


# A Fault Tolerant Multi-Controller Framework for SDN DDoS Attacks Detection

Parisa Valizadeh<sup>a\*</sup>, Ahmad Taghinezhad-Niar<sup>b</sup> 

<sup>a</sup>Ph.D. Candidate, Computer Engineering, Faculty of Engineering, Ferdowsi University of Mashhad, Iran; Parisa29.valizadeh@yahoo.com

<sup>b</sup>Faculty of Electrical and Computer Engineering, University of Tabriz, Tabriz, Iran; a.taghinezhad@tabrizu.ac.ir

## ABSTRACT

Network communication shows a variety of issues with the fast expansion of computer devices, ranging from network administration to traffic engineering. A well-known method for improving these connections is Software-Defined Networking (SDN). The SDN is a networking architecture that separates the control plane from the data plane to ease network administration. The main advantage of the SDN is the central controller. However, it has security flaws like unreachability in Distributed Denial-of-Service attacks (DDoS). Hence, defending SDN against DDoS attacks is critical. We proposed a framework for detecting DDoS attacks and a fault-tolerant method to replace faulty leader controller in distributed multi-controller SDN. We used multi-controllers architecture and leader election algorithm to present a fault-tolerant framework to select a new leader controller, in the case of a leader controller failure. In addition, an early DDoS attack detection algorithm using the entropy of destination IP addresses and the packet window initiation rate is presented. To evaluate our proposed method in various configurations, we simulated exhaustive experiments in Mininet and Floodlight. The results show that our approach outperforms similar algorithms in various network configurations and multi-victim attacks.

**Keywords:** Fault-tolerant, DDoS, Multi victims attack, Control Plane Security, SDN.

## 1. Introduction

Virtual switches, virtual layers, communication protocols, central controllers, and high-level APIs are the main elements of the SDN as a new architecture that facilitates the development of switch and network router control software. It also handles activities at higher levels. By minimizing hardware reliance, SDN improves software capabilities and network intelligence.

The OpenFlow protocol [1] has been chosen as a barrier in SDN because it allows an independent controller to manage switches over a secure channel. SDN provides many advantages, including cost-effectiveness, flexibility, dependability, manageability, and the ability to quickly integrate new technologies, test new ideas, shorten troubleshooting times, and reduce errors. However, these benefits do not imply that SDN addresses all network issues. Despite this, the separation of the control and data planes makes the SDN structure vulnerable to DDoS attacks.

One of the most prevalent methods is a DDoS attack [2], which seeks to deplete computer resources and render the unavailable controller by delivering large amounts of traffic. The most significant disadvantage in a DDoS attack scenario is the centralized structure of the controllers. During the time of a DDoS attack, the controller will initiate several flow entries due to a large number of distinct faked source addresses.

A quick and effective strategy is required to identify early threats successfully. As a result, we recommended deploying many coordinated distributed controllers to enhance the control plane security. We propose the use of distributed


controllers to improve control plane availability. We also provide a technique for early DDoS attack detection that will be implemented within every controller.

Given that a DDoS attack frequently targets multiple hosts, an approach such as [13] for identifying a single victim attack may be ineffective. Therefore, we propose a technique for detecting attacks that use combined packet windows initiation rate (PWI) and entropy methods. In contrast to prior approaches, our method can identify multiple-victim attacks while reducing the number of false-negative and false-positive alarms.

## 2. Background

Traditional networks have been developed and advanced for more than 30 years; nevertheless, their administration is laborious, complex, and lacks the flexibility to suit today's demands. However, in recent years, greater creativity and innovation have pushed network configuration and administration, which has been accomplished through SDN. SDN is a programmable and adaptable network since the controller and data plane are separated.

OpenFlow is a programmable network protocol for SDN environments that allows controllers and OpenFlow switches to communicate and evaluate packets sent from the switch by the controller. Several flow tables make up each OpenFlow switch. There are numerous flow entries in each flow table. When a switch receives a packet, it is compared to the switch flow table, which begins with the initial flow table and could extend to other flow tables in the pipeline for processing [3]. The instructions linked with the specified flow entry are performed if the incoming packet finds a matching entry. It

 <http://dx.doi.org/10.22133/ijwr.2022.345927.1119>

**Citation** P. Valizadeh and A. Taghinezhad-Niar, "A Fault Tolerant Multi-Controller Framework for SDN DDoS Attacks Detection," *International Journal of Web Research*, vol.5, no.1, pp. 1-7, 2022.

\*Corresponding Author

Article History: Received 7 June 2022; Revised 13 July 2022; Accepted 29 July 2022;

Copyright © 2022 University of Science and Culture. Published by University of Science and Culture. This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International license (<https://creativecommons.org/licenses/by-nc/4.0/>). Noncommercial uses of the work are permitted, provided the original work is properly cited.

will be forwarded to the controller if no matching entry can be located in any of the flow tables. It then computes and sends the Packet Out message to the switch to install the new rules in the flow table of the switches in the path. Since then, it is not necessary to send all packets related to that flow to the controller and forward any packets of that flow to their destination. Since it is the main center of decision-making in a network, the SDN controller is frequently targeted by attackers. This makes it a single point of failure. When an attacker tries to compromise the controller, it sends packets to a switch with faked source IP addresses to guarantee that the switch flow table has no matches. A packet will repeatedly request the controller to search for a match if it cannot locate one in the switch flow table. As a result, the controller's requests rise, and the controller's performance drops.

The overflowing of switch flow tables is another impact of a DDoS attack. The controller will add a new flow to the table for each incoming packet. Due to the size and scalability constraints of flow tables, in high-traffic areas, flow tables will be full of fake flows in a short period, causing the switch to fail.

### 3. Related Work

This section reviews the literature on DDoS attack detection in a machine-learning-based approach and prior studies of DDoS attack detection in an entropy-based approach.

#### 3.1 Machine-learning-based DDoS attack detection

On the subject of security, some researchers have employed machine-learning algorithms for DDoS attack detection [4-5].

In [6], the researchers employed the self-organizing map (SOM) machine learning approach for attack detection. It is learned using flow information gathered from OpenFlow switches regularly. Based on the gathered information, the primary factors that are checked for retraining SOM are the average number of packets per flow, average length per flow, percentage of pair flows, average number of bytes per flow, expansion of single-flows, and expansion of different ports. Over time, the SOM improves its vector and collects additional statistics. However, neither the effect of the attack packets on the controller nor their pass through the controller is mentioned in their study. In a DDoS attack, packets are always faked or their source nodes are falsified. As a result, the source address of the packet is new, and it must be sent to the controller for processing. However, this scenario is not covered in their technique.

The researchers in [7] suggested a dynamic multilayer perceptron (MLP) paired with a feature selection approach to identify DDoS attacks. Feedback mechanisms are used to encourage and repair the detector system when detection is insufficiently precise. Some of the chosen features in their model won't be able to discriminate between traffic and regular attacks and pinpoint the failure therein when the traffic network's complexity rises and changes.

In [8], a deep learning strategy is described for DDoS attack detection in SDNs. Their method of deep learning examined the data obtained mostly during the attack and divides it into two groups: malicious and legitimate traffic.

For DDoS detection in SDNs, [9] used a variety of ensemble models, including ensemble CNN, ensemble RNN, ensemble LSTM, and hybrid RL. The outcome demonstrates that CNN as a whole performs better. In [10], several SDN properties are collected, processed, and applied in a machine-learning approach for DDoS attack detection. Statistical-based solutions tend to be more appealing than machine-learning-based alternatives, which impose a significant computing strain on the system and need a substantial quantity of data.

#### 3.2 Entropy-based DDoS attack detection

Entropy-related effort to identify DDoS attacks is mentioned by Qin et al. They employed a 0.1-second window and three levels in the network to eliminate false positives and negatives, although this strategy has drawbacks, such as time consumption and resource usage [11].

A short-term statistics technique based on entropy computations was proposed by Oshima et al. [12]. They conducted a series of tests to determine which window size is suitable for estimating entropy. As a consequence, their approach has discovered that the window size of 50 works better over the other options. For early detection, the window size should not be too large; yet, a small window size will result in a higher processing time. They employed a one-sided test to detect DDoS attacks, as well as to measure the entropy of the source IP address to detect DDoS attacks. When the source IP addresses of all incoming packets are faked, a DDoS attack occurs, resulting in a greater entropy value. However, in this paper, we used a window size of 50 and assessed entropy using destination IP addresses, as opposed to Oshima's work, which used source IP addresses and resulted in lower entropy values in high traffic.

Ra et al. [13] offered a period window-based rapid entropy computation technique. They put numerous datasets to the test to determine an adequate threshold. Their experiment's results demonstrate that false negatives are higher and false positives are lower with their compared methodologies. They claimed to have developed a quicker method of computing entropy. However, these resources are not documented in their paper.

Kia [14] proposed using the entropy value of destination IP addresses to identify attacks. Using a method based on a false-negative rate, their approach improves the attack detection rate. However, they did not consider the time factor of collecting new flows in all processing circumstances of the mechanism. They used one controller in their network topology and depending on network traffic, imposed a threshold, that may result in more false attack detection reports. However, in our proposed solution, multi-controllers have been developed and tested in addition to the single controller to improve the control plane availability. In our approach, if a cluster's main controller fails, a floodlight-based election mechanism selects a new main controller to replace the failed one. Furthermore, we obtained a threshold for the Packet window initiation rate detection technique based on many experiments on a variety of victim hosts and comparisons with the destination IP addresses' entropy values.

Mousavi [15] proposed the use of entropy in the initial stages of DDoS detection in an SDN environment. The entropy of a window is determined using its destination IP

addresses. When a DDoS attack occurs, all newly arriving packets have the same or short defined destination IP addresses, resulting in entropy degradation, whereas, in normal traffic, all packets have various destination IP addresses, resulting in maximum entropy. To evaluate the entropy of window packets, they used a threshold.

The entropy method is a suitable technique for finding DDoS attacks in the initial stages, however, it cannot detect multi-victim attacks. In a DDoS attack, more than three or five hosts are targeted, therefore, single victim attack detection methods could not be effective. We employed the entropy method to detect attacks in their initial stages and paired it with a unique method to overcome the entropy limitation on multi-victim attack detection in this article.

In [16], a mathematical model and survey for DDoS cyberattacks are proposed. They first classified various attack types according to the sorts of networks they target traditional networks, SDN, or virtual networks. According to the type of network they were designed for, the criteria they took into account, and how the authors examined the network environment and the attack patterns, they categorized the attack models they discovered in the literature. They provide a detailed analysis and comparison of the current attack models that might be used against SDN.

The SDN network is also used in the Internet of Things (IoT). In [17], an SDN-based secure IoT framework is proposed that uses IP Payload analysis and session IP counters to identify IoT device vulnerabilities and malicious traffic sent by IoT devices. Their methods could identify the DDoS attack in the SDN-based IoT network. They evaluated their method by creating a lot of traffic from a compromised node, which is later identified and alerted.

Shakil et al. [18] presented a method to handle the common DDoS attack against SDNs. They allege that the likelihood of a data plane attack rises when the control plane is blocked off from the data plane. Therefore, they present a framework to handle the dynamic traffic and defend the network against DDoS attacks. They used the Whale optimization algorithm for DDoS attack detection. However, their proposed method as a meta-heuristic algorithm leads to a massive time complexity algorithm which is not a practical solution for a distributed concurrent system [19-20].

#### 4. Proposed Method

Securing the controller from becoming inaccessible in DDoS attacks is an important concern that must be tackled in SDN.

We presented a solution for preventing DDoS attacks on the controller. A new flow is created when a packet arrives at a controller. Hence, the controller configures a new flow entry in the switch, and the remainder of the flow packets are sent to the destination host without being processed.

Knowing that the arriving packet is new and that the destination IP addresses are in the network, we can calculate the entropy to determine the randomness level of the destination IPs. The entropy technique is a suitable way to identify DDoS attacks since DDoS operations try to transmit high traffic to a host or a selection of hosts. Window size and entropy threshold are the two primary components for identifying DDoS attacks using entropy in the initial stages.

To show the list of network packets, we defined PL in Eq. (1) which denotes the Windows Packets List (WPL) with J elements. The destination IP address and its frequency of occurrence are denoted by  $A_x$  and  $D_x$ , respectively.

We define Eqs. (2) and (3) to determine entropy, where WS is the size of the window. The notion indicates the likelihood of an IP as a target.

$$PL = ((A_1, D_1), (A_2, D_2), \dots, (A_J, D_J)) \quad (1)$$

$$P_x = D_x / WS \quad (2)$$

$$H = - \sum_{i=1}^J P_x \times \log_2 P_x \quad (3)$$

Entropy reaches its maximum value when packets are distributed equally over the network. The entropy of a host will decrease if it receives more packets than others.

All hosts should be able to distribute traffic in normal network mode. During a DDoS attack, the number of packets delivered to a single host or a small group of hosts suddenly surges, reducing the entropy. The entropy reduction alerts the possibility of an attack on the network. The entropy is compared to the threshold value in every window packet. Being smaller than the threshold for five times sequentially, we consider an attack is happening in the system. The sum of five consecutive instances of 50 packets is 250, indicating early detection of a DDoS attack. The detection of entropy is performed in the early stages of an attack. However, it has disadvantages that allow attack detection to fail when the number of victim hosts grows. Because the packets target numerous hosts, entropy may be higher than the threshold value in multi-victim attacks. A PWI is the second approach proposed in addition to the entropy detection algorithm. The second approach is described to facilitate the protection of the SDN controller against DDoS attacks.

First, we determine the time duration of 50 packet window initiation, which is indicated as PWD. In other words, PWD is the time difference between the first and final packet arrivals of the window (i.e., duration between the first and 50th packet per second). Following that, we compute its rate, which is PWI. The PWI is calculated using Eq. (4), where WS represents the number of window packets which is 50 in our experiments.

$$PWI = WS / PWD \quad (4)$$

The PWI value will be compared to the PWI threshold value after it has been computed. A greater value for the PWI in five sequential times means that a DDoS attack is identified. By conducting exhaustive experiments, we set the PWI threshold at 30 flow/sec which resulted in appropriate attack detection. We calculate the PWI threshold by using mean value and standard deviation as shown in Eqs. (5) and (6) respectively. The standard deviation, defined as the square root of the variance, is a statistic that represents the dispersion of a dataset compared to its mean.

$$\mu = \frac{\sum PWI}{|PWI|} \quad (5)$$

$$\sigma = \sqrt{\frac{\sum(PWI-\mu)^2}{|PWI|}} \quad (6)$$

The pseudo-code of the proposed algorithm is depicted in Algorithm 1. When a new packet arrives, the system triggers an event that instructs the suggested algorithm to perform. The following are the variables utilized in this algorithm. When the entropy is less than the entropy threshold (Entropy\_threshold), ER maintains track of the events number. We use PWR to save the number of rounds where PWI is higher than the threshold (PWI\_threshold). The number of new packets for each window is kept by the window counter (WC). In a window, WPL is used to save packet destination IP and its number of occurrence.

In this study, we show how to increase the availability of the SDN control plane in the event of a controller failure. System failures might occur for a variety of causes, including an attack on a controller or switch, a server's operation system failing, etc. When many instances of the controller are running, floodlight offers high availability support. This module publishes and subscribes to updates from many controllers, and stores the data using ISyncService. It synchronizes the state of controllers by allowing all of them to efficiently access changes broadcast by the controller's other modules. Additionally, it performs a leader election process to allow modules to do role-based programming in a distributed system with numerous controllers operating and communicating with each other.

Algorithm 2 illustrates the election algorithm [21] which is used to pick a replacement leader controller in the event of a leader controller failure. As a result, the failed controller is taken by the new leader controller. This results in a high-availability control plane for business networks. Figure 1 shows the distributed controller architecture that we employed to improve the control plane's high availability.

Multiple controllers are running in simultaneously, as shown in Figure 1, each of which might become unavailable due to a DDoS attack. Switches can be reallocated to a new controller in a different domain with a lesser load if they lost connectivity to their leader controller. The new controller which is the leader handles the connection of the network with the switches while updating the other secondary controllers on their current condition. The proposed technique for DDoS attack detection in their initial times can be used as a new module in controllers before the controller becomes unavailable due to these aforementioned attacks. The controller has located in-between application and data layers.

## 5. Simulation and Results

This section contains the simulation tools and the configuration of results. Following that, the results of the experiments will be evaluated. We used Linux Ubuntu 16.04 operation system in these experiments. The Mininet simulator [22], a tool for emulating SDN, was employed to conduct our experiments. Mininet's virtual network enables us to create, interact with, and customize network prototypes for SDN rapidly and simply. Mininet is capable of simulating real-world settings and implementing new attack scenarios.

The Floodlight controller [23], which is employed in this study, is a viable open-source tool for implementing SDN.

We utilized the Scapy program [24] to produce both legal and illegal traffic. It is a strong interactive packet manipulation program and a sophisticated packet decoder and forger. Scapy has two ways of execution: terminal and programmatically from a Python script. To construct our network configuration, the normal and attack traffic programs are built-in in Python. The target IP addresses are incremented from 10.0.0.1 to 10.0.0.64. The RTH is the round threshold which is 5 in our experiment and WS is 50.

To execute the simulation, a network topology with 64 hosts configuration is configured which is shown in Figure 2.

### Algorithm 1: Proposed Algorithm

```

Input: newpacket
1 WS ← 50;
2 RTH ← 5;
3 if WC == 0 then
4   Start PWD;
5   create a new pk.dest_ip for newpacket in WPL which is
   equal 1;
6 else
7   if pk.dest_ip In WPL then
8     find pk.dest_ip then increment it in WPL;
9   else
10    create a new pk.dest_ip for newpacket in WPL (= 1);
11  end
12  if WC == WS then
13    WC ← 0;
14    Stop PWD;
15    PWI ←  $\frac{WS}{PWD}$  ;
16    entropy ← calculate by Eq. (3);
17    if entropy < entropy_threshold then
18      ER += 1;
19      if ER == RTH then
20        Alarm the possible attack detection;
21      end
22    else
23      ER ← 0;
24      if PWI > PWI_threshold then
25        PWR += 1;
26      else
27        PWR ← 0;
28      end
29      if PWR == RTH then
30        Alarm the possible attack detection ;
31      end
32    end
33  end
34 end
35 WC += 1 ;
    
```

Algorithm.1. The proposed method for DDoS attack detection in the SDN

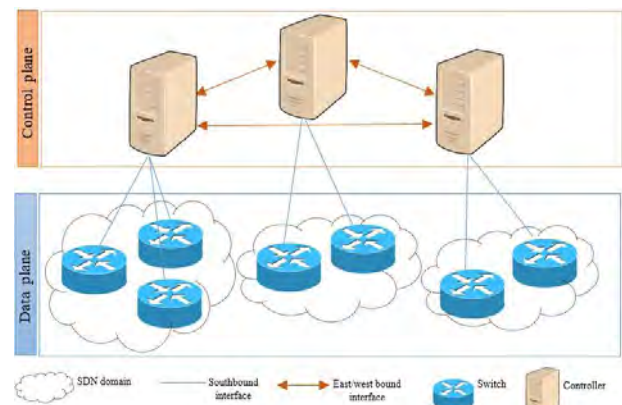


Figure. 1. The architecture of distributed controllers in SDN

**Algorithm 2:** Leader election algorithm

```

Input: SDN – controller
Output: Selected – controller
1 if thisController.role==NULL then
2   while thisController.role==NULL do
3     Multicast thisController.role;
4     Leader.priority;
5   end
6   if !Leader or thisController.priority > Leader.priority ;
7   then
8     thisController.role ← Leader;
9     thisController.IP ← virtual IP;
10  else
11    thisController.role ← follower;
12  end
13 else
14   if thisController.role== Leader then
15     while thisController.role== Leader do
16       Multicast heartbeat packets;
17       Listen newController.priority;
18     end
19     if received newController.priority then
20       send thisController.priority;
21     end
22     if thisController.priority < newController.priority
23     then
24       thisController.role ← follower;
25     else
26       if thisController.role== follower then
27         listen to heartbeat packets;
28         if heartbeat packets == NULL then
29           thisController.role ← NULL;
30         end
31       end
32     end
33 end

```

Algorithm 2. Leader election algorithm

Moreover, Figure 3 shows a second network configuration for multi-controller implementation. We used 1.2 for the entropy threshold and 5 for the entropy detection round (RTH) [14].

The traffic properties of our experiments are shown in Table 1. The regular traffic interval is 0.2 second, whereas the attack traffic period is 0.08 second for single victim attacks and 0.03 second for multiple victim attacks. The notion F/S means flow per second and P/S means packets per second.

In the various counts of victim attacks, Table 2 shows varying entropy rates and PWI rate values. It indicates that when the number of victims attacked increases, the entropy value increase as well. The entropy value is higher than our default threshold in a multi-victim attack with more than eight victims. The entropy method cannot efficiently detect the attack in these circumstances. Another approach based on estimating packet windows initiation rate (PWI) is employed in this research. The PWI threshold value is set to 30 by default. Multiple victim attacks can be detected using this threshold.

Based on Table 1, Figure 4 depicts packets received over time during single victim attacks and normal traffic. It demonstrates that when the attack starts in 103 milliseconds, the packet count extensively increases (i.e., >100 packets per millisecond).

Table 1. Characteristics Of Normal And Attack Traffic

	Multi-Host Attack	Single host Attack	Legitimate Traffic
Packet Load	-	-	21 bytes
Traffic Period	0.03 S	0.08 S	0.2 S
Traffic Rate	33.3 P/S	12.5 P/S	10 P/S
Flow Rate	33.3 F/S	12.5 F/S	2.5 F/S
Types	UDP	UDP	UDP
Transmitted Packets	1	1	4

Table 2. The Pwi Rate And Entropy Value To Calculate The Threshold

Attacker Count	Victim host count	Entropy rate	PWI
0	0	2.46	2.9
1	1	0.48	13.6
2	4	0.66	27.15
3	6	0.87	37.78
4	8	1.05	49.42
5	10	1.35	70.02

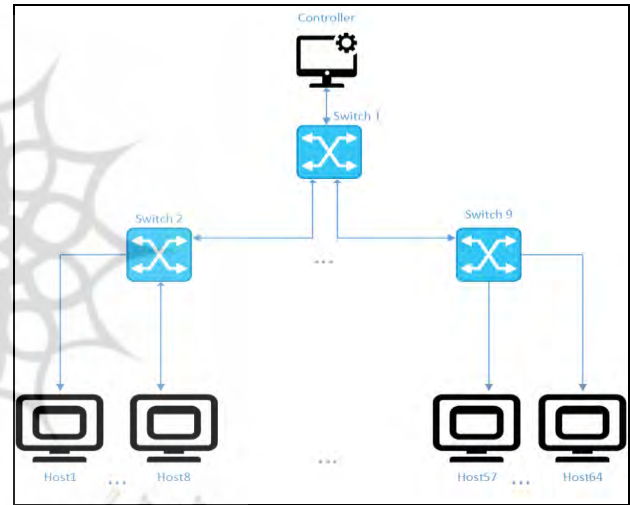


Figure. 2. Mininet network topology with a single controller

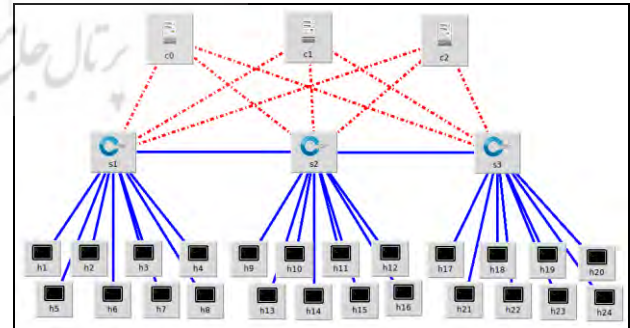


Figure. 3. Mininet network topology with distributed multiple controllers

**5.1 Single victim host attack**

At first, we considered two minutes of normal traffic in 64 hosts before conducting single victim attacks. After these minutes, the network traffic will be stable. We defined four single-victim attack schemes. The traffic rate of a host is simulated with 12.5 P/S in the first scheme. In the second

scheme, two hosts send malicious packets. Consequently, in the third scheme, three hosts send malicious traffic. Moreover, four hosts generate attack traffic in the fourth scheme.

Table 3 displays the number of false-positive and negative reports for a single victim attack based on the traffic pattern in Table 1.

**5.2 Multi-victim host attack**

In this part of the experiments, we conduct 20 experiments in the different topologies of the network. In each experiment, three attacking schemes are conducted. In the first scheme, a host sends regular traffic through the network at a rate of five packets per second, while another sends malicious traffic with 33.3 P/S rates to two individual hosts.

In the second scheme, three malicious hosts generate attack traffics to six individual hosts while a host creates normal traffic. In the third scheme, four hosts create malicious traffic to eight individual hosts and a host generates normal traffic.

Table 4 illustrates the false negative alarm and false positive alarm in the multi-victim attack based on the traffic pattern in Table 1. The detection rate of our proposed method shows 95 percent accuracy. Therefore, our approach outperforms other algorithms in identifying multi-victim attacks.

Figure 5 depicts the proportion of attacks detected on a single host using two entropy approaches as well as the proposed method.

The entropy reduces due to the attack traffic being routed to a destination, the entropy algorithm can detect most attacks on a single victim, as shown in Figure 5.

The entropy cannot be reduced when the attacker distributes traffic among numerous victims, and the attack cannot be detected by this approach. This is due to a small fraction of the entropy of an attack and normal traffic. However, our proposed method which is based on the windows initiation rate and entropy is effective in multi-victim attacks and single-victim attacks. Therefore, based on Figure 5, and Figure 6, our proposed algorithm can use the advantage of both of these algorithms.

Figure 6 depicts the percentage of attacks on a multi-victim host that are detected using two entropy approaches and the proposed method.

Both the presented approach and the entropy algorithm create the same amount of attack reports in the case of single-victim attacks. However, in multi-victim attack detection, the entropy method loses its detectability when the attacker targets more than seven hosts. Hence, entropy and PWI methods can not be individually effective. However, our proposed algorithm which uses both of these methods simultaneously is more effective.

**6. Conclusion**

The central controller is SDN's main advantage; however, it has security disadvantages, such as being unreachable in a DDoS attack. Therefore, protecting SDN from DDoS attacks

Table 3. Alarm Report Of A Single Victim Attack

	<i>Fault likelihood</i>	<i>Fault times</i>
<i>False Negative</i>	0 percent	0
<i>False Positive Alarm</i>	1.7 percent	1
<i>Detection Rate</i>	98.34 percent	
<i>Failure Rate</i>	1.66 percent	
<i>System Attacks</i>	60 times	

Table 4. Alarm Report Of A Multi-Victim Attack

	<i>Fault likelihood</i>	<i>Fault times</i>
<i>False Negative</i>	3.33 percent	2
<i>False Positive Alarm</i>	1.66 percent	1
<i>Detection Rate</i>	95 percent	
<i>Failure Rate</i>	5 percent	
<i>System Attacks</i>	60 times	

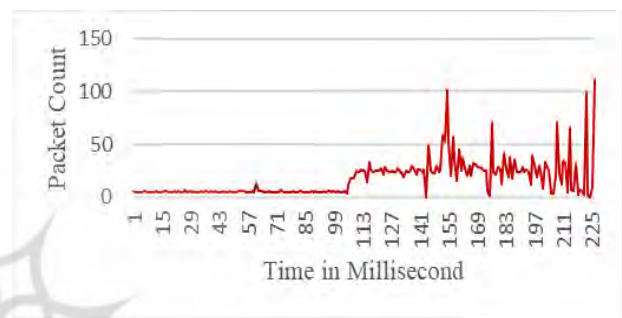


Figure 4. Packets numbers over time

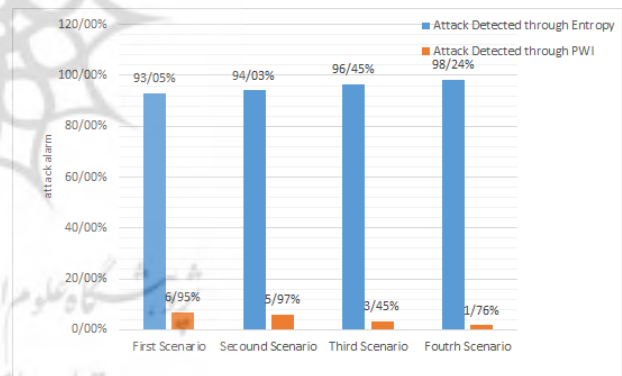


Figure 5. Attack alarm in a single victim attack

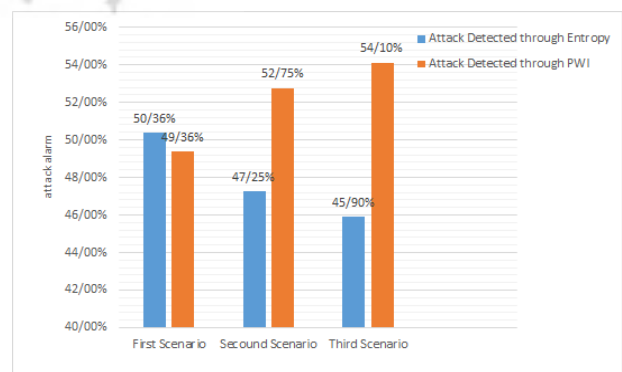


Figure 6. Attack alarm in multi-victim attack

is essential. This paper presented methods to improve the DDoS detection rate and network availability in SDN-based networks. We presented a method that can take the advantage of both the entropy and PWI algorithms to be effective on single and multi-victim attacks. Therefore, our proposed method is fault-tolerant for controller failure due to the use of an election algorithm on a multi-controller SDN. The experiments show that the algorithm's detection rate is satisfactory. In multi-victim attacks, our technique compared algorithms up to 9%. In the future study, we will investigate several protection methods during attack detection at an SDN-enabled wide area measurement system.

## Declarations

### Funding

This research did not receive any grant from funding agencies in the public, commercial, or non-profit sectors.

### Authors' contributions

PV: Study design, acquisition of data, interpretation of the results, statistical analysis, drafting the manuscript.

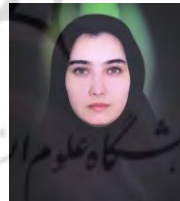
AT: Supervision, design, interpretation of the results, drafting the manuscript, revision of the manuscript.

### Conflict of interest

The authors declare that there is no conflict of interest.

## References

- [1] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software Defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, 2015.
- [2] U. Tariq, M. Hong, and K. Lhee, "A comprehensive categorization of DDoS attack and DDoS defense techniques," *Int. Conf. Adv. Data Min. Appl.*, no. Mic, pp. 1025–1036, 2006.
- [3] B. Heller, "OpenFlow Switch Specification 1.3.0," *Open Netw. Found.*, vol. 3, pp. 1–36, 2012.
- [4] M. Suresh and R. Anitha, "Evaluating Machine Learning Algorithms for Detecting DDoS Attacks BT," *Advances in Network Security and Applications*, pp. 441–452, 2011.
- [5] N. Meti, D. G. Narayan, and V. P. Baligar, "Detection of distributed denial of service attacks using machine learning algorithms in software defined networks," *2017 Int. Conf. Adv. Comput. Commun. Informatics, ICACCI 2017*, vol. 2017-January, pp. 1366–1371, 2017.
- [6] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS Flooding Attack Detection Using NOX / OpenFlow Network-Based Mechanisms Using SDN Network-Based Mechanisms Using SDN," pp. 408–415, 2018.
- [7] M. Wang, Y. Lu, and J. Qin, "A dynamic MLP-based DDoS attack detection method using feature selection and feedback," *Comput. Secur.*, vol. 88, p. 101645, 2020.
- [8] R. U. Rasool, U. Ashraf, K. Ahmed, H. Wang, W. Rafique, and Z. Anwar, "Cyberpulse: A Machine Learning Based Link Flooding Attack Mitigation System for Software Defined Networks," *IEEE Access*, vol. 7, pp. 34885–34899, 2019.
- [9] S. Haider, A. Akhuzada, I. Mustafa, TB. Patel, A. Fernandez, K-KR. Choo, J. Iqbal, "A Deep CNN Ensemble Framework for Efficient DDoS Attack Detection in Software Defined Networks," *IEEE Access*, vol. 8, pp. 53972–53983, 2020.
- [10] H. Polat, O. Polat, and A. Cetin, "Detecting DDoS Attacks in Software-Defined Networks Through Feature Selection Methods and Machine Learning Models," *Sustainability*, vol. 12, no. 3, 2020.
- [11] A. X. Liu, "1 An Advanced Entropy-Based DDOS Detection Scheme Jie Zhang, Zheng Qin, Lu Ou, Pei Jiang , JianRong Liu," *Analysis*, pp. 67–71, 2010.
- [12] S. Oshima, T. Nakashima, and T. Sueyoshi, "Early DoS/DDoS detection method using short-term statistics," *CISIS 2010 - 4th Int. Conf. Complex, Intell. Softw. Intensive Syst.*, pp. 168–173, 2010M.
- [13] G. No and I. Ra, "An Efficient and Reliable DDoS Attack Detection Using a Fast Entropy Computation Method," pp. 1223–1228, 2009.
- [14] M. Kia, "Early detection and mitigation of DDoS attacks in software defined networks." M. Sc. Thesis, 2015.
- [15] S. M. Mousavi and M. St-hilaire, "Early Detection of DDoS Attacks Against Software," *J. Netw. Syst. Manag.*, vol. 26, no. 3, pp. 573–591, 2018.
- [16] S. Kaur, K. Kumar, N. Aggarwal , and G. Singh, "A comprehensive survey of DDoS defense solutions in SDN: Taxonomy, research challenges, and future directions". *Computers & Security*, 110, 102423, 2021.
- [17] J. Bhayo, R. Jafaq, A. Ahmed, S. Hameed, and S. A. Shah, "A time-efficient approach toward DDoS attack detection in IoT network using SDN". *IEEE Internet of Things Journal*, 9(5), 3612-3630, 2021.
- [18] M. Shakil , A. Fuad Yousif Mohammed, , R. Arul, A. K. Bashir, and J. K. Choi, "A novel dynamic framework to detect DDoS in SDN using metaheuristic clustering", *Transactions on Emerging Telecommunications Technologies*, 33(3), e3622., 2022.
- [19] A. Taghinezhad-Niar , S. Pashazadeh, and J. Taheri, "QoS-aware online scheduling of multiple workflows under task execution time uncertainty in clouds". *Cluster Computing*, 8, 2022
- [20] A. Taghinezhad-Niar , S. Pashazadeh, and J. Taheri, "Energy-efficient workflow scheduling with budget-deadline constraints for cloud". *Computing*, 2022.
- [21] A. Rao, S. Auti, A. Koul, and G. Sabnis, "High Availability and Load Balancing in SDN Controllers," *Int. J. Trend Res. Dev.*, vol. 3, no. 2, pp. 2394–9333, 2016.
- [22] "Mininet Walkthrough - Mininet." [Online]. Available: <http://mininet.org/walkthrough/>. [Accessed: 24-Jan-2019].
- [23] "Available Tutorials for Floodlight," [Online]. Available: <https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343514/Tutorials>. [Accessed: 24-Jan-2019].
- [24] "Welcome to Scapy's documentation! — Scapy 2.4.0-dev documentation." [Online]. Available: <https://scapy.readthedocs.io/en/latest/>. [Accessed: 24-Jan-2019].



**Parisa Valizadeh** received her B.Sc. and M.Sc. degrees in Computer Engineering from the University of Tabriz, Tabriz, Iran in 2016 and 2019, respectively. Her research interests include Software Defined Networks, Smart Grid, and the Internet of Things. She is now a Ph.D.

candidate at the Ferdowsi University of Mashhad, Mashhad, Iran.



**Ahmad Taghinezhad-Niar** received his M.Sc. and Ph.D. degrees in Computer Engineering from the University of Tabriz, Tabriz, Iran in 2017 and 2021, respectively. He won awards in different programming competitions in Iran. He has served as a reviewer for several journals

and his research interests include Distributed Systems, Cloud Computing, Scheduling algorithms, and Formal methods. He is currently a lecturer at the University of Mohaghegh Ardabili (Iran).