Federated Learning Based DDoS Attacks Detection in Large Scale Software-Defined Network

Yannis Steve Nsuloun Fotse, Vianney Kengne Tchendji, and Mthulisi Velempini, *Senior Member, IEEE*

***Abstract*—Software-Defined Networking (SDN) is an innovative concept that segments the network into three planes: a control plane comprising of one or multiple controllers; a data plane responsible for data transmission; and an application plane which enables the reconfiguration of network functionalities. Nevertheless, this approach has exposed the controller as a prime target for malicious elements to attack it, such as Distributed Denial of Service (DDoS) attacks. Current DDoS defense schemes often increased the controller load and resource consumption. These schemes are typically tailored for single-controller architectures, a significant limitation when considering the scalability requirements of large-scale SDN. To address these limitations, we introduce an efficient Federated Learning approach, named "FedLAD," designed to counter DDoS attacks in SDN-based large-scale networks, particularly in multi-controller architectures. Federated learning is a decentralized approach to machine learning where models are trained across multiple devices as controllers store local data samples, without exchanging them. The evaluation of the proposed scheme's performance, using InSDN, CICDDoS2019, and CICDoS2017 datasets, shows an accuracy exceeding 98%, a significant improvement compared to related works. Furthermore, the evaluation of the FedLAD protocol with real-time traffic in an SDN context demonstrates its ability to detect DDoS attacks with high accuracy and minimal resource consumption. To the best of our knowledge, this work introduces a new technique in applying FL for DDoS attack detection in large-scale SDN.**

***Index Terms*—Federated learning, distributed denial of service, machine learning, software defined network, virtual network.**

I. Introduction

SOFTWARE-DEFINED networking (SDN) is an emerging networking model which separates the control decisions from the forwarding hardware [1] as shown in Fig. 1. In SDN, the networking devices act as packet forwarding devices, while

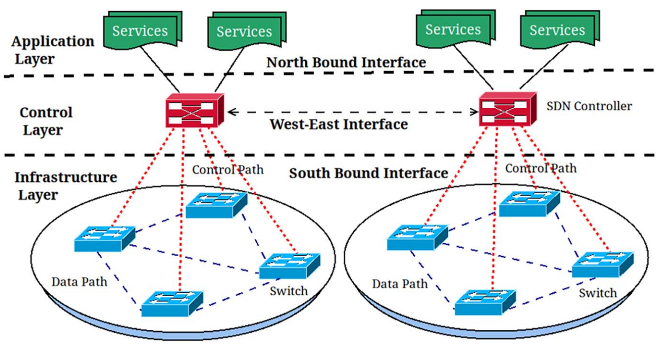


Fig. 1. SDN architecture.

the controller has the intelligence and control logic [1]. The controller is responsible for setting up and managing the flow tables on the switches, including adding, modifying, and removing flow entries. The interaction between the SDN controllers and network switches is shown in Fig. 2, where a new packet forwarding rule is introduced in the switch. Large-Scale SDN involves applying SDN principles to efficiently manage and control extensive network infrastructures. In large-scale SDNs, these principles are implemented across various data centers, campuses, or global-scale deployments, enabling enhanced scalability, flexibility, and agility in managing complex and geographically dispersed networks. SDN offers many advantages over traditional approaches. Firstly, it allows for easy integration of new concepts into the network through software applications, providing more flexibility and convenience compared to using fixed instructions on physical network devices [2]. Secondly, SDN enables a unified strategy for network configuration, which enables network administrators to configure the entire network remotely from a central point. The administrators do not have to configure each device when making network changes. Instead, decisions relating to traffic forwarding can be made from a centralized controller with global knowledge of the network's current state [2].

A review of major security threats in SDN and their existing solutions on different planes, alongside changes in research methods over time in addressing SDN security and privacy concerns is presented in [3]. It results that, despite the manifold benefits of SDN, its centralized structure can be a vulnerability to DDoS attacks, one of the most significant security risks [4].

Received 30 January 2024; revised 20 September 2024; accepted 29 September 2024. Date of publication 4 October 2024; date of current version 12 December 2024. This work was supported by the National Research Foundation of South Africa under Grant 141918. Recommended for acceptance by H. Wu. *(Corresponding author: Vianney Kengne Tchendji.)*

Yannis Steve Nsuloun Fotse is with the Department of Mathematics and Computer Science, University of Dschang, Dschang 67, Cameroon.

Vianney Kengne Tchendji and Mthulisi Velempini are with the Department of Computer Science, University of Limpopo, Mankweng 0727, South Africa (e-mail: vianneykengne@yahoo.fr).

Digital Object Identifier 10.1109/TC.2024.3474180

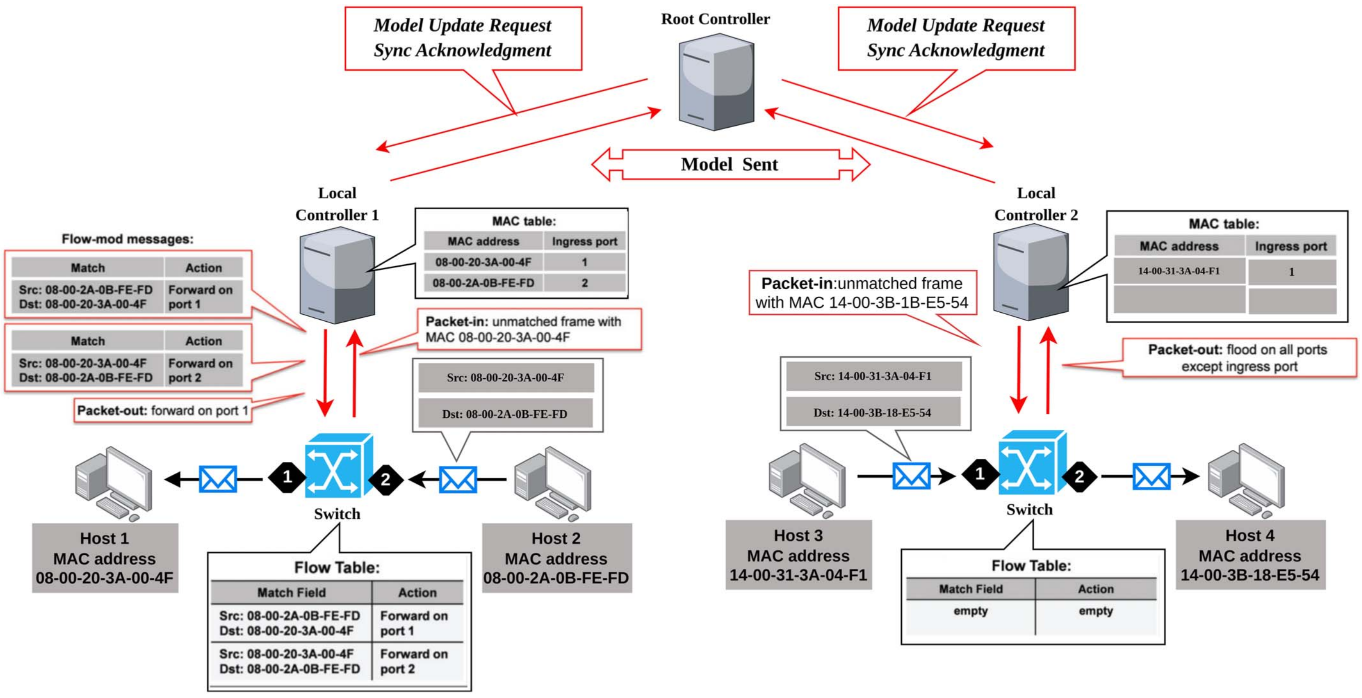

Model Update Request
Sync Acknowledgment
Root Controller
Model Update Request
Sync Acknowledgment
Model Sent
Local Controller 1
Local Controller 2
Flow-mod messages:
Match | Action
Src: 08-00-2A-0B-FE-FD Dst: 08-00-20-3A-00-4F | Forward on port 1
Src: 08-00-20-3A-00-4F Dst: 08-00-2A-0B-FE-FD | Forward on port 2
Packet-out: forward on port 1
MAC table:
MAC address | Ingress port
08-00-20-3A-00-4F | 1
08-00-2A-0B-FE-FD | 2
Packet-in: unmatched frame with MAC 08-00-20-3A-00-4F
Src: 08-00-20-3A-00-4F
Dst: 08-00-2A-0B-FE-FD
Switch
Host 1 MAC address 08-00-20-3A-00-4F
Host 2 MAC address 08-00-2A-0B-FE-FD
Flow Table:
Match Field | Action
Src: 08-00-2A-0B-FE-FD Dst: 08-00-20-3A-00-4F | Forward on port 1
Src: 08-00-20-3A-00-4F Dst: 08-00-2A-0B-FE-FD | Forward on port 2
Packet-in:unmatched frame with MAC 14-00-3B-1B-E5-54
MAC table:
MAC address | Ingress port
14-00-31-3A-04-F1 | 1
Packet-out: flood on all ports except ingress port
Src: 14-00-31-3A-04-F1
Dst: 14-00-3B-18-E5-54
Host 3 MAC address 14-00-31-3A-04-F1
Host 4 MAC address 14-00-3B-18-E5-54
Flow Table:
Match Field | Action
empty | empty

Fig. 2. The interaction of network devices in a hierarchical multicontroller SDN.

DDoS attacks occur when a large number of packets are sent to a network device to disrupt service or normal traffic flow. These attacks often involve forging the source addresses of the packets, making it difficult for switches to locate a match in their communication table. As a result, the packets are sent to the controller for processing [5]. The accumulation of non-malicious and DDoS-spoofed packets can overwhelm the controller's resources, leading to continuous processing until the resources are depleted [5]. There is a need for backup controllers to address these challenges; however, these can also be targeted and rendered inoperable due to the high volume of traffic directed to the controllers. DNS-based DDoS attacks exploit vulnerabilities in the DNS infrastructure by inundating it with a stream of DNS requests [6]. This attack exhausts the available resources such as bandwidth, computational power, and memory within the SDN controller or the entire network.

Attackers change with time, and their techniques evolve, making systems more vulnerable. Detection techniques range from conventional analytical and contemporary Machine Learning (ML) techniques to the integration of multiple approaches, resulting in effective Deep Learning (DL) techniques. All leverage the benefits of the control plane's comprehensive perspective and centralized management capability to detect DDoS attacks effectively. Nevertheless, as a consequence of flow aggregation, metrics, and categorization, amongst other processes carried out on the SDN controller, the controller encounters a significant load of tasks when the network increases, leading to a delay in the detection of attacks [7]. The most unfavorable scenario arises when the controller becomes overwhelmed or when it fails before the DDoS attack is detected [7].

The survey in [8] offers a thorough overview of the work that has been carried out to optimize the processing load of the controller, address malicious attacks, efficiently update flow tables, and enhance flow table management to maintain the stable performance of SDN. The latest countermeasures against DDoS attacks within SDN environments can be categorized into two distinct methodologies, which are ML and DL techniques. In the first methodology, a pre-trained ML model is employed to classify the network flows into normal or malicious based on the features related to traffic characteristics which are collected periodically [9]. However, the performance of the ML model is generally dependent on the dataset used for training. Therefore, the latest datasets with realistic attacks and regular network activity are imperative. Moreover, when applied on a large-scale network, the ML method may increase the controller workload and prolong the detection period.

In the DL approach, the detection mechanism is similar to the ML methodology. Nevertheless, in the case of DL, the feature selection is performed automatically by the DL model [10]. Although DL approach achieves better results than the ML approach, it has some limitations. First, the duration of the training process for the DL model is considerably higher. Secondly, it may require more resources, which can overwhelm the controller as the network scales. Additionally, when constructing a DL model, multiple parameters may be considered.

This paper proposes a new modular SDN architecture that uses a distributed detection system on various network devices to detect DDoS attacks in large-scale multi-controller SDN. The approach consists of a hierarchical structure, taking advantage of the multi-controller design scheme in SDN. Each local controller is equipped with a local model training module, while the root controller is equipped with a model aggregation module to aggregate the different model parameters received from local training by each local controller. This modular architecture allows the proposed approach to be implemented in a large-scale SDN environment. Another advantage is that it does not store personal data, which addresses privacy concerns.

In addition, three current datasets, namely the CICDoS2017 [11], the CICDDoS2019 [12], and the InSDN [13], encompassing the application layer, the transport layer, and the two attacks categories respectively, were employed to train the model. By utilizing these contemporary datasets, we were able to analyze the latest DoS/DDoS attacks. For real-time testing, a simulation was conducted using Mininet and a Ryu controller.

Our contributions are summarized as follows

- A Federated Learning (FL) framework to detect DDoS Attacks (both application and transport-layer attacks) in multi-controller large-scale SDN.
- An improvement in the detection accuracy and the robustness of the model by employing various up-to-date datasets.
- A comprehensive experiment using real-time traffic in a simulated large-scale SDN environment to demonstrate the proposed strategy's efficacy.

The structure of the paper is as follows: Section II discusses the background of DDoS attacks and previous research on DDoS detection in SDN. Section III presents the structure of the proposed federated learning approach. Section IV describes the experimental setup, and Section V evaluates and discusses the performance of the proposed approach. Finally, Section VI concludes the paper.

II. Background

This section briefly describes DDoS attacks and existing works on DDoS attack detection.

A. DDoS Attacks

DDoS attacks involve the misuse of available network resources by multiple interconnected devices. In SDN, DDoS attacks at the data plane layer overwhelm resources like switches or controllers, causing network congestion and service disruption. This attack floods the network with malicious traffic from multiple sources, impeding legitimate traffic flow. Consequently, network downtime, latency, and reduced service performance can occur. DDoS attacks exploit different protocols within the OSI model, with a particular emphasis on vulnerabilities at the transport layer, often utilizing IP spoofing techniques. Despite extensive research, DDoS attacks persist and continue to cause significant harm to networks. Notably, low-volume application-layer DDoS attacks are gaining popularity due to their complex nature, making them more difficult to detect compared to high-volume attacks [14].

1) Transport-Layer DDoS Attacks: Transport-layer DDoS attacks exploit vulnerabilities in the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). There are four categories of these attacks. *Flooding attacks*, such as UDP Flood and DNS Flood, monopolize target systems' bandwidth. *Protocol exploitation flooding attacks*, like SYN Flood, deplete victim resources by exploiting protocol characteristics. *Reflection-based flooding attacks* involve manipulated requests sent to reflectors, depleting victim resources through responses. Examples include DNS and NTP amplification. *Amplification-based flooding attacks* amplify redirected traffic to overwhelm victim resources, often launched with reflection-based attacks. DNS and NTP amplification attacks are both reflection-based and amplification-based.

2) Application-Layer DDoS Attacks: Alternatively referred to as layer 7 (L7) DDoS attacks, they attack the uppermost layer of the OSI model. They exhaust network resources and overwhelm the servers.

High-volume application-layer DDoS attacks involve a massive influx of seemingly legitimate requests that aim to exhaust available resources. In this study, the focus is on HTTP flood attacks, where malicious users overwhelm a server or application with valid HTTP GET or POST requests. These attacks utilize botnets, controlled by malware like Trojan horses, to disrupt the target without using spoofing or reflection techniques. The effectiveness of the attack is maximized when the server or application expends most of its resources in responding to these requests.

Low-volume Application-layer DDoS Attacks entail seemingly authentic traffic transmitted at a noticeably reduced rate. These attacks are challenging to detect as they exhibit characteristics of legitimate traffic with hidden malicious background activities [15]. This study explores three low-volume attack categories, specifically Slowloris, Slow Post, and Slow Read.

In Slowloris, an application layer DDoS attack, the target is flooded and slowed down by repeatedly prolonging an HTTP session for as long as possible.

In Slow Post, a malicious user sends HTTP POST headers with a valid "content-length" field to inform the server of upcoming data. The actual body of the POST message is sent slowly, depleting server resources.

Slow Read involves the attacker sending a valid HTTP request to a server and intentionally taking a long time to process the response. This prevents the server from triggering an idle connection timeout.

Slow HTTP attacks overwhelm a server by using a limited number of open connections. By keeping these connections open for extended periods instead of closing them, the server's resources remain in use, leading to a depletion of network resources.

Attackers are increasingly favoring low-volume DDoS attacks as they are simple, require minimal computation, and can go unnoticed until substantial damage is inflicted. This work examines application and transport layer DDoS attacks, including both low-volume and high-volume attacks.

B. Related Works

Given the evolving characteristics of networks, a responsive intrusion detection system becomes imperative. The demand for systems capable of differentiating between typical and irregular network activities has prompted the integration of ML techniques in network security, particularly in intrusion detection. Recently, comprehensive research has been conducted to implement ML methods for the detection of DDoS attacks within SDN on a global scale. The study in [25] evaluates eight ML algorithms for identifying and handling DDoS attacks in SDN. Decision Tree and Random Forest achieve the highest accuracy rates of 99.86% each. Naive Bayes has the quickest prediction time, whereas Decision Tree has the fastest learning time. It is noted that algorithms performing well on small datasets may not scale well on larger networks due to computational constraints or complexity issues.

A learning model to protect the controller has been proposed in [24]. It focuses on utilizing the Support Vector Machine (SVM) to categorize flows as normal or malicious and perform further analysis using the Self Organizing Map (SOM) algorithm. This approach demonstrates superior performance in terms of accuracy compared to using SVM and SOM individually for DDoS detection. However, it has drawbacks such as increased usage of controller resources and prolonged detection time as the network scales.

To reduce the controller's workload, a defense mechanism against DDoS attacks is designed in [23], utilizing the k-means++ algorithm to organize data into small clusters and the Fast K-Nearest Neighbors (K-FKNN) for classification. This approach reduces the load of the controller and improves the detection speed as compared to algorithms of the same type. However, the authors didn't use an up-to-date dataset, and the detection module working in the application layer is vulnerable to attacks. To improve this mechanism, the work in [22] introduces a framework that leverages cooperative detection techniques on the control plane with a detection trigger mechanism on the data plane, employing switches to monitor the transmission of data packets to counter DDoS attacks in SDN. Their approach was found to be superior, attaining an accuracy rate of 98.85%. Nevertheless, as the network encounters higher levels of traffic, the controller experiences an increased load, resulting in a decline in DDoS detection efficiency.

The study in [18] combines SDN and Moving Target Defense (MTD) to protect against DDoS attacks by redirecting attack traffic to a controlled server and ensuring service continuity. ML algorithms aid in flow classification, with key features being the integration of MTD with SDN, ML-based DDoS diagnosis, and proactive defense tactics. However, performance under varying network conditions and dependency on secondary servers need further examination [16].

Recently, DL techniques have become very popular over ML. Several researchers have utilized deep learning methodologies in detecting DDoS attacks. In [10], [26], and [27], an analysis and comparison of various DL algorithms was performed to detect DDoS attacks in SDN. The results show that deep learning algorithms performed better with high accuracy as compared to related work; however, they need a lengthy processing time resulting in attack detection delay.

According to various works [14], [21], low-rate DDoS (LDDoS) attacks are difficult to detect in SDN because attack traffic behaves like normal traffic. Hence, more complex detection processes are required. The authors in [9] provided a thorough analysis of multiple LDDoS detection algorithms used in SDN. They showed that techniques using DL with a hybrid model, like CNN-LSTM and CNN-GRU, could yield promising results. A hybrid approach that utilizes a combination of long-short term memory (LSTM) and convolutional neural network (CNN) for LDDoS attack detection is suggested in [22]. The analysis solely focused on the data within the packet header to detect these attacks, and the findings indicate that the proposed approach yields superior results. Nevertheless, due to the reliance on deep learning, the processing time is still considerably high. The work in [28] improved the detection method by developing a new combined approach based on gated recurrent units (GRUs) and CNNs. Even though the model worked well, it lacks a real-world implementation. Additionally, the model is not optimized for real-time detection of attacks in live SDNs, as it relies on detecting traffic patterns that may span multiple time intervals of data capture [9].

In [21], a three-level detection system was proposed. The authors collected data flow during a specific time period, and the entropy module determines the target based on the uncertainty levels for each IP address. Subsequently, the classification module utilized various ML algorithms such as Bayesian network, J48, and random tree, achieving high accuracy rates ranging from 99.33% to 99.87%. However, limitations of the study include the additional burden placed on the controller task and the absence of newer types of DDoS attacks in the datasets utilized.

In [15], an adaptive SDN-driven framework is proposed to detect and mitigate LDDoS attacks using ML. The detection mechanism consists of two components: an Intrusion Prevention System (IPS) integrated within the controller to manage network traffic, detect suspected attackers, and an Intrusion Detection System (IDS) that selects the appropriate ML model (RF, REP Tree, Random Tree, MLP, and J48). To train the models, they used CICDoS2017, and their findings demonstrated that the MLP algorithm outperformed alternative models with a 95% accuracy rate. Nevertheless, the accuracy can be improved to achieve better detection. To overcome this limitation, the work in [14] introduced an adaptable framework to detect DDoS attacks at the transport and application layers. The authors examined three ML models (KNN, SVM, and RF) and four DL models (MLP, GRU, and LSTM) using CICDoS2017 and CICDDoS2019 datasets. The outcomes exhibited high detection rates of 99.87% and 99.47%, which is more than the rates achieved in [15]. However, as the network size increases, the controller encounters significant overhead, resulting in the delay of attack detection.

In [19], an LSTM activation function is employed to distinguish between various LDDoS attacks in IoT network traffic by analyzing unique features of these attacks. These features were employed to optimize the parameters, leading to the LSTM model achieving high performance with 98.8% accuracy in classifying LDDoS attacks. However, the authors did not assess the effectiveness of their approach on a large dataset. Still to combat LDDoS attacks, the work in [29] introduced an online ML approach using Stochastic Gradient Descent (SGD) optimizer

TABLE I
COMPARISON OF RELATED WORKS

Year [Ref.]	Detection	Mitigation	DDoS Attacks		Method	Datasets	Limitation
			High-Volume	Low-Volume			
2024 [16]	✓	✓	✓	✓	BernoulliNB, Passive-Aggressive, SGD Classifier, MLP	CICDDoS2019, InSDN, slow-read-DDoS	may demand substantial resources in terms of computing power, storage, and memory, particularly for large-scale networks.
2023 [17]	✓	✗	✓	✗	entropy, LSTM	CICDDoS2019	resource-intensive training and the requirement for evaluation in large-scale network environments.
2023 [18]	✓	✓	✓	✓	GNB,SVM, RF, MLP	CICDDoS2019	sensitive to network conditions and scalability concerns while lacking training diversity.
2022 [19]	✓	✗	✗	✓	LSTM	Edge-IIoT	the effectiveness on massive scale data still need to be assess.
2022 [20]	✓	✗	✓	✗	entropy, packet window rate	Custom dataset	No testing on large datasets which may cause false positive as the network size increase.
2022 [19]	✓	✗	✗	✓	CNN-GRU	CIC DoS	not optimized for real-time detection of attacks in live SDNs.
2021 [14]	✓	✗	✓	✓	KNN, SVM, RF, MLP, GRU,LSTM	CICDoS2017, CICDDoS2019	as the network size increases, the controller encounters significant overhead, resulting in the delay of attack detection.
2021 [21]	✓	✗	✓	✓	Bayesian network, J48, random tree	UNB-ISCX, CTU-13, ISOT	adds extra burden controller and the datasets used lack newer types of DDoS attacks.
2020 [15]	✓	✓	✓	✗	RF, REP Tree, Random Tree, MLP, J48	CICDoS2017	the accuracy can be improved to achieve better detection.
2020 [22]	✓	✗	✗	✓	CNN-LSTM	Custom dataset	the processing time is still considerably high.
2019 [23]	✓	✗	✓	✗	K-means++, K-FKNN	NSL-KDL	didn't use an up-to-date dataset and the detection module working in the application layer is vulnerable to attacks.
2019 [24]	✓	✗	✓	✗	KNN, NB,SVM and SOM	CAIDA 2016	increased usage of controller resources, prolonged detection time, potential for false positives.

and Explainable Boosting Machine (EBM) classifier in SDN-based networks. The method was further improved in [16] to detect and address zero-day DDoS attacks by incorporating a dynamic feature selection mechanism. This model achieved a detection rate of 99.2%, surpassing other similar models on various datasets. However, the implementation and maintenance of an online ML model may demand substantial resources in terms of computing power, storage, and memory, particularly for large-scale networks.

There are a few studies that combined multicontroller and entropy methodologies to identify DDoS attacks in SDN. An algorithm that combines the entropy with the Packet Window Initiation Rate algorithm has been introduced for early detection to improve security of SDN settings in [20]. Although the initial results show a good detection rate, the lack of testing on large datasets could result in many false alarms. Additionally, the algorithm can't be able to quickly adapt to new and emerging DDoS attack techniques, leaving networks vulnerable. The authors in [17] developed a system that detects attacks by analyzing entropy variations in the targeted host's IP address within the first 250 packets of malicious traffic. They then use a DL model to categorize the attacks and share information with other controllers. The system achieved an accuracy of 99.42% on the CICDDoS2019 dataset using an LSTM model. However, challenges like resource-intensive training and the requirement for evaluation in large-scale network environments were noted as limitations.

In reviewing current works on DDoS attack detection, as presented in Table I, it appear that the utilization of ML and DL methods has shown promise in mitigating the impact of these attacks and improving the accuracy of security measures. However, a common limitation across many previous studies is the significant gap that exists in considering the scalability of network infrastructures, leading to increased controller loads and decreased efficacy of existing solutions as network traffic volumes grow. This scalability issue also results in longer attack detection times, particularly in large networks. Moreover, there is potential for further accuracy enhancement by using up-to-date datasets reflecting real-world data on DDoS attacks. To tackle these challenges, we proposes a novel detection framework that is specifically designed for SDN-based networks. This framework leverages the benefits of a distributed learning strategies to not only improve the accuracy of DDoS attack detection but also address the scalability issues faced by existing

TABLE II
COMPARISON OF PROPOSED APPROACH WITH RELATED WORKS

Aspect	**FedLAD**	**Related Works**
Learning Paradigm	Federated Learning	Machine Learning ([14], [15], [21], [22], [23]); Deep Learning ([16], [17], [18], [19])
Scalability	Highly scalable, handles massive data	Limited capabilities ([18], [23], [24])
Response time	Real-time responses	May delay due to massive data ([19], [20])
Resource efficiency	Low resource usage	May incur high resource usage ([14], [16], [17], [21])
SDN-architecture	Multi-controller	Single-controller ([15], [16], [19], [20], [23], [24])

solutions. Table II presents a comparison, highlighting the innovations of our approach.

C. Federated Learning (FL)

Federated learning is a form of decentralized ML techniques that operate without centralized training data and model training [30]. Typically, FL involves a central server and multiple client devices [31]. The iterative process of FL generates models, following four basic steps for a given training task [31]:

- Initializing the training task: the central server sends the training task to each client and shares the current global model to all clients.
- Model training: clients update their local models with the global model they receive and then proceed to train new local models using their own local data. The training iteration continues until it reaches the maximum designated number.
- Model uploading: clients upload their local models to the central server.
- Combining local models: the central server utilizes aggregation algorithms to produce a unified global model from the uploaded local models.

The central server and clients continue the specified process until certain conditions are met. In FL, clients upload their local model rather than sharing their data, ensuring data privacy and security. Due to the limited data owned by each client, the global model aggregated by the central server from all local models results in higher quality. Participation in FL allows clients to obtain a global model with improved accuracy compared to their individual local models.

III. PROPOSED SYSTEM DESIGN

In this section, we present a federated learning scheme designed to detect DDoS attacks in large-scale SDN. Moreover, we discuss datasets used in training and testing.

A. System Architecture

The overall system architecture illustrated in Fig. 3 provides a detailed view of the proposed system's core components tailored to detect both high-volume and low-volume DDoS attacks in large-scale SDN environments. This architecture shows controllers in a distributed control plane arranged in tiers. The tiered distributed control plane typically comprises a root controller and multiple local controllers designed to significantly enhance computational efficiency and scalability. The proposed system consists of three main entities:

1) **Local controllers:** in charge of gathering local data for training local models and the main actors in detecting attacks using the Intrusion Detection Server (IDS).
2) **Intrusion Detection Server (IDS):** employs a learning model to analyze and categorize incoming traffic in real-time, detecting possible DDoS attacks.
3) **Root controller:** functions as the central server that oversees the aggregation of local models from local controllers.

The proposed system architecture aims to improve the scalability of the DDoS detection framework, as well as streamline and optimize the consumption of controller resources. The system workflow is described as follows:

1) Initial setup:
   - The root controller initializes the federated learning process and defines the communication protocols between the root controller and local controllers.
   - The root controller sends model parameters to each local controller.
2) Each local controller through the bandwidth monitoring module monitors network traffic for anomalies and collects local data related to network traffic.
3) The collected network traffic data are store into the Database (DB) server. These data are preprocessed to extracts relevant features and formatted for the detection module and the training module.
4) Each local controller through the local training module trains a local FL model based on the local data stored in the DB Server.
5) The resulting model weights from the local training module of each local controller are sent to the root controller.
6) Global model aggregation:
   - the root controller aggregates the local models from all local controllers periodically;
   - the global model is sent back to the local controllers for updates to their local models. The updated model is used to analyze incoming traffic data and categorize it as normal or DDoS attack.

One of the key features of the proposed architectural design is not just its modular structure, but also its ability to improve model accuracy and diversity. This is because it leverages data from a wider range of sources, leading to a more diverse and robust model. Each local controller contributes its unique perspective to the training process, resulting in models that can better generalize to unseen data.

In the following sections, we will provide comprehensive explanations of how each module operates.

1) Local Controllers: The local controllers are responsible for managing and controlling a specific SDN domain. Each local controller interacts with the switches and hosts in their respective domains. They collect local data about the network

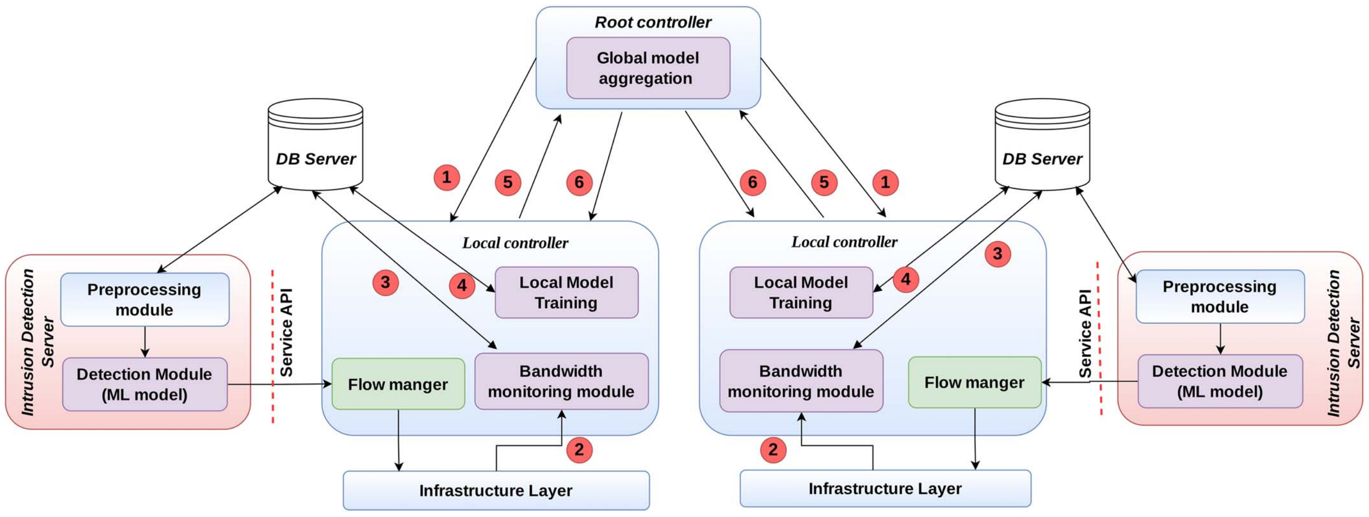


Fig. 3. Proposed FedLAD architecture for DDoS attack detection.

conditions, traffic patterns, and performance metrics within their domain. Local controllers are made up of three components: the bandwidth monitoring module, the local training module, and the flow management module.

The bandwidth monitoring module

By monitoring the incoming traffic and analyzing the bandwidth usage, this module can detect abnormal patterns that indicate a potential DDoS attack. The module continuously monitors the data flow within the SDN network. It collects data about the source and destination IP addresses, packet sizes, protocols used, and timestamps (2). Initially, the module establishes a baseline or normal behavior by analyzing the network traffic during regular operation. It measures and documents the average bandwidth usage and traffic patterns. The module then compares the current incoming traffic with the established baseline. It identifies any significant deviations from normal behavior and flags them as potential anomalies. Unusually high packet rates, sudden surges in traffic volume, or a significant increase in traffic from specific sources can be indicative of a DDoS attack. It later stores (3) the collected data about the flows in the database server (DB server) for further processing. The entire process is described in Algorithm 1.

The local training module

The local data collected and stored in the DB server undergo preprocessing to enhance the classification accuracy and avoid misleading outcomes. A flow containing a series of features from the data is forwarded to the preprocessing module. Utilizing all the features from these flows may lead to increased complexity in training and detection delays [14]. Thus, measures like data cleaning and dimensionality reduction were carried out. The preprocessing steps applied to the collected flow data consist of data preparation, transforming data, feature engineering, and class imbalance mitigation, as described in [14]. The local training module subsequently employs the processed data to train a learning model. Thereafter, the local controllers distribute the resultant model parameters to the root controller for updating the global model, as outlined in the

Algorithm 1: Monitor traffic algorithm

```
Procedure Monitoring()
1   for each controller in network do
2       for each switch do
3           $F_i$ = Incoming_flow;
4           $F_{stats}$ = Flow_statistics;
5           if $Base_{Line} notestablished$ then
6               $Base_{Line}$ = ComputeBaseLine($F_{stats}$);
7           else
8               Compare $F_i$ with $Base_{Line}$ ;
9               if abnormal trafic pattern detected then
10                  Flag as potential DDoS ;
11                  Send $F_i$ to $DB_{server}$;
12                  Start Detection() ;
13              else
14                  Send $F_i$ to $DB_{server}$;
15      Start Preprocesing();
16      Start TRAIN();
```

Algorithm 2. All participating controllers in training receive specific weights assigned to their ML algorithm parameters. For SGD, the learning rate η will serve as a parameter. Each local controller contains its dataset from its DB server, denoted as d_n. The model parameters of the local controller are subsequently modified utilizing the weight obtained from the root controller (1), and the model is trained on the corresponding d_n dataset relating to each local controller (4).

Flow manager module

This module plays a critical role based on the results obtained from the classification phase. After classifying traffic as either legitimate or malicious using the ML model, the flow manager module mitigates the DDoS attacks by redirecting malicious

Algorithm 2: Local training algorithm

```
Procedure Train()
1   $D$ = Local dataset stored in DB server $d_n$ ;
2   $M_{global}$ = Receive(global_model);
3   for each node in K (list of controllers) do
4       Train $M_{global}$ on their respective dataset $d_n$ ;
5       $W_{weight} = W_{weight} - \eta\Delta l(W_{weight}, d_n)$;
6   return $W_{weight}$ ;                // The model's weight
```

Algorithm 3: Aggregation algorithm

```
Procedure Aggregate()
1   $W_{weight}$= ReceiveWeight();
2   $W_{average} = \frac{1}{n}\Sigma_{k=1}^{n} W_k$ ;
3   Set weight of $M_{global} = W_{average}$ ;
4   return $M_{global}$ ;                // The aggregated model
```

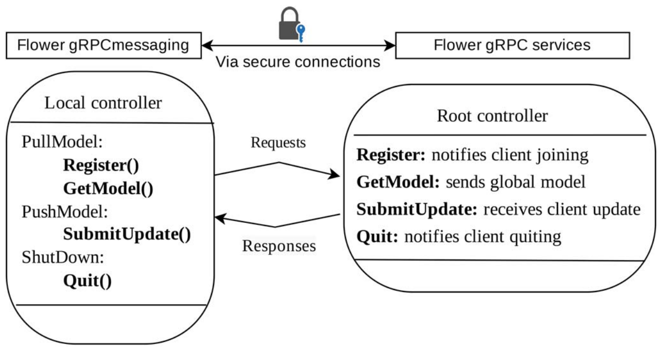


Fig. 4. Communication protocol.

Algorithm 4: DDoS detection algorithm

```
Procedure Detection()
1   for each $F_i$ in $DB_{server}$ do
2       if PredictFlow($F_i$) == Attack then
3           Apply Mitigation for $F_i$ ;
```

flows, implementing traffic rate limiting, and creating blackhole routes. As this task is a component of a wider project, the subsequent action will involve supplementing this module with an effective suppression plan to counter the DDoS attacks in the network using the projections acquired through the FL models.

2) Root Controller: The root controller serves as the central coordinating entity in the multi-controller architecture. Its primary role is to manage the overall federated learning process and coordinate with the local controllers. It performs model aggregation based on parameters received from local controllers after local training. Three aggregation techniques were implemented and analyzed in this work. These are: the federated averaging algorithm proposed in [32], Astraes proposed in [33], and the aggregation based on the ranking of clients' models using their accuracy as proposed in [34]. Astraes considers data distribution to select the client (in this case local controller) that will participate in the training round. Algorithm 3 explains the stochastic gradient descent (SGD) based federated averaging methodology. The weight obtained after the model perfectly fits the data of each local controller is labelled as W_{weight}. Consequently, W_{weight} of every local controller is transmitted to the root controller for averaging purposes (5). The average weight $W_{average}$ is then sent back to all participating local controllers for the subsequent training round (6). Federated averaging and training continue iteratively until the objective function of the global model stabilizes at the desired optimum.

Root to local controller communication

The proposed federated learning system uses the framework that utilizes the Flower framework [35]. It builds upon a contemporary open-source Remote Procedure Call (RPC) framework capable of high performance. This framework facilitates communication between the root controller and local controllers throughout the training phase, as detailed in Fig. 4. During the model training session, there are four fundamental instructions:

- *Register:* The local controller initiates the FL training process by sending a participation request to the root controller. The root controller verifies the local controller's identity and responds accordingly.
- *GetModel:* Retrieves the model for the ongoing round from the root controller. It verifies and returns the global model to the local controller.
- *SubmitUpdate:* Transmits the refined local model following the completion of the current round of FL training to the root controller for aggregation.
- *Quit:* When a local controller is not eligible to engage in the training phase.

3) Intrusion Detection Server:

Detection Module

The detection of DDoS attacks includes an offline (training) stage and an online detection stage. In the first stage, an initial classifier receives a group of normal and malicious flow features to train the model. After training, the derived characteristics are utilized to categorize the network host activities as either normal or malicious, based on traffic type. Subsequently, the acquired model is deployed to classify incoming flows in real time, distinguishing them as either DDoS attacks or normal traffic. The model undergoes continuous updates through the federated learning procedure. The detection process presented in Algorithm 4 relies on the XGBoost algorithm.

XGBoost, short for Extreme Gradient Boosting, is a scalable and distributed library for ML. It employs gradient-boosted decision tree (GBDT) algorithms and supports parallel tree boosting. It is regarded as the best ML library for handling regression, classification, and ranking tasks. The concept was introduced in [36]. Apart from its efficiency, the XGBoost classifier performs well in overfitting and achieves optimal use of computational resources. These benefits arise from streamlining the goal functions, which enables a fusion of predictive and normalized elements, as well as the ability for concurrent processing during the training stage [37]. Regarding the steps involved in the XGBoost algorithm, the primary learner is trained on the complete dataset. Subsequently, the second learner is trained on the errors made by the first learner. This iterative process continues until a specified condition is satisfied, signifying the completion of the final prediction model. The aggregated predictions of all learners are then combined to generate the final prediction model. The forecasting technique at a particular step is denoted by equation 1.

$$Y_i^{(t)} = \Sigma_{n-1}^{t} Y_n = Y_i^{(t-1)} + Y^t(x_i) \tag{1}$$

In this context, Y_i symbolizes the input value, $Y_t(Y_i)$ denotes the prediction made at time step t, and $Y_i^{(t-1)}$ refer to the prediction made at the preceding time step $(t-1)$.

Equation 2 measures the effectiveness of the model.

$$Obj^{(t)} = \Sigma_{k=1}^{n} l(\overline{y}, y) + \Sigma_{k=1}^{t} \alpha(Y_i) \tag{2}$$

In this formula, n sample size, l denotes the loss function, and α denotes the regularization term explicitly defined in equation 3.

$$\alpha(f) = \delta T + \frac{1}{2}\sigma\mu^2 \tag{3}$$

the parameter δ refers to the minimum loss required for further splitting of the leaf node, σ denotes the regularization coefficient, and μ symbolizes the vector of predicted values within the leaf nodes.

The FL model is trained using the latest datasets that encompass a range of DDoS attacks and normal traffic detailed in Section III-B.

B. Datasets

The use of datasets is a requirement for ML/DL intrusion detection techniques. The network traffic contains confidential information, and its availability may compromise the confidentiality of users, organizations, and personal interactions. To address any privacy concerns, many researchers generate their data through simulation. The following three datasets are used in this study:

- *CICDoS2017 dataset [11]:* It presents six attacks at the application layer: DNS queries, slow send body, slow read, DoS HTTP Get flooding, DDoS HTTP Get flooding, improved DoS HTTP Get flooding, and slow send header. However, quality is a main concern of the dataset [38]. Furthermore, the CICDoS2017 dataset is large and includes numerous duplicate entries that appear to be irrelevant for any intrusion detection system (IDS) training [13].
- *CICDDoS2019 dataset* [12]: It consists of various DDoS attacks that can be executed through the application layer protocols based on TCP/UDP. Nevertheless, most attacks are considered to be high-volume attackers in the CICDDoS2019 dataset.
- *InSDN dataset* [13]: It is the most up-to-date dataset which contains the benign and various attack categories (both high-volume and low-volume) that may arise within the various components of the SDN environment.

TABLE III
TRAFFIC DISTRIBUTION FOR EACH DATASET

Datasets	Attack Types	Attack Instances	Benign Instances
CICDoS2017	Slowheaders	5436	2,359,087
	Slowloris	5796	
	Slowread	5499	
	DDos	41835	
CICDDoS2019	SYN	927869	802,503
	UDP	989,492	
	DrDNS	27,864	
InSDN	DoS/DDoS	275515	68424

Specifically, the CICDoS2017 dataset is selected for its inclusion of low-rate DDoS attacks as well as various application layer attacks commonly encountered. The CICDDoS2019 dataset is chosen for its focus on high-volume DDoS attacks across application layer protocols, as well as transport layer attacks. Lastly, the InSDN dataset is preferred as one of the most up-to-date datasets that cover both high-volume and low-volume attacks within the SDN environment, making it a comprehensive choice for training a DDoS detection model. The traffic distribution between benign and normal traffic for each dataset is outlined in Table III.

IV. EXPERIMENTAL SETUP

This part delves into the experimental configuration and technologies used to analyse and evaluate the proposed system in the SDN environment.

A. Environment Setup

To simulate the experiment, we utilized Oracle VirtualBox Manager 6.0. The SDN network layout, as illustrated in Fig. 5, was established on Ubuntu 22.04.1 LTS using the Mininet emulator. The administration and orchestration of the network were managed through the Ryu Python-based SDN controller, a freely available software framework. The SDN environment encompassed controllers, switches, and hosts. In generating both attack and normal network traffic, we employed tools such as Hping3, slowhttptest, and iPerf3. The specifics of the experimental parameters are outlined in Table IV.

B. Performance Metrics

The assessment criteria employed to evaluate the classification model's performance include accuracy, precision, recall, and F1-score. The confusion matrix is a tabular representation that illustrates how well a classification model is performing by comparing its predicted and actual results. The aforementioned performance metrics are calculated using four key elements within the matrix, and their definitions are as follows:

- *True Positive (TP):* the number of instances that are correctly identified as attack traffic;
- *True Negative (TN):* the number of instances that are correctly identified as normal traffic;
- *False Positive (FP):* the number of instances that are incorrectly identified as normal traffic;

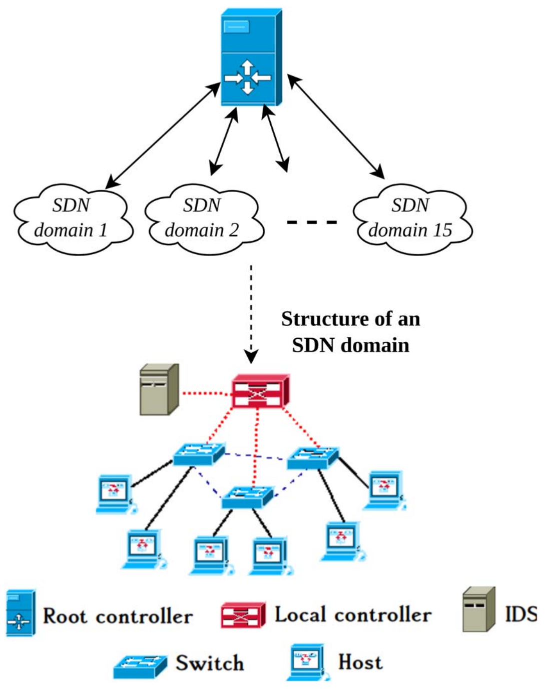


Fig. 5. Network topology of 90 hosts used for simulation.

TABLE IV
THE PARAMETERS USED FOR THE EXPERIMENT

Virtualization Software	VirtualBox
Operating system	Ubuntu 22.04.1 LTS
CPU Cores	5
Main memory	16GB
Storage	50GB
Network simulator	Mininet
SDN controller	Ryu
Switch model	open vSwitches
Number of controllers	16
Number of switches	45
Number of hosts	90
Maximum bandwidth	500Mb/s
IP address span for hosts	10.0.0.1-10.0.0.90
Benign-traffic generator	iperf3
malicious traffic generator	Hping3, slowhttptest
network layout	Tree

- *False Negative (FN):* the number of instances that are incorrectly identified as attack traffic.

Using the provided confusion matrix values, the mentioned metrics and performance measures were computed based on the following definitions:

- *Accuracy (A):* refers to the fraction of samples that were classified correctly, that is:

$$A = \frac{TP + TN}{(TP + TN + FP + FN)} \tag{4}$$

TABLE V
BINARY CLASSIFICATION RESULTS: OFFLINE PERFORMANCE

Dataset	Accuracy	Precision	F1-Score	Recall
CICDoS2017	99.92%	99.95%	99.97%	99.95%
CICDDoS2019	99.82%	99.90%	99.90%	99.90%
InSDN	99.86%	95.00%	95.00%	95.00%

- *Precision (P):* refers to the fraction of samples that were correctly identified as normal among all the samples classified as normal.

$$P = \frac{TP}{TP + FP} \tag{5}$$

- *Recall (R):* refers to the ratio of accurately classifying normal samples out of the total number of normal samples, that is:

$$R = \frac{TP}{TP + FN} \tag{6}$$

- *F1-score or F-score (F1-measure):* is the balanced average of precision and recall, providing a more precise assessment of the classifier's effectiveness compared to just considering precision and recall independently, that is:

$$F1 = 2 * \left(\frac{P * R}{P + R}\right) \tag{7}$$

V. RESULTS AND DISCUSSIONS

The experiments are conducted using both offline and online training with varying numbers of attackers and packet rates, specifically considering scenarios with many attackers. Furthermore, we assessed the model accuracy across various network scales to demonstrate scalability, in addition to the evaluation of the resource consumption on the controller to illustrate the efficiency of the proposed approach.

A. Performance of the Classification Model

This section presents evaluations of both binary and multi-class metrics to accurately measure the model's performance on a centralized and distributed schemes. Binary metrics focus on the model's ability to detect attacks and determine if it can effectively differentiate them from legitimate traffic. On the other hand, multi-class metrics evaluate the model's capability to identify specific attack categories.

In the offline stage, the XGBoost model's binary performance was evaluated on three datasets: CICDoS2017, CICDoS2019, and InSDN, while the multiclass performance was evaluated only on CICDoS2017 and CICDoS2019, as InSDN contains only a binary class. These evaluations were performed using both the centralized and FedLAD scheme for binary and multiclass classification. The findings for the centralized scheme are depicted in Table V and Table VI, while Table VII and Table VIII present the results for the proposed FedLAD derived from the confusion matrices in Fig. 6.

TABLE VI
ATTACK-WISE MULTI-CLASS CLASSIFICATION RESULTS: OFFLINE PERFORMANCE

Datasets	Attacks	Acc.	Prec.	F1-S.	Rec.
CICDoS2017	Slowheaders	99.97%	99.10%	98.20%	97.25%
	Slowloris	99.90%	98.95%	97.85%	96.80%
	Slowread	99.88%	99.03%	98.10%	96.95%
	Dos Get	99.86%	98.92%	97.85%	97.00%
CICDDoS2019	SYN	99.98%	99.14%	98.34%	98.05%
	UDP	99.94%	99.24%	98.11%	98.00%
	DrDNS	99.93%	98.62%	97.86%	97.95%

Upon examining the performance of the centralized and FedLAD schemes in Table V, Table VI, Table VII, and Table VIII, it shows that their performance is relatively similar. In terms of accuracy, both schemes exhibit high rates, with the centralized scheme achieving an average accuracy of 99.87% across all datasets, while the FedLAD scheme achieves an average accuracy of 98.33%. This indicates that both schemes are effective at accurately detecting DDoS attacks.

When considering precision, the centralized scheme demonstrates values ranging from 95% to 99.95%, compared to the FedLAD scheme with values ranging from 97.75% to 98.21%. However, the precision values in the FedLAD scheme are still relatively high, indicating its ability to correctly identify positive instances. The F1-Score also shows a comparable performance between the two schemes. Although the centralized scheme achieves slightly higher F1-Scores, the values in the FedLAD scheme remain strong, highlighting its overall performance.

Examining recall values, the centralized scheme ranges from 90.90% to 99.95%, while the FedLAD scheme ranges from 98.23% to 98.90%. Both schemes demonstrate high recall rates, indicating their capability to capture the majority of DDoS attacks.

B. *Performance of FedLAD in SDN Environment*

Following the assessment of the XGBoost classifier's effectiveness on CICDoS2017, CICDDoS2019, and InSDN datasets, this section presents the evaluation of FedLAD using SDN traffic. The assessment centres evaluate the flexibility and efficiency of FedLAD's methodology in terms of the bandwidth monitoring module and accuracy in detecting DDoS attacks using the global model, and the controller overhead.

1) Efficiency of the Bandwidth Monitoring Module: In Fig. 9, the efficiency of the proposed bandwidth monitoring module in managing the utilization of network capacity can be observed. Two scenarios are carried out in an actual SDN setting, one with the bandwidth monitoring module activated and the other with the bandwidth monitoring module deactivated. In Fig. 9(a) (scenario 1), the flow traffic is more than the threshold without any regulation, resulting in the maximum bandwidth being reached throughout the transmission. The DDoS traffic floods the bandwidth, leading to a significant disruption of network traffic. On the other hand, in Fig. 9(b) (scenario 2), it can be observed that when the bandwidth monitoring module is implemented, the detection and regulation of excessive flow beyond the predefined limit is in place, preventing the bandwidth from being overloaded. Consequently, the adverse impact of the attacks on network traffic is reduced with the implementation of the control algorithm.

2) Overall Model Accuracy: The evaluation of accuracy results in detecting DDoS attacks with the global model in SDN environment focuses on three different aggregation techniques (as mentioned in Section III-A2). These techniques are used to combine the local models trained by the local controllers participating in the federated learning process. By aggregating the contributions from multiple local controllers, FedLAD aims to improve the accuracy and reliability of DDoS attack detection in an SDN environment.

The findings presented in Table IX indicate that FedAVG exhibits lower accuracy, precision, and recall in comparison to Astraes and the ranking client techniques. Astraes and Ranking Client perform the same in most metrics, but Ranking Client has a slightly higher precision and lower recall. Overall, Astraes and Ranking Client outperform FedAVG in most performance metrics. The performance of different aggregation techniques is influenced by the heterogeneity of data across different local controllers. Since the data distribution or characteristics vary significantly across local controllers, Astraes and Ranking Client handle the heterogeneity better and produce more accurate results. Overall, these results highlight the variation in performance achieved by different aggregation techniques in federated learning and emphasize the importance of selecting the most appropriate technique for a specific use case or scenario.

In order to evaluate the scalability of the proposed FedLAD approach, we conducted experiments with varying network scales, using 5, 10, and 15 local controllers (i.e. 30, 60 and 90 hosts respectively). Results presented in Fig. 7 showed that global model accuracy improved with higher controller counts, reaching 96.54% for 5 controllers, 96.78% for 10 controllers, and 97.64% for 15 controllers. This trend highlights the effectiveness of the FedLAD approach in achieving higher accuracy levels as the network scale expands, further supporting its potential for larger network settings.

3) Assessment of Controller Overhead: To assess the decrease in controller resource utilization introduced by the proposed method, we compared the CPU usage and the memory usage of the controller in the proposed scheme to the centralized scheme. The centralized scheme shares similarities with the proposed approach, as they both use a polling mechanism on the control plane for detection. At the same time, no extra processing takes place on the data plane. Instead of encountering new addresses during DDoS attacks, the network suffers from a high volume of fraudulent ICMP packets that disrupt the link. Since the switch already has flow entries for forwarding, it does not need input from the controller, resulting in stable CPU usage without significant fluctuations. Fig. 10(a) illustrates that the proposed scheme has a significantly lower average CPU usage (31.96%) compared to the centralized scheme (48.17%). Specifically, the proposed scheme exhibits a 16.21% decrease in average CPU usage when compared to the central

TABLE VII
FedLAD Multi-Class Classification Results: Offline Performance

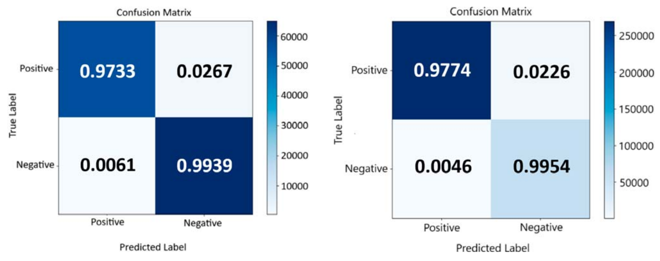
Techniques	FedAVG				Astrea				Ranking Client			
Dataset	Acc.	Prec.	F1-S.	Rec.	Acc.	Prec.	F1-S.	Rec.	Acc.	Prec.	F1-S.	Rec.
CICDoS2017	96.34%	96.18%	96.43%	96.27%	98.73%	98.54%	98.82%	98.82%	98.99%	99.51%	98.95%	98.38%
CICDDoS2019	96.48%	96.38%	96.44%	96.57%	98.87%	98.77%	98.83%	98.86%	98.90%	98.64%	98.87%	98.76%

TABLE VIII
FedLAD Binary Classification Results: Offline Performance

	Binary Metrics			
Dataset	Accuracy	Precision	F1-Score	Recall
CICDoS2017	98.55%	98.21%	98.51%	98.31%
CICDDoS2019	98.38%	97.89%	98.90%	98.90%
InSDN	98.07%	97.75%	98.50%	98.23%

TABLE IX
FedLAD Classification Results: Real-Time Performance

	Performance Metrics			
Techniques Used	Accuracy	Precision	F1-Score	Recall
FedAVG	92.62%	91.90%	93.20%	94.50%
Astraes	97.54%	97.74%	97.70%	97.66%
Ranking Client	97.64%	98.56%	97.60%	97.03%



(a) Confusion matrix CICDDoS2019 (b) Confusion matrix InSDN

Fig. 6. Confusion matrices.

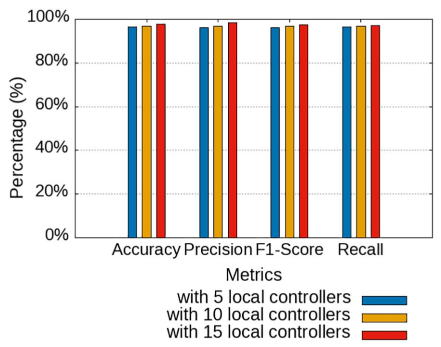


Fig. 7. Evaluation of FedLAD with varying network scales.

scheme. Similarly, the results depicted in Fig. 10(b) reveal that the proposed scheme exhibits a lower average memory usage (5.79%) on each local controller, accounting for approximately 86.85% across all local controllers. This is in stark contrast to the centralized scheme, where a single controller consumes 92.78% of the memory. This suggests that the proposed scheme may be more memory-efficient in detecting DDoS attacks in SDN environments. This is because the load is distributed, and each local controller handles a relatively limited amount of data. In contrast, the centralized scheme operates with only one controller and a single database server. Consequently, this server is solely responsible for storing the flow data of the entire network. As the network grows in size, the memory usage of the centralized scheme will significantly increase as it needs to accommodate and process more flow data on the single database server. Due to hardware limitations on the testing platform, we didn't employ a more bigger network size for our experiments. From the results, it shows that as the network size increases, the merits of the proposed approach become more significant regarding the controller burden.

Resource efficiency is achieved here through various strategies. Local controllers collect and train models within their specific network segments, reducing data transmission and latency while saving computational resources through data filtering. Collaborative learning enables separate model training on different datasets, which are then combined at the root controller to improve accuracy without compromising privacy, thus reducing computational load on individual controllers. Segregating components such as IDS and DB servers from the controller facilitates distributed computational load, optimizing resource usage and avoiding overloading any particular component.

4) Data Privacy: Regarding data privacy, as the flow data is distributed across multiple local controllers, each with its own database server, it becomes challenging for an external attacker to obtain a comprehensive view of the network's data. Even if one database is compromised, the attacker would only gain access to a limited portion of the data. This distributed nature provides a level of data privacy and mitigates the risk associated with a centralized single point of failure. On the other hand, the centralized scheme poses a higher risk in terms of data privacy. If this single server is compromised, an attacker would have access to the entire network's flow data, compromising the privacy of the entire system. Therefore, in terms of data privacy, FedLAD offers more resilience and protection against potential threats.

C. Comparison With Related Works

Numerous research and investigations have been done in DDoS attacks detection. In this section, we provide a

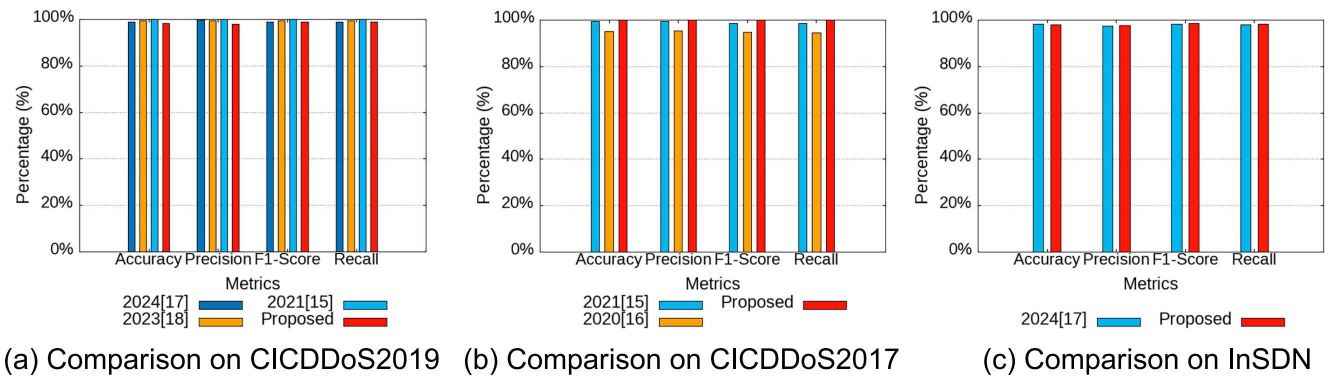


Fig. 8. Comparing the proposed approach with related work on the three datasets.

TABLE X
COMPARISON OF THE PROPOSED FEDLAD SYSTEM WITH THE EXISTING SOLUTIONS

Year [Ref.]	Scalability	Multi-Controller	Adaptability to Emerging Threats	Real-Time Detection	Resource Efficiency	Privacy Preservation
FedLAD	✓	✓	✓	✓	✓	✓
2024 [16]	✓	✗	✓	✓	✓	Not assessed
2023 [17]	✗	✓	✗	✗	✗	Not assessed
2023 [18]	✗	✗	✗	✓	✗	Not assessed
2022 [19]	✗	✗	✓	✓	✓	Not assessed
2022 [20]	✗	✓	✗	✓	✓	Not assessed
2022 [19]	✗	✗	✗	✗	✓	Not assessed
2021 [14]	✗	✗	✗	✓	✓	Not assessed
2021 [21]	✗	✗	✓	✓	✓	Not assessed
2020 [15]	✗	✗	✓	✓	✓	Not assessed
2020 [22]	✗	✗	✗	✗	✗	Not assessed
2019 [23]	✗	✗	✗	✗	✓	Not assessed
2019 [24]	✗	✗	✗	✗	✗	Not assessed

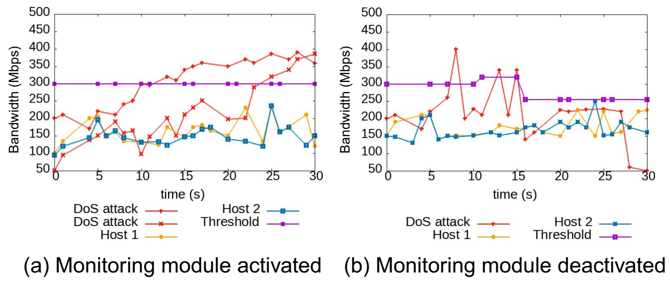


Fig. 9. Evaluating bandwidth monitoring in two scenarios.

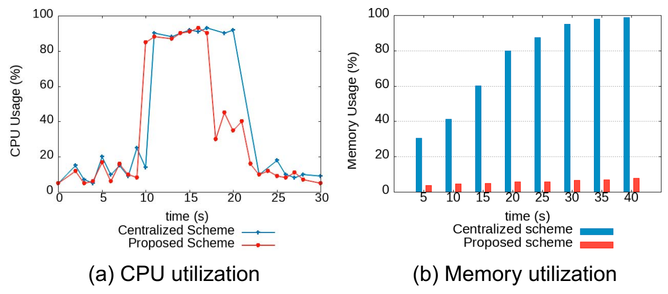


Fig. 10. Evaluating resource consumption.

comprehensive comparison of the proposed FedLAD approach using three datasets compared to existing related works. Furthermore, we critically evaluate different methodologies based on various characteristics and capabilities to highlight the contributions and potential advancements of the FedLAD approach in addressing the effects of DDoS attacks in large-scale SDN.

From the results shown in Fig. 8(b), the proposed approach outperforms the performance of [14] and [15] on CICDoS2017 with an accuracy of 99.92%, compared to 99.43% and 95.01%, respectively. Moreover, the precision and recall rates are significantly higher at 99.95% each, compared to other studies.

On the other hand, the analysis of the performances on CICDDoS2019 in Fig. 8(a) reveals that, although the proposed approach may not rank highest in all categories, it maintains a strong overall performance. Notably, with an F1-score of 98.90% compared to 98.81%, 99.44%, and 99.87% in [14], [16], [17] respectively. FedLAD demonstrates a balanced performance in both precision and recall, indicating its effectiveness in accurately identifying and classifying DDoS attacks.

Upon examining the performances on the InSDN dataset presented in Fig. 8(c), it shows that FedLAD has slightly higher precision, F1-score, and recall compared to [16]. Notably, 97.75%, 98.50%, and 98.23% respectively for precision, F1-score, and recall against 97.51%, 98.27%, and 97.93%. This indicates that the FedLAD method is overall efficient in detecting DDoS attacks.

The evaluation of the comparative indices in Table X demonstrates the effectiveness of the proposed approach, which is robust and adaptable to different DDoS attacks. As attackers constantly change their techniques to evade detection, the proposed scheme can dynamically adjust and learn new attack patterns from various SDN domains, ensuring continuous and effective protection while preserving data privacy. In contrast, existing schemes may increase the controller load as the network scale and may not keep up with evolving attack vectors, making them vulnerable to novel or advanced DDoS attacks.

VI. Conclusion

The SDN concept improves the detection of network-specific attacks compared to classical networks. This study introduced a DDoS attack detection system in large-scale SDN using federated learning, which demonstrates promising efficiency in accurately detecting DDoS attacks in multi-controller large-scale environment. By distributing the detection process across multiple controllers, the system reduces load and enhances speed. However, potential limitations exist, such as the model accuracy being dependent on the aggregation technique and the necessity to safeguard the model against data poisoning threats in order to uphold the accuracy of the detection system.

For future research, avenues for improvement could include improving the model aggregation technique, and developing mechanisms against adversarial ML threat to detect and mitigate data poisoning attacks in federated learning setups.

References

[1] I. A. Mahar, W. Libing, Z. A. Maher, and G. A. Rahu, "A comprehensive survey of software defined networking and its security threats," in *Proc. IEEE 1st Karachi Sect. Humanitarian Technol. Conf. (KHI-HTC)*, Piscataway, NJ, USA: IEEE, Jan. 2024, pp. 1–5.

[2] S. E. Vadakkethil Somanathan Pillai and K. Polimetla, "Integrating network security into software defined networking (SDN) architectures," in *Proc. Int. Conf. Integr. Circuits Communication Syst. (ICICACS)*, Piscataway, NJ, USA: IEEE, Feb. 2024, pp. 1–6.

[3] N. Ahmed et al., "Security and privacy in software defined networks, issues, challenges and cost of developed solutions: A systematic literature review," *Int. J. Wireless Inf. Netw.*, vol. 29, no. 3, pp. 314–340, Jun. 2022.

[4] M. P. Singh and A. Bhandari, "New-flow based DDoS attacks in SDN: Taxonomy, rationales, and research challenges," *Comput. Commun.*, vol. 154, pp. 509–527, 2020.

[5] S. M. Mousavi and M. St-Hilaire, "Early detection of DDoS attacks against SDN controllers," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Garden Grove, CA, USA. Los Alamitos, CA, USA: IEEE Comput. Soc. Press, Feb. 16–19, 2015, pp. 77–81.

[6] G. Schmid, "Thirty years of DNS insecurity: Current issues and perspectives," *IEEE Commun. Surv. Tut.*, vol. 23, no. 4, pp. 2429–2459, Fourth quart. 2021.

[7] S. Yu, J. Zhang, J. Liu, X. Zhang, Y. Li, and T. Xu, "A cooperative DDoS attack detection scheme based on entropy and ensemble learning in SDN," *EURASIP J. Wirel. Commun. Netw.*, vol. 2021, no. 1, p. 90, 2021.

[8] B. Isyaku, M. S. Mohd Zahid, M. Bte Kamat, K. Abu Bakar, and F. A. Ghaleb, "Software defined networking flow table management of openflow switches performance and security challenges: A survey," *Future Internet*, vol. 12, no. 9, p. 147, Aug. 2020.

[9] A. A. Alashhab, M. S. M. Zahid, M. A. Azim, M. Y. Daha, B. Isyaku, and S. Ali, "A survey of low rate DDoS detection techniques based on machine learning in software-defined networks," *Symmetry*, vol. 14, no. 8, p. 1563, Jul. 2022.

[10] J. D. Gadze, A. A. Bamfo-Asante, J. O. Agyemang, H. Nunoo-Mensah, and K. A.-B. Opare, "An investigation into the application of deep learning in the detection and mitigation of DDoS attack on SDN controllers," *Technologies*, vol. 9, no. 1, p. 14, Feb. 2021.

[11] H. H. Jazi, H. Gonzalez, N. Stakhanova, and A. A. Ghorbani, "Detecting HTTP-based application layer DoS attacks on web servers in the presence of sampling," *Comput. Netw.*, vol. 121, pp. 25–36, 2017.

[12] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy," in *Proc. Int. Carnahan Conf. Secur. Technol. (ICCST)*, G. L. Thomas and M. John, Eds., Chennai, India. Piscataway, NJ, USA: IEEE, Oct. 1–3, 2019, pp. 1–8.

[13] M. S. Elsayed, N. Le-Khac, and A. D. Jurcut, "InSDN: A novel SDN intrusion dataset," *IEEE Access*, vol. 8, pp. 165263–165284, 2020.

[14] N. M. Yungaicela-Naula, C. V. Rosales, and J. A. P. Díaz, "SDN-based architecture for transport and application layer DDoS attack detection by using machine and deep learning," *IEEE Access*, vol. 9, pp. 108495–108512, 2021.

[15] J. A. P. Díaz, I. A. Valdovinos, K. R. Choo, and D. Zhu, "A flexible SDN-based architecture for identifying and mitigating low-rate DDoS attacks using machine learning," *IEEE Access*, vol. 8, pp. 155859–155872, 2020.

[16] A. A. Alashhab et al., "Enhancing DDoS attack detection and mitigation in SDN using an ensemble online machine learning model," *IEEE Access*, vol. 12, pp. 51630–51649, 2024.

[17] T. G. Gebremeskel, K. A. Gemeda, T. G. Krishna, and P. J. Ramulu, "DDoS attack detection and classification using hybrid model for multicontroller SDN," *Wireless Commun. Mobile Comput.*, vol. 2023, pp. 1–18, Jun. 2023.

[18] M. A. Ribeiro, M. S. Pereira Fonseca, and J. de Santi, "Detecting and mitigating DDoS attacks with moving target defense approach based on automated flow classification in SDN networks," *Comput. Secur.*, vol. 134, Nov. 2023, Art. no. 103462.

[19] A. A. Alashhab, M. S. M. Zahid, A. Muneer, and M. Abdukkahi, "Low-rate DDoS attack detection using deep learning for SDN-enabled IoT networks," *Int. J. Adv. Comput. Sci. Appl.*, vol. 13, no. 11, pp. 371–377, 2022.

[20] P. Valizadeh and A. Taghinezhad-Niar, "DDoS attacks detection in multi-controller based software defined network," in *Proc. 8th Int. Conf. Web Res. (ICWR)*, Piscataway, NJ, USA: IEEE, May 2022, pp. 34–39.

[21] A. B. Dehkordi, M. Soltanaghaei, and F. Z. Boroujeni, "The DDoS attacks detection through machine learning and statistical methods in SDN," *J. Supercomput.*, vol. 77, no. 3, pp. 2383–2415, 2021.

[22] B. Nugraha and R. N. Murthy, "Deep learning-based slow DDoS attack detection in SDN-based networks," in *Proc. IEEE Conf. Netw. Funct. Virtualization Softw. Defined Netw. (NFV-SDN)*, Leganes, Madrid, Spain. Piscataway, NJ, USA: IEEE, Nov. 10–12, 2020, pp. 51–56.

[23] Y. Xu, H. Sun, F. Xiang, and Z. Sun, "Efficient DDoS detection based on K-FKNN in software defined networks," *IEEE Access*, vol. 7, pp. 160536–160545, 2019.

[24] V. Deepa, K. M. Sudar, and P. Deepalakshmi, "Design of ensemble learning methods for DDoS detection in SDN environment," in *Proc. Int. Conf. Vis. Towards Emerg. Trends Commun. Netw. (ViTECoN)*, Piscataway, NJ, USA: IEEE, Mar. 2019, pp. 1–6.

[25] B. Isyaku, K. B. A. Bakar, M. S. Ali, and M. N. Yusuf, "Performance comparison of machine learning classifiers for DDoS detection and mitigation on software defined networks," in *Proc. IEEE Int. Conf. Autom. Control Intell. Syst. (I2CACIS)*, Piscataway, NJ, USA: IEEE, Jun. 2023, pp. 69–74.

[26] M. S. Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "DDoSNET: A deep-learning model for detecting network attacks," in *Proc. IEEE 21st Int. Symp. "A World of Wireless, Mobile Multimedia Netw." (WoWMoM)*, Piscataway, NJ, USA: IEEE, Aug. 2020, pp. 391–396.

[27] N. Ahuja, G. Singal, and D. Mukhopadhyay, "DLSDN: Deep learning for DDoS attack detection in software defined networking," in *Proc. 11th Int. Conf. Cloud Comput., Data Sci. Eng. (Confluence)*, Piscataway, NJ, USA: IEEE, Jan. 2021, pp. 683–688.

[28] W. Sun, S. Guan, P. Wang, and Q. Wu, "A hybrid deep learning model based low-rate dos attack detection method for software defined network," *Trans. Emerg. Telecommun. Technol.*, vol. 33, no. 5, pp. 1–17, 2022.

[29] A. A. Alashhab, M. S. Mohd Zahid, M. Alashhab, and S. Alashhab, "Online machine learning approach to detect and mitigate low-rate

DDoS attacks in SDN-based networks," in *Proc. IEEE Int. Conf. Artif. Intell. Eng. Technol. (IICAIET)*, Piscataway, NJ, USA: IEEE, Sep. 2023, pp. 152–157.

[30] J. Konečný, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," 2016, *arXiv:1610.02527*.

[31] X. Ma, L. Liao, Z. Li, R. X. Lai, and M. Zhang, "Applying federated learning in software-defined networks: A survey," *Symmetry*, vol. 14, no. 2, p. 195, Jan. 2022.

[32] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016, *arXiv:1610.05492*.

[33] M. Duan, D. Liu, X. Chen, R. Liu, Y. Tan, and L. Liang, "Self-balancing federated learning with global imbalanced data in mobile systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 1, pp. 59–71, Jan. 2021.

[34] J. Zhang, C. Luo, M. Carpenter, and G. Min, "Federated learning for distributed IIoT intrusion detection using transfer approaches," *IEEE Trans. Ind. Inform.*, vol. 19, no. 7, pp. 8159–8169, Jul. 2023.

[35] D. J. Beutel et al., "Flower: A friendly federated learning research framework," 2020, *arXiv:2007.14390*.

[36] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, New York, NY, USA: ACM, Aug. 2016, pp. 785–794.

[37] A. Natekin and A. Knoll, "Gradient boosting machines, a tutorial," *Front. Neurorobot.*, vol. 7, pp. 1–21, 2013.

[38] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "*A Detailed Analysis of the CICIDS2017 Data Set*," in *Information Systems Security and Privacy*, P. Mori, S. Furnell, and O. Camp, Eds., Cham, Switzerland: Springer International Publishing, 2019, pp. 172–188.

Yannis Steve Nsuloun Fotse received the M.Sc. degree in computer science in 2022 from the University of Dschang, Cameroon, where he is working toward the Ph.D. degree in computer science. His ongoing research focuses on cybersecurity, artificial intelligence, and cloud computing.

Vianney Kengne Tchendji received the Ph.D. degree in computer science from the University of Picardie Jules Verne, Amiens, France, in 2014. He is currently an Associate Professor in computer science with the University of Dschang, Dschang, Cameroon. Since 2023, he has been a Research Fellow with the University of Limpopo, Mankweng, South Africa. He has co-authored more than 40 journal and conference papers in computer science. He has served as a Program Committee Member of international conferences and as a reviewer for many international computer science journals. His research interests include parallel algorithms and architectures, network virtualization, Internet of Things, ad hoc networking, and cyber security.

Mthulisi Velempini (Senior Member, IEEE) is an Active Researcher in medium access control protocols, routing protocols, and security in computers. He is an Emerging Researcher with Wireless Access Network Technologies.