Detecção de Ataques DDoS em Redes SDN Utilizando Aprendizado de Máquina: Uma Abordagem em Microsserviços

Victor Dias[1], Murilo Silva[1], Matheus Gomes[1], Lucas B. Oliveira[1], Diego Abreu[1], João Ferreira[3], Antônio Abelém[1]

[1] Programa de Pós-Graduação em Ciência da Computação (PPGCC)
Universidade Federal do Pará (UFPA) – Belém – PA – Brasil
[2]Instituto de Tecnologia (ITEC)
Faculdade de Engenharia da Computação e Telecomunicações - UFPA

murilosilva@itec.ufpa.br,matheus.cordovil@itec.ufpa.br,
lucas.borges@itec.ufpa.br,joao.bastos.junior@itec.ufpa.br

diego.abreu@itec.ufpa.br,victor.leite@ig.ufpa.br,abelem@ufpa.br

Abstract. *Even today, security remains a critical challenge in Software Defined Networks (SDN), including threats such as Distributed Denial of Service (DDoS) attacks. In this scenario, the use of machine learning holds promise for detecting and mitigating such attacks, where not only the model's performance must be considered, but also its impact on the network controller's performance. This study proposes a microservices-based approach, evaluating five machine learning models for detection. The results identified Random Forest as the most effective with an F1-Score of 98.65%. Furthermore, the microservices approach enabled the use of more complex models without compromising the performance of the SDN controller.*

Resumo. *Ainda hoje, a segurança é um desafio crítico nas Redes Definidas por Software (SDN), incluindo ameaças como ataques de negação de serviço distribuído (DDoS). Nesse cenário, o uso de aprendizado de máquina é promissor para detectar e mitigar tais ataques, onde devem ser considerados não apenas o desempenho do modelo, como também, o seu impacto no desempenho do controlador da rede. Este trabalho propõe uma abordagem baseada em microsserviços, avaliando cinco modelos de aprendizado de máquina para detecção. Os resultados identificaram o Random Forest como mais eficaz com F1-Score de 98.65%. Além disso, a abordagem de microsserviços permitiu a utilização de modelos mais complexos sem prejudicar o desempenho do controlador SDN.*

1. Introdução

O paradigma de Redes Definidas por Software (SDN) emergiu como uma abordagem promissora para melhorar a eficiência e flexibilidade no gerenciamento de redes de computadores. No entanto, com o aumento da complexidade e dependência da infraestrutura de SDN, a segurança tornou-se uma preocupação crítica. Diante desse cenário, os ataques de negação de serviço distribuídos (DDoS) representam uma das grandes ameaças e impactam diretamente na disponibilidade das aplicações.

Os ataques DDoS consistem em sobrecarregar uma rede, serviço ou aplicação por meio de um intenso fluxo de tráfego malicioso. O objetivo desses ataques é exceder a capacidade dos recursos disponíveis, resultando na interrupção ou indisponibilidade dos serviços legítimos. Essa prática prejudica a operação normal e causa uma série de prejuízos, afetando não apenas a disponibilidade dos serviços, mas também a confiança e a reputação das organizações envolvidas [Dayal et al. 2017].

A arquitetura de microsserviços proporciona a construção de sistemas distribuídos por meio de serviços independentes e altamente escaláveis [Francesco et al. 2017]. No contexto da detecção de ataques DDoS em redes SDN, essa abordagem se mostra especialmente relevante para lidar com a complexidade e a escalabilidade necessárias na detecção desses ataques. Ao adotar a arquitetura de microsserviços, é possível implementar estratégias de detecção de ataques DDoS de forma mais eficiente, mitigando as limitações associadas aos controladores SDN. Esses controladores, embora ofereçam benefícios em termos de centralização e gerenciamento da rede, muitas vezes apresentam desafios em relação à escalabilidade e complexidade quando se trata de detectar e responder a ataques de larga escala, como os DDoS. Através da distribuição de funcionalidades de detecção e resposta em microsserviços independentes, cada um dedicado a uma tarefa específica, é possível contornar as limitações dos controladores SDN, permitindo uma detecção mais ágil e uma resposta mais precisa a ataques DDoS.

Embora as soluções baseadas em aprendizado de máquina tenham sido propostas, muitas delas enfrentam desafios ao integrar-se e acompanhar a evolução de outras tecnologias presentes no paradigma SDN, a exemplo do P4, uma linguagem de programação de alto nível que permite a programabilidade do plano de dados. Além disso, mesmo com o surgimento de controladores como o ONOS, desenvolvido pela Open Network Foundation (ONF), que estão cada vez mais presentes em ambientes de nuvem, ainda existem obstáculos para aproveitar plenamente a modularidade e o desacoplamento oferecidos pelos orquestradores de contêineres, como o Kubernetes. Esses desafios podem limitar a adoção eficiente e integrada de diferentes tecnologias no contexto das redes SDN, prejudicando a capacidade de inovação e adaptação aos desafios em constante evolução.

Nesse contexto, este trabalho propõe explorar a adoção de microsserviços para realizar detecção de ataques DDoS utilizando o aprendizado de máquina em redes SDN. O objetivo é investigar e explorar a adoção dessas novas tecnologias, examinando de forma aprofundada uma solução base [Macías et al. 2021]. Além disso, o desempenho dos diferentes modelos de aprendizado de máquina e o impacto da utilização de microsserviços na distribuição de recursos computacionais durante o monitoramento do fluxo de pacotes na rede são analisados. Os resultados desse trabalho permitiram comprovar a viabilidade da adoção de uma arquitetura modular e distribuída, a de microsserviços, para detecção de ataques DDoS utilizando AM.

Esses resultados se enquadram tanto no âmbito do desempenho dos modelos, com os melhores modelos *Random Forest* (RF) e *Decision Tree* (DT) obtendo resultados de acurácia 98.50% e 98.06% e *F1-Score* de 98.65% e 98.25% respectivamente, quanto na distribuição na utilização de recursos durante o período de detecção, desse modo, permitindo que o controlador opere utilizando em média 0,25 CPU durante os ataques independentemente do modelo de AM utilizado, que por sua vez em modelos mais robustos chegam a consumir até 0,5 CPU de forma isolada.

Com isso, se espera que com os resultados que vão ser apresentados e discutidos durante todo o trabalho e os repositórios que permitam a sua reexecução, o trabalho em questão possa contribuir com o fortalecimento da segurança em redes SDN possibilitando a detecção eficiente de ataques DDoS. Além disso, a abordagem baseada em microsserviços oferece *insights* valiosos para o desenvolvimento de soluções de segurança mais resilientes e adaptáveis a diferentes tipos de ataques em redes definidas por software. Com isso, abre-se o caminho para avanços na proteção de redes SDN, tornando-as mais robustas e capazes de enfrentar os desafios de segurança emergentes.

O documento está organizado da seguinte forma: a Seção 2 aponta os trabalhos relacionados, abordando a ideia base adaptada por este trabalho. A Seção 3 mostra o desenvolvimento do microsserviço para a detecção dos ataques, bem como, a implementação deste no cenário de redes SDN. A Seção 4 apresenta os resultados obtidos e realiza a sua análise. Por fim, Seção 5 é destinada às conclusões e sugere alguns trabalhos futuros.

2. Trabalhos relacionados

A utilização de aprendizado de máquina para detecção de ataques DDoS vem se mostrando uma área de grande fomento, os mais diversos modelos de aprendizado de máquina como SVM (*Support Vector Machine*), DT, KNN (*K-Nearest Neighbors*), RF, XGBoost, entre outros, vêm sendo utilizados em pesquisas [Sudar et al. 2021]. Essa ampla adoção se caracteriza devido a bons resultados de desempenho. Uma grande preocupação está presente na definição das características ou *features* que melhor se encaixem na classificação desejada. Nesse sentido, trabalhos como de [Sharafaldin et al. 2018] agrupam as características de tráfego mais pertinentes em diferentes categorias como pacote, fluxo, conexão, intra-fluxo e múltiplos fluxos. Como visto, cada categoria representa uma abordagem diferente para extrair informações relevantes do tráfego de rede e permitir a classificação do comportamento do tráfego em legítimo ou DDoS.

Outro passo fundamental para o treinamento desses modelos está na obtenção do conjunto de dados ou *dataset* que são utilizados como base para representar a ocorrência real. Trabalhos como [Aslam et al. 2022] realizam a criação de um próprio *dataset* que tem a ocorrência de alguns tipos de ataque como *TCP SYN flood, UDP flood, ICMP flood e HTTP flood*. Ademais, o trabalho realizado por [Sharafaldin et al. 2018] culminou na criação de um *dataset*, chamado CIC-IDS2017, que apresenta uma série de ataques de rede que envolvem DDoS. Este trabalho culminou para a definição de uma série de características chave para treinamento de modelos para a detecção de ataques DDoS. A Tabela 1 mostra as principais características validadas pelo trabalho.

Tabela 1. Tabela de características (Adaptada [Macías et al. 2021]).

#	Características	Descrição	Nível / tipo
F1	FlowDuration	Duração do fluxo em microssegundos.	Fluxo/Bidirecional
F2	FlowIATStd	Desvio Padrão dos Tempos de Intervalo entre Pacotes do Mesmo Fluxo.	Intra-Fluxo/Bidirecional
F3	avgPacketSize	Média dos tamanhos das cargas úteis dos pacotes do mesmo fluxo.	Fluxo/Bidirecional
F4	BwdPktLenStd	Desvio padrão dos tamanhos das cargas úteis dos pacotes na direção BWD.	Intra-Fluxo/Unidirecional

Na literatura, artigos que investigam a distribuição da carga de trabalho para a detecção de ataques apresentam propostas que se baseiam tanto em sistemas reais [Aslam et al. 2022], quanto em abordagens virtualizadas utilizando virtualização das fun-

ções de rede (NFV) [Singh et al. 2020]. Contudo, até a presente data, não foram identificados estudos que explorem a integração da abordagem de microsserviços em um ambiente Kubernetes para otimizar a detecção de ataques DDoS no contexto do controlador ONOS.

Nesse sentindo, utilizou-se o trabalho intitulado ORACLE DDoS [Macías et al. 2021] como base para abordagem de microsserviços, que consiste na colaboração entre o plano de dados e o plano de controle SDN para realizar a detecção de ataques DDoS. Esse projeto foi escolhido pois utilizava os principais parâmetros de detecção de ataques, mostrados na Tabela 1, além de fazer a utilização do controlador ONOS, no qual, os autores já possuem familiaridade. Neste, foi percebida a oportunidade de melhorar alguns de seus aspectos com relação a acurácia dos modelos e a otimização dos recursos computacionais, através da utilização de microsserviços.

A Figura 1 ilustra a estrutura empregada no trabalho a qual, é composta por dois módulos, o primeiro sendo o plano de dados e o segundo o plano de controle. O primeiro utiliza o *switch* programável para retirar as características e estatísticas necessárias dos fluxos que está processando (1), após isso ele realiza o envio desses dados via P4Runtime para o controlador ONOS (2). O segundo módulo extrai essas informações de fluxo e utilizando uma aplicação que realiza o cálculo para transformá-las nas características que serão utilizadas para classificação, em seguida essa aplicação envia para o classificador via API REST (3) que por sua vez avisa o operador caso algum tráfego suspeito comece a ocorrer (4). Grande parte das ferramentas utilizadas pela solução estão presentes no repositório [SEBITAS 2021].

O artigo relata a melhor acurácia alcançada entre os modelos KNN e RF, sendo de 96% utilizando o KNN. Além do artigo, a tese de mestrado do autor [Gómez Macías 2020] mostra utilização de CPU durante o processo de detecção dos dois modelos utilizados (Figura 2) onde é visível o aumento considerável de consumo de CPU no modelo mais robusto RF.

3. Detecção de ataques DDoS em Redes Definidas por Software

Esta seção detalhará a proposta para detecção de ataques DDoS em redes SDN utilizando a abordagem de microsserviços desenvolvida neste trabalho. São descritas as contribuições realizadas em relação a solução utilizada como base, em seguida são detalhados os processos de recriação dos modelos e implementação da proposta no Kubernetes.

3.1. Arquitetura da proposta

A arquitetura proposta neste trabalho é apresentada na Figura 3, que utiliza como base a proposta descrita na Seção 2, no qual, os elementos do plano de controle foram implementados em um ambiente de nuvem, utilizando o Kubernetes e suas abstrações para proporcionar um melhor isolamento de recursos para cada microsserviço, onde o classificador é um dos diversos modelos de aprendizado de máquina.

Como visto na Seção 2, o projeto utiliza de forma distribuída diversas ferramentas que trabalham em sincronia para obter melhores resultados. Diante disso, algumas melhorias foram feitas em relação à arquitetura base, como trazer a estrutura para o Kubernetes, onde foram retirados alguns possíveis limitantes de desempenho.

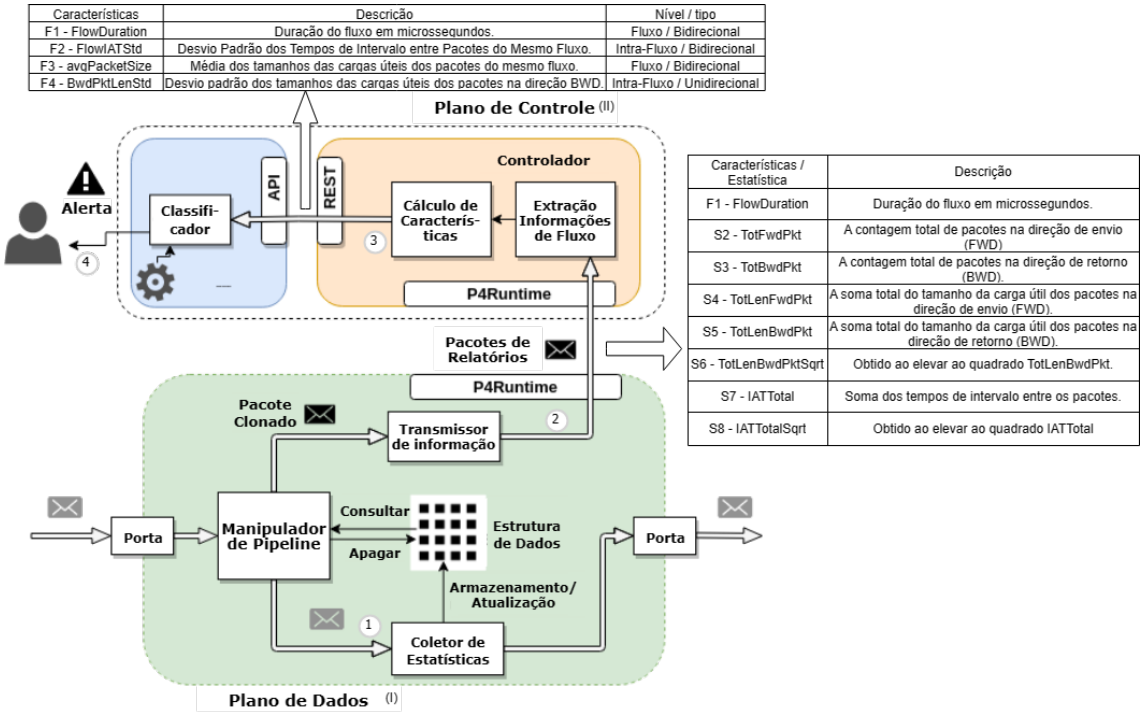
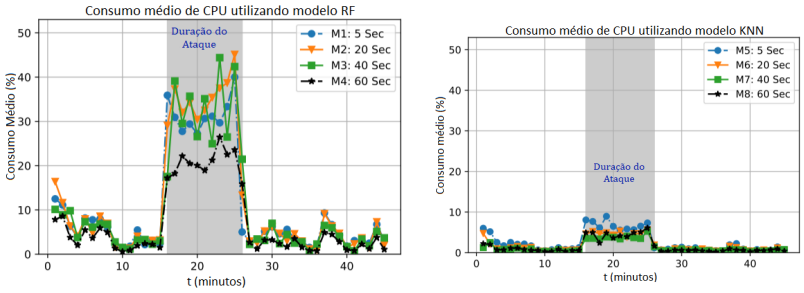


Figura 1. Funcionamento solução ORACLE DDoS (Adaptada [SEBITAS 2021]).



(a) Consumo de CPU gerado pelo modelo RF.

(b) Consumo de CPU gerado pelo modelo KNN.

Figura 2. Consumo de CPU ORACLE DDoS [Gómez Macías 2020].

A primeira contribuição com a adaptação realizada está presente na estrutura do plano de controle, que antes atuava em conjunto ao classificador de tráfego na mesma máquina, para receber as *features* geradas pela aplicação via *API REST* pela porta 8181 do ONOS, o que pode acarretar em problemas de alto consumo de recursos dependendo do tipo de modelo de aprendizado de máquina utilizado como é ilustrado na Figura 2(a). Desse modo, a adoção do padrão de microsserviços no Kubernetes, combinada com os conceitos da solução [Macías et al. 2021] podem trazer resultados valiosos no que tange à melhor distribuição de recursos, possibilitando a utilização de modelos mais robustos sem impacto direto no desempenho controlador.

Outra contribuição se deu na atualização de aspectos relacionados ao versionamento. Na solução base, fazia-se uso de bibliotecas específicas com versões antigas, o que gerava problemas tanto na implementação quanto na virtualização da solução. Na solução proposta basta que o usuário tenha instalado as versões mais recentes das biblio-

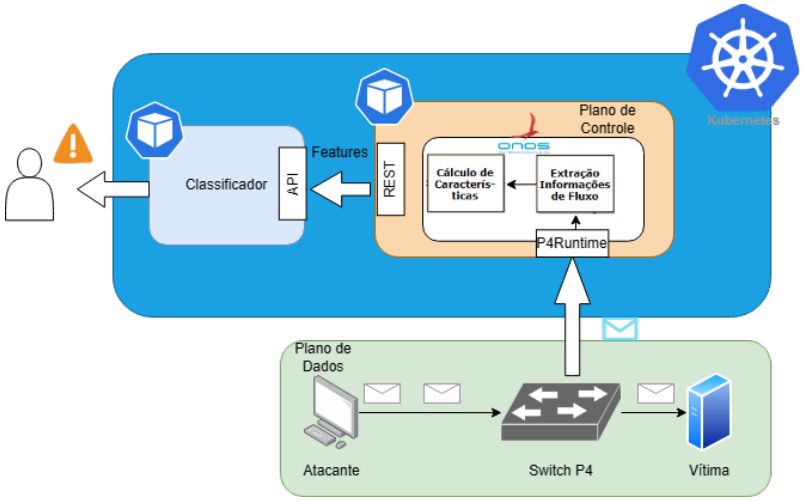


Figura 3. Proposta de detecção de DDoS utilizando microsserviços em SDN.

tecas do Python. No que se refere aos modelos de aprendizado de máquina, foi adicionada uma quantidade maior de modelos de classificação e seus desempenhos foram analisados não só em relação às métricas de performance como também à utilização de recursos do controlador. Na solução base, eram apenas analisados os modelos RF e KNN.

3.2. Treinamento de Classificadores: Do Pré-processamento ao Modelo Final

O conjunto de dados utilizado no treinamento dos modelos foi obtido a partir da execução de dois arquivos PCAP (*Packet Capture*), retirados do repositório [SEBITAS 2021] que por sua vez são uma adaptação do *dataset* CIC-IDS2017[Sharafaldin et al. 2018] que contém uma série de ataques, entre eles o ataque DDoS.

As características de tráfego brutas são a base para o cálculo das *features* que melhor representam o ataque, que são escolhidas com base em um processo de *feature selection* para definir quais dessas *features* são mais representativas e está esquematizada na Tabela 1. Para obter o *dataset* foram utilizados os passos presentes no GitHub do projeto ORACLE DDoS [SEBITAS 2021], os dois arquivos PCAP foram executados em um dos *hosts* da topologia para simular o tráfego real (1), para que o *switch* P4 retirasse as características necessárias e as enviasse para o controlador ONOS via P4Runtime (2), para que desse modo a aplicação presente no controlador após realizar os cálculos criasse os arquivos .csv com as características da tabela e algumas métricas de tempo que seriam filtradas posteriormente (3). As partes referentes à obtenção do *dataset* desta subseção assim como da próxima subseção, pré-processamento, são ilustradas na Figura 4.

3.2.1. Pré-processamento

Com relação ao pré-processamento do *dataset*, foi verificado se não existiam variáveis nulas para serem substituídas, e se continha *features* não numéricas para realizar a transformação de variáveis categóricas nominais para ordinais, sendo que para os dois casos a resposta foi negativa. Em seguida, foi feita a separação dos atributos previsores e atributos alvo, que são as características da tabela e as suas *labels*. Após isso, foi realizado o *split*

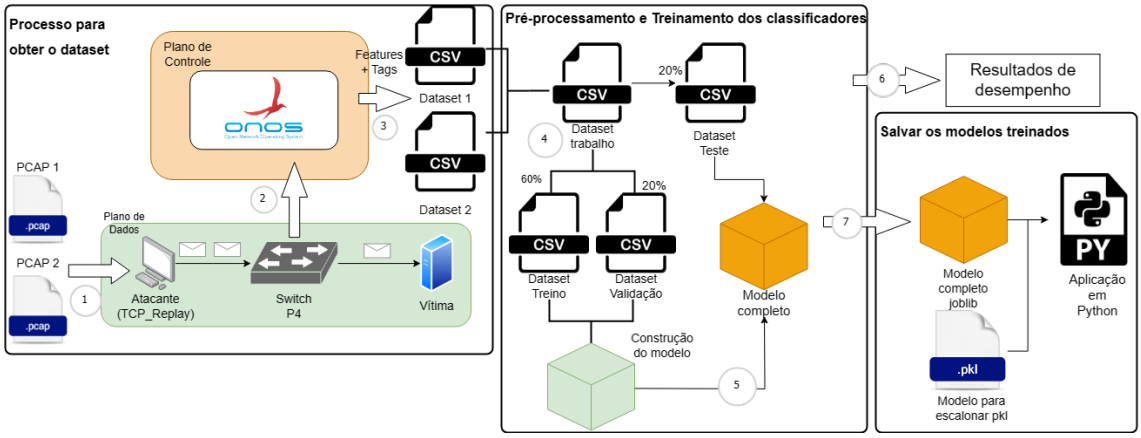


Figura 4. Esquematização do processo de obtenção do *dataset* e treinamento.

do *dataset*, onde o *dataset* foi dividido em conjunto de treinamento, validação e teste na proporção de 60% , 20% e 20%, respectivamente (4).

Após a separação dos atributos necessários, foi utilizada a biblioteca *sklearn.preprocessing* que fornece várias funções utilitárias de uso geral e classes transformadoras para converter recursos brutos em representações mais adequadas, para realizar o escalonamento dos atributos. Esse processo se dá pelo ajuste do escalonador aos dados de treinamento utilizando o método *fit transform*, permitindo que ele faça o cálculo de suas estatísticas, esse cálculo irá depender do método de escalonamento utilizado. Em seguida os dados de treinamento e validação são transformados utilizando os mesmo parâmetros de transformação, garantindo que todos os dados sejam tratados de forma consistente e estejam na mesma escala.

3.2.2. Treino dos Classificadores

A Figura 4 resume todo o processo de treinamento dos 5 modelos, ilustrando a utilização do conjunto de dados de treinamento e validação para ajustar os hiperparâmetros dos modelos (5). Após obter os melhores resultados com o *dataset* de validação, é feita a utilização do *dataset* de teste para retirada das métricas de desempenho dos modelos (6).

Após treinar todos os modelos, foi utilizada a biblioteca joblib para exportar os modelos pré-treinados para aplicação do classificador dos fluxos que seriam recebidos. O processo de normalização foi salvo em formato pkl, a fim de padronizar as entradas de fluxo para classificação, possibilitando assim, um melhor desempenho dos classificadores (7).

3.3. Implementação no Kubernetes

Para implementação do método de classificação de fluxos em um ambiente Kubernetes foi utilizado o Docker para criação de um contêiner com o classificador, o projeto ONOS *Classic* da ONF já foi portado para o ambiente Kubernetes e pode ser facilmente implementado utilizando o instalador de pacotes Helm. Desse modo, só foi necessário criar um *deployment* do classificador para receber os dados de fluxo. Todos os arquivos utilizados para a implementação no Kubernetes estão presentes no repositório [Dias 2023b]. Este

deployment, utiliza como base uma imagem Docker que foi criada utilizando um arquivo Dockerfile. Essa imagem serviu como base para criar e executar contêineres isolados, cada um contendo um classificador específico.

No Dockerfile, foi incluída uma série de instruções para criar o ambiente de execução necessário para os classificadores. Essas instruções podem incluir a instalação de dependências e a cópia dos arquivos do classificador para dentro do contêiner. Após isso, depois de criados os contêineres, eles foram enviados a um repositório no DockerHub [Dias 2023a].

Com a criação dos contêineres que seriam utilizados no trabalho e o *cluster* que serviria como infraestrutura, se deu início a implementação do classificador como um microsserviço do Kubernetes, essa implementação consistiu na criação de uma estrutura de controle chamada *deployment*. A criação do *deployment* se deu utilizando a linguagem YAML, padrão do orquestrador Kubernetes. Nela foram definidos os parâmetros cruciais para implementação do microsserviço como número de réplicas, imagem Docker que será utilizada para criação do contêiner e as portas que ele irá utilizar. Se procurou deixar as especificações mais genéricas possíveis para poder utilizar o manifesto como base para todos os classificadores.

Para implantação da proposta foi necessário realizar a comunicação entre o controlador ONOS e o microsserviço, para realizar essa comunicação no Kubernetes foi preciso realizar a criação de serviços, que são objetos que fornecem um abstração para expor aplicações que são executadas em um ou vários *pods*, permitindo a comunicação externa e entre eles independente de seus endereços de IP atuais, dada a natureza efêmera dos *pods* os serviços acabam oferecendo uma maneira consistente de acesso a eles. A criação dos serviços também se deu por manifestos YAML, que foram criados tanto para o controlador, permitindo a conexão com a porta 8181 responsável pela comunicação *API REST*, quanto para o classificador possibilitando receber os dados para classificação. Além de que para que essa nova abordagem funcionasse, foi necessário realizar a mudança na aplicação presente no ONOS, inserindo o endereço do serviço correspondente a classificação.

Após a implementação do ambiente de controle se seguiu os passos apresentados no repositório dos autores, onde primeiro a aplicação ONOS foi compilada, com as respectivas mudanças necessárias para se comunicar com o endereço do serviço do classificador e não mais utilizando *localhost*. Após o sucesso do processo de compilação da aplicação, ela foi instalada no controlador presente no *cluster*.

O último passo para realizar os testes se deu na implantação do plano de dados, utilizando o mininet para criar uma topologia constituida por 1 *switch* P4 e 2 *hosts* que seriam utilizados para execução do PCAP afim de simular o comportamento da rede. O cenário de testes obtido, assim como todo o processo descrito nesta subseção atual, é ilustrado na Figura 5.

4. Resultados

Nesta seção serão apresentados os resultados obtidos pelo trabalho e em seguida será feita a análise para avaliar a proposta e sua contribuição. Esses resultados foram divididos em duas partes, a Subseção 4.1 consiste na análise de desempenho dos modelos de classificadores e a Subseção 4.2 na análise da utilização de recursos computacionais desses classificadores.

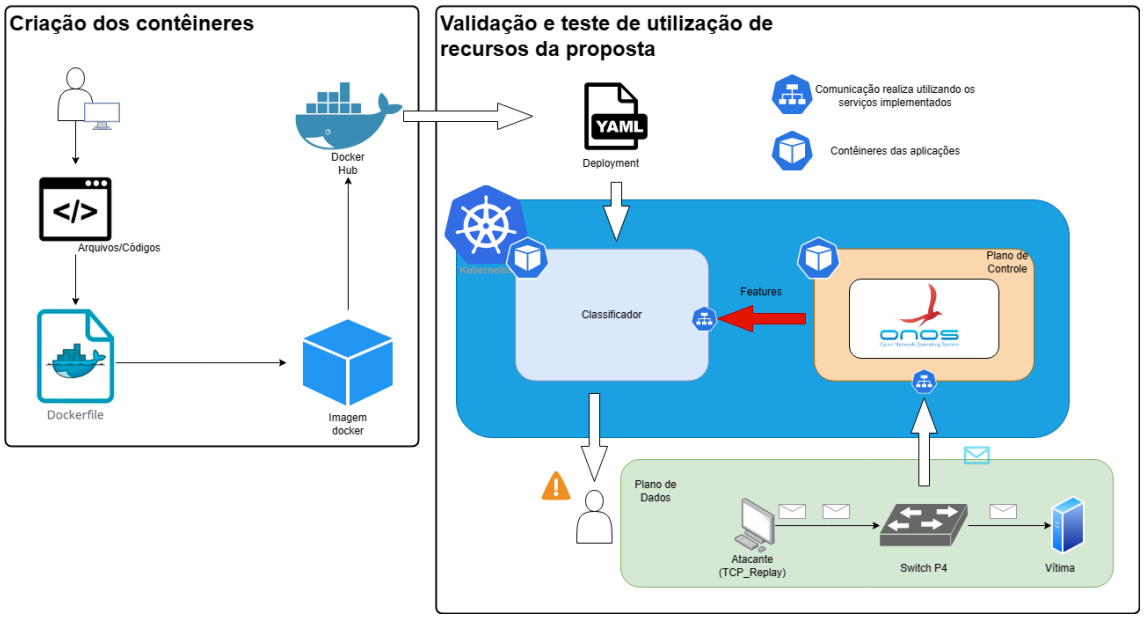


Figura 5. Ilustração da implementação no Kubernetes.

4.1. Métricas de desempenho dos modelos

Primeiro, o *dataset* de validação foi utilizado para se obter os melhores hiperparâmetros, após a definição dos melhores hiperparâmetros, os modelos foram submetidos ao *dataset* de teste, obtendo-se as métricas de desempenho de classificação mostradas na Tabela 2. Os resultados de acurácia durante o período de teste apresentaram 3 modelos com uma alta taxa de acerto geral, sendo KNN, DT e RF. Porém, outras métricas além da acurácia foram levadas em conta para definir qual o melhor modelo.

Tabela 2. Tabela contendo os Hiperparâmetros e as métricas de acurácia, precisão, recall e f1-score.

Modelos/Hiperparâmetros	Acurácia	Precisão	Recall	F1-score
SVM (C: 3000, gamma: 300, kernel: rbf)	94.93%	93.44%	93.44%	95.52%
RL (max iter: 15000, solver: lbfgs)	79.11%	76.30%	90.28%	82.70%
KNN (n° neighbors: 3, leaf size: 30, p: 2)	97.78%	97.74%	98.26%	98.00%
DT (criterion: entropy, splitter: best)	98.06	98.14%	98.37%	98.25%
RF (criterion: entropy, n° estimators: 100)	**98.50%**	**98.50%**	**98.85%**	**98.65%**

Como pode ser visto, os modelos KNN, DT e RF, apresentam os melhores resultados, com um foco no modelo RF que apresenta uma taxa de 98,65% de *F1-score* para casos de ataques DDoS, sendo essa métrica uma média harmônica da precisão e do *recall*, possibilitando uma análise tanto de falsos positivos quanto de falsos negativos durante a classificação de DDoS. Esses resultados permitiram definir que o melhor modelo durante o período de teste foi o RF.

Os resultados de desempenhos obtidos pelo trabalho possibilitaram a definição do modelo RF como a melhor escolha de classificador, diferindo do trabalho base que teve como melhor classificador o modelo KNN. Este resultado já era esperado devido o modelo RF ser mais robusto pois utiliza diversas DTs, se julga que essa diferença entre os dois

trabalhos ocorreu pelo ajuste dos hiperparâmetros, porém com a mudança do processo de treinamento é difícil realizar uma comparação direta entre esses desempenhos. Além disso, a utilização de modelos mais robustos é recomendável quando o conjunto de dados se torna mais próximo do real.

A utilização de modelos mais robustos como RF podem ajudar a lidar com algumas problemáticas:

- O *dataset* utilizado muitas vezes pode ser pouco representativo em relação a realidade, o que diminui a capacidade de generalização dos modelos, ou seja, diante da variedade de possibilidades de fluxos de dados reais o modelo não obteria resultados tão bons.
- Em conjuntos de dados mais complexos, classificadores mais simples como o KNN possivelmente não apresentariam um resultado tão satisfatório.

4.2. Métricas de utilização de recursos

Para obter os resultados referentes à utilização de recursos como CPU e memória dos modelos, foi utilizado o PCAP de teste via tcpreplay no *host* 1 (h1) da topologia, que consiste em uma carga de trabalho de 45 minutos, onde durante 10 minutos são executados ataques DDoS.

A Tabela 3 apresenta a média de consumo de CPU durante os testes realizados, é possível perceber um aumento de utilização de CPU no controlador ONOS durante a execução dos ataques DDoS, sendo esta a média para todas os modelos utilizados. O controlador ONOS segundo o manifesto de *deployment* definido pela ONF deve utilizar no máximo 0.5 CPU, sendo assim, é visível que esse limite é respeitado durante o período de ataque, onde o controlador utiliza em média 0.261 de CPU.

Tabela 3. Tabela com a média de consumo de CPU durante os testes de classificação.

Consumo de CPU	Período Sem Ataque (35 min)	Período Ataque DDoS (10 min)
ONOS	0.13 CPU	0.261 CPU
SVM	0.088 CPU	0.415 CPU
RL	0.006 CPU	0.0475 CPU
KNN	0.014 CPU	0.130 CPU
DT	0.008 CPU	0.049 CPU
RF	0.10 CPU	0.46 CPU

A Tabela 3 apresenta também o comportamento dos serviços de classificação, nas quais se tem um aumento considerável na utilização durante os 10 minutos de ataques que ocorrem. Os modelos KNN, RL (Regressão logística) e DT apresentaram um impacto de desempenho baixo, pois são classificadores relativamente mais simples e têm menos complexidade computacional em comparação com outros modelos, como também, não precisam realizar cálculos complexos ou processar grandes quantidades de dados. Porém, os modelos SVM e DT tiveram um consumo por vezes até maior que o controlador, devido sua complexidade a exemplo do RF que utiliza uma série de DTs para realizar a sua predição.

Essa grande diferença de utilização de recursos entre os classificadores com seus desempenhos relativamente próximos, como pode ser observado nos DT e RF podem acarretar na adoção de modelos mais simples em detrimento dos mais robustos, pois, métricas de desempenho tão próximas podem justificar a adoção de um classificador que utilize menos recursos. Embora modelos mais simples utilizem menos recursos, isso pode ser acompanhado por limitações em termos de desempenho quando lidamos com problemas mais complexos. O alto consumo de modelos como RF está diretamente ligado com a sua capacidade de lidar com esses problemas e alcançar um bom desempenho.

Logo, a abordagem de microsserviços se mostrou eficaz em realizar a divisão de recursos computacionais entre o controlador SDN e um classificador robusto como o RF, que se estivessem trabalhando no mesmo ambiente haveria um consumo maior de CPU afetando diretamente o controlador. Os resultados obtidos são importantes à medida que, com a evolução do conjunto de dados para mais próximos de casos reais, é possível utilizar classificadores mais complexos sem gerar impacto direto no plano de controle.

5. Conclusões e Trabalhos Futuros

A segurança da internet é um tema extremamente importante, pois ela é parte fundamental do nosso cotidiano, e ataques, como os de DDoS, que vem se intensificando, tanto em arquiteturas tradicionais quanto em SDN, são um problema crucial, pois afetam diretamente o seu desempenho e disponibilidade. Devido a esses fatores, em SDN diversos estudos foram e estão sendo realizados. Entre os mais promissores estão os que utilizam AM, pois é uma tecnologia que permite a detecção mais precisa de padrões maliciosos. Controladores SDN, como o ONOS, evoluíram com o tempo e atualmente estão inseridos em ambientes de nuvem e 5G, oferecendo suporte a tecnologias como P4 que possibilita uma melhor programabilidade no plano de dados.

Nesse sentido, novos padrões que contemplem esses avanços estão surgindo. A partir desse trabalho, foi possível adaptar uma solução para o contexto de microsserviços na nuvem e desse modo tirar proveito de suas vantagens, permitindo um melhor isolamento e consequentemente, melhor desempenho das aplicações que são utilizadas. Com os resultados mostrados por esse trabalho, foi possível perceber que a abordagem de microsserviços possibilita a utilização de métodos de classificação mais robustos sem gerar um impacto direto no funcionamento de um controlador SDN, o que é imprescindível durante um ataque a nível de DDoS. Desse modo, almejasse que além das ideias que este artigo propõe e os repositórios cedidos novos estudos nessa área possam ser realizados afim de contribuir com o avanço dos estudos na área de segurança de redes SDN em nuvem.

Com os resultados obtidos e o *framework* acessível no GitHub do autor, espera-se que esse trabalho possa servir como base para novos estudos. Como trabalhos futuros, estima-se a implementação de outros modelos de classificação como *XGBoost* e *LightGBM*, que atualmente vem apresentando resultados promissores na área de ML, bem como, outros mais robustos utilizando Redes Neurais Profundas. Além disso, pretende-se adaptar a solução para outros tipos de ataques utilizando outros *datasets* assim como, realizar uma análise comparativa com outras abordagens como NFVs em relação ao tempo de resposta e desempenho de detecção.

Referências

Aslam, N., Srivastava, S., and Gore, M. (2022). Onos flood defender: An intelligent approach to mitigate ddos attack in sdn. *Transactions on Emerging Telecommunications Technologies*, 33.

Dayal, N., Maity, P., Srivastava, S., and Khondoker, R. (2017). Research trends in security and ddos in sdn. *Security and Communication Networks*, 9.

Dias, V. (2023a). Repositório Docker Victor GERCOM. `https://hub.docker.com/u/victor98dl`.

Dias, V. (2023b). Repositório Victor GERCOM. `https://github.com/victor98dl/Tcc_victor.git`.

Francesco, P. D., Malavolta, I., and Lago, P. (2017). Research on architecting microservices: Trends, focus, and potential for industrial adoption. In *2017 IEEE International Conference on Software Architecture (ICSA)*, pages 21–30.

Gómez Macías, S. (2020). Sistema de detección de ataques de ddos basado en modelos de aprendizaje de máquina para la arquitectura sdn. Master's thesis. Disponível em: https://hdl.handle.net/10495/17420.

Macías, S. G., Gaspary, L. P., and Botero, J. F. (2021). Oracle: An architecture for collaboration of data and control planes to detect ddos attacks. In *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 962–967. IEEE.

SEBITAS (2021). ORACLE_ddos. `https://github.com/sebitas0623/ORACLE_ddos.git`. Acesso em: 01/04/2023.

Sharafaldin, I., Habibi Lashkari, A., and Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. *Proceedings of the 4th International Conference on Information Systems Security and Privacy*.

Singh, A. K., Jaiswal, R. K., Abdukodir, K., and Muthanna, A. (2020). Ardefense: Ddos detection and prevention using nfv and sdn. In *2020 12th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, pages 236–241.

Sudar, K., Beulah, M., Deepalakshmi, P., Nagaraj, P., and Chinnasamy, P. (2021). Detection of distributed denial of service attacks in sdn using machine learning techniques. In *2021 International Conference on Computer Communication and Informatics (ICCCI)*, pages 1–5.