0.5	0	0.722	0.556	0
loadmodule	0	0	0	0	0.222	0.111	0
multihop	0	0	0.143	0	0.286	0.429	0
neptune	1	0.832	1	0.762	1	1	0.696
nmap	0	0.001	0.981	0.142	0.948	0.983	0.177
normal	0.944	0.729	0.998	0.391	0.993	0.999	0.438
perl	0	0	0	0	0	0	0
phf	0	0	0.75	0	0.75	0.75	0
pod	0	0	0.98	0	0.796	0.945	0
portsweep	0	0.813	0.989	0.876	0.994	0.994	0.728
rootkit	0	0	0	0	0	0	0
satan	0	0.462	0.989	0.434	0.985	0.991	0.569
smurf	0	0.661	0.998	0.589	0.884	0.997	0.676
spy	0	0	0	0	0	0	0
teardrop	0	0.358	0.999	0.098	0.888	0.996	0
warezclient	0	0.23	0.982	0.187	0.931	0.982	0.389
warezmaster	0	0	0.7	0	0	0.75	0
Média	**0.085**	**0.248**	**0.646**	**0.216**	**0.655**	**0.700**	**0.220**

4.4. Emulação do ambiente SDN: Identificação de anomalias e remediação

Foi criado um cenário emulado com o Mininet[1] que possibilita criar uma rede virtual com *switches* que utilizam o protocolo *OpenFlow*. O controlador SDN utilizado foi o POX[2]. A rede foi criada com 5 *switches*, que correspondem ao IP de origem e destino do *dataset* de IoT. Os switches estão interligados de forma circular S1 to S5.

Na prova de conceito foi simulado o processo de detecção de uma anomalia DoS e a remediação automática através de um programa que envia os comandos necessários ao controlador SDN para isolar o *switch* origem do DoS. A Figura 3 detalha este processo. É importante notar que cada modelo de IC será especialista em detectar um ou mais tipos de anomalias, e com isto terá que ser ajustado para descartar qualquer outra anomalia que não seja sua especialidade.

O componente *Data Profile IoT* da figura é a representação de dados monitorados em tempo de execução e analisados pelo modelo de IC previamente treinado. Cada KPI monitorado é associado a um ou mais dispositivos, que por sua vez estão associados a uma topologia de rede juntamente com seu tipo e possivelmente dados como tipos de clientes que utilizam estes recursos. Associado a estes perfis de dados se tem os modelos de IC

[1] `http://mininet.org/`

[2] `https://github.com/noxrepo/pox`

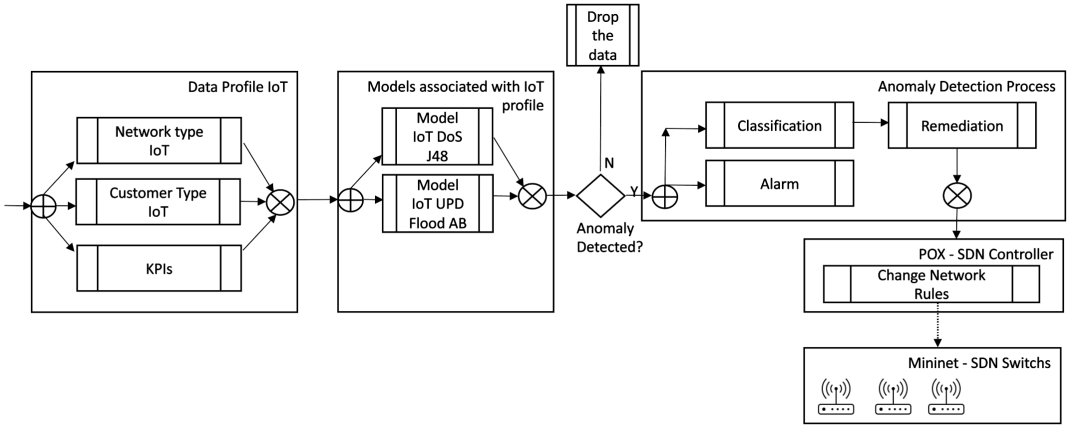

Figura 3. Detecção do DoS em tempo de execução

especializados em cada perfil. Ainda na mesma figura, o componente *Models associated with IoT profile* representa os dois modelos treinados pela prova de conceito. Estes dois modelos são responsáveis por analisar os dados de monitoramento enviados e informar se detectou ou não alguma anomalia. Se não é uma anomalia, o dado é descartado. Se é uma anomalia, inicia o processo de gerar alarme, classificação da anomalia e caso exista um fluxo de remediação automática, executa-o.

A remediação automática envia mensagens para o controlador SDN que então envia mensagens para os *switches* na rede Mininet, reconfigurando suas regras com intenção de mitigar a anomalia. O passo a passo deste processo é detalhado no diagrama de sequência da Figura 4. Neste diagrama de sequência se pode ver que os dados monitorados são enviados para os modelos de IC, o modelo *IoT DoS J48* detecta uma anomalia DoS e envia em paralelo a detecção da anomalia para o componente responsável por gerar o alarme e associá-lo ao(s) recurso(s) de rede e para o classificador de anomalias. Este último componente é o responsável por identificar se já existe um fluxo de remediação automática associado ao tipo da anomalia e disparar a execução do processo de remediação automática. Este por sua vez envia os comandos para o controlador SDN que faz as solicitações aos *switches* para reconfigurar suas regras de encaminhamento.

Com relação a reconfiguração da rede SDN, no total ocorreram 6 mudanças nas regras em 3 diferentes *switches*. A intenção destes comandos foi isolar o *switch* de rede S4, que é a porta de entrada do DoS, e criar um novo link entre os *switches* S3 e S5. A Tabela 9 mostra estas regras, em vermelho estão as regras que foram modificadas para descartar pacotes que tenham o S4 como origem ou destino. Já as regras em verdes foram adicionadas para criar o link de comunicação entre o S3 e S5. A Figura 5 mostra a comparação da topologia da rede Mininet de antes e depois da execução do fluxo de remediação. É importante frisar que este é um exemplo de utilização simples e que regras bem mais complexas poderiam ser implementadas pelos administradores de rede que tenham experiência em segurança.

4.5. Consideração sobre a prova de conceito

Para a prova de conceito da arquitetura foram selecionados dois *datasets* de tipos de redes distintas, um de IoT e outro de tradicional. Estes foram utilizados como dados de

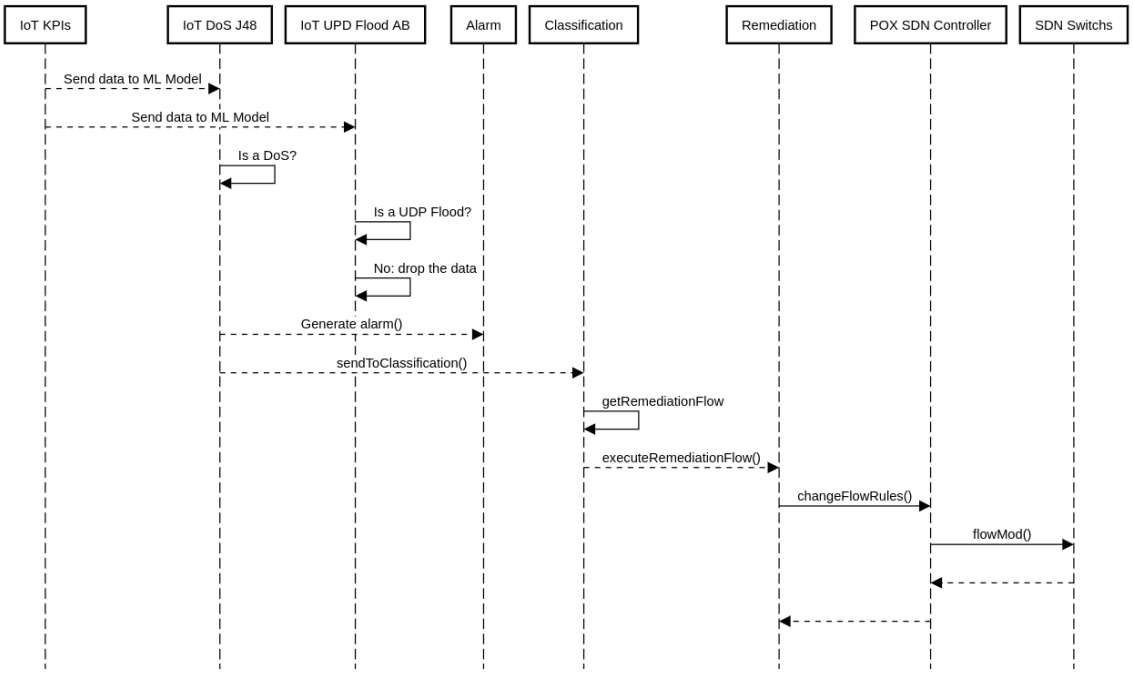

Figura 4. Diagrama de sequência da detecção de anomalia e remediação

Tabela 9. Reconfiguração de rotas nos switches SDN

Src IP	Dst IP	Protocol	Action
S1	S2	tpc	accept
S1	S3	tpc	accept
S3	S4	tpc	drop
S4	S5	tpc	drop
S2	S5	tpc	accept
S2	S1	tpc	accept
S3	S1	tpc	accept
S4	S3	tpc	drop
S5	S4	tpc	drop
S5	S2	tpc	accept
S3	S5	tpc	accept
S5	S3	tpc	accept

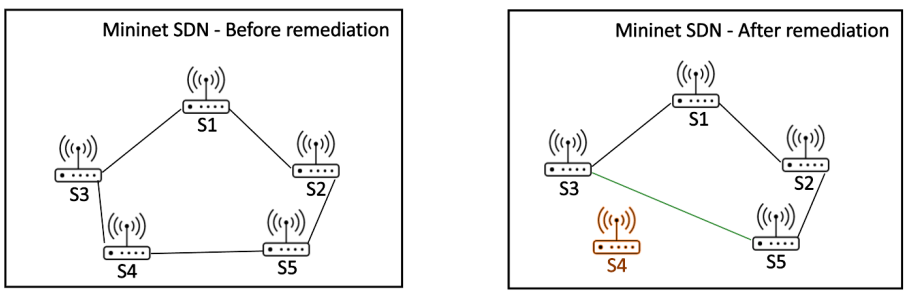

Figura 5. Diferença da topologia SDN do antes e depois da remediação do DoS

treinamento no Weka para 7 diferentes algoritmos de IC, 4 destes com treinamentos supervisionados e 3 com não-supervisionados. Depois foram avaliadas as predições de cada modelo utilizando *Stratified Cross-validation*. Distintos algoritmos tiveram o melhor resultado para cada uma das redes, além disso, dentro do resultado de cada rede distintos algoritmos obtiveram o melhor resultado para diferentes anomalias.

Este comportamento comprova a escolha de criar uma arquitetura de referência que seja flexível para adicionar e remover diferentes modelos de aprendizado de máquina para diferentes tipos de anomalias e redes. Foi utilizado o modelo treinado e especializado na rede IoT, que depois de detectar a anomalia, executou com sucesso o fluxo de remediação que alterou os *switches* da SDN remediando o DoS.

5. Conclusão

A utilização de redes definidas por *software* possibilita estratégia de gestão centralizada dos dispositivos de rede, esta característica habilita a criação de redes com uma quantidade bem maior de dispositivos do que as redes tradicionais. Esta grande quantidade de dispositivos gera muitos KPI de monitoramento, com o qual a IA tem se mostrado uma ferramenta adequada para analisar e classificar estas informações e com isto ajudar na gestão da rede implementando inteligência computacional na camada de aplicações do paradigma SDN e assim também apoiar na detecção de anomalias.

A arquitetura proposta neste estudo é abrangente e integradora, reunindo diversas estratégias antes isoladas em diferentes trabalhos. O artefato arquitetural é fundamentado na resiliência e escalabilidade de microsserviços hexagonais e adota padrões de interoperabilidade de mercado (TM Forum ODA), garantindo legitimidade institucional e alinhamento com a Teoria dos Sistemas de Trabalho. A principal contribuição da pesquisa reside não apenas na concepção desta arquitetura robusta, mas na validação experimental da sua premissa central: a necessidade de flexibilidade e agnosticismo algorítmico.

A Prova de Conceito (PoC), utilizando um subconjunto representativo dos componentes (SDN Controller, Runtime Execution/ML NetApp, Anomaly Classification e Remediation Module), demonstrou a viabilidade operacional do fluxo completo, desde a detecção de uma anomalia DoS até a reconfiguração automática e bem-sucedida das regras de encaminhamento no controlador SDN para isolar o ataque.

Além disso, a arquitetura proposta permitiu incorporar informações de negócios, dados de topologia de rede, tipos de usuários e sistemas de CRM, além de oferecer flexibilidade para adicionar ou remover diferentes modelos de detecção de anomalias, otimizar janelas de treinamento, aplicar padrões de mercado como SID/TMForum, seguir princípios de Separation of Concerns, utilizar SDN distribuído e microsserviços, aplicar arquitetura hexagonal e trazer interpretabilidade aos modelos.

Adicionalmente, a prova de conceito demonstrou que a arquitetura é funcional e capaz de suportar o treinamento e avaliação de múltiplos modelos de inteligência computacional, promovendo os melhores modelos e executando remediação automática de anomalias. Também foi validado que não existe um único algoritmo ideal para todos os cenários, reforçando a necessidade de estratégias adaptativas para diferentes tipos de anomalias. Com isso, a proposta contribui para a evolução da gestão automatizada de dispositivos de rede em larga escala, promovendo redes mais seguras, resilientes e confiáveis, e estabelece uma base sólida para estudos futuros e aprimoramentos na área.

Como futuros seria interessante criar provas de conceito para validar as escolhas arquiteturais realizadas por este trabalho, mas que ficaram de fora deste primeiro estudo, como a utilização de IC para definir o tamanho das janelas deslizantes, implementar o *Data ingestion* para fazer a transformação dos dados para o modelo único, implementar o XAI e validar sua eficácia, criar modelos de IC para gerar amostras adversariais que serão utilizadas para evoluir os modelos de detecção de anomalias e implementar um controlador SDN distribuído.

Agradecimentos

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

Referências

Bagaa, M., Taleb, T., Bernabe, J. B., and Skarmeta, A. (2020). A machine learning security framework for iot systems. *IEEE Access*, 8:114066–114077.

Das, T., Shukla, R., and Sengupta, S. (2021). The devil is in the details: Confident & explainable anomaly detector for software-defined networks. pages 1–5.

Devi, A. U., Vishal Kumar, R., and et al (2023). A data mining approach on the performance of machine learning methods for share price forecasting using the weka environment. In *2023 Fifth International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, pages 01–06.

Dinh, P. T. and Park, M. (2021). R-edos: Robust economic denial of sustainability detection in an sdn-based cloud through stochastic recurrent neural network. *IEEE Access*.

Hu, F., Hao, Q., and Bao, K. (2014). A survey on software-defined network and openflow: From concept to implementation. *IEEE Communications Surveys and Tutorials*, (4).

Immich, R., Borges, P., Cerqueira, E., and Curado, M. (2014). Adaptive motion-aware fec-based mechanism to ensure video transmission. In *IEEE Symposium on Computers and Communication (ISCC)*, pages 1–6.

Immich, R., Cerqueira, E., and Curado, M. (2015). Shielding video streaming against packet losses over vanets. *Wireless Networks*, pages 1–15.

Kang, H., Ahn, D. H., Lee, G. M., Yoo, J. D., Park, K. H., and Kim, H. K. (2019). Iot network intrusion dataset.

Le, D.-H., Tran, H.-A., Souihi, S., and Mellouk, A. (2021). An ai-based traffic matrix prediction solution for software-defined network. In *ICC 2021 - IEEE International Conference on Communications*, pages 1–6. IEEE.

Mathas, C. M., Segou, O. E., Xylouris, G., Christinakis, D., Kourtis, M.-A., Vassilakis, C., and Kourtis, A. (2018). Evaluation of apache spot's machine learning capabilities in an sdn/nfv enabled environment. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, New York, NY, USA.

Neto, E. P., Silva, F. S. D., Schneider, L. M., Neto, A. V., and Immich, R. (2021). Seamless mano of multi-vendor sdn controllers across federated multi-domains. *Computer Networks*, 186:107752.

Nobakht, M., Sivaraman, V., and Boreli, R. (2016). A host-based intrusion detection and mitigation framework for smart home iot using openflow. In *2016 11th International Conference on Availability, Reliability and Security (ARES)*, pages 147–156.

ODA, T. (2023). Open digital architecture (oda). Maio, 2023. Disponível em: `https://www.tmforum.org/oda/`. Acesso em Maio 02, 2023.

Oliveira, I., Neto, E., Immich, R., Fontes, R., Neto, A., Rodriguez, F., and Rothenberg, C. E. (2021). dh-aes-p4: On-premise encryption and in-band key-exchange in p4 fully programmable data planes. In *2021 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 148–153.

Pan, X., Yang, H., Xu, Z., and Zhu, Z. (2022). Adversarial analysis of ml-based anomaly detection in multi-layer network automation. In *Journal of Lightwave Technology*, volume 40, pages 4934–4944. IEEE.

Phan, T. V., Nguyen, T. G., Dao, N.-N., Huong, T. T., Thanh, N. H., and Bauschert, T. (2020). Deepguard: Efficient anomaly detection in sdn with fine-grained traffic flow monitoring. *IEEE Transactions on Network and Service Management*, 17(3).

Protogerou, A., Kopsacheilis, E. V., Mpatziakas, A., Papachristou, K., Theodorou, T. I., Papadopoulos, S., Drosou, A., and Tzovaras, D. (2022). Time series network data enabling distributed intelligence. a holistic iot security platform solution. In *Electronics (Switzerland)*, volume 11. MDPI.

Qi, Q., Shen, R., Wang, J., Sun, H., Guo, S., and Liao, J. (2021). Spatial-temporal learning-based artificial intelligence for it operations in the edge network. In *IEEE Network*, volume 35, pages 197–203. IEEE.

Queiroz, W., Capretz, M. A. M., and Dantas, M. (2019). An approach for sdn traffic monitoring based on big data techniques. volume 131, pages 28–39. Cited By :49.

Silva, F. S. D., Bessa, A., Silva, S., Ferino, S., Paiva, P., Medeiros, M., Silva, L., Neto, J., Costa, K., Santos, C., Maciel, D., Silva, L., Inoue, A., Immich, R., Aranha, E., Martins, A., Sousa, V., Kulesza, U., Fernandes, M., Salvador, M., Pupio, G., Fontes, R., and Neto, A. (2023). Proactive ML-assisted and quality-driven slice application service management to keep QoE in 5G mobile networks. In *2023 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*.

Starke, A., McNair, J., Trevizan, R., Bretas, A., Peeples, J., and Zare, A. (2018). Toward resilient smart grid communications using distributed sdn with ml-based anomaly detection. In Chowdhury, K. R., Di Felice, M., Matta, I., and Sheng, B., editors, *Wired/Wireless Internet Communications*, Cham. Springer International Publishing.

Tsogbaatar, E., Bhuyan, M. H., Taenaka, Y., Fall, D., Gonchigsumlaa, K., Elmroth, E., and Kadobayashi, Y. (2020). *SDN-Enabled IoT Anomaly Detection Using Ensemble Learning*. IFIP Advances in Information and Communication Technology. Springer.

Ullah, I. and Mahmoud, Q. H. (2020). A scheme for generating a dataset for anomalous activity detection in iot networks. In Goutte, C. and Zhu, X., editors, *Advances in Artificial Intelligence*, pages 508–520, Cham. Springer International Publishing.

Zhao, Y., Yan, B., Liu, D., He, Y., Wang, D., and Zhang, J. (2018). Soon: self-optimizing optical networks with machine learning. *Opt. Express*, 26(22):28713–28726.