Int. J. Simul. Multidisci. Des. Optim. **16**, 29 (2025)
© N. Nisa et al., Published by EDP Sciences, 2025
https://doi.org/10.1051/smdo/2025031

IJSMDO
Available online at:
https://www.ijsmdo.org

RESEARCH ARTICLE OPEN ACCESS

SVM driven approach for detecting DoS attacks in SDN environment

Najmun Nisa[1,2,*], Adnan Shahid Khan[1], Azman Bin Bujang Masli[1], and Nusrat Shaheen[2]

[1] Faculty of Computer Science and Information Technology, Universiti Malaysia Sarawak, Malaysia
[2] Department of Computer Science, COMSATS University Islamabad, Pakistan

Received: 22 September 2025 / Accepted: 10 October 2025

Abstract. Software-Defined Networking (SDN) reveals a significant progression in networking technology, offering improved management and operational oversight of network infrastructures. Even though the control plane offers benefits, it is still susceptible to Denial of Service (DoS) attacks, and this poses a significant threat to system security. By taking advantage of the network's centralized architecture, these attacks pose serious dangers and can overload controllers, leading to severe packet loss and significant downtime in the network. To address this challenge, we propose a novel approach that efficiently detects DoS attacks by implementing a packet inspection process using a queuing mechanism, followed by machine learning classification using SVM and KNN algorithms. These algorithms were rigorously evaluated using the CICDoS 2017 dataset and integrated into an SDN threat-detection framework. The results of extensive testing in SDN environment demonstrated higher efficiency measures, such as enhanced network performance by reducing latency and resource consumption, maintaining a false-positive rate under 5%, and achieving a detection accuracy of 99%. These results demonstrate how well our proposed approach works to successfully detect DoS attacks in SDN systems. Moreover, the novel approach, the thorough end-to-end solution exhibited, and the importance of the experimental outcomes all work together to establish a solid basis for future studies in this area.

Keywords: Software defined networking / Denial-of-Service (DoS) attacks / Distributed Denial-of-Service (DDoS) attacks / Mininet / CICDoS 2017 dataset / OpenFlow / RYU controller / Machine Learning (ML) / Support Vector Machine (SVM) / K-Nearest Neighbour (KNN) / First-In-First-Out (FIFO)

1 Introduction and SDN workflow

Traditional network management faces significant challenges due to the complexity of its architecture and the evolving market demands. The widespread influence of the Internet is reshaping communication and computing technologies. In traditional networks, operations and traffic routing are primarily managed by routers and switches, as illustrated in Figure 1. This integration increases the complexity and cost of network administration. Consequently, incorporating additional features into conventional networks becomes more difficult, particularly with the growing requirements for scalability, security, flexibility, dependability, and reliability [1].

Traditional networks have separate parts for the data plane and the control plane. SDNs, on the other hand, use a disaggregated approach by giving each operational duty to a separate entity [2]. The SDN controller acts as the control plane in SDN, which decides on different policies and traffic routing. The data plane, which includes network devices like routers and switches, is responsible for forwarding packets according to the control plane's directives. In SDN, the control plane, data plane, and even the application plane are all able to communicate with one another through the southbound and northbound interfaces as shown in Figure 1. This allows for the efficient management and operation of the network [3].

In order to centralize control within an SDN controller and distribute the packet forwarding responsibilities across network devices such as switches and routers, SDN separates the control plane from the data plane. This centralized control provides a comprehensive perspective of the network, making management and policy enforcement easier. Nevertheless, this architectural design presents considerable data vulnerabilities, particularly the possibility of the SDN controller being a sole point of failure. In the event of the controller being compromised, there is potential for extensive network disruption or illegal access to sensitive data. In order to reduce these dangers,

* e-mail: najam_nisa@comsats.edu.pk

This is an Open Access article distributed under the terms of the Creative Commons Attribution License (https://creativecommons.org/licenses/by/4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

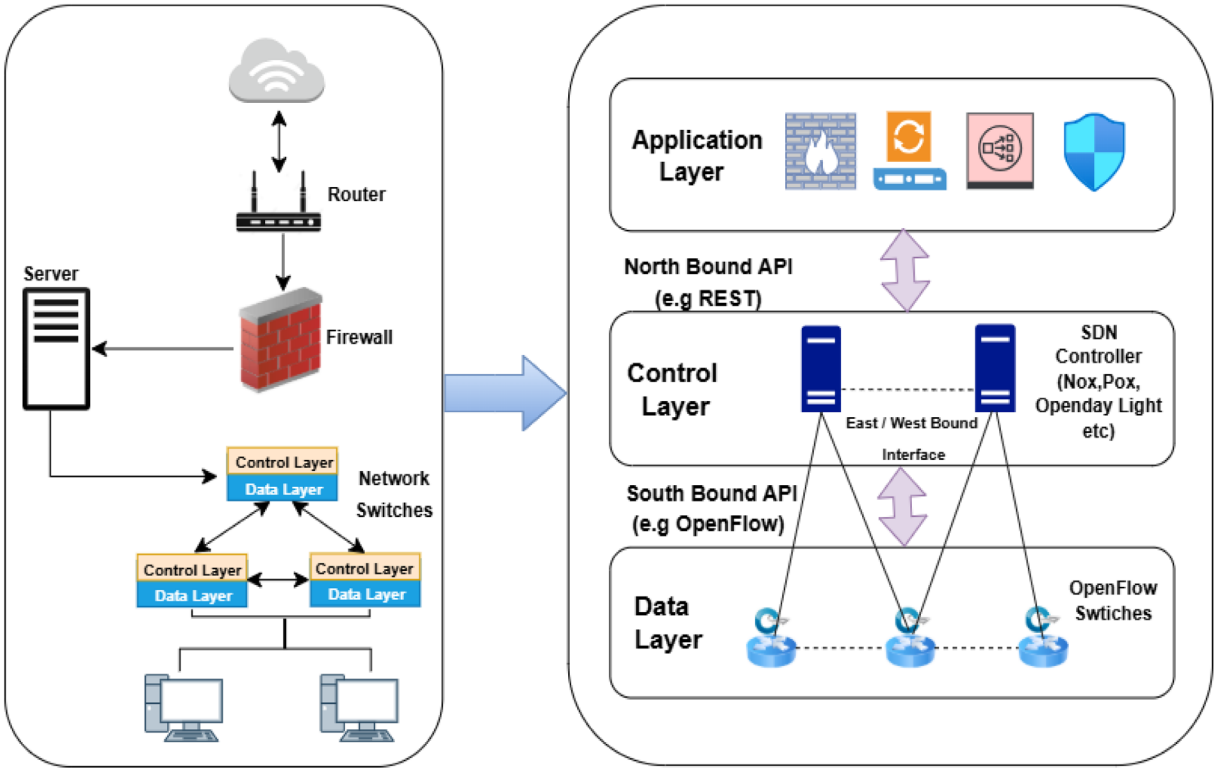

Fig. 1. Transformation from traditional network to SDN.

it is essential to establish strong security measures, such as implementing rigorous authentication, encryption, and redundancy systems, to guarantee the reliability and security of the controller [4].

Using hierarchical controller topologies and dynamic load balancing, SDN makes networks more scalable and faster. These methods disperse control tasks, reduce congestion, and optimize resources. To maintain steady performance, you need to carefully build the system and check it often. SDN's programmability reduces human error and effort by enabling automated, dynamic network management. However, in order to avoid misconfigurations, automation scripts must be thoroughly validated [5].

To ensure the safety of SDN, it is essential to centralize the administration of security policies and to isolate traffic. The SDN controller can implement measures that enhance overall security by providing a comprehensive view of the network. Flow rules facilitate precise traffic control, prevent malicious activities, and enforce security policies [6].

In SDN, separating the control and data planes makes the system more flexible, but it also makes it much less secure because the controller is in one place. A hacked controller can disrupt networks or manipulate traffic. Also, DDoS attacks, are quite likely to hit SDN controllers, which can slow down the network and make services unavailable [7,8]. There are threats to the SDN data plane, which includes routers and switches, such as data leaks, illegal access, and packet modification by spoofing. These kinds of attacks can stop services from working and put data at risk, which can cause legal and security problems. Although SDN provides a flexible approach to network management, inadequate interface protection can lead to network instability and unauthorized modifications [9].

There are three main interfaces in the SDN architecture. The east/westbound interface lets distributed controllers talk to each other; the southbound interface connects the control and data planes; and the northbound interface connects the control plane to the application plane. OpenFlow commonly serves as the underlying communication protocol, facilitating the interaction between the control plane and the data plane [10,11].

The data plane forwards packets in SDN according to the controller's flow table rules; if no match is discovered, a 'packet_in' request starts a table-miss procedure. The performance of the network can be impacted, and bandwidth consumption can increase when entire packets are sent to the controller during periods of high traffic [12,13].

After receiving the 'packet_in' request, the controller evaluates the route for the new stream. It then constructs a flow rule that specifies to the switch how it should direct the subsequent data flow. In SDN, the controller handles requests in the order they come in. But during DoS/DDoS attacks, too many fake packets cause table-miss events to happen all the time. This sends a lot of 'packet_in' requests to the controller, which uses up all of its bandwidth and memory. This causes packets to be dropped and performance to suffer [11,14] When there are a lot of attacks, the control-data plane link gets too busy, which makes communications drop and the controller fail. This stops legitimate traffic from being processed. Because of this,

legitimate flow requests are either delayed or thrown away, which has a big effect on the network's availability, reliability, and performance [15].

The fact that an SDN network can fail or develop problems at the controller level shows how bad things can get to the point where the entire infrastructure can completely collapse [16,17]. The vulnerability that has been acknowledged shows how important it is to have a quick and effective way to find and stop DoS attacks before they have a chance to hurt network operations. It is hard to protect the SDN controller from different and harmful DoS attacks. This requires careful planning and resources to avoid big problems with network performance [18,19].

To make SDN more resistant to DoS attacks, it is important to know how SDN works and how these attacks mess up internal processes. To fix these security holes, SDN needs a full security plan that is made just for its unique structure. Machine learning techniques can tell the difference between bad and good traffic by looking at flow features without looking at the contents of the packets. This strategy leads to a decreased computing load in comparison to conventional packet inspection methods [20,21].

In SDN networks attackers can flood switches with packets aimed at random destinations causing them to constantly contact the controller. This heavy traffic strains key resources such as bandwidth, memory, and CPU power, which in turn slows down responses and leads to packet loss. So, the main challenge here is finding an effective way using ML to detect and limit DoS attacks without compromising the overall performance of the network. Therefore, in this study, the ML approach has been used for detecting DoS attacks, as it can efficiently analyze network traffic patterns and identify abnormal behavior with higher efficiency and lower computing burden. More precisely, the approach utilizes the centralized viewpoint of the SDN controller, which has a complete understanding of the network's structure. By implementing the proposed detection system directly on the controller, it takes advantage of this comprehensive viewpoint, making it extremely efficient in identifying and reducing the impact of attacks. The strategic use of the controller's extensive network visibility improves the effectiveness of the proposed system in detecting and addressing DoS incidents in SDN systems.

This article is broken up into the following sections: Section 2 provides a thorough examination of the techniques employed in SDN systems to identify and address DoS attacks. In Section 3 we describe in detail the design of the proposed system to detect DoS attacks. Section 4 evaluates the effectiveness of the proposed system by conducting a comparative analysis with existing benchmarks. Section 5 outlines the limitations of the proposed system in detail. Section 6 contains the paper's conclusion and offers recommendations for additional research.

2 Related work

This section contains an in-depth investigation of the diverse strategies employed in recent times to detect and protect SDN from DoS/DDoS attacks. Machine learning algorithms have demonstrated significant efficiency in detecting DoS attacks in SDN-based systems, consistently exhibiting robust performance. These techniques make it feasible to identify and counteract DoS attacks that particularly damage the SDN controller's data plane and control plane.

The control plane, which serves as the core component of SDN, is particularly susceptible to security breaches due to its singular point of failure. Detecting DoS and DDoS attacks in the control plane in SDN requires advanced detection mechanisms like behavioral analysis, rate limiting, flow-based detection, hybrid approaches, signature-based detection and anomaly detection.

The control layer may be subject to DDoS attacks if particular applications generate a large number of flow rules that are in conflict with one another. It is the controller's job to decide how a network device should forward a packet, after the device has received the packet, in the data plane, and has not found any matching forwarding rule in its flow table. To process the request, the controller either receives the entire packet or a segment of the packet header. Transmitting a complete packet to the controller can consume a significant amount of data bandwidth due to the large volume of network traffic [22]. Mousavi [23] proposes an entropy technique to reduce the effects of DOS attacks in centralized SDN systems by examining changes in entropy at an IP address. This technique identifies DoS attacks by examining the fluctuations in incoming data packet solicitations. Entropy is related to uncertainty and can be computed with the IP address associated with the packet. Nevertheless, this approach is constrained to centralized SDN and may solely be deployed on an individual host. The adaptive algorithm is described in references [24] and [25] in these threshold is adjusted whenever the calculated entropy approaches the threshold value. Santos et al. [26] introduces an efficient ML Entropy method which makes it easier to find DDoS attacks in SDN by using both entropy-based methods and machine learning. This method teaches a Support Vector Regression model how to use network traffic entropy to automatically change detection thresholds with real word data sets.

Yang et al. [27] developed a new way to spot and stop SYN flooding attacks by using programmable switches. Their system keeps an eye on SYN/ACK packets and automatically updates lists of trusted and malicious devices to block attackers. When tested on a P4 software switch, it cut harmful traffic in half compared to older methods like TCP resets and worked better than the common Bloom filter technique. It not only detected attacks about 2% more accurately but also used only half the memory needed by Bloom filters, making it both efficient and effective under different attack conditions.

Avant Guard [28] and Flood Guard [29] represent two strategies for mitigating DDoS attacks targeting OpenFlow switches. Avant Guard employs a TCP handshake mechanism to identify potential attack sources and create flow rules for data transmission. To detect overload threats and divert traffic towards the data plane cache, Flood Guard employs a proactive flow rule analyzer in conjunction with a packet migration mechanism. While both

approaches prioritize DDoS protection, their detection capabilities are sometimes underestimated. Furthermore, Avant Guard and Flood Guard are not compatible with the standard OpenFlow protocol. FloodDefender [19] is an adaptable and freely available solution created to identify and lessen DoS attacks on SDN systems. The system uses frequency-based characteristics to accurately select and implement mitigation methods such as table-miss engineering, packet filtration, and flow rule planning. Performance testing of a prototype is conducted in both software and hardware settings. Nevertheless, elevated attack rates can lead to erroneous elimination of benign packets, thereby impacting the efficiency of network communication. FloodDefender is a scalable system for OpenFlow networks that helps detect and reduce DoS attacks using frequency features and smart traffic control. It can drop valid packets during heavy attacks and slow communication between planes. Unlike AvantGuard which only works with TCP traffic and needs switch updates, the proposed method offers a simpler flexible way to cut controller load and delay while keeping accurate detection using queuing mechanism.

Tan et al. [30] propose a hybrid K-Means-KNN hybrid machine learning strategy for better detection accuracy when dealing with traffic asymmetry and rapid fluctuations. Detection of DDoS attacks in SDN was explored in [22]. Support vector machines (SVM), genetic algorithms (GA), and kernel principal component analysis (PCA) are some of the ML computational techniques that are combined in [31]. In order to segregate data points, it makes use of kernel functions, generalized arithmetic, and hyperplanes. This helps to improve efficiency of DoS/DDoS attack detection.

For SDN, Peng et al. [32] established a centralized management-free method for detecting DDoS attacks. This method is made up of pre-processing and outlier detection modules. The pre-processing function normalizes and standardizes the flow feature vectors before the anomaly detection technique assigns labels indicating whether or not the features are authentic. Dehkordi et al. [33] propose an approach that utilizes statistical analysis and machine learning to tackle the issue of identifying malicious flows in network traffic. On the other hand, Ye et al. [22] use a SVM to find DDoS attacks in SDN by looking at six-tuple feature values. It's important to be able to accurately classify attacks with as little information as possible and find out where the attack is coming from that is targeting the controller. The SVM classifier is frequently utilized for the purpose of identifying DDoS attacks due to its outstanding accuracy rate and low false-positive rate [34]. Using entropy to determine the degree of unpredictability in flow data, Kumar et al. [35] and Sahoo et al. [36] provide techniques for early identification of DDoS attacks. However, since threshold values do not account for potential variations, Revathi et al. [37] proposed a solution utilizing principal component analysis (PCA). This approach utilizes the collected data to generate new models capable of predicting attacks. DoS attacks may be lessened indirectly by SDN's centralized control, according to Yasin et al. [38], which also enhances connection resilience and handles network problems. Ashraf et al. [39] suggest optimizing routing and improving the scalability of the control plane through the use of deep neural networks alongside additional ML models, thereby strengthening the network's defense against DoS attacks.

Imran et al. [40] suggested a parallel flow deployment method that sets up forwarding rules on all switches from the source to the destination with just one request. This method cuts down on traffic on the control channel and makes better use of controllers during DoS attacks. P. Zhang [41] proposed multi-layer fair queueing (MLFQ) to mitigate DoS attacks by fairly distributing resources among the SDN controller. MLFQ uses queues designated for each switch and port to differentiate attack traffic and ensure equitable scheduling. Employing a Weighted Moving Rate methodology, the controller monitors queue lengths to ensure network stability.

To make security work well and be flexible, our SDN DoS attack detection system uses a queue mechanism to inspect packets and then uses ML classifiers like SVM and KNN for accurate detection. Unlike FloodDefender, which discards legitimate packets and slow down communication, or entropy-based approaches with set thresholds, our system constantly reacts to network conditions. It doesn't need any extra hardware like Avant-Guard and Flood-Guard, and it works perfectly with regular SDN systems.

The literature study focuses on the utilization of machine learning techniques for the purpose of identifying and reducing the impact of DoS attacks in SDN. These systems increase detection accuracy, data handling capacity, and adaptability in real-time to changing hazards as well as their capacity to manage great volumes of data. ML helps lower the risks that come with centralized management and programming, making sure that the network environment is strong and safe. An accurate way to distinguish between legitimate and malicious communications can be achieved with the use of SVM and KNN. These methods provide a rapid response by enabling exact and dependable detection. For detection of DoS attacks, the proposed system is compared to existing systems as illustrated in Table 1.

3 Proposed work and methodology

The main goal of the proposed system (as shown in Fig. 2) is to identify DoS attacks while using as few resources as possible. This proposed system incorporates a detection module into the control layer of the SDN architecture. In this architectural framework, the detection module utilizes a two-step process to identify and address potential attacks. At the first stage, packet filtering is performed to examine the packets and detect any malicious behavior. Then Machine learning classifier models, particularly KNN and SVM, are subsequently utilized to accurately detect attacks. This approach is better than direct classification as packet filtration will reduce the traffic load for the classifiers and to improve efficiency at the controller.

We have selected SVM and KNN for classification, as both models exhibit outstanding performance in flow-based traffic statistics and are comparatively lightweight. SVM works well in high-dimensional feature spaces

Table 1. Generic comparison of prior systems with proposed system.

Feature	What prior systems does	Proposed system
Packet Queuing Strategy	Queuing is mostly rule-based or static; suspicious traffic is handled after table-miss events or via fixed thresholds; the queue itself is not steered by a learned model.	A controller resident queuing design that separates normal and suspicious flows before classifying, cut down unnecessary packet drop and there is a time limited blacklist that automatically unblocks.
Any Feature Selection Method	Feature sets are either fixed (hand-engineered or filter-only) or not adjusted for the specific classifier	Wrapper Based forward feature selection process for choosing compact features that makes the controller more accurate and less time consuming for performance.
SDN Integration	Detection often runs off the controller (as a separate module or service) and is loosely connected to mitigation, and this raises delays and controlling impact.	Ryu Controller: packet filtration to queuing mechanism and then classification from alert to block/unblock, which will simplify all the operations.
System Architecture	Detection and other parts are divided among components (like neighbor-switch detours, caches, and proxies) and may need changes to the device or protocol.	Unified pipeline with low overhead on SDN; has a reconnect policy so that benign hosts can start up again when things go back to normal. A cohesive pipeline cuts down on false positives and control plane load while making things better for real users.

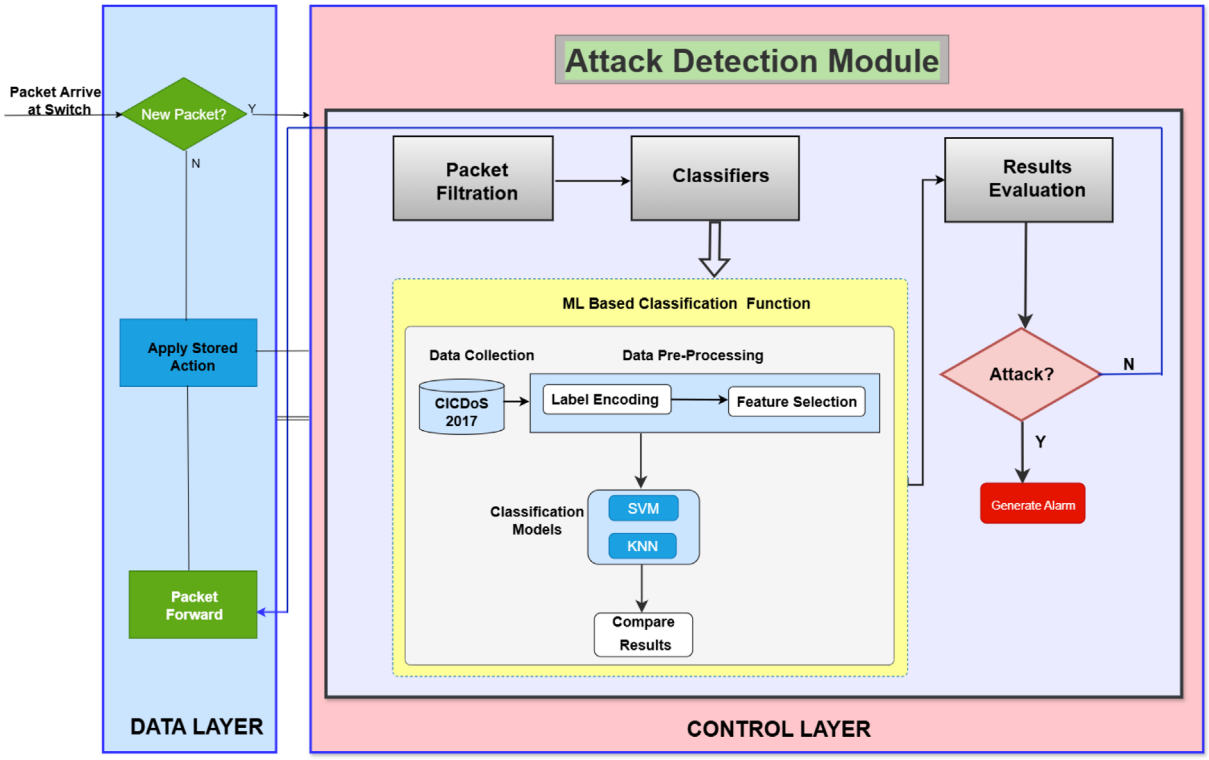

Fig. 2. Proposed methodology for identification and detection of DoS attack.

because it generalizes well and is stable. KNN, on the other hand, may observe nonlinear data patterns without making any assumptions about the distribution. The two of them served as a positive test of how well the suggested framework could identify things.

The KNN algorithm, as described in [42] is a supervised learning technique that can be applied to both classification and regression tasks. It exhibits a lack of motivation. The learning algorithm employs a memory-based storage mechanism during the training phase, wherein all training

data is utilized for classification purposes. By using a majority vote, it assigns a label to each new instance of data based on how similar it is to previous instances in the dimensional space. If the majority of the neighboring instances have class 'y', a new instance will be labeled 'y'. The calculation of similarity involves the utilization of a distance metric.

SVM is a supervised learning algorithm that is universally applicable for both classification and regression tasks. The primary objective of a support vector machine is to identify the optimal hyperplane in an n-dimensional space (where n represents the number of features) that accurately classifies the data points. The process of two-step learning is implemented through the following methods. Initially, the process involves graphing the inputs in a space with n dimensions, where each individual attribute coordinate is represented by support vectors. Additionally, a determination will be made regarding the optimal hyperplane construction for the separation of instances. SVM employs kernel techniques to effectively represent intricate non-linear functions, thereby minimizing computational complexity [42].

We choose these models because they can reduce the computational load on the control plane, both in terms of data storage and processing power. The functioning of the proposed system entails a series of consecutive processes. When an SDN switch receives a data packet, it initially verifies if the packet corresponds to any previously handled packets by comparing its source, destination, and receiving port details with the entries in a flow database. When a match is found, the packet is processed in a manner that is appropriate for the situation. If table-miss (the switch can't find a matching entry in its table) takes place, the switch will transmit an OpenFlow packet-in message to the SDN controller.

Due to this, the controller will be prompted to broadcast OpenFlow Flow Mod messages to configure the requisite forwarding rules based on the packet's destination address. Additionally, the controller conducts data filtering to remove any abnormal traffic. The proposed method combines both packet filtering and classifiers to create a model that can clearly tell the difference between normal SDN flow packets and those that show a denial-of-service attack.

3.1 SDN's controller attack detection module

When data plane switches are configured with their default settings, they utilize a single First-In-First-Out (FIFO) buffer. This buffer lacks the ability to distinguish between legitimate and illegitimate traffic. In response to this difficulty, a novel technique has been suggested, which entails creating a very efficient approach for configuring the SDN controller.

This technique seeks to improve the overall operating efficiency of the system. Figure 2 depicts the incorporation of the detection module into the SDN RYU controller as part of the manufacturing process. RYU is a freely available SDN controller that aims to simplify the management of networks and the customization of traffic, hence improving the flexibility of the network. The RYU controller serves as the main element in the SDN environment. It coordinates network operations and promotes data interchange with switches and routers via southbound APIs. Additionally, it allows communication with applications and business logic through northbound APIs. Mininet, one of the most well-known network emulators in the industry within the SDN domain, is the one used in the research.

In order to identify and prevent DoS attacks, our detection module is implemented within the SDN environment. The first phase involves the utilization of packet filtration technologies to identify abnormal traffic by analyzing packet flows. After identifying patterns, classifier models use machine learning techniques to identify and stop malicious attacks based on the data linked to these patterns. In order to mitigate the burden on the control plane and sustain a consistent communication capacity, it is feasible to employ established models such as KNN and SVM. These models are widely recognized for their remarkable precision in differentiating between malicious and benign packet flows. This strategic approach aims to safeguard and optimize the utilization of existing communication resources.

3.2 Attack detection functions

3.2.1 Packet filtering

In an SDN environment, the packet filtering process is designed to identify and isolate malicious packets that may indicate a DoS attack. To ensure network security and optimize performance, this process involves continuous monitoring, packet analysis, and adaptive decision-making.

The system operates in an endless loop, tracking all incoming packets. It initializes two primary queues, one for malicious packets and another for normal packets, along with timing variables, frequency thresholds, and counters to monitor packet frequency. Additionally, a blacklist records IP addresses associated with malicious or invalid transmissions (as shown in Algorithm 1). The system verifies its IP address immediately upon getting a packet. When a packet arrives from the same IP address as an earlier one, the system keeps track of when and how often it arrived and adds one to the IP address counter. The system verifies the legitimacy of a packet originating from an unfamiliar IP address. In order to stop further malicious activity, unverified IPs are instantly blacklisted, their data are recorded, and the blacklist is updated.

Packets originating from allowed IP addresses are sent to the normal queue, allowing traffic to be transmitted to the SDN controller without interruption. The time variable in Algorithm 1 shows how long it takes for packets from the same source IP address to arrive one after the other. It helps figure out if packets are coming in too quickly, which could mean that they are behaving strangely. The system looks for strange traffic patterns by comparing how often packets from the same IP address arrive and how long it takes for them to arrive. If either the frequency goes above the set limit (for smaller networks, that's 40–50 packets per second; for larger networks, it's 90–100 packets per second), or the time between packets drops below the

set limit, the traffic is marked as abnormal. In these cases, the packet will be sent to the malicious queue for more investigation or blocking. The proposed queuing mechanism has a blacklisting process that changes over time. When an IP address attacks many times, it is temporarily blocked and put to the blacklist. After a certain amount of time has passed, the IP address is reassessed regarding each individual entry. As soon as the IP address is no longer seen as malicious, it is removed from the blacklist and communication returns to normal. By taking this technique, we can keep persistent attackers under check and avoid permanently blocking legitimate users.

The packet filtration method effectively separates anomalous packets, which are suggestive of DoS attacks, from genuine traffic by continuously monitoring and analyzing packet traffic in this manner. The dynamic and real-time processing of data in the SDN environment ensures the security and integrity of the system. It instantly identifies and isolates bad traffic while allowing the uninterrupted flow of valid traffic. The technique of queue based packet filtration is mentioned in Algorithm 1.

Algorithm 1: Queue based Packet Filtration in Attack Detection

Input: Incoming new packet, switch Id, IP address
Output: Abnormal packets in the malicious queue

1. Initialize counter, time interval, frequency, threshold amount, malicious queue, normal queue, Banned list time threshold = predefined minimum inter-arrival time between packets, frequency threshold = predefined maximum allowed packet rate (packets/second)
2. **while** true **do**
3. Collect packet from buffer
4. **If** packet from same IP **then**
5. Increment counter
6. Record time and frequency
7. **else**
8. **If** invalid IP **then**
9. Block communication
10. Add that IP to Banned list
11. Log packet and source IP
12. **else**
13. Send packet to normal queue and then to controller for action
14. Packet send to destination
15. **end if**
16. **end if**
17. **If** time interval < time threshold OR frequency> frequency threshold amount **then**
18. Send this abnormal packet to malicious queue
19. **Continue** it for next iteration
20. **else**
21. Send packet to normal queue and then to controller for action
22. Packet send to destination
23. **end if**
24. **end while**

3.2.2 Attack detection classifiers

The process begins with a labeled dataset of network traffic, which is pre-processed to ensure consistency and readiness for analysis. An iterative loop is then used to continuously monitor incoming packets. Within this loop, a wrapper-based forward feature selection method is applied to enhance detection performance by identifying the most relevant features from packets in the malicious queue. This iterative process builds the feature subset step by step while continuously evaluating the performance of the classifier itself rather than relying only on statistical measures such as correlation or variance. We began with no features and progressively added one feature at a time. This method systematically evaluates different combinations of features derived from packet headers and network traffic statistics to determine the most effective subset for distinguishing between normal and malicious traffic. The original dataset contains 79 features, but feature selection is employed to reduce dimensionality and improve model efficiency by eliminating less significant attributes while retaining those most critical for accurate attack detection.

During each iteration of the loop, the program computes pertinent characteristics from packet headers and network statistics. The selected features are meticulously chosen to capture essential aspects of packet behavior that signify possible DoS attacks.

After extracting the features, they are fed into classifier models like SVM and KNN. These models assess the feature sets to find out whether the packet has behaviors that specify a DoS attack. As a result, attackers are unable to overrun the control plane. This ensures that only abnormal packets are halted, preventing any disruption to normal traffic.

3.2.2.1 SVM based classification

To identify DoS attacks, our SVM system sorts network data according to an ideal decision boundary that distinguishes between benign and malicious packets. To do this, it finds an optimal hyperplane that better separates the two classes, which increases the accuracy of the classification. The algorithm is trained here with specific traffic features obtained through the forward feature to guarantee accurate detection. The decision function is employed using equation (1).

$$I(k) = a^T k + m. \tag{1}$$

Here the variable k denotes the feature vector of the incoming packet, which is derived from the optimized feature set during feature selection process. Another variable a is the weight vector that determines the orientation of the hyperplane within the respective feature space. T denotes the transpose of the weight vector. The bias term, denoted as m, is responsible for moving the hyperplane away from the origin.

SVM uses the standard Polynomial Kernel Function for data that is not linearly separable. So in order to determine if incoming packets are malicious or benign, the SVM decision function is used in real-time filtering. Once identified as malicious, a packet is immediately directed

to the malicious queue for remediation. Assuming a normal evaluation, the packet is forwarded to the usual processing queue without interruption. This approach improves network security and performance by using kernel-based improvements to increase the accuracy of DoS detection, guarantee effective classification, and decrease the number of false positives.

3.2.2.2 KNN based classification

To find DoS attacks, our KNN-based function looks at the feature vector k_p of incoming packets and compares it to a dataset that has already been processed. The Euclidean distance metric is utilized in order to determine the measure of similarity that exists between the incoming packet as well as the data points that are already there. This method makes sure that nearest-neighbor searches work well and is good for numerical traffic attributes because it figures out the straight-line distance between feature vectors.

We construct the feature vector for each incoming packet and input it into the KNN classifier model. For our proposed attack detection system, we employ the Euclidean distance function to compute the distance between the feature vector and each data point in the pre-processed training dataset, as mentioned in equation (2).

$$d(k_p, k_q) = \sqrt{\sum_{n=1}^{s} (k_{pn} - k_{qn})^2}. \quad (2)$$

The variable s represents the number of features in the feature vector, while n serves as an index for summation and does not represent the nearest neighbors. The method identifies the k nearest neighbors to the incoming packet's feature vector (denoted as k_p) based on the smallest Euclidean distances. A majority voting mechanism among these k nearest neighbors is used to determine the classification of the incoming packet. In our experiments, we set $k = 5$, meaning the five closest data points in the feature space are considered for classification. If the majority of these neighbors are designated as malicious, the packet is categorized as a DoS attack and routed to the malicious queue for subsequent action. Alternatively, it is transmitted without any problems and handled like normal traffic.

This attack detection module operates continuously, ensuring the immediate identification and response to potential DoS attacks within the SDN environment. By utilizing the wrapper feature selection method in combination with advanced machine learning algorithms, it improves network security by rapidly detecting threats while ensuring the uninterrupted flow of authorized network data. Algorithm 2 outlines the primary processes carried out for the attack detection procedure by using Classifiers. The classifiers technique is described in Algorithm 2.

Algorithm 2: Attack Detection Module (Classifiers)

Input: Malicious queue packets with features, Dataset
Output: DoS Attack Detected
1. Provide dataset
2. Pre-process dataset
3. **while** true **do** // Apply Feature selection method
4. Feature list = wrapper ()
5. Input Feature list to classifier models
6. Choose best results
7. **If** Model detects Attack **then**
8. Add that IP to Banned list
9. Log packet and source IP
10. Generate Alarm
11. Send this identified attack packet to SDN controller
12. **else**
13. Send packet to normal queue
14. Packet send to destination
15. **end while**

By using these two classification strategies, the likelihood of incorrectly labelling normal packets as malicious is reduced. In the end, after evaluation, SVM was chosen because its detection accuracy was better than KNN.

4 Performance evaluation

4.1 Dataset CICDoS 2017

The CICDoS 2017 was employed as the primary dataset for training and subsequently assessing the various classification models utilized in this research. Previous research efforts have demonstrated the dataset's capability to overcome various challenges and limitations effectively. We retrieved and computed a comprehensive set of twenty network traffic features out of 79 features (some of these features are provided in Tab. 1), encompassing both benign and DoS flows. These features were integrated into the categorization process utilizing the CICDoS 2017 dataset, which comprises observations obtained from network traffic analysis, along with flows tagged with specific attributes. Information such as timestamps, protocols, attack types, source and destination IP addresses, ports, and protocols is all part of the labeling process. We removed any values deemed outliers, infinite, or empty before shuffling the data.

The dataset was divided into a training set and a testing set using random flow-level sampling rather than host-based separation. By employing this random division, it was demonstrated that each subset displayed an adequate proportion of attacked and normal traffic flows. Random division was used to eliminate host-level bias, which might have led to findings that were too optimistic if many flows from the same host were included in both groups. This bias was eliminated using random division. For conducting an evaluation that is both fair and accurate regarding the precision of detection, the classifiers were trained and evaluated on a variety of traffic patterns using random sampling.

To enable binary classification, category labels were converted into an integer format, with a value of 1 denoting a safe category and 0 indicating a dangerous one. The data underwent a two-step process involving indexing string values and subsequent normalization to mitigate misclassification. The collection consists of two distinct categories: legitimate records and malicious records. The normal records group comprises 2,153,720 instances, whereas the malicious records group accounts for 49.98% of the dataset.

4.2 Data preparation and feature selection

Data preparation is a critical phase in machine learning that involves important tasks like cleaning, pre-processing, wrangling, and feature engineering. The primary goal is to convert unprocessed data into a structure that can be easily utilized by machine learning algorithms, facilitating the identification of valuable information and the ability to make predictions. One of the critical challenges in preparing the CICDoS2017 dataset involves addressing missing or incomplete data. Correcting errors in the CICDoS2017 dataset is a vital aspect of the data cleaning process. Eliminating columns with missing or incomplete data can enhance the performance of machine learning models. Addressing the absence or incompleteness of data in the CICDoS2017 dataset is essential for ensuring accurate analysis and reliable results. Many columns in the dataset do not have sufficient data, which undermines the effectiveness of machine learning models. Attributes such as forward bytes per second, packets per second, bytes per block rate average, and bytes per block rate average sometimes have null values in many records, so they need to be removed from the dataset. Within situations involving continuous categorization, a value of 0 indicates a secure state, whereas a value of 1 represents the occurrence of a DoS attack. Furthermore, machine learning models face difficulties when dealing with categorical data found in the CICDoS2017 dataset, which requires their removal in order to train the models effectively. To streamline data processing, the CICDoS2017 dataset removes columns with a blank percentage over 50% and rows with a blank percentage above 5% in any column. Furthermore, to safeguard the authenticity of the data and enhance the accuracy of the model, any erroneous data, such as negative values, within the CICDoS2017 dataset is detected and eliminated. This is done to guarantee the completeness and accuracy of the data.

The wrapper feature selection method's forward feature selection approach starts with a feature set that is empty and then adds the features that are the most relevant one at a time in an iterative manner. The feature that yields the greatest substantial enhancement to model performance is chosen and incorporated into the set. This process will continue until there are no additional improvements in performance detected for any reason. Out of the 79 features in the dataset, only twenty were chosen for analysis. The selected features are employed to classify network traffic flow and differentiate between legitimate and malicious operations. Some of these selected features can be seen in Table 2.

The overall classification methodology proposed for both models is shown in Figure 3.

We train the model using the selected set of features (characteristics) at each stage of the analysis, which involves processing and filtering the most relevant traffic attributes. To comprehensively assess the performance of the trained model, we use four key evaluation metrics: accuracy, precision, recall (sensitivity), and F1-score. Accuracy measures the overall correctness of the model in classifying both normal and malicious traffic. Precision evaluates the proportion of true positive detections among

Table 2. Packet features.

Feature name	Description
Protocol	Protocol Type
Flow Duration	Duration in milliseconds
Total Fwd Packets	Total number of forwarded packets per flow
Source IP	IP address of source
Destination IP	IP address of destination
Source Port	Source port of flow
Destination Port	Destination port of flow
Fwd Packets Length Total	Total length of forwarded packets
Bwd Packets Length Total	Total length of backward packets

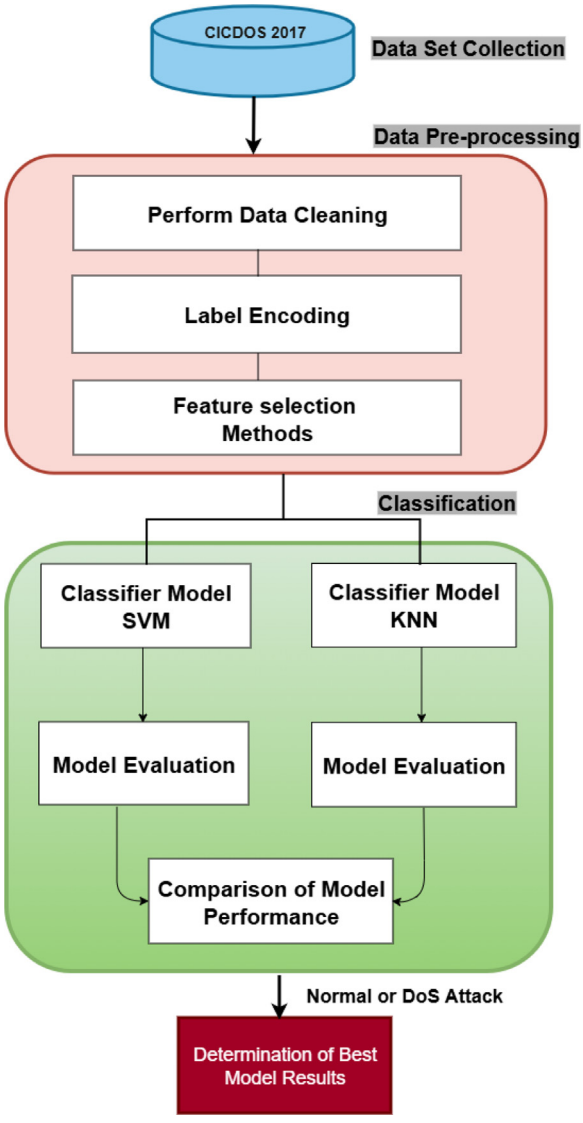

Fig. 3. Proposed classification methodology.

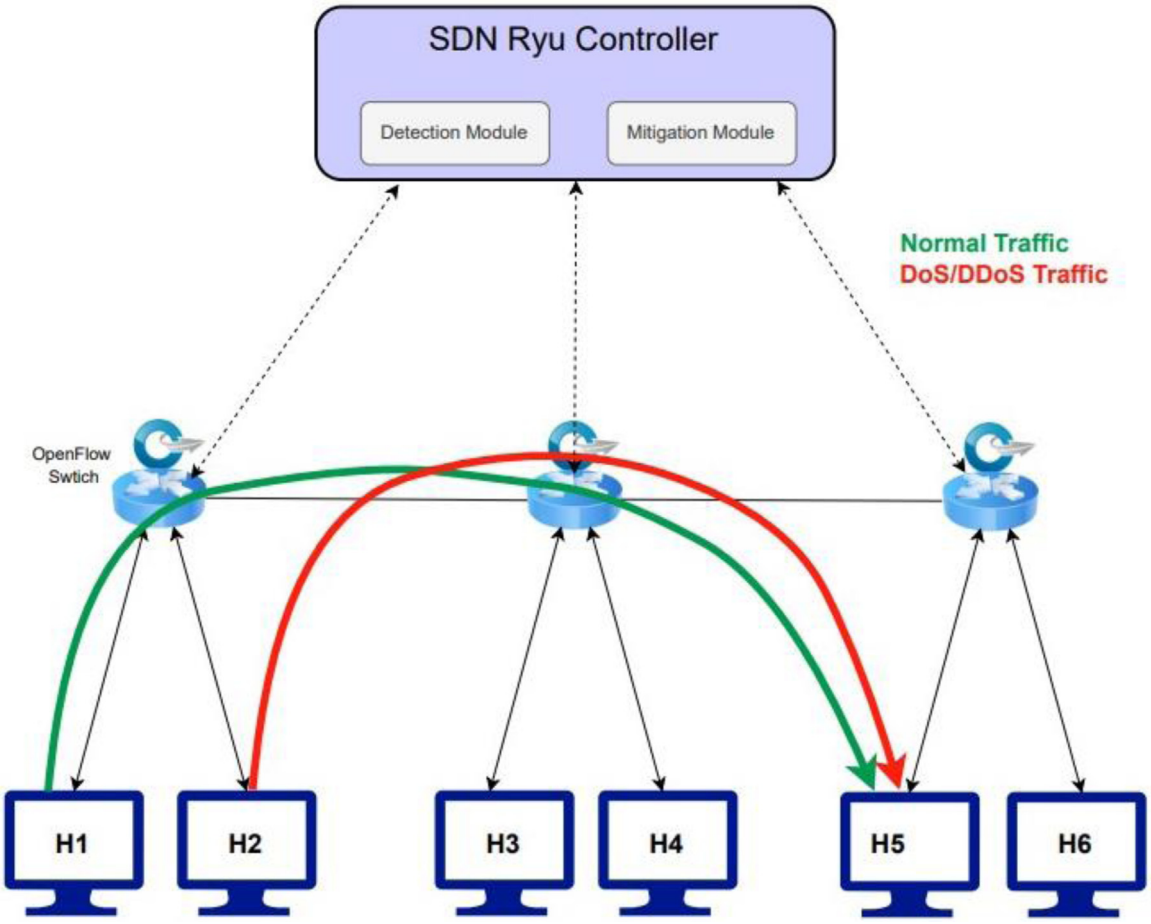

Fig. 4. Network model of the proposed system.

all instances classified as malicious, indicating how well the model avoids false positives. Recall measures the model's ability to correctly identify actual malicious instances, minimizing false negatives. Finally, the F1-score provides a balanced metric by combining precision and recall, offering a more meaningful assessment when dealing with imbalanced datasets, such as network traffic where attack packets may be fewer than legitimate ones. This comprehensive evaluation ensures the model performs reliably under various traffic conditions.

4.3 Experimental setup

In order to carry out the research experiments, a personal computer (PC) that was running the Ubuntu 18.04.2 LTS operating system was utilized. The PC is equipped with a dual-core Intel Core i7 8th Generation CPU operating at a frequency of 2.4 GHz. While SDN simulations can utilize many controllers such as RYU [43], FloodLight [44], and NOX [45], our study exclusively employed the RYU controller.

A tree-type network with a depth of 2, consisting of 3 switches and 6 hosts, was built using Mininet, as depicted in Figure 4. The Scapy library [46] was employed to generate packets, taking advantage of its effectiveness in producing, searching, sniffing, attacking, and forging packets. Two categories of traffic were produced: normal traffic and aggressive traffic, with Scapy being utilized for altering the source IP address using spoofing techniques. Attack traffic was routed through six compromised hosts, while the remaining hosts transmitted ordinary data packets. To successfully install the Ubuntu operating system, it is advisable to set up a virtual machine (VM) environment. Within this environment, the deployment of the SDN network and switches is conducted.

Additionally, the following installations are necessary to streamline the experimental process:

- Begin by establishing the Ryu controller application, which is implemented in Python.
- It is advisable to utilize an integrated development environment (IDE) like PyCharm when implementing the Ryu controller. The integration of Ryu with switches enables the efficient routing of incoming packets for processing.
- After establishing a connection, the Ryu controller engages in packet analysis in order to extract conclusions from the data it receives.

4.4 Performance analysis

The performance of the proposed system has been evaluated and compared with prior ones [10,13,19,37]. The results of the comparison are presented below.

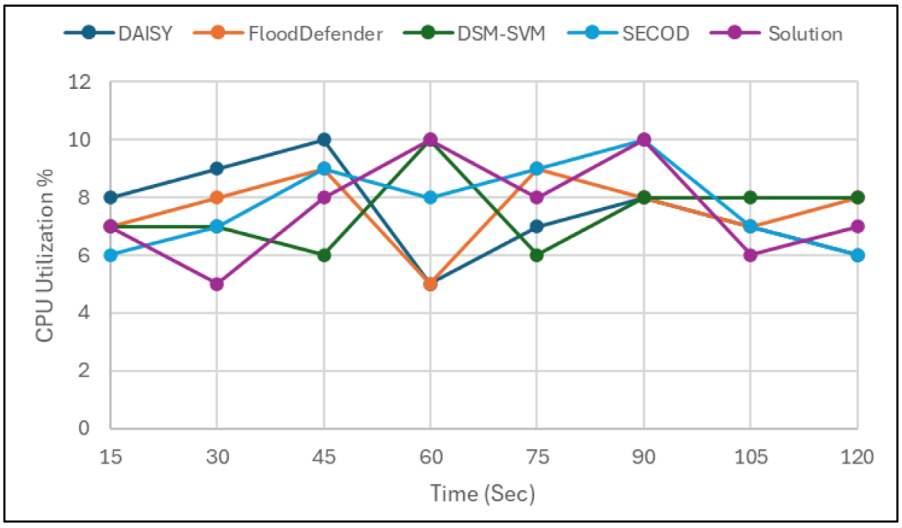

Fig. 5. Analysis of CPU utilization without DoS attack.

4.4.1 CPU utilization

A crucial statistic that reveals how computational resources are distributed while dealing with malicious activities specifically with respect to DoS attacks is the CPU utilization in SDN. In a typical workload, CPU use tends to be somewhat constant. However, when attacks are taking place, there is a possibility of an increase in CPU utilization, as SDN components attempt to detect, respond to, and mitigate threats. Because of this, it is an essential factor to monitor and manage in order to ensure the network's performance and safety. The average amount of CPU use that our system and benchmarks offer as a reference can be seen in Figure 5. This level is achieved during regular traffic levels. Overall, the average CPU utilization rates of DAISY, FloodDefender, DSM-SVM, SECOD, and our proposed system are nearly the same, coming in during normal traffic, as shown in Table 3.

In contrast, Figure 6 illustrates and compares the typical CPU utilization of the proposed system during a DoS attack scenario, following standard benchmarks. During the attack, the controller's CPU experiences increased load due to the processing of excessive malicious flow requests, which reduces its capacity to handle legitimate traffic. Once the attack is detected, CPU utilization rises as the system filters abnormal packets and redirects them to the malicious queue. This filtering process effectively reduces the controller's workload over time, leading to a noticeable decrease in CPU utilization. The proposed system demonstrates greater stability and efficiency in managing CPU resources compared to existing methods, ensuring better handling of legitimate flow requests even under attack conditions.

In Figure 6, it can be observed that the average CPU consumption during DoS attacks is approximately identical for the systems studied by DAISY., FloodDefender, DSM-SVM, SECOD, and our proposed system. The results indicate that our system effectively optimizes the computational resources of the control layer. Table 3 depicts the average percentage of CPU time spent actively processing data during a DoS attack, as measured by several benchmarks. While Figures 5 and 6 illustrate the variation in CPU utilization (%) over time at multiple intervals (15 to 120 s) under different methods. This approach provides both a detailed time-based performance visualization in Figures 5 and 6 and an overall performance summary in Table 3 for easy comparison.

Table 3. Analysis of CPU average utilization.

Methodology	Without DoS attack (%)	With DoS attack (%)
DAISY	7.50	8.50
FLOODDEFENDER	7.63	14.87
DSM-SVM	7.52	9.75
SECOD	7.75	11.50
Proposed System	7.62	9.13

4.4.2 Control channel bandwidth

In SDN, the bandwidth available on the control channel is critical to the smooth operation of the controller and the network components. Figure 7 illustrates the comparison between our system and benchmarks in terms of the proportion of control channel bandwidth utilized during normal traffic. It shows that DAISY., Flood Defender, DSM-SVM, SECOD, and our proposed system have approximately equal control channel bandwidths for normal traffic also shown in Table 4.

However, the presence of both malicious and benign network traffic adds complexity to the challenge of differentiating between them. The OpenFlow specification states that a switch must store a packet in its flow buffer if it comes across one for which it does not have any predefined instructions.

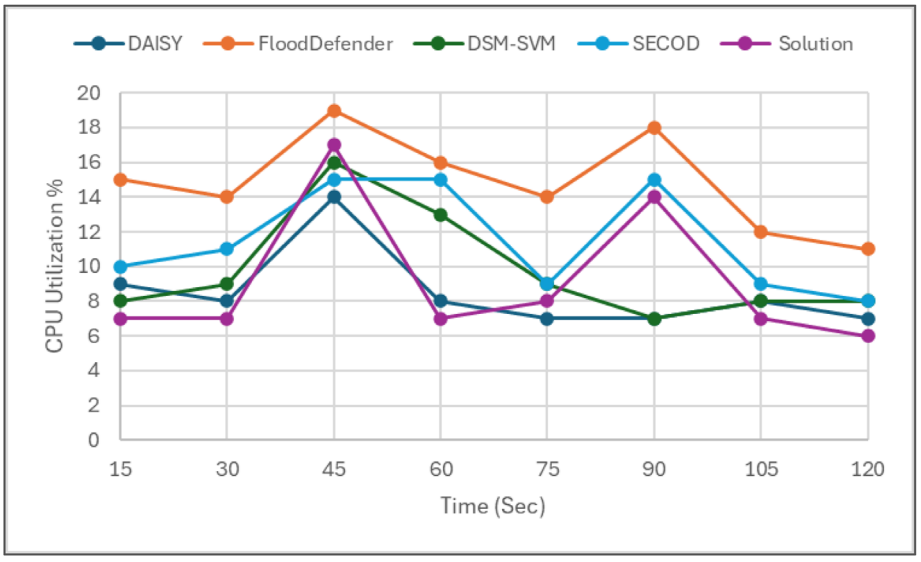

Fig. 6. Analysis of CPU utilization with DoS attack.

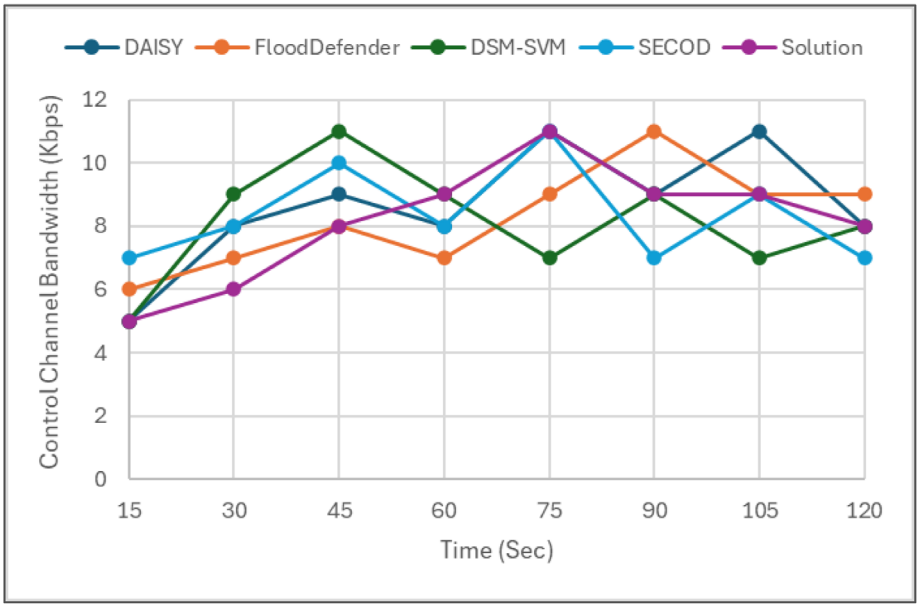

Fig. 7. Analysis of control channel bandwidth without DoS attack.

Table 4. Analysis of control channel average bandwidth.

Methodology	Without DoS attack (%)	With DoS attack (%)
DAISY	8.63	15.38
FLOODDEFENDER	8.25	15.13
DSM-SVM	8.13	15.54
SECOD	8.34	15.25
Proposed System	8.13	14.13

Following this, the switch transmits a Packet-In message to the controller in order to inquire about additional instructions regarding the handling of packets. This procedure is essential in software-defined networking environments to ensure switches effectively manage incoming packets. Nevertheless, it introduces latency to packet processing, impacting network responsiveness. Efficient packet processing techniques are crucial for minimizing latency and ensuring uninterrupted network functionality.

To assess bandwidth, a series of packets are transmitted from a single IP address within a 5 s timeframe. Observations indicate that the influx of packet flooding results in a maximum rise of 30 (kbps) in bandwidth. However, this increase is transient, as the bandwidth subsequently declines. After a duration of 30 s, the bandwidth once again experiences an upward trend, as illustrated in Figure 8.

The controller might suffer from exhaustion of resources in the case of a DoS attack, which is essential for handling normal network traffic. This situation arises when flooded traffic comes to the network with a

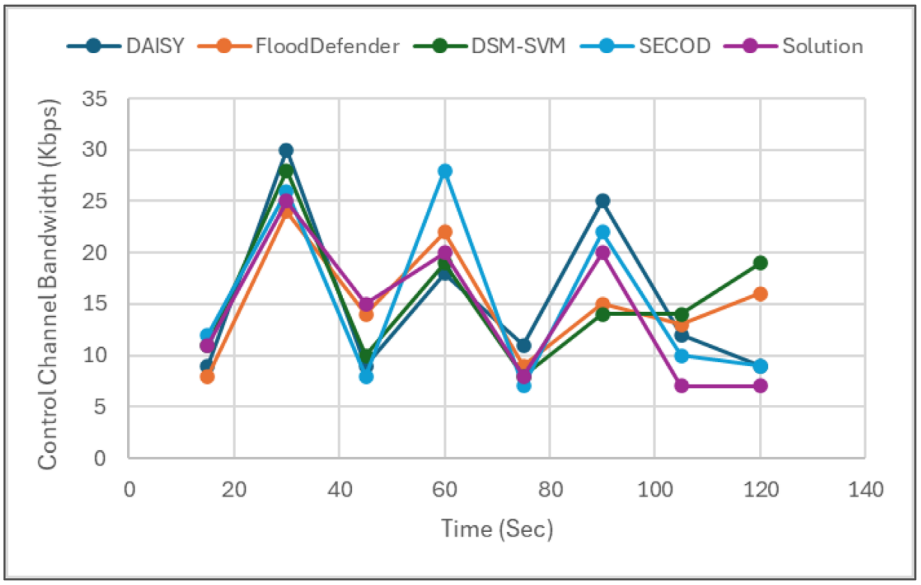

Fig. 8. Analysis of control channel bandwidth with DoS attack.

substantial volume of Packet-In messages within a short timeframe. Moreover, offensive traffic has the potential to fully use the accessible bandwidth connecting the controller and the switches, resulting in a significant degradation of network performance. In the event of a DoS attack, the control channel may experience congestion, thereby impeding the passage of requests for new flows.

It is advised to impede specific network traffic at switches. If an IP address that is on the blacklist is received for a flow request four times, the controller will discard the incoming packet, and the switches will prevent any further communication from that IP address and that the incoming traffic from that IP will be restricted at switches.

The proposed system offers the additional benefit of improved consuming control channel management by achieving 14.13% bandwidth, in contrast to benchmark solutions that experience bandwidth of 15.38%, 15.13%, 15.54%, and 15.25% respectively as shown in Table 4, It clearly shows approximately 1% reduction in bandwidth.

4.4.3 Packet delivery ratio

Switches in an OpenFlow network employ flow rules to effectively handle different messages originating from SDN controller. These switches ascertain the correspondence between incoming packet headers and their flow tables, with each correspondence initiating a distinct action. When the flow table does not contain any matches, it signifies that the packet does not conform to any pre-existing rule. This procedure is crucial for overseeing the movement of data in SDN.

Under typical traffic conditions, the rate in which incoming packets are send to the controller generally falls within the range of 200 to 400 packets per minute, as observed across different benchmarks as shown in Figure 9. Nevertheless, in the context of DoS attacks, the influx of data into a switch escalates.

As a reaction, the controller must incorporate additional regulations into the flow table. The controller performs a comparison between incoming packets and its database. Packets that are included in the banned list within the packet filtration module are automatically dropped. Alternatively, in the absence of a match, a fresh entry is generated in the flow table.

During DoS attacks as shown in Figure 10, there is a notable increase in the average flow rate of packets per minute (PPM), with values such as 5565, 5209, 5874, and 5272 observed across various benchmarks.

Nevertheless, the implementation of the proposed system, which incorporates a packet filtration module, results in a reduced average flow rate of 5197 packets per minute(PPM). The decrease in the number of new flow table entries can be attributed to the effective management of incoming packets.

Detailed comparison of packet delivery ratio with benchmarks is mentioned below in Table 5.

4.4.4 Evaluation of latency

Figure 11 shows the average response times for the proposed system and the tested solutions during normal traffic. H1 is used as a baseline reference point in the latency evaluation figures because the originating host always has low response times. H2, H3, and H4 are used to see how well the proposed system works under normal traffic and attack conditions.

The proposed system has an average response time of 4.25 (ms) better than the others because its queue management system separates normal and malicious packets into separate queues, which lowers latency. Figure 11 shows the average response times for the proposed system and the tested solutions during normal traffic. The proposed system has an average response time of 4.25(ms) better than the others because its queue management system separates normal and malicious packets into separate queues, which lowers latency.

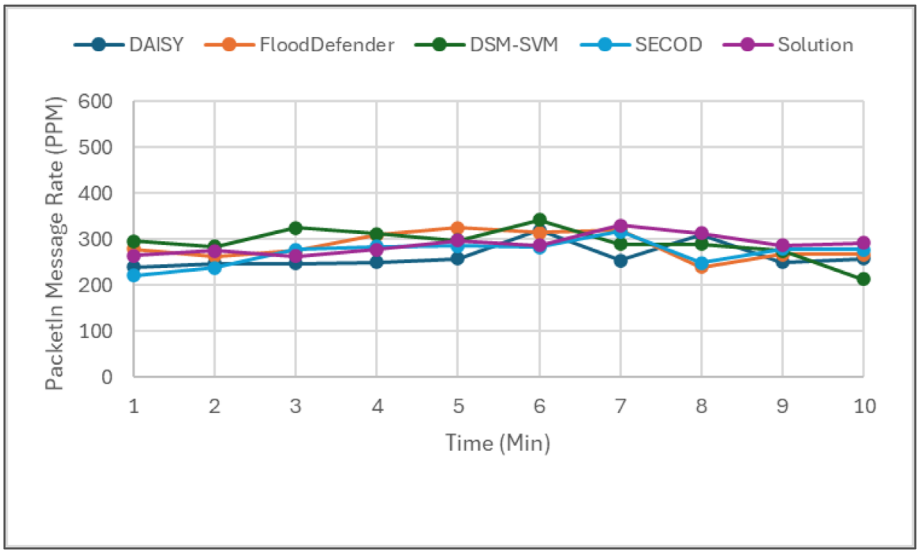

Fig. 9. Analysis of packet delivery ratio without DoS attack.

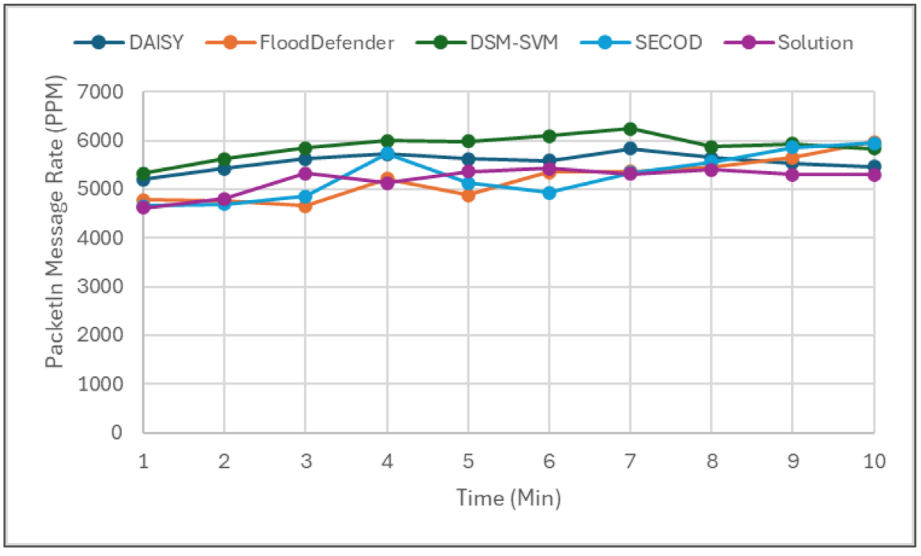

Fig. 10. Analysis of packet delivery ratio with DoS attack.

Table 5. Packet delivery analysis (packets per minutes).

Methodology	Without DoS attack	With DoS attacks
DAISY	263	5565
FLOODDEFENDER	286	5209
DSM-SVM	293	5874
SECOD	271	5272
Proposed System	289	5197

In comparison, the benchmark systems tend to show higher average response times, with values of 4.50(ms) for H1, 4.7(ms) for H2, 5.7(ms) for H3, and 5.0(ms) for H4. In our case, H1 consistently shows faster response times because it mainly sends out requests and doesn't really get affected by queuing delays or filtering steps. On the other hand, H2, H3, and especially H3 and H4, show noticeable improvements with our proposed system, where the effects of heavy traffic and filtering are much more significant. In comparison, benchmarks indicate higher average response times detailed in Table 6.

As shown in Figure 12, the proposed system keeps an average response time of about 6.50(ms) under DoS attack conditions, which is similar to some of the benchmarks. The two-phase filtering mechanism makes this possible by blocking attacked traffic and keeping the controller's performance high for legitimate hosts. The response times of the benchmarked systems during DoS attacks stay at 6.0, 6., 6.5, and 5.5 from H1 to H4. In general, the proposed system always keeps response times fast, whether there is normal or attack traffic. This is especially helpful for hosts that have to deal with a lot of traffic.

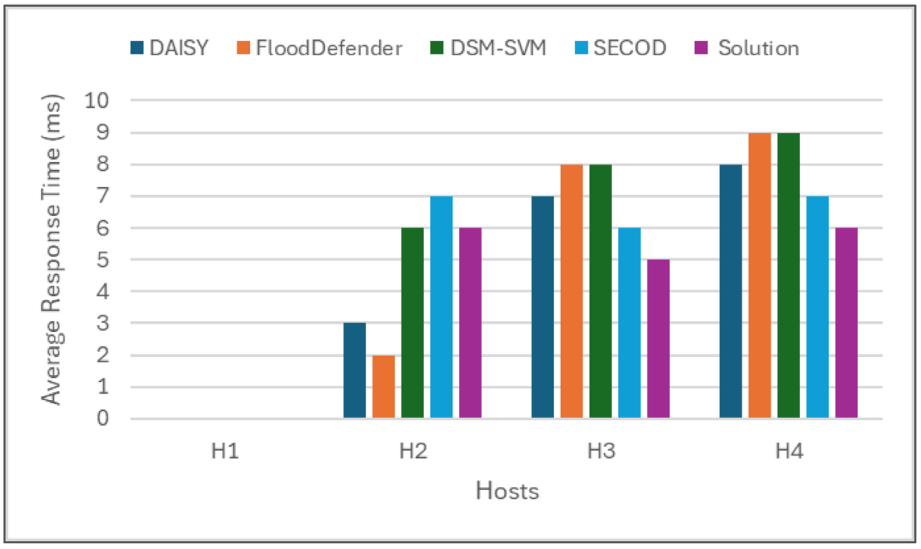

Fig. 11. Average latency analysis without DoS attack.

Table 6. Average latency analysis (ms).

Benchmark	Normal traffic	With DoS attacks
DAISY	4.50	6.0
FLOODDEFENDER	4.75	6.5
DSM-SVM	5.75	6.5
SECOD	5.00	5.5
Proposed System	4.25	6.5

When analyzing DoS attacks, as illustrated in Figure 12, our system's average response time increases by 6.50(ms), which is comparable to the two other methods depicted in Figure 12 and Table 6. This data shows that our system works just as well as others during attacks. Overall, it gives a more balanced result by keeping response times low during normal traffic and handling malicious traffic well when the network is under attack.

4.4.5 Attack identification false rate

The effectiveness of the proposed approach in identifying DoS attacks is demonstrated in Figure 13. When there is a spike in the number of attacks, there is also a corresponding increase in the percentage of false positives that are causing harm to the system. What causes this phenomenon is the relationship between a higher attack rate and a higher frequency of the same flow over a given time period.

Applying a tougher threshold to packet filtration algorithms effectively mitigates attack traffic by rerouting it to a malicious queue. As the number of attacks rises, so does the number of attack packets that are incorrectly classified as regular traffic. However, there has been no change in the percentage of false positives, suggesting that the recall rate has stayed constant. In general, the proposed system's filtering method consistently maintains a low false-positive rate of less than 5% while identifying over 99% of DoS attacks with a remarkable level of accuracy.

4.4.6 Accuracy of attack detection

Our system utilization is evaluated using CICDoS 2017 dataset. Furthermore, it involves assessing packet filtering and classification techniques. We found throughout our evaluation process that a few of the reference implementations generate false positives. However, the proposed approach not only restricts access from the IP address but also unblocks after a certain period of time when no more malicious packets come from that blocked IP address. The method we have suggested exhibits a greater level of detection accuracy, specifically 99.56%, compared to the other benchmarks which achieved accuracies of 99.05%, 97.15%, 99.12%, and 98.64% as displayed in Figure 14.

The proposed system's accuracy is significantly enhanced by the integration of SVM and KNN, which guarantees that the packet filtration module's assessments are substantiated by two reputable classification methods. The integration of these two classification methods diminishes the likelihood of misclassifying benign packets as malicious, thereby reducing the false-positive rate. Table 7 shows that the SVM classifier was more accurate, precise, and had a better F1-score than the KNN model. SVM can produce clear decision boundaries when multiple features overlap.

This separates normal traffic from attack traffic, indicating what caused this improvement. However, complicated or noisy data can impact KNN because it is dependent on distance-based categorization. So, the SVM was selected as the better alternative following the analysis due to its higher accuracy compared to the KNN and it's more flexible and reliable. Table 7 illustrates the performance of the employed machine learning models.

5 Limitations

The proposed system has some limitations in terms of its design and functionality, which are described here. The proposed approach exclusively identifies DoS attacks that

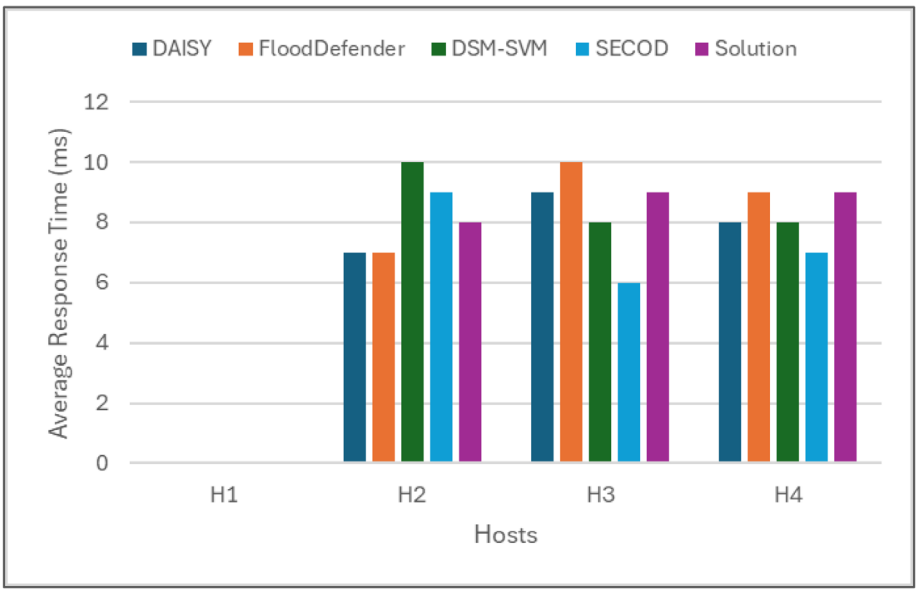

Fig. 12. Average latency analysis with DoS attack.

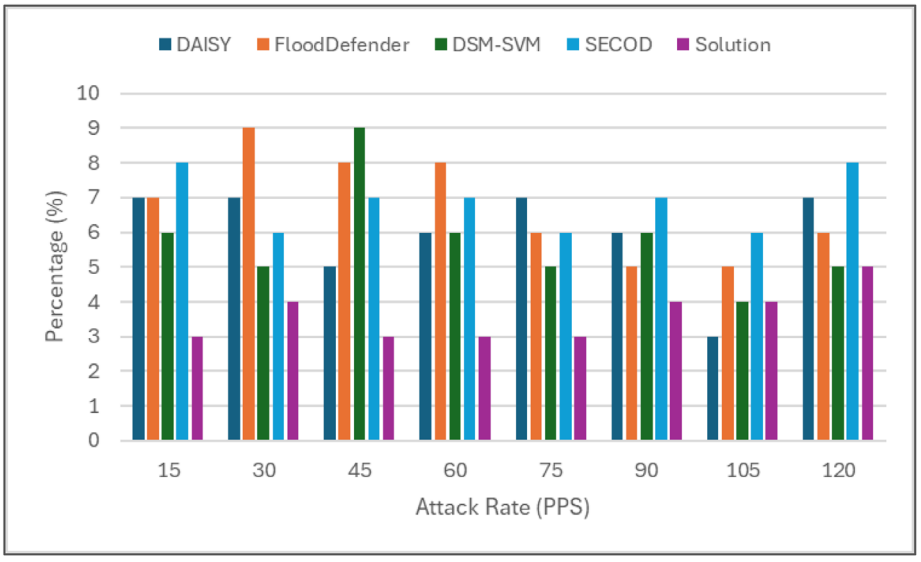

Fig. 13. Attack identification false rate.

specifically aim at the SDN infrastructure so there is no intention of protecting different servers that make use of the SDN environment. The performance of the proposed system has not been assessed in larger and more complicated network situations, so the scalability of the system, particularly in managing large amounts of traffic and multiple simultaneous attacks, has not been tested. Furthermore, when networks grow in size, the computational cost of ongoing feature extraction and model changes may rise, particularly in real-time deployments. The research doesn't cover how the proposed approach can adjust to different kinds of DoS attacks as threats change over time. An efficient plan for reducing cyber risks has to be flexible enough to be updated when new threats emerge.

6 Conclusion and future work

There are several security concerns with SDN, with DoS attacks being particularly problematic. Since the SDN controller manages the entire network, it is especially vulnerable to such attacks. Therefore, efficient detection of SDN DoS attacks is crucial. This article looks at the best machine learning classifiers for finding SDN DoS attacks quickly and correctly. Both the machine learning classifiers and the advantages of SDN work together to protect the SDN controller from these threats. A packet filtering process in the attack detection module is the first step in the proposed method for finding DoS attacks. Because machine learning is becoming more and more important for quick identification, SVM and KNN algorithms are used to

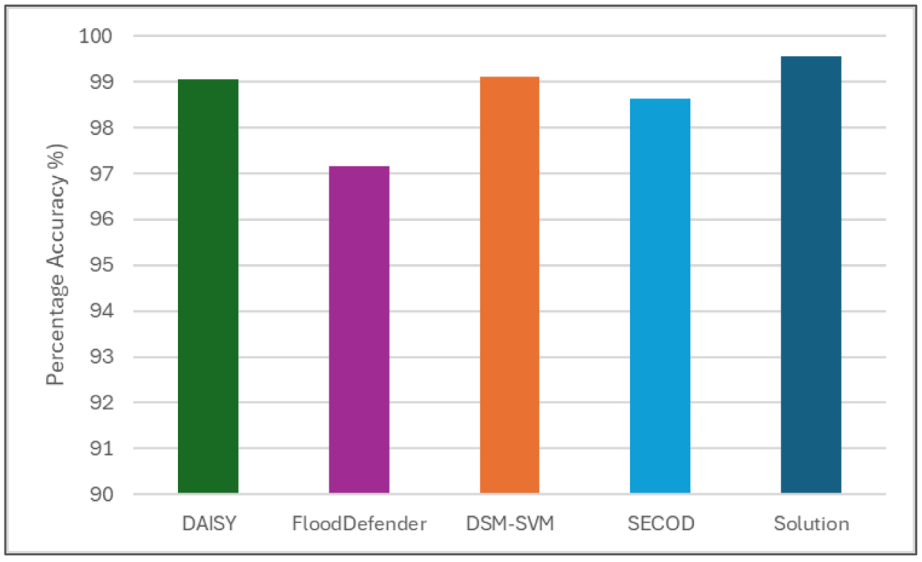

Fig. 14. Accuracy of attack detection.

Table 7. Performance analysis of SVM and KNN.

ML algorithms	Accuracy	Sensitivity	Precision	F1-Score
SVM	99.23%	98.78%	99%	98%
KNN	96.99%	97.13%	95%	95%

make DoS attack detection more accurate. The methodology was tested using the CICDoS 2017 dataset. The trained network model was evaluated within an SDN environment, with experiments conducted on an Ubuntu virtual machine hosted on VMware. Benchmark tests show that using our strategy in SDN improves the accuracy of detection. This is shown by lower false alarms, latency, and resource use that can be achieved without adding more hardware. In future work, we plan to add to and test our proposed system in real SDN environments to make sure it can handle more users and is safe from DoS attacks that are getting more complicated and varied.

Funding

This study was carried out without any financial support or grants from any organization.

Conflicts of interest

The authors confirm that there are no financial or personal conflicts of interest that could have influenced the work presented in this paper.

Data availability statement

All data used in this research are available from the corresponding author upon request.

Author contribution statement

- Najmun Nisa: Developed the main idea, designed the methodology, formal analysis and prepared the first draft of the manuscript.
- Adnan Shahid Khan: Provided supervision, guidance, and reviewed and refined the manuscript.
- Azman Bin Bujang Masli: Managed data collection and assisted with software related work.
- Nusrat Shaheen: Provided assistance in machine learning part and participated in reviewing and editing the paper.

References

1. D. Gao, Z. Liu, Y. Liu, C.H. Foh, T. Zhi, H.C. Chao, Defending against Packet-In messages flooding attack under SDN context, Soft. Comput. **22**, 6797–6809 (2018)
2. Z.A. Bhuiyan, S. Islam, M.M. Islam, A.B.M.A. Ullah, F. Naz, M.S. Rahman, On the (in)Security of the control plane of SDN architecture: a survey, IEEE Access **11**, 91550–91582 (2023)
3. R. Masoudi, A. Ghaffari, Software defined networks: a survey, J. Netw. Comput. Appl. **67**, 1–25 (2016)
4. L.F. Eliyan, R. Di Pietro, DoS and DDoS attacks in software defined networks: a survey of existing solutions and research challenges, Future Gener. Comput. Syst. **122**, 149–171 (2021)
5. B. Isyaku, M.S.M. Zahid, M.B. Kamat, K.A. Bakar, F.A. Ghaleb, Software defined networking flow table management of OpenFlow switches performance and security challenges: a survey 147 (2020), https://doi.org/10.3390/FI12090147

6. J. Xie et al., A survey of machine learning techniques applied to software defined networking (SDN): research issues and challenges (Institute of Electrical and Electronics Engineers Inc., pp. 393–430 (2019), https://doi.org/10.1109/COMST.2018.2866942
7. S. Soltani, A. Amanlou, M. Shojafar, R. Tafazolli, Security of topology discovery service in SDN: vulnerabilities and countermeasures, IEEE Open J. Commun. Soc. (2024). https://doi.org/10.1109/OJCOMS.2024.3406489
8. M. Rahouti, K. Xiong, Y. Xin, S.K. Jagatheesaperumal, M. Ayyash, M. Shaheed, SDN security review: threat taxonomy, implications, and open challenges (Institute of Electrical and Electronics Engineers Inc., 45820–45854 (2022) https://doi.org/10.1109/ACCESS.2022.3168972
9. S. Gao, Z. Li, B. Xiao, G. Wei, Security threats in the data plane of software-defined networks, IEEE Netw. **32**, 108–113 (2018)
10. S. Wang, K. Gomez, K. Sithamparanathan, M.R. Asghar, G. Russello, P. Zanna, Mitigating ddos attacks in sdn-based iot networks leveraging secure control and data plane algorithm, Appl. Sci. (Switzerland) **11**, 1–27 (2021)
11. M. Yue, H. Wang, L. Liu, Z. Wu, Detecting DoS attacks based on multi-features in SDN, IEEE Access **8**, 104688–104700 (2020)
12. S. Siddiqui et al., Toward software-defined networking-based IoT frameworks: a systematic literature review, taxonomy, open challenges and prospects (Institute of Electrical and Electronics Engineers Inc., 70850–70901 (2022), https://doi.org/10.1109/ACCESS.2022.3188311
13. M. Imran, M.H. Durad, F.A. Khan, H. Abbas, DAISY: a detection and mitigation system against denial-of-service attacks in software-defined networks, IEEE Syst. J. **14**, 1933–1944 (2020)
14. L.F. Eliyan, R. Di Pietro, DeMi: a solution to detect and mitigate DoS attacks in SDN, IEEE Access **11**, 82477–82495 (2023)
15. E. Kaljic, A. Maric, P. Njemcevic, DoS attack mitigation in SDN networks using a deeply programmable packet-switching node based on a hybrid FPGA/CPU data plane architecture, in: *ICAT 2019-27th International Conference on Information*, Communication and Automation Technologies, Proceedings, 1–6 (2019), https://doi.org/10.1109/icat47117.2019.8938862
16. M.B. Jimenez, D. Fernandez, J.E. Rivadeneira, L. Bellido, A. Cardenas, A survey of the main security issues and solutions for the SDN architecture, IEEE Access **9**, 122016–122038 (2021)
17. H. Elubeyd, D. Yiltas-Kaplan, Hybrid deep learning approach for automatic DoS/DDoS attacks detection in software-defined networks, Appl. Sci. (Switzerland), **13**, (2023). https://doi.org/10.3390/app13063828
18. N. Nisa, A.S. Khan, Z. Ahmad, S. Aqeel, J. Asim, S. Afzal, Conceptual review of DoS attacks in software defined networks, in: *Proceedings – AiIC 2022: 2022 Applied Informatics International Conference: Digital Innovation in Applied Informatics during the Pandemic, Institute of Electrical and Electronics Engineers Inc.*, 2022, pp. 154–158
19. S. Gao et al., Detection and mitigation of DoS attacks in software defined networks, IEEE/ACM Trans. Netw. **28**, 1419 (2020)
20. J. Liu, Q. Xu, Machine learning in software defined network, in: *Proceedings of 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference, ITNEC 2019*, no. Itnec, 2019, pp. 1114–1120
21. A. Ahmad, E. Harjula, M. Ylianttila, I. Ahmad, Evaluation of machine learning techniques for security in SDN, in: *2020 IEEE Globecom Workshops, GC Wkshps 2020 – Proceedings, Institute of Electrical and Electronics Engineers Inc.*, Dec. 2020
22. J. Ye, X. Cheng, J. Zhu, L. Feng, L. Song, A DDoS attack detection method based on SVM in software defined network, Secur. Commun. Netw. 2018, (2018). https://doi.org/10.1155/2018/9804061
23. S.M. Mousavi, M. St-Hilaire, Early detection of DDoS attacks against software defined network controllers, J. Netw. Syst. Manag. **26**, 573–591 (2018)
24. Z.R. Alashhab, M. Anbar, M.M. Singh, I.H. Hasbullah, P. Jain, T.A. Al-Amiedy, Distributed denial of service attacks against cloud computing environment: survey, issues, challenges and coherent taxonomy, (MDPI, 2022)
25. G. Fioravanti, M.G. Spina, F. De Rango, Entropy based DDoS detection in software defined networks, in: *Proceedings – IEEE Consumer Communications and Networking Conference, CCNC, Institute of Electrical and Electronics Engineers Inc.*, 2023, pp. 636–639
26. M.J. Santos-Neto, J.L. Bordim, E.A.P. Alchieri, E. Ishikawa, DDoS attack detection in SDN: enhancing entropy-based detection with machine learning, in: *Concurrency and Computation: Practice and Experience*, John Wiley and Sons Ltd, May 2024
27. C.H. Yang, J.P. Wu, F.Y. Lee, T.Y. Lin, M.H. Tsai, Detection and mitigation of SYN flooding attacks through SYN/ACK packets and Black/White lists, Sensors **23**, (2023). https://doi.org/10.3390/s23083817
28. S. Shin, V. Yegneswaran, P. Porras, G. Gu, AVANT-GUARD: scalable and vigilant switch flow management in software-defined networks, in: *Proceedings of the ACM Conference on Computer and Communications Security*, 2013, pp. 413–424
29. H. Wang, L. Xu, G. Gu, FloodGuard: A DoS attack prevention extension in software-defined networks, Proc. Int. Conf. Depend. Syst. Netw. **2015**, 239–250 (2015)
30. L. Tan, Y. Pan, J. Wu, J. Zhou, H. Jiang, Y. Deng, A new framework for DDoS attack detection and defense in SDN environment, IEEE Access **8**, 161908–161919 (2020)
31. K.S. Sahoo et al., An evolutionary SVM model for DDOS attack detection in software defined networks, IEEE Access **8**, 132502–132513 (2020)
32. H. Peng, Z. Sun, X. Zhao, S. Tan, Z. Sun, A detection method for anomaly flow in software defined network, IEEE Access **6**, 27809–27817 (2018)
33. A. Banitalebi Dehkordi, M.R. Soltanaghaei, F.Z. Boroujeni, The DDoS attacks detection through machine learning and statistical methods in SDN, J. Supercomput. **77**, 2383–2415 (2021)
34. N.N. Tuan, P.H. Hung, N.D. Nghia, N. Van Tho, T. Van Phan, N.H. Thanh, A DDoS attack mitigation scheme in ISP networks using machine learning based on SDN, Electronics (Switzerland), **9**, 1–19 (2020)
35. P. Kumar, M. Tripathi, A. Nehra, M. Conti, C. Lal, SAFETY: early detection and mitigation of TCP SYN flood utilizing entropy in SDN, IEEE Trans. Netw. Serv. Manag. **15**, 1545–1559 (2018)
36. K.S. Sahoo, D. Puthal, M. Tiwary, J.J.P.C. Rodrigues, B. Sahoo, R. Dash, An early detection of low rate DDoS attack to SDN based data center networks using information distance metrics, Future Gener. Comput. Syst. **89**, 685–697 (2018)

37. M. Revathi, V.V. Ramalingam, B. Amutha, A machine learning based detection and mitigation of the DDOS attack by using SDN controller framework, Wirel. Pers. Commun. **127**, 2417–2441 (2022)
38. Q. Yasin, Z. Iqbal, M.A. Khan, S. Kadry, Y. Nam, Reliable multipath flow for link failure recovery in 5G networks using SDN paradigm, Inf. Technol. Control **51**, 5–17 (2022)
39. A. Ashraf, Z. Iqbal, M.A. Khan, U. Tariq, S. Kadry, S. Park, Scalable offloading using machine learning methods for distributed multi-controller architecture of SDN networks, J. Supercomput. **78**, 10191–10210 (2022)
40. M. Imran, M.H. Durad, F.A. Khan, A. Derhab, Reducing the effects of DoS attacks in software defined networks using parallel flow installation, Hum. −Centric Comput. Inf. Sci. **9**, 1–19 (2019)
41. P. Zhang, H. Wang, C. Hu, C. Lin, On denial of service attacks in software defined networks; on denial of service attacks in software defined networks, IEEE Netw. **30**, (2016). https://doi.org/10.1109/MNET.2016.1600109NM
42. R. Santos, D. Souza, W. Santo, A. Ribeiro, E. Moreno, Machine learning algorithms to detect DDoS attacks in SDN, in: *Concurrency and Computation: Practice and Experience*, John Wiley and Sons Ltd, Aug. 2020
43. S. Asadollahi, B. Goswami, M. Sameer, *Ryu controller's scalability experiment on software defined networks*. 1–5 (2018), https://doi.org/10.1109/ICCTAC.2018.8370397
44. S. Rowshanrad, V. Abdi, M. Keshtgari, Performance evaluation of SDN controllers: floodlight and opendaylight, 47–57 (2016), https://doi.org/10.31436/iiumej.v17i2.615
45. N. Gude et al., NOX: Towards an Operating System for Networks. [Online]. Available: http://www.noxrepo.org
46. Scapy A Python tool for security testing, J. Comput. Sci. Syst. Biol. **8**, (2015). https://doi.org/10.4172/jcsb.1000182

Cite this article as: Najmun Nisa, Adnan Shahid Khan, Azman Bin Bujang Masli, Nusrat Shaheen, SVM driven approach for detecting DoS attacks in SDN environment, Int. J. Simul. Multidisci. Des. Optim. **16**, 29 (2025), https://doi.org/10.1051/smdo/2025031