








Research Article

Toward a Real-Time TCP SYN Flood DDoS Mitigation Using Adaptive Neuro-Fuzzy Classifier and SDN Assistance in Fog Computing

Radjaa Bensaid ¹, Nabila Labraoui ², Ado Adamou Abba Ari ^{3,4,5},
Leandros Maglaras ⁶, Hafida Saidi ¹, Ahmed Mahmoud Abdu Lwahhab ⁷,
and Sihem Benfriha ¹

¹STIC Laboratory, Abou Bekr Belkaid Tlemcen University, Tlemcen, Algeria

²LRI Laboratory, Abou Bekr Belkaid Tlemcen University, Tlemcen, Algeria

³DAVID Lab, Université Paris Saclay, University of Versailles Saint-Quentin-en-Yvelines, 45 Avenue des États-Unis, Versailles Cedex 78035, France

⁴Creative, Institute of Fine Arts and Innovation, University of Garoua, P.O. Box 346, Garoua, Cameroon

⁵LaRI Lab, University of Maroua, P. O. Box 814, Maroua, Cameroon

⁶School of Computing, Edinburgh Napier University, Edinburgh, UK

⁷Department of Electronics and Communications, Dakahlia Mansoura University, Mansoura, Egypt

Correspondence should be addressed to Leandros Maglaras; leandrosmag@gmail.com

Received 14 June 2023; Revised 12 January 2024; Accepted 14 February 2024; Published 23 February 2024

Academic Editor: Andrea Michienzi

Copyright © 2024 Radjaa Bensaid et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The growth of the Internet of Things (IoT) has recently impacted our daily lives in many ways. As a result, a massive volume of data are generated and need to be processed in a short period of time. Therefore, a combination of computing models such as cloud computing is necessary. The main disadvantage of the cloud platform is its high latency due to the centralized mainframe. Fortunately, a distributed paradigm known as fog computing has emerged to overcome this problem, offering cloud services with low latency and high-access bandwidth to support many IoT application scenarios. However, attacks against fog servers can take many forms, such as distributed denial of service (DDoS) attacks that severely affect the reliability and availability of fog services. To address these challenges, we propose mitigation of fog computing-based SYN Flood DDoS attacks using an adaptive neuro-fuzzy inference system (ANFIS) and software defined networking (SDN) assistance (FASA). The simulation results show that the FASA system outperforms other algorithms in terms of accuracy, precision, recall, and F1-score. This shows how crucial our system is for detecting and mitigating TCP-SYN floods and DDoS attacks.

1. Introduction

The growing number of connected objects, from millions to billions in various fields, is leading to an explosion in the amount of data. These huge volumes of data cause a lack of latency and make real-time analysis complex and difficult. To solve these issues, the deployment of computing models such as cloud and fog computing is crucial [1]. Technologies of cloud computing enable an extremely powerful computer resource over the network. Nevertheless, due to several concerns about

data privacy and security, attaching more diverse types of objects immediately to the cloud is extremely difficult, as are network latency difficulties [2, 3]. Therefore, the need to introduce a new paradigm is necessary to solve these problems.

Recently, fog computing has emerged to expand the cloud computing paradigm from the core to the network's periphery. The purpose of fog computing is to bring computer capabilities closer to IoT devices, offering real-time processing with low latency [4]. Aside from this, fog computing also provides mobility support, location awareness, and decentralized

infrastructure. Fog computing has a local data storage infrastructure, which makes it more secure than cloud computing. Despite this, IoT devices are limited in terms of storage capacity and battery life. Thus, they can be easily hacked, destroyed, or stolen, and fog computing may become unavailable and unable to handle normal user requests. Therefore, it is necessary to apply security mechanisms to identify and block unauthorized requests on network systems. However, fog computing is still susceptible to various security and privacy gaps. It can be a point of vulnerability, and it is easily overwhelmed by a massive number of malicious requests, primarily intended for distributed denial of service (DDoS) attacks [5]. DDoS attacks can be divided into two types, depending on the protocol level addressed. The first one is known as “network-level flooding,” when TCP, UDP, ICMP, and DNS packets are used to overload intended clients’ network capabilities and resources. Whereas, the second protocol level is referred to as “application-level DDoS flooding” which is typically done on an HTTP web page when attacks are launched to deplete server resources such as sockets, CPU, ports, memory, databases, and input/output bandwidth [6]. Regarding the rapid growth and the harm caused by DDoS attacks, several kinds of research have been conducted on these attacks, and various approaches have been presented in the literature to prevent these attacks using fog computing [7, 8]. Most of them proposed a defensive fog computing that operates as a filtering layer between the user layer and the cloud computing layer. However, these defensive approaches miss DDoS detection mechanisms, and detailed computation is not discussed. Also, they do not identify any infrastructure to protect fog computing which is particularly susceptible to DDoS attacks and may disrupt network services.

For this purpose, proposing software defined networking (SDN) technology-based solutions could provide an innovative framework to deal efficiently with this insidious attack. SDN enables us to define logic control and instructs the forwarding plane to act appropriately by isolating the control and data planes. This programmability provides more control over network traffic, which wasn’t conceivable before the development of SDN [9].

Considerable research has been done within SDN-based IoT-fog networks using task scheduling techniques like threshold random walk with credit-based connection (TRWCB) and rate limiting. These techniques are deployed for detecting anomalies and mitigating DDoS attacks, which effectively reduces average response times [10]. However, it can result in excessive CPU and RAM consumption. This scheduling-based approach only focuses on secure scheduling periods, leaving the network vulnerable during idle times when no tasks are scheduled [11]. Moreover, the previous approach incorporates both fuzzy logic and multiobjective particle swarm optimization. Nevertheless, as the number of variables and rules increases, designing and fine-tuning the fuzzy logic system can become highly complex.

Recently, machine and deep learning algorithms have gained attention for their effectiveness in detecting DDoS attacks by analyzing data patterns [12, 13]. Hence, merging fuzzy systems with neural networks combines the benefits of

neural learning with the interpretability offered by fuzzy systems. An adaptive neuro-fuzzy inference system (ANFIS) empowers fuzzy systems to acquire knowledge from data. This synergy enhances fuzzy systems through neural networks. ANFIS’s hybrid approach facilitates adaptability to diverse attack patterns and network conditions.

Moreover, employing various approaches, such as the neuro-fuzzy classifier on the KDD CUP99 dataset [14], has been a common practice. This dataset includes numerous recognized attack variations and has traditionally been utilized in intrusion detection. Nevertheless, the KDD CUP99 dataset is now regarded as outdated, as it presents several unresolved issues that fail to meet the updated criteria for DDoS identification [15]. In our work, we focus exclusively on the TCP-SYN flood attack. Since it is the most effective DDoS attack in fog computing, in order to exhaust the system’s resources or overwhelm the target server, the attackers typically infect several devices that behave as bots and synchronize suspicious traffic or requests, leading to an incomplete three-way handshake procedure [16]. Consequently, legitimate users cannot reach the desired fog server. In this paper, we suggest a novel fog computing-based SYN Flood DDoS attack detection and mitigation using an adaptive neuro-fuzzy inference system (ANFIS) and SDN assistance (FASA). Compared to previous works, FASA utilizes the ANFIS model for network traffic classification, incorporates SDN support to enable real-time mitigation, and relies on the newly released CIC-DDoS2019 dataset. The proposed model demonstrates exceptional performance across multiple metrics, including accuracy, precision, recall, and F1-score. In addition, it exhibits a notably low rate of false positives. In brief, our significant contributions are outlined as follows:

- (1) We propose a novel model FASA to detect and mitigate a SYN flood DDoS attack in fog computing using SDN assistance.
- (2) We implement the ANFIS model to self-train the fog servers and make the difference between normal and malicious packets.
- (3) The ANFIS model is implemented at the SDN controller and deployed at the fog server using a dataset captured from the SDN environment. Its main objective is to allow benign packets access while rejecting malicious ones to release a secure and dependable SDN controller that ensures fog service availability.
- (4) The proposed evaluation method uses both the newly released dataset CIC-DDoS2019 and the SDN dataset. It is experimentally analyzed from the data availability and the algorithm operating efficiency and it can improve the performance

The paper is structured in the following manner: Section 2 examines and discusses previous work to tackle the issue of DDoS attacks. Section 3 contains background knowledge. In Section 4, we formally define the proposed model, and in Section 5, we introduce our proposed framework followed by the evaluation outcomes and discussion in Section 6. Finally, the conclusion is given in Section 6.

2. Related Work

In this section, we have provided an extensive overview of DDoS attacks. Especially, TCP-SYN flood attack detection. Besides, these works have been grouped into three sections. The initials represent statistical methods. The second and third ones highlight a few works based on machine/deep learning (ML/DL) algorithms.

2.1. Statistical Methods. Statistical methods constantly evaluate user/network activities to identify abnormalities [17]. Hence, due to their capacity to analyze the behavior of data packets, they are commonly utilized in DDoS attack detection systems. If the data flow does not match with some test statistics and measures, it is thought to be illegal. Ahalawat et al. [18] suggested a detection method for DDoS attacks based on Renyi entropy and a mitigation solution for SDN based on the packet drop approach, using several probability distributions. They can examine network traffic fluctuations. However, the necessity of setting an optimal detection threshold is a typical limitation of various entropy-based approaches. Hoque et al. [19] presented a novel correlation measure using standard deviation and mean to detect DDoS attacks, the traffic is then classified as attack traffic or normal by comparing the collected traffic to the profiled traffic. However, the suggested metric's use in identifying low-rate attempts is unclear. A DDoS detection-based multivariate correlation analysis was discussed by Jin and Yeung [20] in their work, and they provided a covariance analysis method for recognizing SYN flood attacks. The experimental results demonstrate that this technology accurately and efficiently detects DDoS attack traffic in networks of varied levels of intensity. However, using the correlation approaches consumes a lot of processing in real-time to detect DDoS attacks. As a result, they are unable to operate in real-time. A novel framework was suggested by Bhushan [7] using fog for detecting DDoS attacks even before they reach the cloud by using an efficient resource provisioning algorithm to service cloud requests through intermediate fog servers. Furthermore, an entropy DDoS detection method and mitigation system designed for cloud computing environment using SDN have been proposed by Tsai et al. [21]. An entropy-based DDoS detection approach was implemented to protect the virtual machines and controller from malicious attacks. As a result, the detection rate is significantly affected by the threshold value. Javanmardi et al. [10] proposed FUPE, a security-driven task scheduling algorithm for SDN-based IoT-Fog networks. FUPE uses fuzzy logic and multiobjective particle swarm optimization to assign tasks to fog nodes, balancing security and efficiency objectives. However, managing and interpreting extensive rule sets poses challenges to maintaining and validating the fuzzy logic framework. Nonetheless, multiobjective optimization with PSO requires parameter tuning and could be computationally intensive, particularly in large-scale environments. Furthermore, it incorporates techniques like threshold random walk with credit-based connection (TRWCB) and rate limiting to detect malicious nodes and utilizes the SDN controller for mitigation by blocking attackers, ultimately leading to a reduction in average response times. Nevertheless, this approach

may lead to elevated CPU and RAM usage. FUPE exclusively identifies and addresses anomalies during the scheduling phase, leaving the network susceptible to threats in the absence of scheduling requests [11].

2.2. Machine Learning Methods (ML). Machine learning-based methods are used to identify DDoS attacks such as decision trees, deep learning, support vector machine (SVM), K-means clustering, and so on [22]. These methods might be unsupervised machine learning (a label for training is not required) or supervised machine learning (a label for training normal/malicious) algorithms. Moreover, the dataset, which contains numerous network and traffic features, is used to train and learn automatically how to recognize suspicious behavior patterns. Rajagopal et al. in [23] provided a meta-classification strategy that integrates many classifiers for both binary and multiclass classification. The decision jungle serves as the meta learner, combining numerous learners to obtain the best prediction performance. This proposed method has a precision of 99%. Tuan et al.'s [24] idea was about proposing a novel TCP-SYN flood attack mitigation by tracing back IP sources of attack in SDN networks using K-nearest neighbors (KNN) machine learning based on SDN. The testbed's experimental findings reveal that 97% of attack flows are identified and blocked. Priyadarshini et al. [25] demonstrated a new source-based DDoS mitigation approach in order to prevent these attacks in both fog and cloud computing environments. It deploys the defender module that presents at the SDN controller which is based on machine learning (SVM, KNN, and Naive Bayes algorithm). However, the classical ML techniques cannot handle the amount of data. However, the "real world" application of classical ML algorithms is limited due to network attack issues. In addition, these approaches need a lot of time to learn, so they cannot be used in real-time.

2.3. Deep Learning (DL) Methods. In recent studies, there has been a particular emphasis on evaluating the performance of DL models in DDoS detection. This is primarily due to their ability to effectively analyze large volumes of data and identify complex patterns within it. de Assis et al. [26] proposed a near-real-time solution by applying convolutional neural networks (CNN) to cover and defend victims' servers from DDoS attacks at the end source, the detection model reached a precision rate above 95.4%. Novaes et al. [27] employed the generative adversarial network (GAN) architecture to mitigate the damage of DDoS attacks on SDNs. For experiment assessments, the accuracy obtained using the published datasets, namely, CIC-DDoS2019, and the emulation was about 94.38%. The authors compared the GAN framework's findings against those of other deep learning algorithms, such as LSTM, CNN, and MLP. The authors of [28] employed a variety of machine learning (ML) algorithms to identify low-rate DDoS attacks. They found that the multilayer perceptron (MLP) performs the best among the assessed algorithms, with a detection rate of up to 95%. Other ML models, such as random tree, random forest,

and support vector machines, have shown useful in detecting and mitigating DDoS attacks. Deep learning has already been used to identify SYN flood attacks by Brun et al. [29], in which a random neural network was built to classify and differentiate whether the packet is normal traffic or SYN attacks. Evmorfos et al. [30] use a random neural network for identifying typical SYN attacks on Internet-connected equipment including edge devices and gateways, and fog servers, with limited processing capability. Devi et al. [31] presented an intrusion detection system (IDS) approach based on the SUGENO-based fuzzy inference system ANFIS to identify security concerns on relay nodes in a 5 G wireless network. The model was tested and trained using the KDD Cup 99 datasets. Boroujerdi and Ayat [12] developed a novel ensemble of Sugeno-type adaptive neuro-fuzzy classifiers to identify DDoS attacks based on the Marliboost boosting approach. It was tested on the NSL-KDD dataset. However, the data in the NSL-KDD or KDD Cup 99 datasets were considered unsuitable for the new requirement of a DDoS attack since they comprise packet traces rather than flows, implying that the DDoS detection methods may become computationally difficult as the network expands in size. As a consequence, there have been various studies published in recent years on how to identify DDoS attacks, particularly TCP SYN flood using machine and deep learning. However, few of them have addressed using ANFIS to detect such attacks in fog computing based on SDN technology.

In order to address the limitations of the previous studies, in this paper, we propose an ANFIS classifier, implemented in the SDN controller to classify network traffic and deployed at the fog server using the recently published dataset CICDDoS2019. The inclusion of various types of DDoS attacks in this dataset bridges the gaps found in previous databases. In addition, we employ ANFIS using the SDN dataset for real-time mitigation.

3. Background Knowledge

This section highlights the required context for our proposed model. First, we give an overview of DDoS attacks and the different methods used for detection. Then, we introduce the Adaptive Network-based Fuzzy Inference System (ANFIS) detection algorithm. Finally, we present the Software Defined Networking (SDN) technology.

3.1. DDoS Attacks and Fog Computing. The DDoS attack is a highly progressed type of DoS attack. It differs from other attacks in that it may be deployed in a “distributed” manner. A DDoS attack’s primary purpose is to inflict harm on a target for personal reasons, financial gain, or popularity [1]. It is an attack based on availability and aims to make the victim system inaccessible to authorized users [32]. Moreover, it is done by a combination of a huge amount of hacked and dispersed devices known as bots or zombie devices that have been infected with malicious malware or compromised by an attacker [33]. Hence, an attacker centrally controls and coordinates these machines to launch an attack on the target machine [34].

3.1.1. Types of DDoS Attacks on Fog Computing. Several DDoS attack types are used to bring down the functionality or availability of network services on fog computing [32], as illustrated in Figure 1.

(1) *Application-Bug Level DDoS.* These sorts of attacks, like HTTP POST and HTTP PRAGMA, deplete the application system, causing it to fail or temporarily close down.

(2) *Infrastructural Level DDoS.* The key purpose of these threats is to exhaust network bandwidth, buffers, CPU, and storage, preventing legitimate users from using them. Thus, the only requirement for this attack is the victim’s IP address. It is categorized into two types: direct and reflector attacks.

(i) *Direct Attack*

This attack is carried out with the assistance of compromised devices or bots. It sends malicious queries to the target using bots in order to deplete its resources, bandwidth, and services, rendering them inaccessible to authenticated users. This attack can be further subdivided into network-layer and application-layer DDoS attacks.

- (a) Network Layer DDoS: This attack type employs various network and transport layer protocols, including TCP SYN, UDP, and ICMP, among others.
- (b) Application Layer DDoS: In this attack, HTTP flood traffic is adopted widely to exhaust the victim. This kind of vulnerability is difficult to detect, raising security issues.

(ii) *Reflector Attack*

In this attack, the IP address is spoofed and requests are delivered to a vast range of reflector hosts. Following the receipt of the requests, a response is provided in order to flood the target.

3.1.2. DDoS Defense Mechanisms. In this section, we discuss various defense mechanisms used for DDoS attack detection and mitigation for the security of fog computing [35]. Nearer-to-edge devices, fog computing, offers computing capabilities in the form of fog nodes which creates a heavy load on network management. To address this issue, SDN technology can be implemented to guarantee the safety of fog computing in the following aspects:

- (i) Monitoring the network: If the network is monitored permanently and continuously, any suspicious data attempting to disrupt services may be recognized and rejected. As this is performed at fog nodes, legitimate users will have no difficulty accessing the services.
- (ii) Priority-based and isolated traffic: It implies the process of prioritizing legal and illegitimate network traffic, hence requiring the use of shared knowledge resources such as CPU or I/O. As a result, SDN can

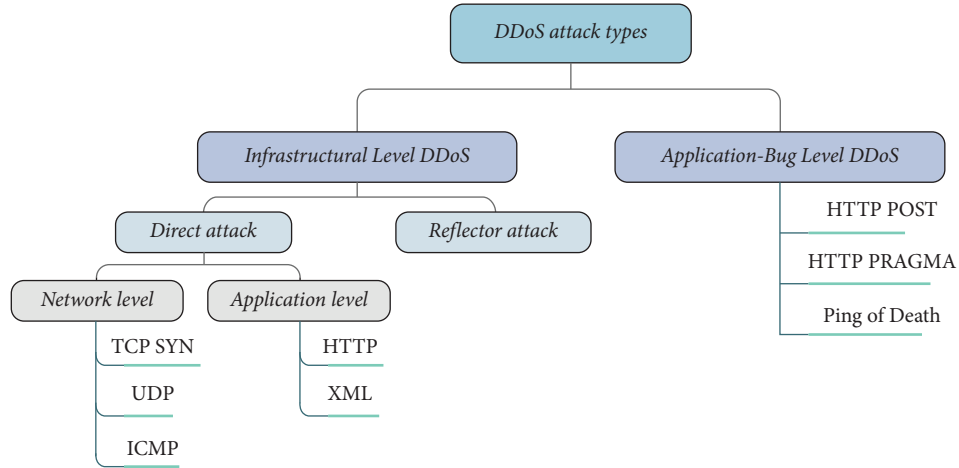


FIGURE 1: Types of DDoS attack in fog computing.

reject damaging traffic by separating it through a VLAN ID/tag.

- (iii) Access control mechanism for resources in the network: To prevent DDoS attacks, an effective access control system should be implemented.
- (iv) Shared network: The shared network is the crucial condition since anyone can access it, holding security at risk.

In addition, two distinct assessments are used to identify DDoS defense mechanisms. The first classification divides the DDoS defense systems into the following four groups based on the activity carried out:

- (i) Intrusion prevention,
- (ii) Intrusion detection,
- (iii) Intrusion tolerance and mitigation,
- (iv) Intrusion response.

Further, the second categorization mainly classify DDoS defenses into the following three groups based on where they are deployed:

- (i) Victim network,
- (ii) Intermediate network,
- (iii) Source network.

3.1.3. TCP-SYN Flood Attack. The SYN flooding attack is a specific type of DoS attack that targets hosts that operate TCP server processes. It became well-known in 1996 [36]. The concept of the three-way handshake that initiates a TCP connection serves as the mainstay of this attack [37]. It exploits a TCP protocol process characteristic and may be used to restrict server functions from responding to normal user demand to establish new TCP connections. As a result, each service that connects and waits on a TCP socket is highly susceptible to TCP SYN flood attacks. Although several techniques to counteract SYN flood attacks may be found in modern operating systems and equipment.

3.2. The Adaptive Network-Based Fuzzy Inference System (ANFIS) Detection Algorithm. ANFIS is a network model that combines a Sugeno-type fuzzy system with neural learning capability [38]. Neuro-fuzzy systems are ways to learn fuzzy systems from data that use neural network-derived learning algorithms. Therefore, due to their learning capabilities, neural networks are an ideal choice for combining with fuzzy systems [39], which are used to automate or simplify the process of developing a fuzzy system for specific usage. The initial neuro-fuzzy techniques were primarily explored within the field of neuro-fuzzy control, although the approach is now broader because it is used in a number of domains, including control, data analysis, decision support, and so on [40]. ANFIS is based on two parameters (premise and consequent parameters) which are used to connect the fuzzy rules. Moreover, ANFIS is made up of five layers in total, as illustrated in Figure 2. The square nodes have parameters, whereas the circular nodes do not.

The considered fuzzy inference system contains two inputs, y considered as nonlinear parameters, and one output f . Also, each input variable is described by two linguistic terms: A_1 and A_2 for the variable x , and B_1 and B_2 for the variable y , respectively.

The following two *IF-THEN* rules construct the Sugeno fuzzy model [40]:

- (i) Rule 1: If x is $A_1 \wedge y$ is B_1 , then $f_1 = p_1x + q_1y + r_1$
- (ii) Rule 2: If x is $A_2 \wedge y$ is B_2 , then $f_2 = p_2x + q_2y + r_2$

Where p_i , q_i , and r_i , $i=1, 2$, correspond to the linear parameters of the conclusion part to be adjusted during the training.

- (i) Layer 1: O1 represents the membership function μ of a fuzzy set A_i (or B_i). $O_{1,i} = \mu_{A_i}(x) \quad i = 1, 2$
 $O_{1,i} = \mu_{B_{i-2}}(y) \quad i = 3, 4$

Because of their smoothness and simple syntax, Gaussian membership functions are preferred approaches for defining fuzzy sets. The advantage of these curves is that they are smooth and nonzero at all locations. The Gaussian membership function is

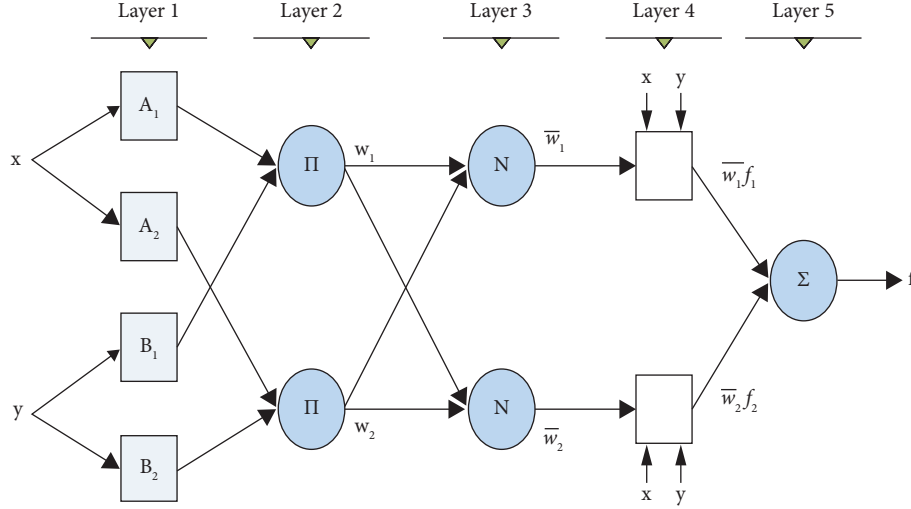


FIGURE 2: ANFIS architecture layers.

used in this study; it is frequently used to reduce the uncertainty of real-world measurement and is represented by the equation (1) where c and σ represent the mean and standard deviation respectively. Here, c represents the center, and σ represents the width. a , c are called premise parameters (nonlinear).

$$\mu_A(x) = ae^{-(x-c)^2/(2\sigma)^2}. \quad (1)$$

- (ii) Layer 2: the fuzzification layer w determines the degree of membership function satisfaction of each input; the output is the product Π of all the entering signals; it is determined using the following equation:

$$O_{(2,i)} = w_i = \mu_A(x) \cdot \mu_B(x), \quad i = 1, 2. \quad (2)$$

The output of every node shows the firing strength of a rule. The node function in this layer can be any other fuzzy AND T-norm operator, such as min.

- (iii) Layer 3: the normalization layer, in which the i -th node determines the proportion of the firing strength of the i -th rule to the total firing strength of all rules, as demonstrated in the equation:

$$O_{3,i} = \bar{w}_i = \frac{w_i}{w_1 + w_2}. \quad (3)$$

The outputs of this layer are referred to as normalized firing strengths.

- (iv) Layer 4: In the defuzzification layer, parameters are named consequent parameters. Each node has a function where \bar{w}_i is a normalized firing strength from layer 3 and p_i, q_i, r_i are the set of linear node parameters and are defined as consequent parameters of this node and f_i denotes the output of the rule, as shown in the following equation:

$$O_{4,i} = \bar{w}_i f_i = \bar{w}_i (p_i x + q_i y + r_i). \quad (4)$$

- (v) Layer 5: in this layer, the single node adds up all of the incoming signals to compute the overall output, as demonstrated in equation:

$$O_{5,i} = \text{overalloutput} = \sum_{i=0}^n \bar{w}_i f_i = \frac{\sum w_i f_i}{\sum w_i}. \quad (5)$$

An adaptive network's nodes are related to parameters that may affect the final output. To adapt the parameters in an adaptive network, ANFIS typically uses a hybrid learning algorithm that associates gradient descent and the least squares approach [41]. The hybrid algorithm comprises a forward pass and a backward pass. To optimize the consequent parameters, the least squares method (forward pass) is used; node outputs are passed forward until Layer 4, and the least squares determine the consequent parameters. In our work, for optimizing the premise parameters, the ADAM method [42] is employed during the backward pass. Error signals are propagated backward, and the premise parameters are updated using ADAM. This hybrid learning approach offers faster convergence by reducing the search space dimensions compared to the original backpropagation method [31]. It has been demonstrated that this hybrid algorithm is extremely effective in training ANFIS systems [43]. The ANFIS training technique begins by defining the number of fuzzy sets, the number of sets of each input variable, as well as the shape of their membership function. The primary goal of ANFIS is to improve input-output data sets and a learning mechanism to enhance the parameters of a comparable fuzzy logic system. The difference between the intended and actual outputs is minimized as much as feasible during parameter optimization.

3.3. Software-Defined Networking (SDN). SDN is a network paradigm that enables users to directly manage network resources by orchestrating, controlling, and using software applications [44]. Moreover, the control and data planes are divided by SDN, making it most commonly used to improve

network efficiency. When the data plane forwards packets from one location to another, the control plane determines whether or not the packets should propagate through the network. Thus, SDN is formed by the combination of a controller and switches; these switches follow the forwarding rules that are defined by the controller, which can dynamically manage network flows and implement different configurations based on network circumstances. The three fundamental layers of SDN architecture are as follows: (i) The application layer which contains the general network functions including intrusion detection systems, firewalls, and security applications. (ii) The control layer which is the centralized software controller that serves as the SDN's brain. The network policies and traffic flows are managed by this controller. (iii) The infrastructure layer contains a variety of networking equipment, including switches and routers [45], as shown in Figure 3. The communication between the controllers and switches is outlined throughout the OpenFlow protocol [46], which serves as the communication standard for SDN networks. It is referred to as SDN networks' southbound communication. The controller can deal with open flow switches (OF-switch) with existing flow tables by using an open flow protocol. When a packet's flow entry is found in the OF-switch's table, the packet is forwarded in the usual manner; otherwise, the controller receives it for additional evaluation. Thus, SDN controllers with OpenFlow-enabled switches are widely used for SDN networking. They are especially suitable for light traffic communication and control.

3.4. System Model. This study aims to develop a distributed FASA framework to mitigate SYN flood attacks in the network environment by recognizing and avoiding attacks close to the attacking sources. To enable quicker and more accurate attack detection using the ANFIS model, fog computing is suitable for deploying SDN for mitigating SYN flood attacks by assigning compute power near the operation process and spreading the burden in the system through a FASA mitigation scheme. In this section, we first outline SYN flood DDoS attacks in fog computing. Then, we discuss the FASA network architecture.

3.5. SYN Flood DDoS Attack. As shown in Figure 4, when a standard TCP three-way handshake has been initiated, the end user (EU) transmits the SYN packet to the fog server. Then, the fog server responds with a SYN/ACK packet. Next, the EU should send an ACK packet to the fog server. So, when all of these processes are completed, the connection is established [47]. However, the main drawback of TCP connections is the inability to maintain half-open connections. The fog server is in a half-open connection state because it is standing in line for the EU's reply to acknowledge the three-way handshake. Furthermore, IoT devices have limited computation, storage capacity, and short battery life, and they can easily be compromised, damaged, or kidnapped. Therefore, due to the aforementioned limitations, an attacker may simply hack IoT devices and utilize them as botnets to generate and send excessive SYN request packets with a fake source IP address to fog

servers. As a result, the ACK packet will never reach the fog server which is in the open port state waiting for the ACK packet. Moreover, the SYN/ACK packets are transmitted to the faked host, and the three-way handshake procedure will never be completed. Also, the connection registration is kept in the connection delay buffer till time expires, preventing legitimate users from accessing the services [48].

3.6. FASA Network Architecture. To effectively deal with the SYN flood DDoS attack concerns in the network systems, attack prevention must be built into fog computing based on SDN. Indeed, in this paper, we propose a novel distributed fog defensive system for SYN flood DDoS attacks using ANFIS and SDN Assistance (FASA). The FASA architecture has three layers, the cloud layer, the SDN-based fog (SDFN) layer, and the things layer, as shown in Figure 5.

3.6.1. Cloud Layer. Cloud computing, as a computing model, defines a method of managing a pool of configurable computing resources, offers elastic, on-demand services, and has access to the system anywhere and at any time. Therefore, users can use resources according to their demands. The salient features provided by cloud technology are immediate flexibility and measurable services [1]. SDN and cloud technology can be combined to automate, and cloud application provisioning must be completely integrated with the network. Hence, in the FASA system, cloud computing refers to the application plane which consists of many useful applications that communicate with the controller to abstract a logically centralized controller to make coordinated decisions.

3.6.2. SDN-Based Fog Network (SDFN) Layer. This layer combines the fog computing and SDN paradigm to identify and respond to DDoS attacks. With recent advances in SDN, it opens up new opportunities for providing intelligence within networks. The benefits of SDN, including logically centralized control, software-based traffic analysis, an entire network view, and flexible forwarding rule updates, help to improve and facilitate machine learning applications [49]. Therefore, the SDFN layer provides new trends in DDoS attacks in fog computing environments using SDN. This layer is formed of two sublayers, SDFN-server and SDFN-node.

- (i) SDFN-server: This sublayer refers to the control plane deployed at fog servers where an intelligent ANFIS classifier is integrated into the control network to classify traffic flow decision and consequently, policies are managed to depend on its decisions. Moreover, the SDFN server communicates with the cloud layer (application) via the northbound interface and with the SDFN-node layer via the southbound interface.
- (ii) SDFN-node: This sublayer refers to the data plane of physical equipment in the network such as switches and routers. It forwards the network traffic to its destinations using the OpenFlow protocol.

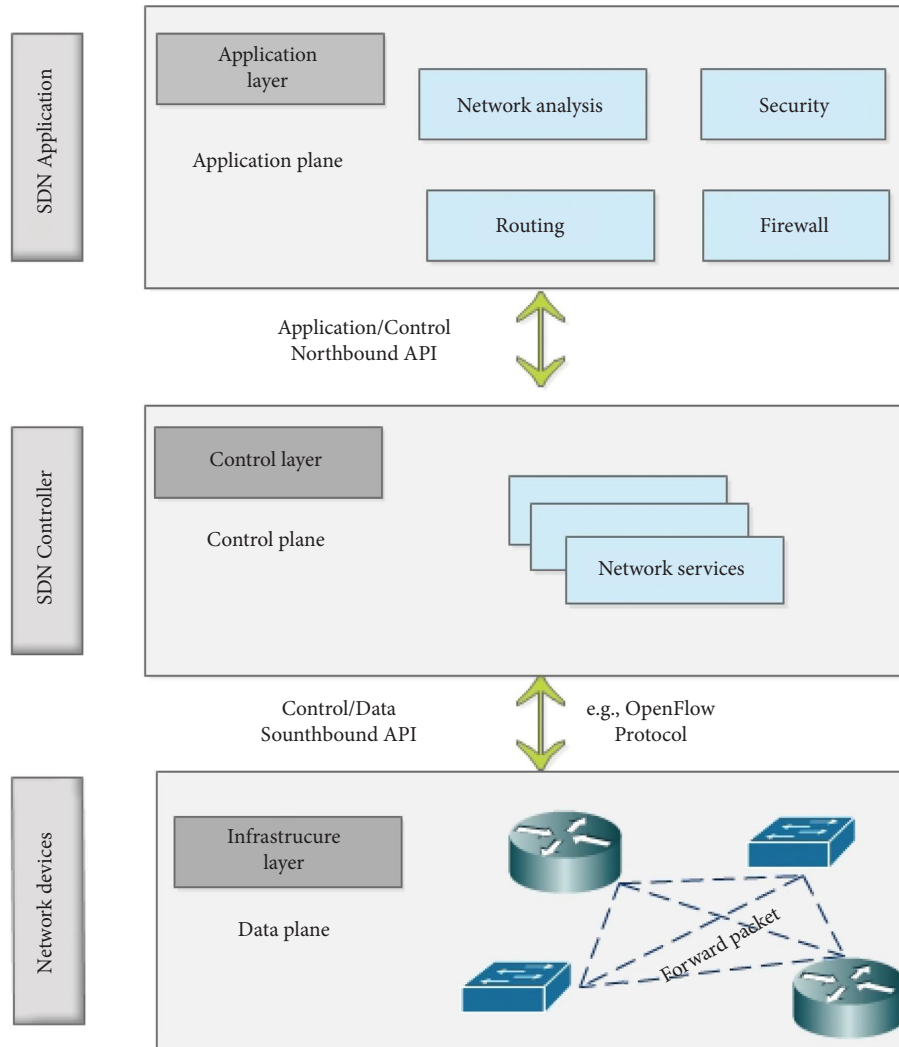


FIGURE 3: Software defined networking.

3.6.3. Things Layer. This layer serves the purpose of sensing, collecting, and uploading data from wireless sensors and end-users to fog computing. The transmitted packet can be classified as either benign or malicious.

The following assumptions are made in order to better explain the SYN flood DDoS attack identification and defense framework:

- (i) The SDN-based Fog Network server (SDFN-server) is susceptible to being compromised
- (ii) DDoS attacks are TCP SYN flood attacks against SDFN servers.
- (iii) The SDN controller and the switch are not compromised.
- (iv) IoT devices can be hacked.

4. Proposed FASA Framework

SYN flood DDoS attacks can instantly bring down a network, and it is difficult to detect them since they can be carried out in a very short time. Therefore, detecting and

mitigating such attacks is critical. A detection approach for such threats is needed in fog computing to filter and block the malicious requests before the attack produces a negative impact on the fog services. Consequently, our FASA framework can be used to identify and immediately mitigate SYN flood attacks in real-time through fog computing, as illustrated in Figure 6.

4.1. The detection Process. FASA is based on the ANFIS model and the SDN network to guarantee service availability in the fog network. To attain recognition and detection purposes, a fog layer is established among both the cloud layer and the things layer. Thus, the recognition techniques deployed on the fog layer can handle and process malicious traffic. Also, the SDN controller deployed on the fog layer controls packets arriving from every system node to enhance security and network management. In addition, the SDFN-server is prior trained with ANFIS algorithms and tested using two different datasets, CIC-DDoS2019, and SDN dataset. After a successful data preprocessing step, the most important features will be extracted. Then, these features will

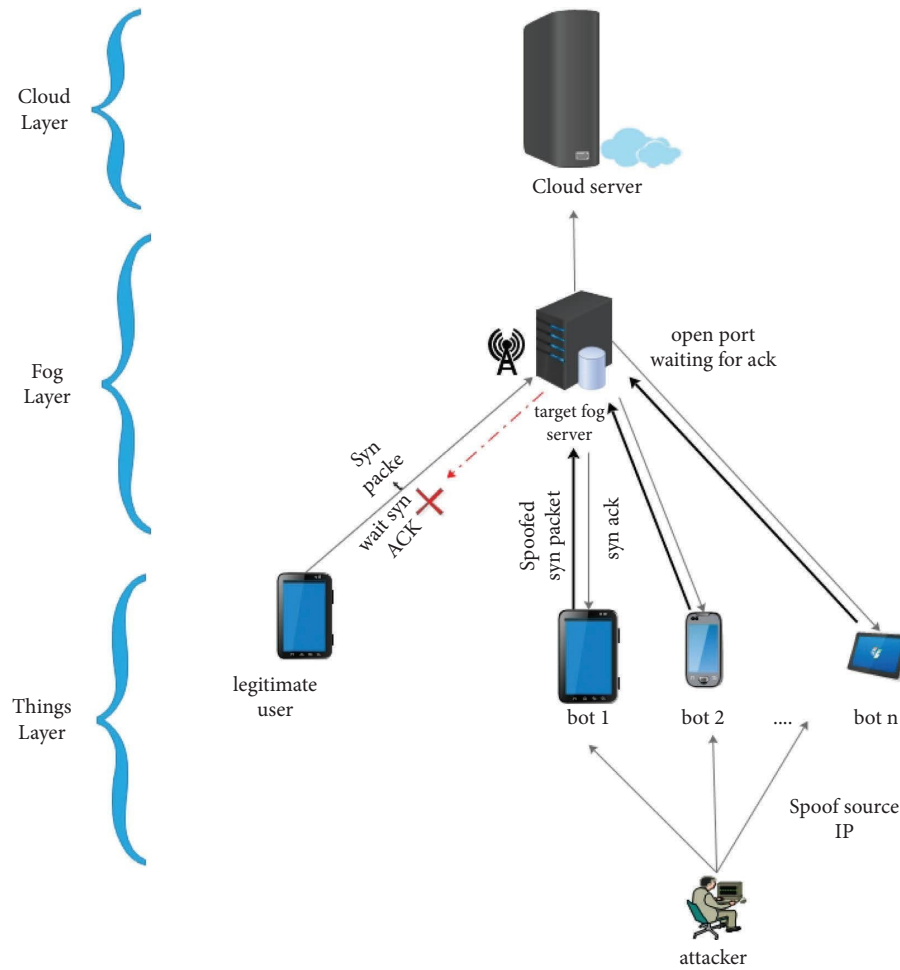


FIGURE 4: SYN flood DDoS attack in fog computing.

be divided into training data and testing data to self-train the SDFN-server to identify the SYN flood attack. Once that is done, the ANFIS model will be able to determine whether an incoming packet is legitimate or not. then, the controller’s decision is based on that, as presented in the flowchart of Figure 7.

4.2. *The Mitigation Process.* SDN simplifies the implementation of complex mitigation models. When an OpenFlow switch gets a packet, it compares it to the matching rule in its flow table and decides whether to act by forwarding packets to the destination according to the found rule or seek assistance from the controller if the rule is not matched. In addition, the OpenFlow switch initiates this request through the SB-API of the OpenFlow agent in the switch, as demonstrated in the flowchart in Figure 7. Although, the attack may be identified by determining a threshold value, which is the maximum value of serving capacity defined by the availability of computational resources. If the number of service requests exceeds the limit, a malicious packet is sent out [50]. Otherwise, if it is less than the threshold capacity, it will pass through the ANFIS classifier for prediction on the fog server. Therefore, the real-time mitigation phase starts when

the ANFIS model detects an SYN flood packet. This phase aims to perform defensive functions to limit the damage caused by an exploit. So, the packet passes through the OpenFlow protocol which takes action by executing the updated rule in the flow table whether it is a legitimate user to allow access. Otherwise, the controller looks for the most often occurring source address Mac with different source address IPs and uses it to determine the infected port number. By correlating the identified Mac address with the corresponding port on the switch, the controller determines the port through which the attack traffic is entering the network. To prevent further damage, the controller instructs the OF switches to drop all packets obtained from the host associated with the identified Mac address. Then, the controller also directs the switch to block traffic on the specific port associated with the infected host, effectively preventing any communication through that port. Next, the controller updates the flow table of the switch to modify the rules related to receiving or forwarding packets to the identified port. This ensures that any packets destined for that port are dropped or redirected to mitigate the attack.

As a result, TCP SYN flooding attacks may be identified and prevented by instantly blocking the switch port that is connected directly to the attacker’s host (see Algorithm 1).

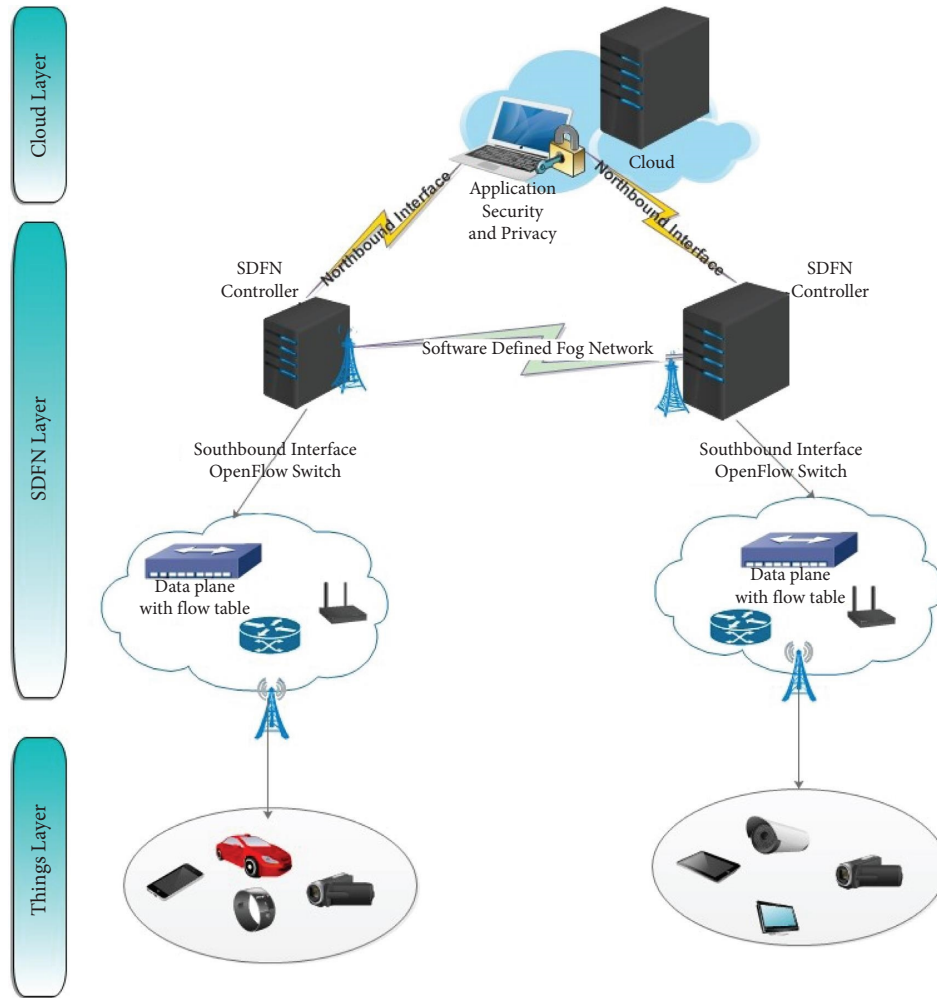


FIGURE 5: Network model: SDN-based fog network (SDFN).

5. Experiments and Results

5.1. Experimental Setup. In this part, we will go over the various tools that were used to build up the experimental setup for detecting SYN flood attacks in the simulated SDN and fog computing environments, using Wireshark [51] to capture and analyze network traffic in real-time. The entire experiment is carried out on the Windows 10 OS with an Intel i3 processor and 8 GB of RAM. To emulate the network behavior, the SDN Mininet network emulator [52] was used with the Ryu controller [53]. Ryu is an open-source platform that provides transparency and flexibility, enabling customization and extension of functionalities. Its Python-based architecture promotes accessibility and ease of development, facilitating rapid implementation of SDN applications. In addition, support for multiple protocols, including OpenFlow, ensures seamless communication with diverse network devices. Ryu's compatibility with various networking technologies and hardware makes it suitable for heterogeneous infrastructures, rendering it particularly well-suited for this research [54].

For training and testing our ANFIS model, the Python programming language has been used with libraries for deep learning such as Keras [55], and TensorFlow [56]. In addition, to prevent overfitting, the stratified K-Fold cross-validation [57] was also employed in the ANFIS algorithm. Due to the fact that Stratified k-fold cross-validation guarantees that each fold has a class distribution that is identical to the original dataset, resulting in a more accurate and reliable model assessment, along with binary crossentropy, a classic loss function used in binary classification. Also, we set the default Keras learning rate to 0.001. Furthermore, Adam optimizer [42] was selected as an adaptive algorithm for optimizing learning rates in neural network models. Moreover, by using two different scenarios in this study, we examine the performance and efficiency of the FASA system.

- (i) Scenario 1: Evaluate the performance of the FASA system by employing the SDN environment.
- (ii) Scenario 2: Evaluate the performance of the FASA system by using the public dataset CIC-DDoS 2019 [58].

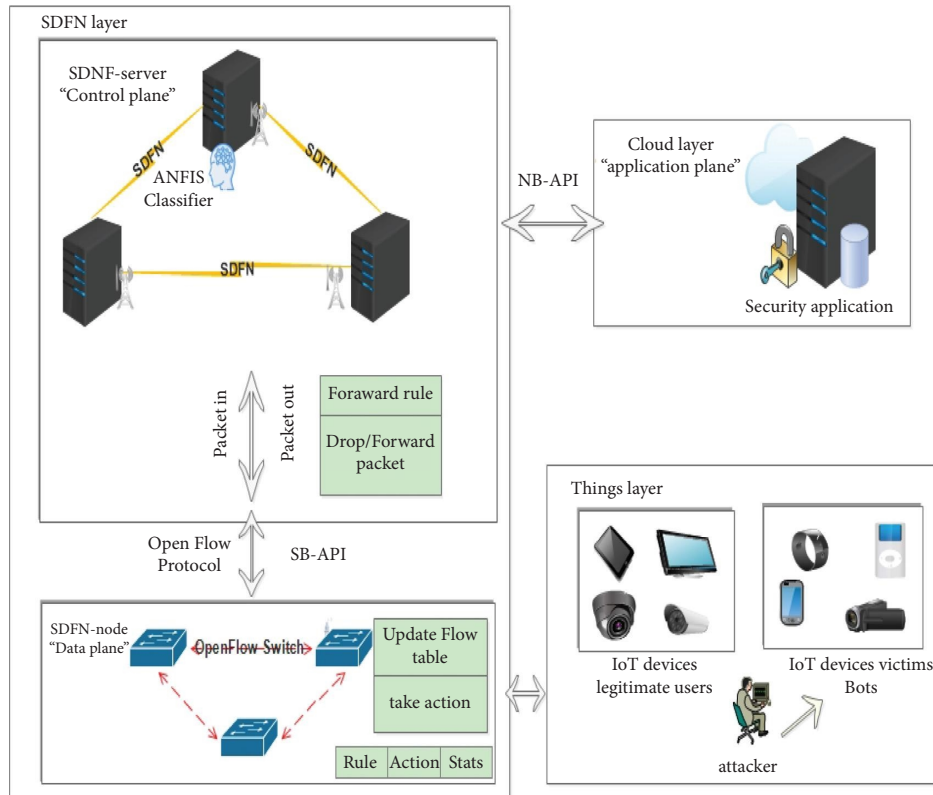


FIGURE 6: The proposed FASA framework.

5.2. *Experimental Analysis.* In our next subsection, we discuss each test scenario and provide the studies' results.

5.2.1. *Scenario 1.* In our experiment, the Mininet network emulator [48] was used to design virtual network topologies consisting of controllers, hosts, links, and switches. Therefore, to run Mininet and Ryu controllers [53], we have used two virtual machines based on the Linux operating system. Ryu controller is based on a Python program and supports several network management protocols such as OpenFlow switches. Moreover, the FlowManager is a Ryu controller program that allows the user to manipulate the flow tables in an OpenFlow network manually. We have used the Ryu controller for SDN networking environments due to its ease of deployment, expansion, and simple architecture. Hence, Ryu controllers with OpenFlow-enabled switches are widely used for SDN networking. They are especially suitable for light traffic communication and control. In addition, the Ryu controller provides a routing link to OpenFlow switches to ensure that the topology can perform data analysis. Thus, to emulate our network structure, a linear topology is used on Mininet, in which 8 switches are connected to the Ryu controller, and each switch is connected to 8 hosts. In total, 64 hosts are linked to the OpenFlow virtual switches, as shown in Figure 8. The IP address of the Ryu controller is 192.168.162.133. Likewise, each host is assigned an IP address. For example, the IP address of Host1 = "10.0.0.1/24"

and the mac address starting from 00:00:00:00:00:01 converted from hexadecimal to an integer.

In general, the following processes are involved in scenario 1: the data generation and collection process, the detection process, and the mitigation process. These processes are deployed using Mininet VM and Ryu controller VM based on Python programming language.

- (i) *Data generation and collection process:* The SDN dataset is created using both the Mininet emulator and Ryu controller. The normal traffic is collected using the "iperf" command, and we consider one host (Host1) as a simple HTTP server listening on port 80. In addition, we collect the SYN flood traffic data using the Hping3 tool with random IP addresses. Hping3 is an open-source TCP/IP protocol used as a packet generation tool; that is, written in the TCL language. Hping3 enables programmers to create scripts for TCP/IP packet handling and analysis in a restricted period. MAC addresses are an important criterion to mitigate SYN flood attacks because layer 2 switches forward incoming traffic based on Mac addresses. Also, it helps to identify the infected source port. Moreover, the layer 4 switch depends on the source and the destination ports that are essential in the flow table with the following features: datapath id, source IP, source Mac, destination IP, destination Mac, IP protocol, ICMP

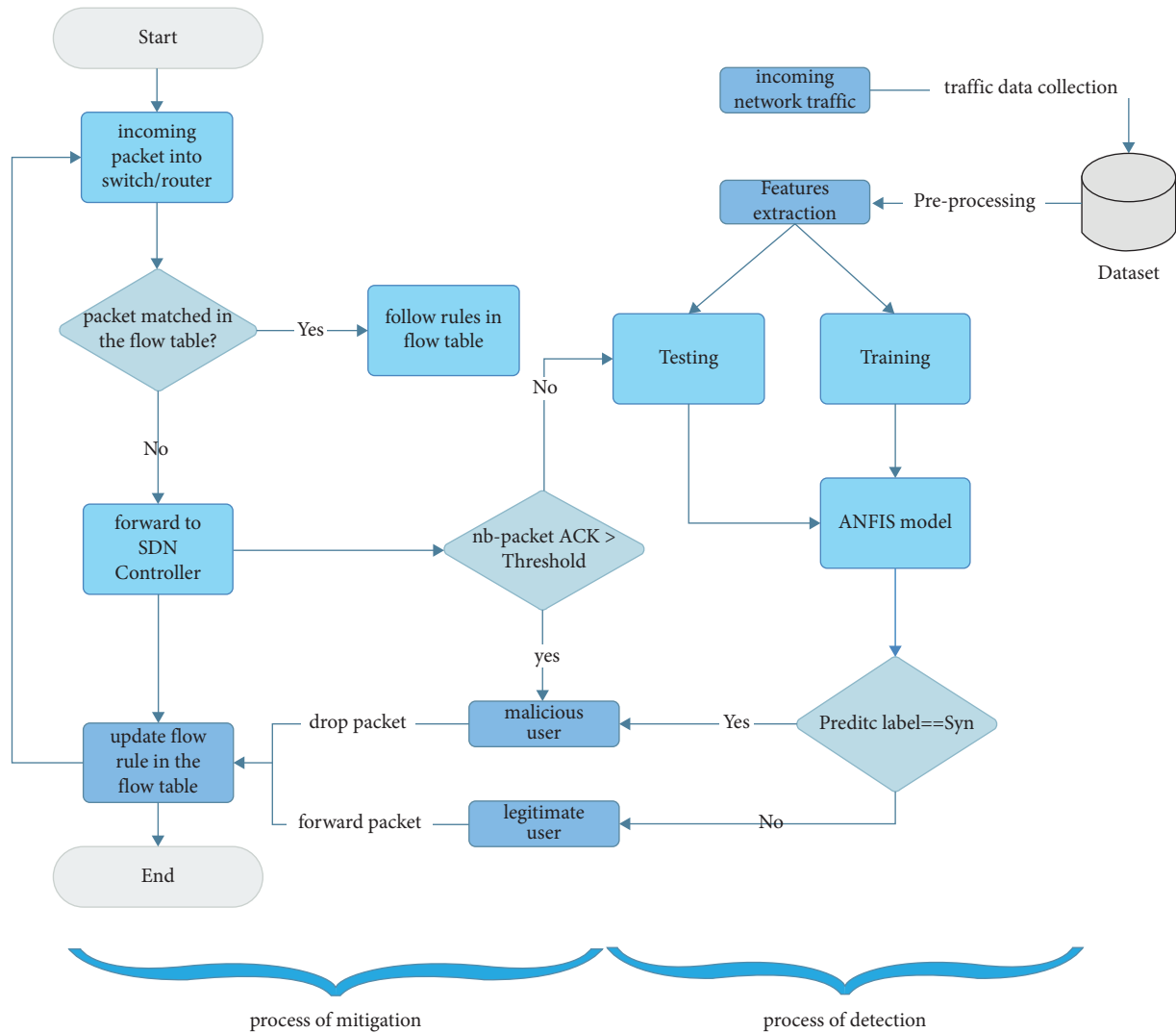


FIGURE 7: Flowchart of the proposed model FASA.

code, ICMP type, packet counts, and flags. Table 1 provides detailed information about the collected dataset.

- (ii) The detection process: After the pre-processing of the collected data presented in I, we will split the dataset as follows: The training set contains 80% of the dataset, whereas the testing set contains 20% of the dataset. Then, we use the ANFIS algorithm with cross-validation to avoid overfitting and train the collected dataset to achieve an accuracy of 100%. Next, once the packet-in is received in various forms of regular traffic and attack traffic, the Ryu-controller collects the features and assigns their values to the predicted dataset. For the prediction process, the detection module (ANFIS algorithm) examines each flow entry.
- (iii) The mitigation process: DDoS attacks are difficult to mitigate because of IP spoofing; therefore, blocking the suspected attacker's IP is ineffective in mitigating; To achieve our objective of obtaining a list of

edge switches directly connected to each host, we will store the Mac address, port number, and switch ID for each host in a Python dictionary. This dictionary will serve as a data structure to retrieve the required parameters for creating mitigation rules.

Every flow entry passes the detection process to check if it is a normal packet or a malicious packet. Then, it will be sent to the Ryu controller to make a decision based on the result of the prediction. Therefore, if the flow entry's predicted value is 1, it indicates a SYN flood attack in which the attacker transmits both the real source Mac address and a random false source IP. Repeating the higher Mac address with different IPs in each flow entry indicates that the hacker is the host of this Mac address. In this case, we use the assigned Mac address to get the port number and switch id from the dictionary. The Ryu controller then responds by enforcing the rule that rejects all packets originating from that attacker. This rule is then sent to the affected switch, instructing it to block the specific port that is directly connected to the attacker's host. By implementing this rule,

```

input: incoming packet of traffic flow to the switch
output: response with flow classification and decision
if packet matched in the flow table
    Apply the rule in the flow table;
else
    Forward packet to SDFN-server;
    Apply ANFIS classifier;
    if flow classified as malicious packet then
        Retrieve the Mac address of the attacker;
        Update rule table in flow table with a malicious user;
        Make a decision:
        Drop the packets with this source Mac address;
        Block the infected switch port;
    else
        Update rule table in SDN with the legitimate user;
        Make decision: Forward the packet to destination;
    end if
end if

```

ALGORITHM 1: FASA framework process.

the switch effectively prevents any communication from the attacker's host through that particular port, helping to mitigate the impact of the attack. Both the hard timeout and the idle timeout are essential parameters that must be adjusted for the mitigation process:

- (i) Idle time means the flow rule will be deleted if no match occurs with incoming packets within the idle timeout value.
- (ii) Hard timeout means the flow rule will be deleted automatically after the hard timeout expires since the rule was created.

In the case of an attack, the Ryu controller blocks the packet on the OF switch with idle time = 0 sec and hard time = 300 sec. With a high priority, we used priority 1000 for our model. As a result, the switch continues to block the source port for 300 seconds without notifying the controller. Otherwise, if the detection result is 0, this signifies normal traffic. The idle time will be 200 seconds, and each flow entry has a fixed priority of 10. If no matching happens throughout this time period, the flow rule will be removed after 200 seconds. The hard time will be 400 seconds, after which all flow entries will be deleted.

During this experiment, the real-time flow traffic captured by Wireshark is represented in Figure 9 which displays the packets per second versus the time plot. In addition, Table 2 presents the parameters employed in this experiment.

Initially, normal traffic is sent out at time 0 seconds. Next, a Syn flood attack is initiated, and at time 60, the packet rate reaches a threshold value close to 700 packets per second. The ANFIS detection module identifies the attack when Once the attack is detected, the mitigation module takes over. The controller utilizes appropriate flow rules to mitigate the attack by dropping packets, blocking the source ports involved in the attack, and informing the switches to update the flow table accordingly. The attack is successfully

mitigated in less than 5 seconds, resulting in a significant drop in the packet rate. The graph shows the continued normal traffic flow without any breakdown until the end of the experiment 140 seconds. This period is crucial as it represents the controller's capability to receive packets effectively. Figure 10 can demonstrate that during the attack, we observed a decrease in bandwidth consumption, reaching as low as 90 Mbits/sec. Fortunately, it quickly recovered to its pre-attack state and remained relatively stable at around 100 Mbits/sec. This demonstrates the effectiveness of our model in mitigating the impact of the attack and restoring normal network performance.

5.2.2. Scenario 2. In the second scenario, we evaluate the proposed model's capability to identify the TCP SYN flood DDoS attacks using the CIC-DDoS dataset produced by Sharafaldin et al. [58] for detecting DDoS attacks and classifying attack types. This dataset is in a CSV format. It includes both benign and current popular DDoS attacks launched in 2019. It is collected on the first and second days and reflects the actual real-world data (PCAPs). It also provides the findings of a network traffic analysis performed with CICFlowMeter-V3 that includes labeled traffic flows. This dataset originally had 88 features.

In this scenario, we use the SYN flood dataset presented in Table 3. TCP SYN flood is a type of exploitation category-based DDoS attack that exploits vulnerabilities in TCP connection protocols. It is composed of data from two days, each with a different attack category and a wide range of imbalanced class distribution.

(1) Resampling Data. Both training and testing datasets have a minority class "BENIGN" with a little sample, resulting in an imbalanced classification, which has an impact on a model's capacity to learn and decide and, furthermore, can cause overfitting in our model. To accomplish this, we build a new dataset in which we take all samples labeled

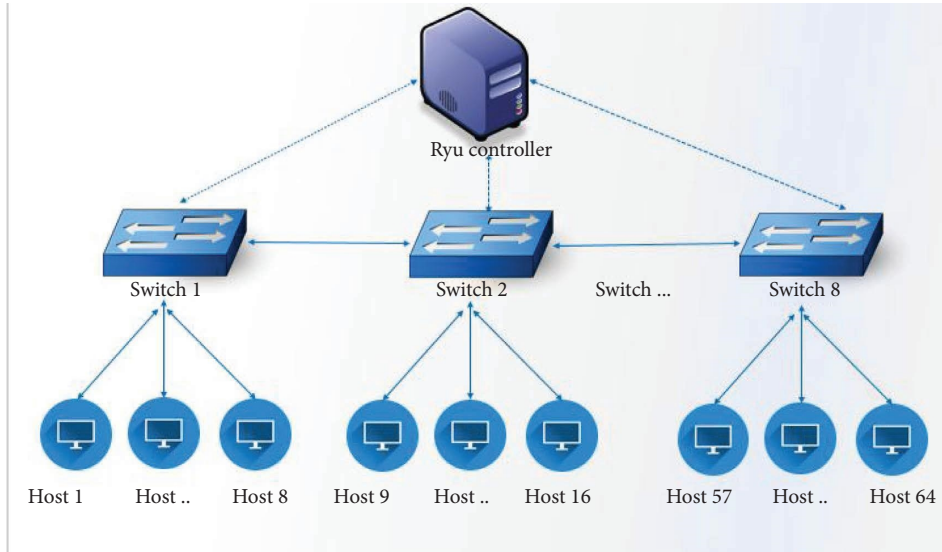


FIGURE 8: Emulated SDN network on scenario 1.

TABLE 1: The collected dataset using SDN.

SDN dataset	All samples	BEGINN	SYN
Train/test	737156	816156	1553311

“BENIGN” from the training and testing datasets, forming 20% of the total dataset and 80% of samples labeled “SYN” as shown in Table 4.

(2) *Data Pre-Processing*. In this section, we will go over the techniques used to analyze our dataset, which contains 88 features. The data will be cleansed and prepared for use in our suggested ANFIS algorithms once certain undesirable attributes have been removed and adjusted. As a result, the implementation of a data preprocessing step, as shown in Figure 11, provides more reliable training and, thus, a more accurate model.

- (i) First, we removed features that have a unique value in the entire dataset that do not affect the training phase (“Bwd PSH Flags,” “Fwd URG Flags,” “Bwd URG Flags,” “FIN Flag Count,” “Fwd Avg Bytes/Bulk,” “Fwd Avg Packets/Bulk,” “Fwd Avg Bulk Rate,” “Bwd Avg Bytes/Bulk,” “PSH Flag Count,” “ECE Flag Count,” “Bwd Avg Packets/Bulk,” “Bwd Avg Bulk Rate”).
- (ii) Some values of “Init Win bytes forward” and “Init Win bytes backward” of flow data from the Syn csv file were set to -1 . Nevertheless, it is inconceivable to initiate a byte window of size -1 . This problem was caused by a software issue with CICFlowmeter and should be set to 0 or removed to not disrupt the training phase.
- (iii) The need to cope with missing data threw off the model’s training. The lines containing “infinity” and “NaN” were removed from “Flow Bytes/s” and “Flow Packets/s.”

- (iv) We removed categorical features that can change from one network to another (“Source Port,” “Destination Port,” “Source IP,” “Destination IP,” “Flow ID,” “SimilarHTTP,” “Unnamed: 0,” “Timestamp”).
- (v) To properly distinguish important features, delete columns with a correlation higher than 0.8 (“Total Backward Packets,” “Total Length of Bwd Packets,” “Fwd Packet Length Std,” “Bwd Packet Length Min,” “Bwd Packet Length Mean,” “Bwd Packet Length Std,” “Flow IAT Mean,” “Flow IAT Std,” “Flow IAT Max,” “Fwd IAT Total,” “Fwd IAT Mean,” “Fwd IAT Std,” “Fwd IAT Max,” “Fwd IAT Min,” “Bwd IAT Std,” “Bwd IAT Max,” “Fwd Header Length,” “Bwd Header Length,” “Max Packet Length,” “Packet Length Mean,” “Packet Length Std,” “Packet Length Variance,” “RST Flag Count,” “Average Packet Size,” “Avg Fwd Segment Size,” “Avg Bwd Segment Size,” “Fwd Header Length.1,” “Subflow Fwd Packets,” “Subflow Fwd Bytes,” “Subflow Bwd Packets,” “Subflow Bwd Bytes,” “Active Max,” “Active Min,” “Idle Mean,” “Idle Max,” “Idle Min”).
- (vi) In order to detect and classify DDoS attacks, the dataset is split into two classes. The label “BENIGN” is coded as “0” and the label “Syn” is coded as “1” in the dataset created to detect a SYN flood DDoS attack on the network traffic.
- (vii) Feature selection is used to discover key data features and decrease the amount of data required for detection. We use the XGBoost technique that provides an importance score to each feature based on its

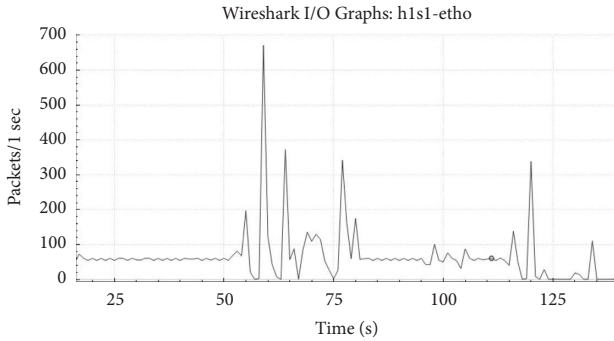


FIGURE 9: Real-time detection and mitigation of Syn flood attack.

influence in making crucial decisions using boosted decision trees [58]. Then, depending on the rated feature, we removed features that were of negligible importance “Protocol,” “Flow Duration,” “Total Fwd Packets,” “Fwd Packet Length Max,” “Bwd Packet Length Max,” “Flow IAT Mean,” “Flow IAT Min,” “Bwd IAT Total,” “Bwd IAT Mean,” “Bwd IAT Min,” “Fwd PSH Flags,” “Fwd Packets/s,” “Bwd Packets/s,” “Min Packet Length,” “SYN Flag Count,” “CWE Flag Count,” “Down/Up Ratio,” “Init Win bytes backward,” “act data pkt fwd”, “Active Mean,” “Active Std,” “Idle Std”, and choose nine ideal feature subsets, as presented in Table 5.

- (a) We normalize the data by scaling all features in the range of 0-1 value. As previously described, the dataset was divided into two parts training data and testing data. by using cross-validation to avoid overfitting in training steps.
- (b) Finally, we put the ANFIS model to the test for making predictions on unseen data. The next section discusses the performances and results.

5.3. *Performance Metrics.* Using the right performance metrics is the key to correctly evaluating models. Therefore, in this section, we explore the following performance metrics to evaluate the FASA framework:

- (i) *True Negatives (TN)*: Normal flow data is appropriately identified as such.
- (ii) *True Positives (TP)*: malicious flow data is accurately identified as such.
- (iii) *False Positives (FP)*: Normal flow data is mistakenly labeled as malicious traffic.
- (iv) *False Negatives (FN)*: malicious flow data is classified as normal flow data when it is not.

In addition, we provide the confusion matrix to describe our model’s classification performance. It can resume the correct and false predictions obtained using our proposed approach, as demonstrated in Figure 12.

Accurately distinguishing the Benign class within our model is of utmost importance, as elevated false positive rates can result in unnecessary complexity and unwarranted alerts. Our main objective is to minimize the false rate. Hence, our

TABLE 2: Experiment parameters.

Parameters	Value
Traffic generation tool	Iperf, Hping3
Simulation time	140 sec
Bandwidth	100 Mbits/sec
Data collection interval	5 sec
Server	Host 1

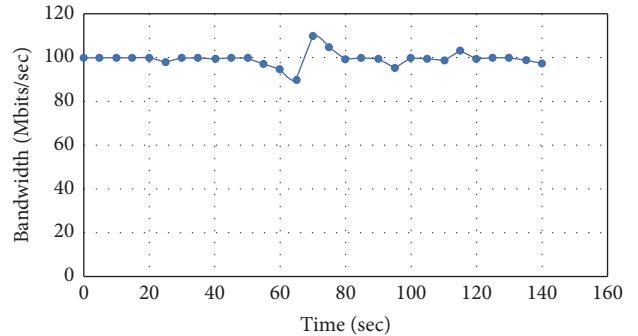


FIGURE 10: Bandwidth usage.

TABLE 3: SYN flood CIC-DDoS 2019 dataset.

All new dataset	All samples	BEGINN	SYN
Training day	1582681	392	1582289
Testing day	4320541	35790	4284751

framework achieves a rate of 0% false positives in both CIC-DDoS2019 and SDN datasets. Otherwise, it obtains 0.058% false negatives in the CIC-DDoS2019 dataset and 0% in the SDN dataset. The receiver operating characteristic (ROC) curve is performed. It represents the relationship between both the True and False parameters. The area under the ROC curve (AUC) measures whether it is possible to distinguish false positives from true positives. As illustrated in Figure 13, our model has an AUC of 99.96% using the CIC-DDoS2019 dataset and 100% using the SDN dataset and there are two extremely similar values, indicating that our suggested model separates correctly positive from negative classes. By employing established techniques like k-fold cross-validation, the model ensures generalizability and guards against overfitting. Furthermore, the meticulous selection and optimization of impactful traffic features enhance the model’s proficiency in distinguishing between normal and attack behaviors. In addition, the fusion of fuzzy logic and neural learning components proves effective in capturing complex traffic patterns. Lastly, training on diverse attack data distributions further enhances the model’s robustness. We have also used a variety of measures to assess our suggested model, including accuracy, precision, recall, and F-score, to conduct an in-depth comparative assessment with some other relevant methods. These metrics, which are often employed in SYN flood DDoS detection systems, are described in the following:

- (1) Accuracy refers to the ratio of the number of samples correctly classified to the overall number of samples observed. It is computed as follows:

TABLE 4: The new balanced SYN flood dataset.

New dataset	All samples	BEGNIN	SYN
Train/test	180910	36182	144728

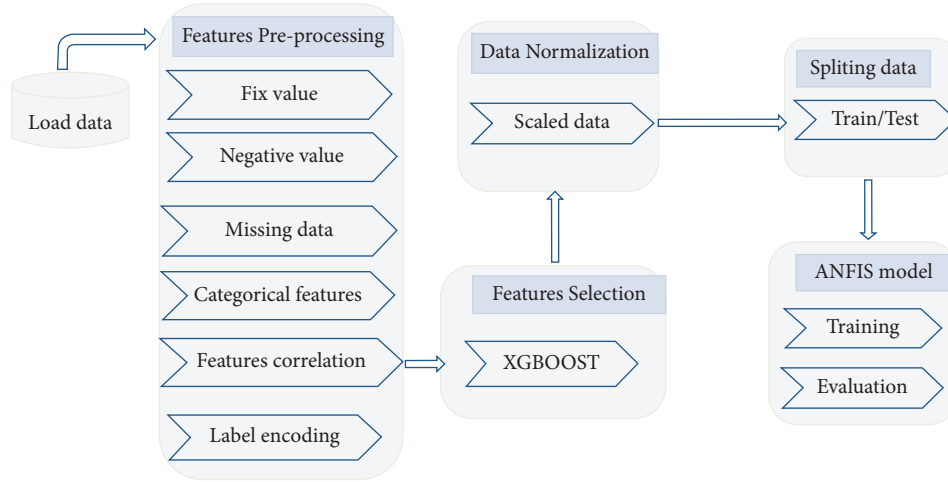


FIGURE 11: Data preprocessing.

TABLE 5: Features selected with XGBoost.

Feature name	Description
Total Length of Fwd Packets	Overall size of packet in the forward direction
Fwd Packet Length Mean	Mean size of packet in forward direction
ACK Flag Count	Number of packets with ACK
URG Flag Count	Number of packets with URG
Init Win bytes forward	Number of bytes sent in initial window in the forward direction
min seg size forward	The observed minimum segment size in the forward direction
Inbound	The direction in which traffic moves between networks
Label	Type of packets for classification

$$\text{accuracy} = \frac{tp + tn}{tp + tn + fp + fn}. \quad (6)$$

- (2) The precision is the ratio of correctly predicted positive samples, it is calculated as follows:

$$\text{precision} = \frac{tp}{tp + fp}. \quad (7)$$

- (3) The false positive rate is determined by calculating the proportion of negative samples that were incorrectly classified as positive using the following formula:

$$fp - \text{rate} = \frac{fp}{fp + tn}. \quad (8)$$

- (4) The recall also called the true positive rate is calculated with the ratio of correctly discovered positive samples. It is determined using the equation:

$$\text{recall} = tp - \text{rate} = \frac{tp}{tp + fn}. \quad (9)$$

- (5) Good precision may be more relevant in certain situations, whereas high recall might be more critical in others. In many cases, though, we aim to enhance both

values. The f1-score is the combination of these values, and it is commonly stated as the harmonic mean:

$$f1 - \text{score} = \frac{2 \times \text{precision} \times \text{recall}}{(\text{precision} + \text{recall})}. \quad (10)$$

5.4. Evaluation Results. To validate our system, we have compared the FASA framework to the FUPE [10] method and other DDoS attack detection systems that were employed on SDN and used the CIC-DDoS 2019 dataset, as illustrated in Table 6.

The first method is FUPE [10] which puts forward a fuzzy-based multiobjective particle swarm optimization approach and a security-aware task scheduler in IoT-fog networks. The second method is the convolutional neural network (CNN) [26], a low-cost based supervised classifier designed to identify suspicious events in a data center. The next approach is based on the generative adversarial network GAN [27] for identifying DDoS threats in SDN environments. Finally, the multilayer perceptron (MLP) [28] is adopted to identify and prevent low rate-DDoS attacks in SDN settings. Figure 14 depicts a comprehensive analysis of the metric findings of the comparative approaches.

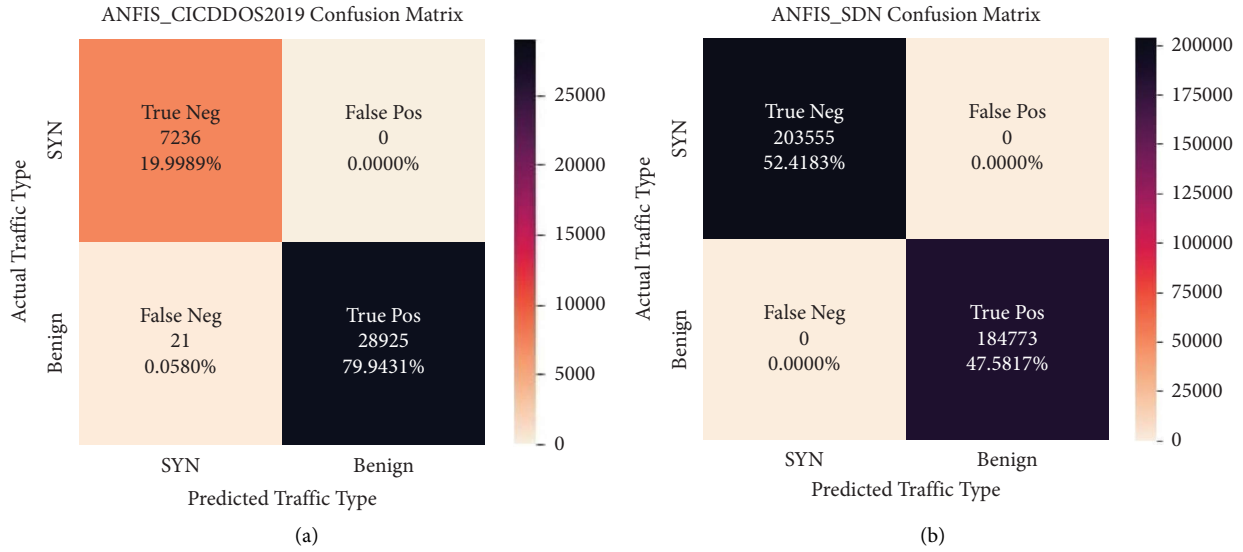


FIGURE 12: Confusion matrix of the ANFIS model. (a) Using CIC-DDoS dataset. (b) Using SDN dataset.

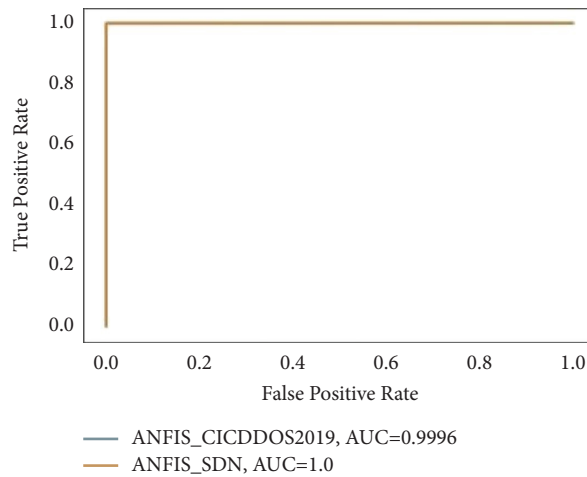


FIGURE 13: ROC curve of ANFIS model.

TABLE 6: The evaluated metrics were used to compare the results of ANFIS with other methods.

Method	Accuracy	Precision	Recall	F1-score
ANFIS SDN	100	100	100	100
ANFIS CIC-DDoS2019	99.95	100	99.94	99.95
FUPE [10]	98.2	96.08	98	N/A
CNN [26]	95.4	93.3	92.4	92.8
GAN [27]	94.38	94.08	97.89	95.94
MLP [28]	95.01	95.46	94.51	94.98

As shown in Figure 14, we can observe that the performance of our model using the SDN dataset outperforms all previous techniques with 100% accuracy, precision, recall, and F1-score in each case, and it closely resembles the outcome obtained using the CIC-DDoS2019 dataset. In addition, the accuracy of every learning algorithm is assessed. As a result, the ANFIS achieved the highest accuracy rating of 99.95% across all classifiers, then the FUPE approach with 98.2% followed by the CNN algorithm with

94.83%. Furthermore, MLP and GAN classifiers attained an accuracy of 95.4% and 95.01%, respectively. It also illustrates the precision of each algorithm in identifying legal and malicious traffic. Thus, the ANFIS reached 100% precision, and FUPE with a precision of 96.08%, and the MLP attained a precision value of 95.46%. Next, the GAN, and CNN algorithms with a precision of 94.08%, and 93.3%, respectively. Furthermore, Figure 14 displays the recall values of all methods used in the performance evaluation. The

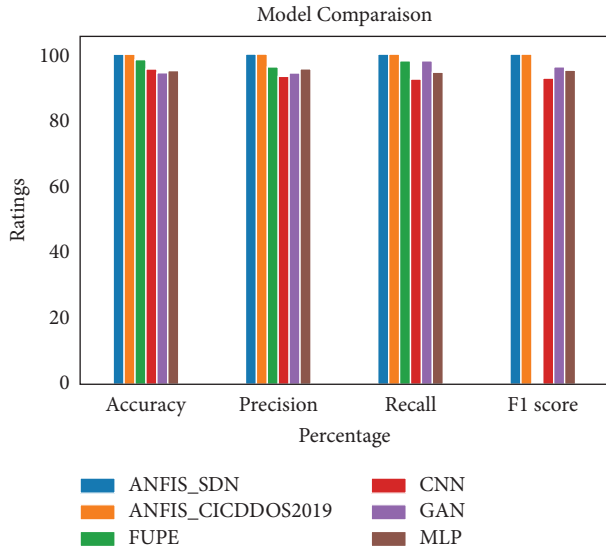


FIGURE 14: Evaluation metrics for comparative methods.

ANFIS algorithm had a 99.94% recall value followed by FUPE with 98%, whereas GAN had a 97.89% recall rating. In comparison to the other algorithms tested, the CNN achieved the lowest recall value of 92.4% while the MLP had a recall of 94.51%. It also illustrates the $F1$ -score of the classifying methods with 99.95%, the ANFIS received the highest $F1$ -Score. On the other hand, GAN, MLP, and CNN received $F1$ -scores of 95.94%, 94.98%, and 92.8%, respectively, while the FUPE's $F1$ -Score is not mentioned. In conclusion, our FASA framework outperforms the other evaluated approaches. The promising test results indicate that it is an effective approach for identifying SYN flood DDoS attacks.

6. Conclusion and Future Work

In this work, FASA, a fog computing-based SYN flood DDoS attacks mitigation using an adaptive neuro-fuzzy inference system (ANFIS) and software defined networking (SDN) assistance was proposed. The choice of the integration of SDN and fog environment with the ANFIS machine learning algorithm brings intelligence to the SDN controller. Also, it makes our framework suitable, efficient, and more secure against SYN flood attacks. We trained and evaluated our framework on the newly released CIC-DDoS2019 dataset that contains the most recent and extensive SYN flood DDoS attacks. The findings of the performance assessment indicate that the suggested model has a high detection accuracy and a low rate of false positive and negative rates, which is a remarkable result, and it also offers the highest evaluation metrics in regards to precision, recall, and F -score when compared to well-known machine learning algorithms. Our future work is to focus on how well our proposed model performs on various datasets. In the current experiments, we have employed a binary classification approach that is implemented on SDN to distinguish between legitimate and malicious input traffic in fog computing. Thus, in future work, we will try to investigate the utility of the suggested

approach for other multi-class classification systems. Furthermore, we plan to assess the performance of ANFIS using additional regression metrics, including R -Square, RMSE (root mean square error), and MAE (mean absolute error), beyond the classification metrics presented. This expanded evaluation will offer a more comprehensive understanding of the model's capabilities. In addition, to create a diversified dataset that truly represents actual Internet traffic, we will emulate the SDN network under various scenarios and with various attack traffic. In addition, we will also consider expanding our work to include the SoDIP6-based ISP/Telcom network, including edge computing network scenarios. This will allow us to evaluate the performance of our proposed model in a more complex and realistic environment. We will also investigate the use of our model for other network security applications, such as intrusion detection and prevention.

Data Availability

Data used in this study are available upon request to the corresponding author.

Conflicts of Interest

All authors declare that there are no conflicts of interest.

Acknowledgments

The authors conducted this research while affiliated with Abou Bekr Belkaid Tlemcen University, Paris-Saclay University, Edinburgh Napier University, and Dakahlia Mansoura University. An early version of the article appears in arxiv [59]. Open Access funding was enabled and organized by JISC.

References

- [1] H. Saidi, N. Labraoui, A. A. A. Ari, and D. Bouida, "Remote health monitoring system of elderly based on fog to cloud (f2c) computing," in *Proceedings of the 2020 International Conference on Intelligent Systems and Computer Vision (ISCV)*, pp. 1–7, Fez, Morocco, June 2020.
- [2] M. Babaghayou, N. Labraoui, and A. A. A. Ari, "Location-privacy evaluation within the extreme points privacy (epp) scheme for vanet users," *International Journal of Strategic Information Technology and Applications*, vol. 10, no. 2, pp. 44–58, 2019.
- [3] M. Babaghayou, N. Labraoui, A. A. Abba Ari, M. A. Ferrag, L. Maglaras, and H. Janicke, "Whisper: a location privacy-preserving scheme using transmission range changing for internet of vehicles," *Sensors*, vol. 21, no. 7, p. 2443, 2021.
- [4] H. Saidi, N. Labraoui, A. A. A. Ari, L. A. Maglaras, and J. H. M. Emati, "Dsmac: privacy-aware decentralized self-management of data access control based on blockchain for health data," *IEEE Access*, vol. 10, pp. 101011–101028, 2022.
- [5] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks," *IEEE communications surveys and tutorials*, vol. 15, no. 4, pp. 2046–2069, 2013.
- [6] Y. Yigit, C. Chrysoulas, G. Yurdakul, L. Maglaras, and B. Canberk, "Digital twin-empowered smart attack detection

- system for 6g edge of things networks,” in *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, Kuala Lumpur, Malaysia, December 2023.
- [7] K. Bhushan, “Ddos attack defense framework for cloud using fog computing,” in *Proceedings of the 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information and Communication Technology (RTEICT)*, pp. 534–538, IEEE, Bangalore, India, May 2017.
 - [8] B. Paharia and K. Bhushan, “Fog computing as a defensive approach against distributed denial of service (ddos): a proposed architecture,” in *Proceedings of the 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1–7, IEEE, Bangalore, India, July 2018.
 - [9] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, “A survey of software-defined networking: past, present, and future of programmable networks,” *IEEE Communications surveys and tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.
 - [10] S. Javanmardi, M. Shojafar, R. Mohammadi, A. Nazari, V. Persico, and A. Pescapè, “Fupe: a security driven task scheduling approach for sdn-based iot-fog networks,” *Journal of Information Security and Applications*, vol. 60, Article ID 102853, 2021.
 - [11] S. Javanmardi, M. Shojafar, R. Mohammadi, M. Alazab, and A. M. Caruso, “An sdn perspective iot-fog security: a survey,” *Computer Networks*, vol. 229, Article ID 109732, 2023.
 - [12] A. S. Boroujerdi and S. Ayat, “A robust ensemble of neuro-fuzzy classifiers for ddos attack detection,” in *Proceedings of 2013 3rd International Conference on Computer Science and Network Technology*, pp. 484–487, IEEE, Dalian, China, October 2013.
 - [13] P. Arun Raj Kumar and S. Selvakumar, “Detection of distributed denial of service attacks using an ensemble of adaptive and hybrid neuro-fuzzy systems,” *Computer Communications*, vol. 36, no. 3, pp. 303–319, 2013.
 - [14] KDD, “Kdd Data Set,” 1999, <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>.
 - [15] A. Aldweesh, A. Derhab, and A. Z. Emam, “Deep learning approaches for anomaly-based intrusion detection systems: a survey, taxonomy, and open issues,” *Knowledge-Based Systems*, vol. 189, Article ID 105124, 2020.
 - [16] S. Fichera, L. Galluccio, S. C. Grancagnolo, G. Morabito, and S. Palazzo, “Operetta: an openflow-based remedy to mitigate tcp synflood attacks against web servers,” *Computer Networks*, vol. 92, pp. 89–100, 2015.
 - [17] G. Ramadhan, Y. Kurniawan, and C.-S. Kim, “Design of tcp syn flood ddos attack detection using artificial immune systems,” in *Proceedings of the 2016 6th International Conference on System Engineering and Technology (ICSET)*, pp. 72–76, IEEE, Bandung, Indonesia, October 2016.
 - [18] A. Ahalawat, K. S. Babu, A. K. Turuk, and S. Patel, “A low-rate ddos detection and mitigation for sdn using renyi entropy with packet drop,” *Journal of Information Security and Applications*, vol. 68, Article ID 103212, 2022.
 - [19] N. Hoque, H. Kashyap, and D. K. Bhattacharyya, “Real-time ddos attack detection using fpga,” *Computer Communications*, vol. 110, pp. 48–58, 2017.
 - [20] S. Jin and D. S. Yeung, “A covariance analysis model for ddos attack detection,” in *Proceedings of the 2004 IEEE International Conference on Communications (IEEE Cat. No. 04CH37577)*, pp. 1882–1886, IEEE, Paris, France, July 2004.
 - [21] S.-C. Tsai, I.-H. Liu, C.-T. Lu, C.-H. Chang, and J.-S. Li, “Defending cloud computing environment against the challenge of ddos attacks based on software defined network,” in *Advances in Intelligent Information Hiding and Multimedia Signal Processing: Proceeding of the Twelfth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pp. 285–292, Springer, Kaohsiung, Taiwan, 2017.
 - [22] A. Bhardwaj, V. Mangat, R. Vig, S. Halder, and M. Conti, “Distributed denial of service attacks in cloud: state-of-the-art of scientific and commercial solutions,” *Computer Science Review*, vol. 39, Article ID 100332, 2021.
 - [23] S. Rajagopal, P. P. Kundapur, and K. Hareesha, “Towards effective network intrusion detection: from concept to creation on azure cloud,” *IEEE Access*, vol. 9, pp. 723–742, 2021.
 - [24] N. N. Tuan, P. H. Hung, N. D. Nghia, N. Van Tho, T. V. Phan, and N. H. Thanh, “A robust tcp-syn flood mitigation scheme using machine learning based on sdn,” in *Proceedings of the 2019 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 363–368, IEEE, Jeju, Korea, October 2019.
 - [25] R. Priyadarshini, R. Kumar Barik, and H. Dubey, “Fog-sdn: a light mitigation scheme for ddos attack in fog computing framework,” *International Journal of Communication Systems*, vol. 33, no. 9, p. e4389, 2020.
 - [26] M. V. de Assis, L. F. Carvalho, J. J. Rodrigues, J. Lloret, and M. L. Proença Jr, “Near real-time security system applied to sdn environments in iot networks using convolutional neural network,” *Computers and Electrical Engineering*, vol. 86, Article ID 106738, 2020.
 - [27] M. P. Novaes, L. F. Carvalho, J. Lloret, and M. L. Proença Jr, “Adversarial deep learning approach detection and defense against ddos attacks in sdn environments,” *Future Generation Computer Systems*, vol. 125, pp. 156–167, 2021.
 - [28] J. A. Perez-Diaz, I. A. Valdovinos, K.-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. 155 859–155 872, 2020.
 - [29] O. Brun, Y. Yin, E. Gelenbe, Y. M. Kadioglu, J. Augusto-Gonzalez, and M. Ramos, “Deep learning with dense random neural networks for detecting attacks against iot-connected home environments,” in *Security in Computer and Information Sciences: First International ISCIS Security Workshop 2018*, pp. 79–89, Springer International Publishing, London, UK, 2018.
 - [30] S. Evmorfos, G. Vlachodimitropoulos, N. Bakalos, and E. Gelenbe, “Neural network architectures for the detection of syn flood attacks in iot systems,” in *Proceedings of the 13th ACM International Conference on Pervasive Technologies Related to Assistive Environments*, pp. 1–4, Corfu, Greece, June 2020.
 - [31] R. Devi, R. K. Jha, A. Gupta, S. Jain, and P. Kumar, “Implementation of intrusion detection system using adaptive neuro-fuzzy inference system for 5g wireless communication network,” *AEU-International Journal of Electronics and Communications*, vol. 74, pp. 94–106, 2017.
 - [32] O. Osanaiye, K.-K. R. Choo, and M. Dlodlo, “Distributed denial of service (ddos) resilience in cloud: review and conceptual cloud ddos mitigation framework,” *Journal of Network and Computer Applications*, vol. 67, pp. 147–165, 2016.
 - [33] S. Khattak, N. R. Ramay, K. R. Khan, A. A. Syed, and S. A. Khayam, “A taxonomy of botnet behavior, detection, and defense,” *IEEE communications surveys and tutorials*, vol. 16, no. 2, pp. 898–924, 2014.

- [34] N. Hoque, D. K. Bhattacharyya, and J. K. Kalita, "Botnet in ddos attacks: trends and challenges," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 4, pp. 2242–2270, 2015.
- [35] B. Paharia and K. Bhushan, "A comprehensive review of distributed denial of service (ddos) attacks in fog computing environment," *Handbook of Computer Networks and Cyber Security: Principles and Paradigms*, pp. 493–524, Springer, Berlin, Germany, 2020.
- [36] C. L. Schuba, I. V. Krsul, M. G. Kuhn, E. H. Spafford, A. Sundaram, and D. Zamboni, "Analysis of a denial of service attack on tcp," in *Proceedings of the 1997 IEEE Symposium on Security and Privacy (Cat. No. 97CB36097)*, pp. 208–223, IEEE, Oakland, CA, USA, May 1997.
- [37] R. Santos, D. Souza, W. Santo, A. Ribeiro, and E. Moreno, "Machine learning algorithms to detect ddos attacks in sdn," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 16, p. e5402, 2020.
- [38] J.-S. Jang, "Anfis: adaptive-network-based fuzzy inference system," *IEEE transactions on systems, man, and cybernetics*, vol. 23, no. 3, pp. 665–685, 1993.
- [39] S. Benfriha and N. Labraoui, "Insiders detection in the uncertain iod using fuzzy logic," in *Proceedings of the 2022 International Arab Conference on Information Technology (ACIT)*, pp. 1–6, IEEE, Abu Dhabi, United Arab Emirates, November 2022.
- [40] N. Walia, H. Singh, and A. Sharma, "Anfis: adaptive neuro-fuzzy inference system-a survey," *International Journal of Computer Application*, vol. 123, no. 13, pp. 32–38, 2015.
- [41] S. Ghosh, S. Biswas, D. Sarkar, and P. P. Sarkar, "A novel neuro-fuzzy classification technique for data mining," *Egyptian Informatics Journal*, vol. 15, no. 3, pp. 129–147, 2014.
- [42] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," 2014, <https://arxiv.org/abs/1412.6980>.
- [43] V. Bureva, "Generalized net model of information security activities in the automated information systems," in *Advances and New Developments in Fuzzy Logic and Technology: Selected Papers from IWIFSGN'2019-The Eighteenth International Workshop on Intuitionistic Fuzzy Sets and Generalized Nets Held on October 24-25, 2019 in Warsaw, Poland*, pp. 280–288, Springer, Berlin, Germany, 2021.
- [44] A. Aguado, M. Davis, S. Peng et al., "Dynamic virtual network reconfiguration over sdn orchestrated multitechnology optical transport domains," *Journal of Lightwave Technology*, vol. 34, no. 8, pp. 1933–1938, 2016.
- [45] K. Bakshi, "Considerations for software defined networking (sdn): approaches and use cases," in *Proceedings of the 2013 IEEE Aerospace Conference*, pp. 1–9, IEEE, Big Sky, MT, USA, March 2013.
- [46] D. Samociuk, "Secure communication between openflow switches and controllers," *AFIN 2015*, vol. 39, 2015.
- [47] S. Sathyadevan, K. Achuthan, R. Doss, and L. Pan, "Protean authentication scheme—a time-bound dynamic keygen authentication technique for iot edge nodes in outdoor deployments," *IEEE Access*, vol. 7, pp. 92 419–492 435, 2019.
- [48] J.-P. A. Yaacoub, O. Salman, H. N. Noura, N. Kaaniche, A. Chehab, and M. Malli, "Cyber-physical systems security: limitations, issues and future trends," *Microprocessors and Microsystems*, vol. 77, Article ID 103201, 2020.
- [49] Q. Yan, F. R. Yu, Q. Gong, and J. Li, "Software-defined networking (sdn) and distributed denial of service (ddos) attacks in cloud computing environments: a survey, some research issues, and challenges," *IEEE communications surveys and tutorials*, vol. 18, no. 1, pp. 602–622, 2016.
- [50] R. Vishwakarma and A. K. Jain, "A survey of ddos attacking techniques and defence mechanisms in the iot network," *Telecommunication Systems*, vol. 73, no. 1, pp. 3–25, 2020.
- [51] A. Nath, *Packet Analysis with Wireshark*, Packt Publishing Ltd, Birmingham, UK, 2015.
- [52] M. team, "mininet overview," 2023, <http://mininet.org/overview/>.
- [53] RYU, "Ryu Sdn Framework," 2023, <https://ryu-sdn.org/>.
- [54] S. Bhardwaj and S. N. Panda, "Performance evaluation using ryu sdn controller in software-defined networking environment," *Wireless Personal Communications*, vol. 122, no. 1, pp. 701–723, 2022.
- [55] Keras, "Keras.io," 2023, <https://keras.io>.
- [56] M. Abadi, "Tensorflow: learning functions at scale," in *Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming*, p. 1, Nara, Japan, September 2016.
- [57] S. Prusty, S. Patnaik, and S. K. Dash, "Skcv: stratified k-fold cross-validation on ml classifiers for predicting cervical cancer," *Frontiers in Nanotechnology*, vol. 4, Article ID 972421, 2022.
- [58] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (ddos) attack dataset and taxonomy," in *Proceedings of the 2019 International Carnahan Conference on Security Technology (ICCST)*, pp. 1–8, IEEE, Chennai, India, October 2019.
- [59] R. Bensaid, N. Labraoui, A. A. Ari et al., "Toward a real-time tcp syn flood ddos mitigation using adaptive neuro-fuzzy classifier and sdn assistance in fog computing," 2023, <https://arxiv.org/abs/2311.15633>.
- [60] Y. Xia, C. Liu, Y. Li, and N. Liu, "A boosted decision tree approach using bayesian hyper-parameter optimization for credit scoring," *Expert Systems with Applications*, vol. 78, pp. 225–241, 2017.