International Journal of Electrical and Computer Engineering (IJECE)
Vol. 13, No. 1, February 2023, pp. 972~986
ISSN: 2088-8708, DOI: 10.11591/ijece.v13i1.pp972-986

Q-learning based distributed denial of service detection

Hiba Salah Yaseen, Ahmed Al-Saadi
Department of Computer Engineering, University of Technology, Baghdad, Iraq

Article Info

Article history:

Received Jan 3, 2022
Revised Sep 1, 2022
Accepted Sep 26, 2022

Keywords:

Distributed denial of service attacks
Network operating system
OpenFlow
Pythonic network operating system
Software defined network
User datagram protocol flood

ABSTRACT

Distributed denial of service (DDoS) attacks the target service providers by sending a huge amount of traffic to prevent legitimate users from getting the service. These attacks become more challenging in the software-defined network paradigm, due to the separation of the control plane from the data plane. Centralized software defined networks are more vulnerable to DDoS attacks that may cause the failure of all networks. In this work, a new approach is proposed based on q-learning to enhance the detection of DDoS attacks and reduce false positives and false negatives. The results of this work are compared with entropy detection in terms of the number of received packets to detect the attack and also the continuity of service for legitimate users. Moreover, these results indicate that the proposed system detects the DDoS attack from flash crowds and redirects the traffic to the edge of the data center. A second controller is used to redirect traffic to a honeypot server that works as a mirror server. This guarantees the continuity of service for both normal and suspected traffic until further analysis is done. The results indicate an increase of up to 50% in the throughput compared to other approaches.

Corresponding Author:

Hiba Salah Yaseen
Department of Computer Engineering, University of Technology
Alsinaa Street, Baghdad, Iraq
Email: ce.19.01@grad.uotechnology.edu.iq

1. INTRODUCTION

Security threats are growing rapidly, causing great damage for large companies, data centers, and governments. Distributed denial of service (DDoS) attacks are one of the most common threats that cause damage to the web and networks [1]. The attacker generates a huge amount of fake traffic targeting a specific target, for instance, a web server. The DDoS attacks are achieved through generating traffic from multiple devices distributed in different locations. The objective of such attacks is to stop the services of some companies or governments as long as possible to prevent users from using them as shown in Figure 1 [2].

The number of attacks on important websites is increasing every day and has become more dangerous, especially DDoS attacks because they are causing huge economical loss. For example, one of the largest attacks was in February 2018 on the GitHub site, which is a well-known code hosting site. This attack resulted in stopping the service for several hours by sending huge traffic that reached 1.35 Tb/s [3]. Another big attack happened last year in November on the Azure Microsoft site in Asia, which is a well-known cloud computing site. This attack took place for 15 minutes and happened by sending huge traffic of user datagram protocol (UDP) flood that reached 3.47 Tb/s [4]. Due to the rapid increase in network size and the use of heterogeneous devices, a new programmable network has been developed to simplify network management. This network is known as software defined network (SDN), which is considered an evolution in network architecture, and provides centralized management [5]. This technology provides unique features like centralization, programmability, scalability, and flexibility. These benefits are obtained from the separation

of the data plane and the control plane, while in the traditional network, these planes were on the same device, as shown in Figure 2 [6]. The control plane is responsible for monitoring and maintaining forwarding tables by using algorithms and sending them to switches. While the data plane contains switches that are only forwarding packets to appropriate output based on decisions taken from the controller [7].

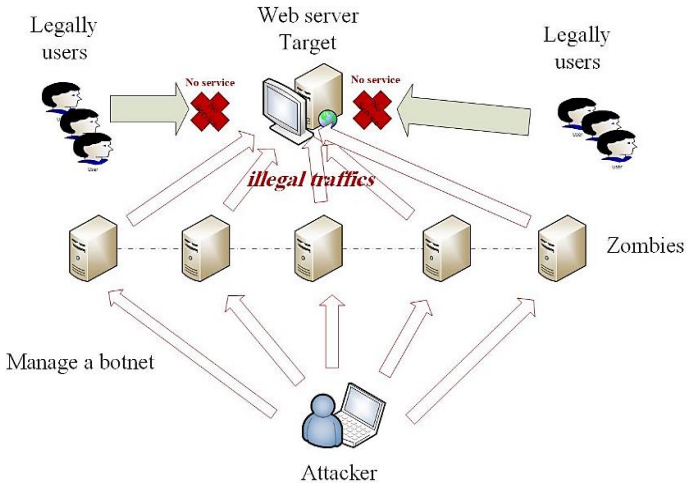

Figure 1. DDoS attacks [2]

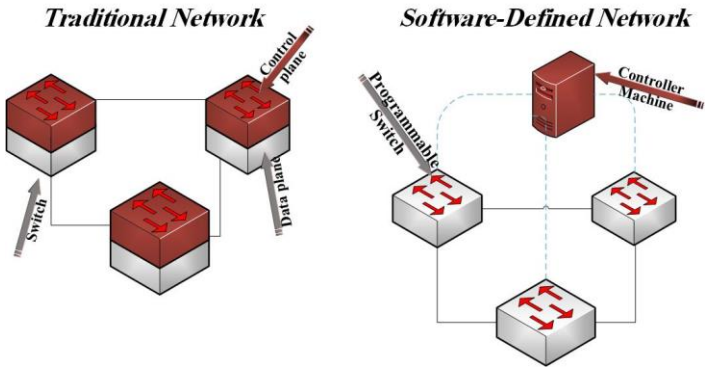

Figure 2. The difference between the two networks [6]

SDN network architecture consists of 3 layers and three important interfaces as shown in Figure 3.

- The application layer provides the possibility to develop various applications such as security, load balancing and layer 3 switch. These applications can be programmed using multiple programming languages such as python, java, and C++ depending on the type of controller being used in the network; therefore, the SDN is considered programmable [8].
- The control layer is the brain of the network that is responsible for managing and creating routing tables or flow tables. It also has a global view for all the network components [8].
- The infrastructure layer consists of switches, routers which are responsible for forwarding traffic, and the OpenFlow protocol is one of the most common protocols used in SDN [8].
- Southbound application program interface (API) exists between the infrastructure and the control layer. It is responsible for delivering control messages between the controller and OpenFlow switch [8].
- Northbound (API) is located between the control and the application layer. It is used as an interface for programmers to update and develop applications then deliver them to a controller. Moreover, it helps the controller to understand and execute commands on all network devices rather than configuring them individually [8].
- West/East (API) is located between controllers in the control plane in order to synchronize and manage the work of the network using multiple controllers [8].

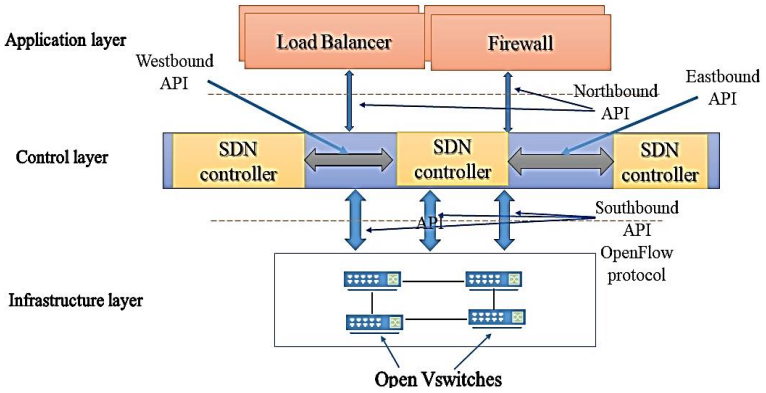

Figure 3. SDN architecture [9]

Network breaches are very widespread, some attacks target important information without permission, while other attacks make the services inaccessible. Therefore, cybersecurity is getting more attention from big companies due to the economic consequences of attacks [9]. The security threats in SDN networks are more challenging due to centralization of these networks. The centralized control of SDN networks provides the opportunity to develop new detecting and mitigating mechanisms easier at the controller [10]. Therefore, security engineers are studying and using networks.

There are various types of DDoS attack like UDP flood, transmission control protocol (TCP) flood, and ping of death. The Azure location, the most widespread attack until the second half of 2021 was the UDP spoof flood which is 55% of DDoS attacks. Thus, this research focuses on studying this type of attack [4]. However, there are many detection methods to find these attacks, each one has its advantages and disadvantages. That means there is no optimal solution to detect and mitigate DDoS attacks. The most popular DDoS detection methods used in SDN networks are entropy, machine learning, traffic pattern analysis, and connection rate [2], [11].

Moreover, another important component in SDN design is the OpenFlow protocol because it allows the separation between control and data planes in modern networks. This protocol can be very useful to detect and mitigate attacks by managing and controlling the flow tables in the switches to forward data. This is done by using the entities in flow tables to select the forwarding port [12]. These entities contain information that is used to manipulate and forward the incoming packets to the desired destination in the network as shown in Figure 4 [13].

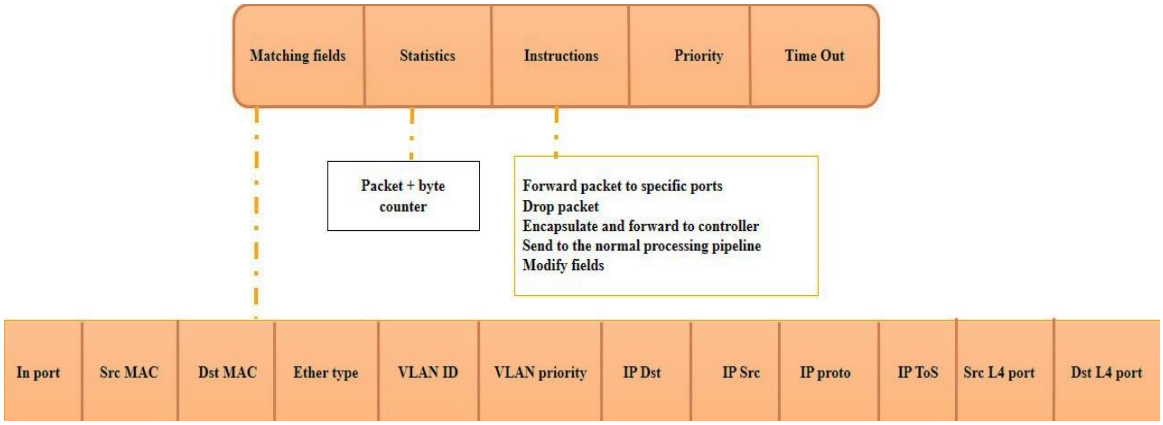

Figure 4. Flow entry [13]

There are various DDoS detection techniques, some based on the expected rate of bandwidth consumption and network activity patterns during normal circumstances. Rapid change in traffic flow, delay, central processing unit (CPU) consumption, or a sudden drop in the performance of network devices will be considered abnormal [14]. Several types of DDoS detection are based on statistical analysis and machine

learning. Statistical analyses depend on extracting statistical information of traffic, for instance, a number of bytes, packets, and flows. Examples of statistical methods are entropy and chi-square, which rely on extracting information from packet headers such as packet type, destination internet protocol (IP) address, and source IP address. The entropy method detects unexpected and unknown abnormal traffic, while the chi-square method is used to find the abnormal traffic in a specific intrusion type with a known type of packet header. For instance, if the transmission control protocol-synchronize (TCP-SYN) flood is the expected traffic, take a sample of data and count the number of TCP-SYN headers. The average number of such headers will be displayed as a pattern. Then any deviation beyond the limits of this pattern is considered abnormal [14]. Another method for detecting DDoS attacks is machine learning, which uses dynamic filters to detect suspected traffic. These algorithms are taught to update their filtering criteria constantly based on the events in the network. They depend only on extracting the incoming flow features and comparing them with the attack features library in a dataset. If a match occurs, the flow belongs to DDoS attack traffic; otherwise, it is not an attack. These approaches never result in false positives, but they are ineffective against variants outside of the library [15].

Braga *et al.* [16], proposed an algorithm to detect DDoS flooding attacks using the self-organizing method (SOM) which is unsupervised machine learning. This algorithm is employed to classify the received flow to either DDoS attack or legitimate traffic based on 6-tuples. These features are the average packet per flow, average bytes per flow, average duration per flow, percentage of pair flows, growth of single flow, and growth of single ports. The work is implemented using network operating system (NOX) [17] controllers and gives a good detection rate with a very low rate of false alarms. However, every time the algorithm needs to be changed, it must be retrained, which takes time and resources, while the proposed work utilizes two algorithms together, one is based on entropy, and the other is q-learning. Q-learning is reinforcement machine learning that does not depend on a specific filter or pattern library to train. The training is accomplished through the interactions of a learning agent with the environment. The suspected traffic will be redirected to another server at the edge of the data center using a second controller.

Another research used the entropy method to detect suspicious traffic and then employed a mapping method to distinguish between the flash crowd and DDoS attacks [18]. In order to indicate the significant difference between normal high traffic and DDoS traffic, IP addresses are mapped with media access control (MAC) addresses in a network. Normal traffic comes from actual hosts, which means it has one IP address related to one MAC address. In contrast, DDoS attacks come from many devices that employ spoofed source IP addresses and reuse the same MAC address. The results employ a packet rate to check whether attacks happened or not.

Another approach proposes an early detection method to detect DDoS attacks; it uses the entropy detection method based on the destination IP address and different window size [19]. The window size is the number of packets sent in a period of time. The entropy method used the randomness of packets to detect abnormal traffic in the network. This approach detects the DDoS attack within 260 to 550 packets with a window size of 50 because it does not impose a computational burden on the CPU and memory. Furthermore, it immediately blocks the suspect port. This method has some drawbacks, such as the fact that it does not conduct additional analysis to determine whether this is a genuine attack.

Another work [20] proposes an early DDoS detection method which is developed using the RYU controller. It compares the performance of three topologies (single, tree, and linear) with one and multi-controllers in terms of detection time for different attack rates. The used mitigation method is dropping the suspected port without further analysis to ensure that the traffic is an actual attack or not.

The work in [21] detects the DDoS attack at an early stage based on using the mean entropy for the destination IP address compared with a specific threshold. It is also using the percentage of the dropping packets in the controller during the attack for detection operation. The topology used for the experiment was a single controller with eight switches and 49 hosts.

Another study [22] proposes a detection method of the DDoS attack by using a statistical method with machine learning algorithms. The statistical method used is entropy based on the destination IP address with a threshold to detect the suspected traffic. Then, the features of this traffic are extracted and given as input to a number of classification algorithms to detect the attack. Moreover, the work was done using one controller which is the Floodlight controller and there is no mitigation method for detecting attacks.

Another work is comparing the performance of different topologies (single, tree, and linear) and controllers (pythonic network operating system (POX), RYU, libfluid, open network operating system (ONOS), OpenDaylight in terms of some of the quality of service (QoS) measurements like average round-trip time (RTT), throughput, and jitter [23]. The mitigation method used in [23] for suspected traffic ports is direct blocking without additional investigation to ensure that the suspected traffic is an actual attack or not.

Carvalho *et al.* [24], proposes an approach to monitor the network and protect it against UDP flooding and TCP-SYN flood DoS attacks. The technique used is the entropy method with a specific threshold to identify the unusual traffic. Destination or source IP address, window size, and threshold are

considered as detection fields that can be configured to match the values desired by the network operator. This strategy has no mitigation techniques; instead, the attacked port is immediately blocked.

Another research direction implements a small-scale network test to detect UDP flood DDoS attacks. This system uses a Raspberry Pi as Open V Switch (OVS) which is connected with at least two hosts and POX as the controller [25]. Although the detection method used is the entropy of a system that calculates the number of incoming packets and a destination IP address in a specific window. Then the computed entropy is compared with a specific threshold to find out whether there is a DDoS attack or not. The OpenFlow protocol is used to mitigate the effect of the attack by modifying the flow table of the OVS switch to direct the flow to a non-existing port. This approach handles multiple victims, and the result of detection was within three to ten seconds and blocked the detected port.

A multi-level detection approach is proposed in multi-controllers SDN networks to avoid single point of failure [26]. The lightweight detection method used on multi controllers is the entropy detection which is based on the destination IP address to find suspicious traffic. Then, k-means clustering, and support vector machine (SVM) is used to do more analysis on suspicious traffic.

An early detection method was proposed on a distributed SDN multi-controller [27]. The method used is entropy detection based on destination IP address and specific threshold for each controller. When suspect traffic is detected, the controller will send a block message to the related port in the switch. Furthermore, it will inform the other controllers about the suspect port in order to update their switches about the traffic.

Pandikumar *et al.* [28] used entropy detection method to protect network devices against DDoS attacks. The method computes entropy for destination IP address and threshold for all hosts in the network. Moreover, this method is applied on multi-controller SDN to detect the attack in an early stage. The mitigation method used is blocking the port that the traffic comes from directly. In order to do this, a flow rule is installed by the controller on the switches for every spoofing packet. However, there is a drawback in installing rules in switch for every new packet which is the limited capability of switch. Thus, in order to handle this problem, it should change the default value of flow idle time to a smaller value. This is done for every attack traffic, which leads to faster deleting the rule in flow table of switch.

Badrinath *et al.* [29] introduces an approach to detect the DDoS attack based on entropy detection that uses the flow statistics from switches to work. Other measures are implemented to prevent these attacks like "completely automated public turing test to tell computers and humans apart" (CAPTCHA), installation of drop entry for blacklisted IPs, and honeypot mechanism. When the DDoS attack is detected by using the entropy, the controller will locate the victim server and inform it about the attack. In that case, the server enters the CAPTCHA mode to locate the source of the attack and save it in a blacklist. This list will be sent to the controller to install drop entry for the list content. The results show the algorithm can detect the attack at early intervals, and the attack solution was scalable and optimal for a campus network.

The work in this paper addresses the shortcuts of prior research by combining the entropy method with q-learning to optimize the detection of DDoS and avoid false positives. This type depends on the principle of reward and punishment based on entropy value for a specific server in the network. The proposed approach detects suspicious traffic and redirects them to the edge of a data center through a second controller. This results in reducing the impact of an attack on the network and allows additional analysis. The mitigation method used in some research like blocking ports or deleting packets could cause losing some legal traffic which leads to increasing in false-positive results. As a result, this study isolates suspected traffic away at the edge of the data center through the use of a honeypot server and a second controller. This allows further analysis using traffic latency to distinguish DDoS attacks from flash crowds. Previous studies have used a drop or block traffic method, which leads to shutting down that port on the switch and making it unconnected.

2. ENTROPY AND Q-LEARNING DETECTION (EQD)

In this paper, two methods are combined to detect DDoS attacks in SDN networks. First is entropy, which is a well-known concept in the field of information theory. Entropy determines the degree of dispersion or concentration of a random variable [24]. When there is a lot of randomness or disorder, the entropy increases, and vice versa. This was important in ordinary networks for detecting and categorizing network anomalies that were typically minor and hidden in a network traffic volume. These abnormalities are measured in terms of the number of flows, packets, or bytes which are used to detect any change in traffic volume [30]. As a result, entropy can be utilized to detect DDoS attacks based on the randomness of packet properties received by the network. Source IP, destination IP, source port, destination port, and size are examples of these attributes [24]. This study utilizes destination IP, window size, and a threshold for detecting DDoS attacks. The size of a window is the number of packets sent in a certain amount of time. In

order to identify DDoS attacks, entropy is calculated by measuring the uncertainty of packets inside a system window. A threshold of entropy is used to detect whether any attacks occur if the entropy falls below that threshold. Entropy is defined as (1) [31].

$$H = -\sum_{i=1}^{n} p_i \log_{10} p_i \quad (1)$$

Let p_i be the probability of occurrence of y_i Which is the number of packets in the window W for each destination IP address; *x*, and *n* be the number of different occurrences in the window [24].

$$W = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots\ldots\ldots\ldots (x_n, y_n)\} \quad (2)$$

$$p_i(x_i) = \frac{y_i}{n} \quad (3)$$

When a new packet is received, the proposed algorithm checks the flow table. If there is no match in the flow table, the switch will encapsulate the packet and send it as a *Packet_In* message to the controller. Then, 50 packets are collected with different destination IP addresses, and stored in a two-column hash table. The first column contains the destination IP address, while the second counts the number of times this IP is used. The entropy and probability of occurrence for these IP addresses in normal and attack traffic are calculated to determine the threshold. Then compare the entropy to the threshold to check if it was higher or lower than the specified value, as shown in Figure 5.

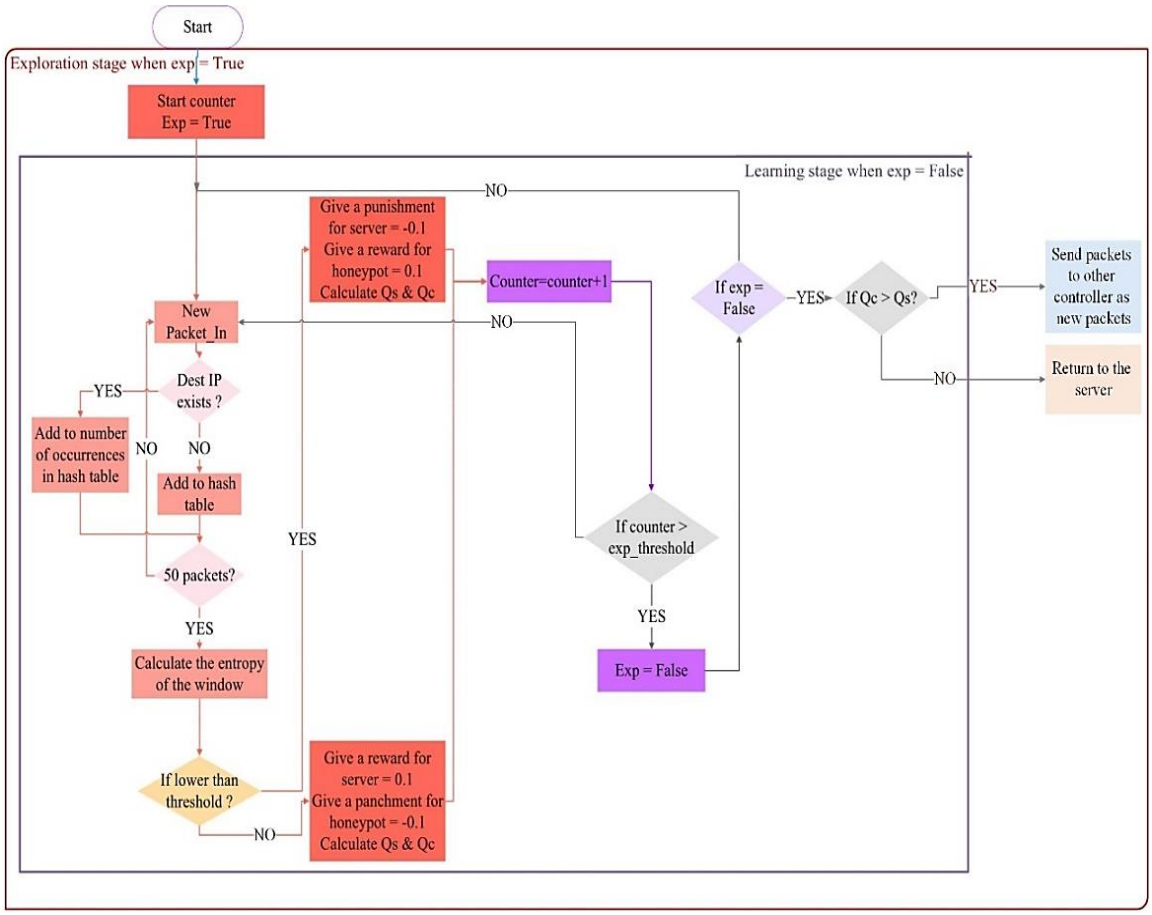

Figure 5. EQD detection for the first controller

The second contribution in this paper is using q-learning reinforcement learning. Q-learning is a type of machine learning that employs a reward-guided behavior. Reinforcement learning is an agent-based learning algorithm that utilizes trial-and-error interactions with the environment to obtain reward or punishment. Reinforcement learning systems (RLS) adapt parameters dynamically and achieve the highest

possible reinforcement signal. Instead of informing the system how to create the proper action, the reinforcement signal is delivered by the environment with a positive or negative evaluation of the actual action [32].

One of the most well-known reinforcements learning algorithms is the q-learning algorithm. It does not require a model of its surroundings; instead, it forecasts the future benefits of doing a specific action. In q-learning, a reward $R(t_i)$ is determined each time (t_i) action is performed based on feedback from the environment. In (4) [33] recalculates the q-value, then utilized to determine the optimum action [34]:

$$Q(t_i) = (1-\alpha)Q(t_{i-1}) + \alpha[R(t_i) + \gamma Q(t_{i+1})Q(t_{i-1})] \quad (4)$$

where t_i is the current time, t_{i-1} is the previous time for $i>1$, and γ is the discount value. If α=0, the algorithm has no learning; if γ=0, reinforcement learning is opportunistic, maximizing only the immediate short-term reward [34]. This work assumes that α=0.2 and γ=0 to make the agent depend on reward, punishment, and previous network actions only (these values computed impartially).

The q-learning algorithm allows the proposed DDoS detection system to learn from previous entropy values in order to detect a real attack or just normal high traffic. Each time the entropy algorithm detects a potential attack the q-learning algorithm punishes the q-value of using the main server and rewards using the honeypot on the edge data center. Then if the entropy value indicates the traffic does not possess threats on the network the q-value rewards the main server and punishes the honeypot server. The algorithm selects the appropriate action based on q-values for the main server and the honeypot server. Furthermore, the rule lifetime of the OpenFlow table is calculated based on the reward and punishment done using the q-learning algorithm. If the entropy of the main server is higher than the threshold, the main server will get a reward and longer rule lifetime. When the entropy of the main server is lower, it will be punished and have a shorter lifetime. The reward value Rs is selected based on the following formula:

$$Rs = \begin{cases} -0.1 \; entropy\ value\ for\ a\ specific\ server < threshold \\ 0.1 \; otherwise \end{cases}$$

Moreover, the agent in the controller will monitor the entropy in (1) for the server and use it to get the q-learning of the server using (4) as shown in (5):

$$Qs(t_i) = (1-\alpha)Qs(t_{i-1}) + \alpha[Rs(t_i) + H - Qs(t_{i-1})] \quad (5)$$

where t_i is the current time, t_{i-1} is the previous time for $i>1$, α=0.2 (empirically selected), $Qs(t_i)$ represents the current probability of entropy for the server, $Qs(t_{i-1})$ represents the previous probability of entropy for the server, $Rs(t_i)$ reward of the server in current time, and H is the entropy value of the server.

Furthermore, another q-learning agent is proposed to monitor the second controller based on entropy. It gets a reward, when entropy is lower than the threshold and punishment if it is upper. As shown in the following formula Rc which is reward for second controller:

$$Rc = \begin{cases} 0.1 \; entropy\ value < threshold \\ -0.1 \; otherwise \end{cases}$$

$$Qc(t_i) = (1-\alpha)\mathrm{Q}c(t_{i-1}) + \alpha[\mathrm{R}c(t_i) + \mathrm{H} - \mathrm{Q}c(t_{i-1})] \quad (6)$$

where t_i is the current time, t_{i-1} is the previous time for i>1, α=0.2, $Qc(t_i)$ represents the current probability of entropy for second controller, $Qc(t_{i-1})$ represents the previous probability of entropy for second controller, $Rc(t_i)$ a reward of the second controller in the current time, and H is the entropy value of the server.

The flowchart in Figure 6 is divided into two parts: the exploration and exploitation. The first stage is an exploration, in which the q-learning has no previous memory to use in selecting actions. The second stage is exploitation in which the q-learning algorithm has built a memory and can perform learning in selecting actions. The exploration stage started when *exp=True*. In this stage, the algorithm initializes entropy parameters and calculates q-value of the network until the counter reaches *exp_threshold*. After that, the value of *exp* is changed to False and enters the next stage of learning. This stage is starting to evaluate the detection and learn how to deal with the entropy of a specific server. The system learns when $Qc(t_i) > Qs(t_i)$ the traffic is suspected and redirects the traffic to another controller for further analysis. Otherwise, the traffic is routed to the main server because there is no attack. Isolating the suspected traffic using a second controller at the edge of the data center is to protect the infrastructure of a network from these attacks.

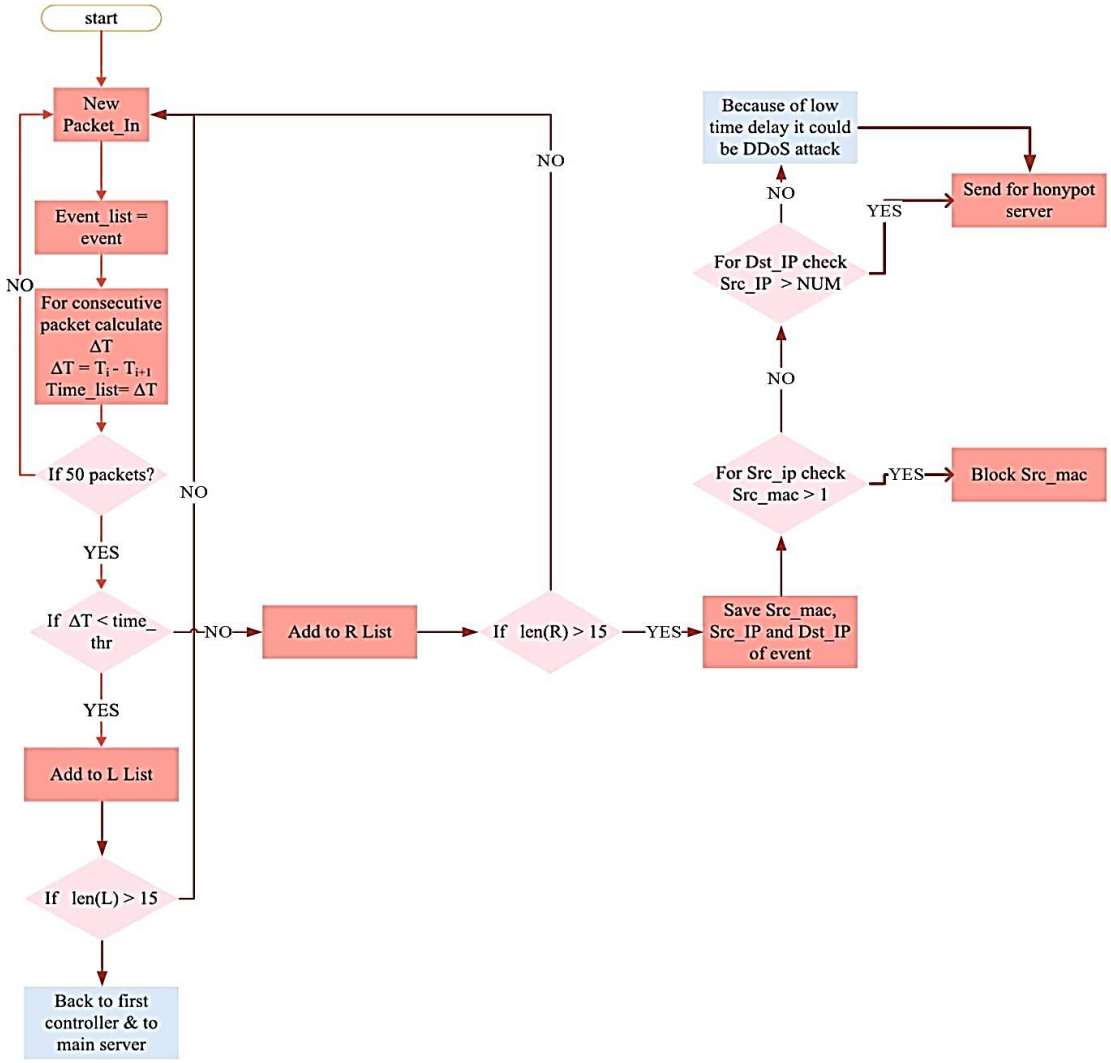

Figure 6. Suspected traffic detection for the second controller

Further analysis can be done on the suspected traffic at the second controller as illustrated in Figure 6. The controller collects the incoming traffic for every 50 packets and analyzes them based on the time interval between packets. Time intervals in attack traffic are usually less than normal traffic. When the time is less than the specific threshold *time_thr*, the traffic is considered suspicious. Then, the traffic features are saved to ensure the traffic source is not spoofing IP. Due to the fact that most DDoS attacks use a fake source IP address with a common media access control (MAC) address to create attacks. When there is more than one IP address with the same MAC address, then the controller blocks all traffic from the MAC address. This approach helps to distinguish between DDoS attacks and flash crowd traffic. Furthermore, if no spoofing IP is found, the controller checks whether the source IP is associated with more than one destination IP address to detect attacks. Then, forward the traffic to a honeypot server to log the source address of these attacks for further actions in the future. Moreover, the traffic with a low time delay between packets is still considered suspicious and could be a DDoS attack. Therefore, it will be sent to the honeypot server for further analysis in the future. When no attack is found in the traffic, it can be returned to the main server or sent a report to the network admin because the traffic is just a flash crowd.

3. RESULTS AND DISCUSSION

Here entropy and q-learning detection (EQD) is evaluated and tested using the Mininet [35] tool emulator, which is extensively used in SDN research and POX controller. The results are compared in terms of time and throughput with the entropy detection [19], [23]. Moreover, there is some statistical tools are used to show the difference in results as shown next section.

3.1. Simulation setup

In order to assess the proposed DDoS detection and mitigation strategy in SDN networks, Mininet is utilized to imitate the Open vSwitch [36] which supports the OpenFlow protocol. The network topology used in this study is created using the Mininet emulator. which is installed on Oracle Virtual Box version 6.0.18. A real programmable SDN controller (POX) [37] is used as a remote controller. The proposed algorithm is implemented using two remote POX controllers on port 6633, on two virtual machines, one server, a honeypot server, and four scenarios are applied to test the performance of the proposed system, as shown in Figure 7. Wireshark tool is used to monitor, and analyze the throughput of the server in different scenarios [38]. Scapy Python library is used to generate TCP and UDP traffic [39]. Hping3 is used to generate UDP flooding attacks [40].

Normal traffic is generated by a set of hosts in Mininet, and at the controller, the entropy is calculated every 50 packets, which is the window size. In this paper, the window size is set to 50 with 64 hosts, divided into two networks; each one of them contains 32 hosts. If small window sizes are used, such as 5, 10, and 20, it will be too small to determine the threshold [19]. The threshold is selected by running several attacks on the server and controller. This experiment has been repeated 10 times to test the system under various attacks and loads.

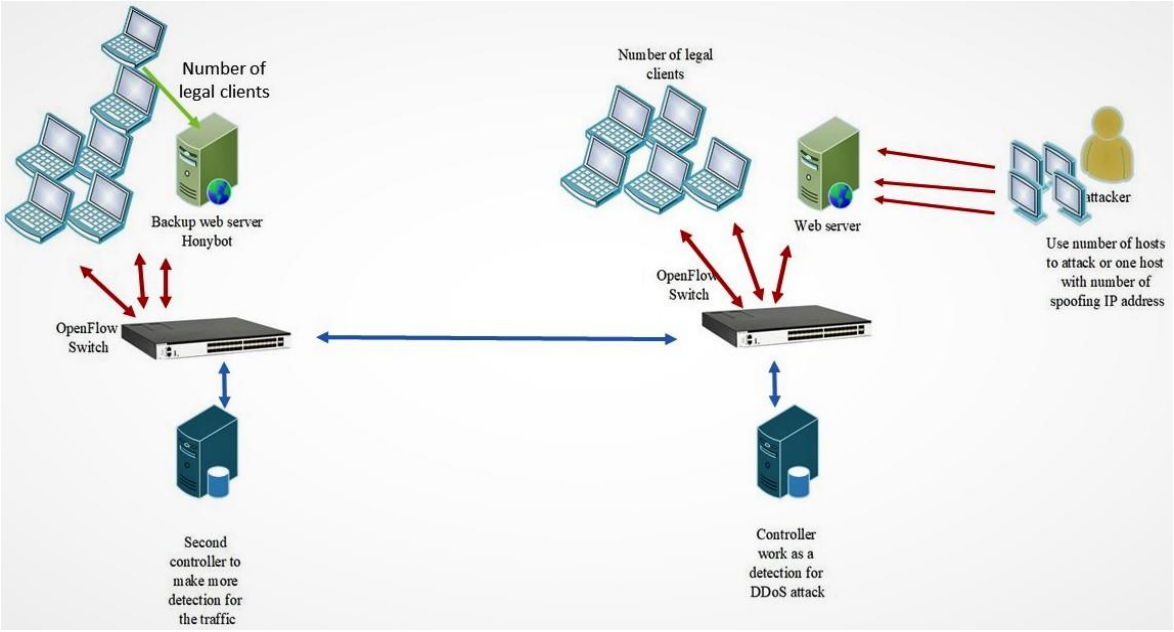

Figure 7. Network topology

3.2. Experiments results

The proposed approach in this work is examined and compared with DDoS detection method that utilizes entropy and uses direct block as a mitigation method to deal with the suspected traffics [19], [23]. The server performance will not be affected because the suspected traffic will be sent to the honeypot server which reduces the load on the server. Furthermore, it will reduce the load on data center infrastructure and create load balancing in case the attack is a false positive. The results indicate that the throughput of a server will remain normal despite the attacks. This research utilizes an analysis of variance statistical (ANOVA) statistical test [41], which is a tool used to ensure that the compared mitigations are statistically distinct using $F>F_{crit}$. Tables 1 and 2 show the ANOVA and least significant difference (LSD) results for the normal, normal with attack conditions, and two approaches in terms of received normal traffic in the server.

Table 1. ANOVA results

	ANOVA Test		
	F	F_{crit}	P < 0.05
Spoofing IP attacks	515.980296	2.866265551	1.27E-29
No spoofing IP attacks	806.543587	2.866265551	4.75E-33

Table 2. LSD results

	Average of received normal traffic (Packets)				LSD
	Normal	Normal + attack	EQD	Entropy detection	
Spoofing IP attacks	36.4	4.8	33.8	1	2.359
No spoofing IP attacks	36.4	4.8	38.7	1	2.029

Figures 8 and 9 illustrate how the throughput of non-attack traffic is affected when an attack of UDP flood occurs. The graph type of Box and Whisker is used to view the results. The box is divided by median so it can show the average of UDP traffic higher and lower than the median, while Whisker represents the minimum and maximum values. For instance, in Figure 8, data is divided into four parts, and each part is a quartile. The first quartile starts from the lower value and is called Q1, which represents 25% of the data. The second one, which is the median, represents 50% of data and is called Q2. While the third quartile above the median (Q3) has a percentage from 50.1% up to 75%, and the last one (Q4) represents the highest quartile of data up to the maximum value. Figure 8 shows the received traffic in normal circumstances is between 30 and 40 packets when the throughput of a server is 0.384 packets/sec. During UDP flooding attacks using spoofing IPs, the non-attack traffic average drops to 2 packets due to high number of packets being sent as fast as possible. When the proposed algorithm detects a new DDoS, it redirects the suspected traffic to the edge of the data center through a second controller. During the attack, the traffic from normal users will remain routed through the main server using the main controller. Therefore, normal traffic will not drop during the attack while the second controller drops or block the DDoS attacks. Figure 8 shows the EQD method reaches 35 packets and is more than 50% greater than the entropy detection approach that blocks the port of attack traffic. In Figure 9 more than one host sends 2,500 packets/sec without spoofing IP, the algorithm detects an attack and sends it to the second controller. The second controller forwards the suspected traffic to the honeypot server for further analysis. This scenario shows an increase in accepting normal traffic reach to 40 packets during attacks. Both figures note that normal entropy detection, which uses a direct block to deal with suspect traffic, has stopped all traffic that comes to the server due to failure of the controller itself. Moreover, entropy detection cannot distinguish between IP spoofing attacks. Entropy detection approach is more interested in protecting the controller from DDoS attacks than servers, while the proposed EQD method is protecting both of them. Moreover, the proposed work performs better in securing the normal traffic when the attack is performed from multiple devices. This happens due to the different speed of attacks from various devices while one device can send attacks at one constant speed.

A number of different results are generated in each scenario, as shown in the Figures 8 and 9. The average value of these results is calculated for (EQD_{avg}, $entropy_{avg}$, $normal_{avg}$, $normal+attack_{avg}$). If the absolute value of (EQD_{avg}–$entropy_{avg}$) is greater than LSD, then the two averages are statistically different.

Another scenario is a false positive attack in which the traffic received by the server is identified as an attack, but it is actually normal traffic. In entropy detection approach [19], the traffic is identified as a DDoS attack and blocks it directly. In the EQD method, the algorithm detects this traffic as suspected traffic and redirects it to the edge of the data center through a second controller. The second controller further checks the traffic using delay between packets and checks if the traffic is generated using spoofing IP. Then if the traffic has the characteristics of a DDoS attack it will block this traffic. However, if the delay between traffic is not categorized as attack and spoofing is being used then a report message is sent to the security team to further check the traffic, as shown in Figure 10.

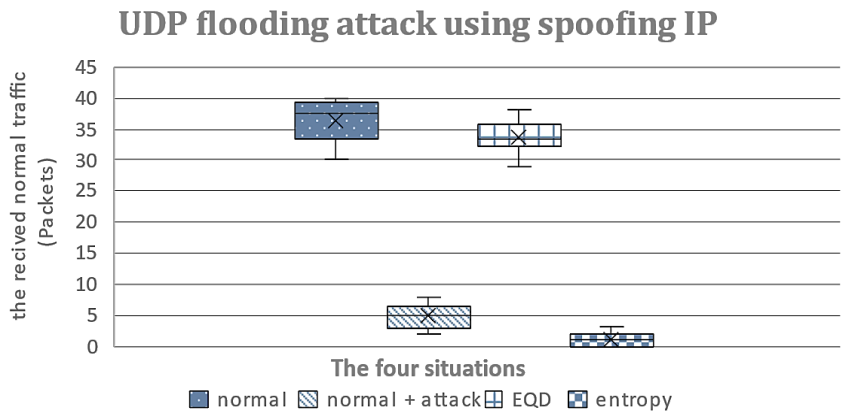

Figure 8. Received traffic during an attack from one device with spoofing source IPs

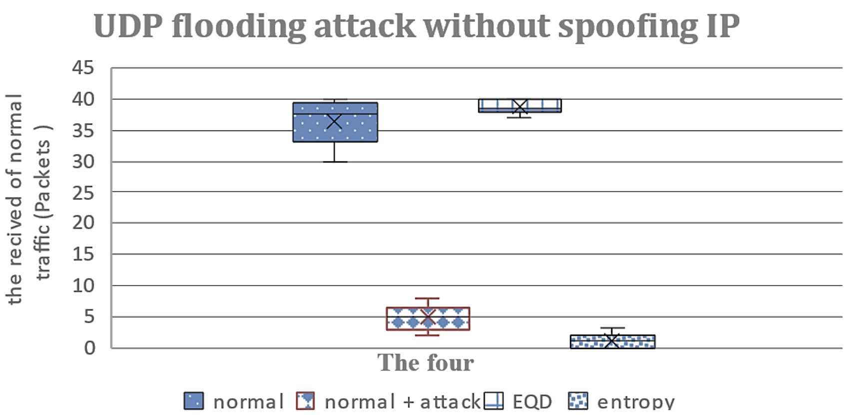

Figure 9. Received traffic during an attack with no spoofing Ips

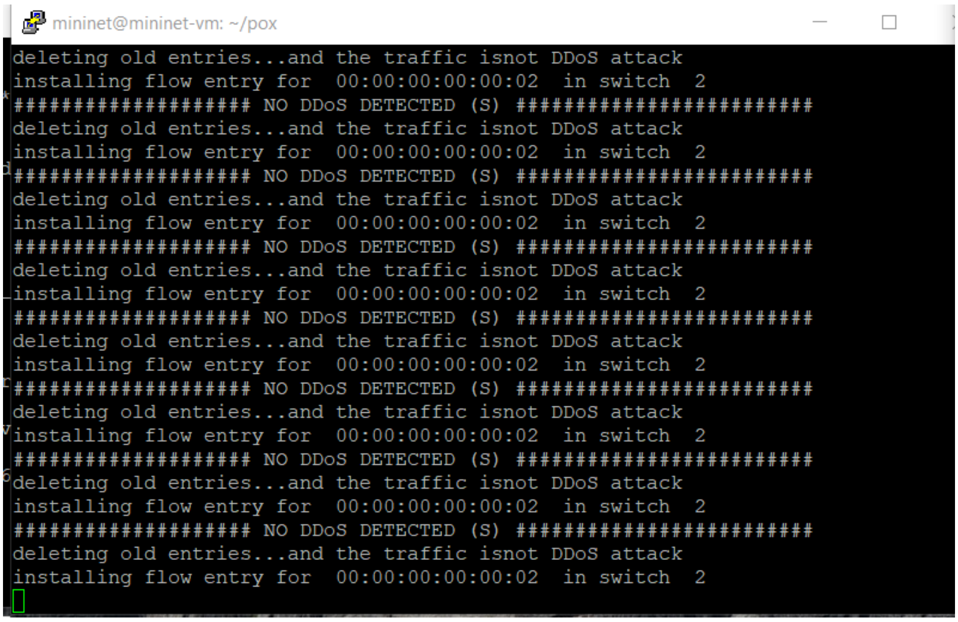

Figure 10. Second controller report

Figures 11 and 12 illustrate the difference between the EQD and entropy detection methods in performance to detect and mitigate the impact of suspected traffic from multiple hosts. In Figures 11(a) and 11(b), the EQD method is starting to detect the traffic at the server in less than one second and redirect the suspected traffic to the honeypot server at the second controller. The detection is based on the drop of entropy value, which gives punishment to the q-value of the server and reward to the honeypot server. The rule time for re forwarding the traffic to the second controller is 40 seconds as shown in Figure 11(b), While traffic from other ports can be received by the server as shown in Figure 11(a). When the entropy remains below the threshold, the EQD method blocks the ports of attack for the time period 40 to 60 sec and the honeypot server stops receiving traffic from the first controller. Furthermore, when the entropy is rising, the EQD method learns to return the traffic to the server from 60 to 75 sec. The EQD method learns how to deal with the entropy of traffic and choose the appropriate action. In Figure 12, the entropy method detects the suspected traffic and blocks the port permanently in less than one second. Due to using a very fast flood of attack traffic, the controller and the switch cannot handle the huge traffic. Then, the controller goes down which means the traffic from other ports will not be received by the server as shown in Figure 12. Furthermore, the entropy method cannot distinguish the DDoS attack from either multiple devices or one device using spoofing IPs. While the EQD method can deal with these attacks after sending them to the second controller.

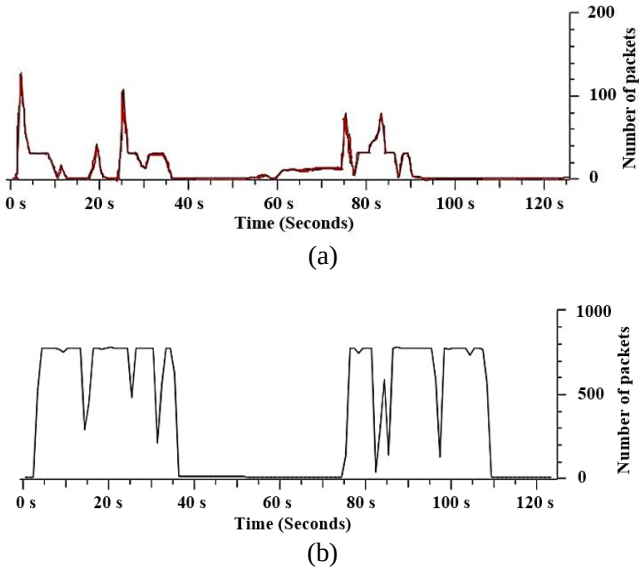

Figure 11. EQD detection methods (a) server and (b) honeypot

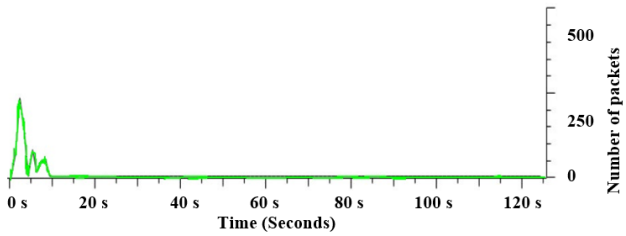

Figure 12. Entropy detection methods for server

Another scenario is used to show the accuracy of this method by using the Canadian Institute for Cyber Security Intrusion Detection System dataset (CICIDS2017) [42]. This dataset consists of real traffic which is classified into 80 network traffic features extracted for all normal and attack flows. These features are computed by using software named CICFlowMeter which is available on Canadian Institute for Cybersecurity websites. The first six columns labeled for each flow are *FlowID, SourceIP, DestinationIP, SourcePort, DestinationPort*, and protocol with more than 80 network traffic features [42], [43]. This work employs the second column as spoofing IPs to generate traffic by using Scapy and Panda library [44]. The detection time is less than one second in this experiment as shown in Figure 11(a). Moreover, this research is employed to classify attacks based on using multiple IP addresses with one MAC address which does not exist in this dataset besides time delay between packets. The results show how to classify between spoofing UDP flood attack and flash crowd based on delay by using the CICIDS2017 as normal traffic. A number of hosts send normal traffic about 10 packets/sec and one host sends a number of spoofing IP flood traffic about 500 packets/sec where other hosts send a flash crowd of traffic about 62 packets/sec. As shown in Figure 13, the results indicate 150 packets are classified as normal traffic in the second controller based on the threshold of time interval. While entropy detection classifies these packets as attacks and directly blocks the traffic. The threshold can be chosen from the range between minimum and maximum time interval for the traffic. Moreover, any host will be blocked if the spoofing UDP flood attack comes from it as shown in Figure 14.

The EQD method is considered a better solution for two reasons: First, increasing the throughput for legal users during DDoS attacks compared with entropy detection. The second one is providing load balance for the huge traffic between two controllers and two servers. Moreover, this work guarantees the continuity of accessing the services as long as possible until further analysis is performed on the suspected traffic. Another benefit of this approach is to decrease false positives and increase true positives by redirecting suspected traffic to other servers on the edge of the network through a second controller. The SDN network makes the work easier in programming because it can use any language to program while a traditional network must use a specific language to work.

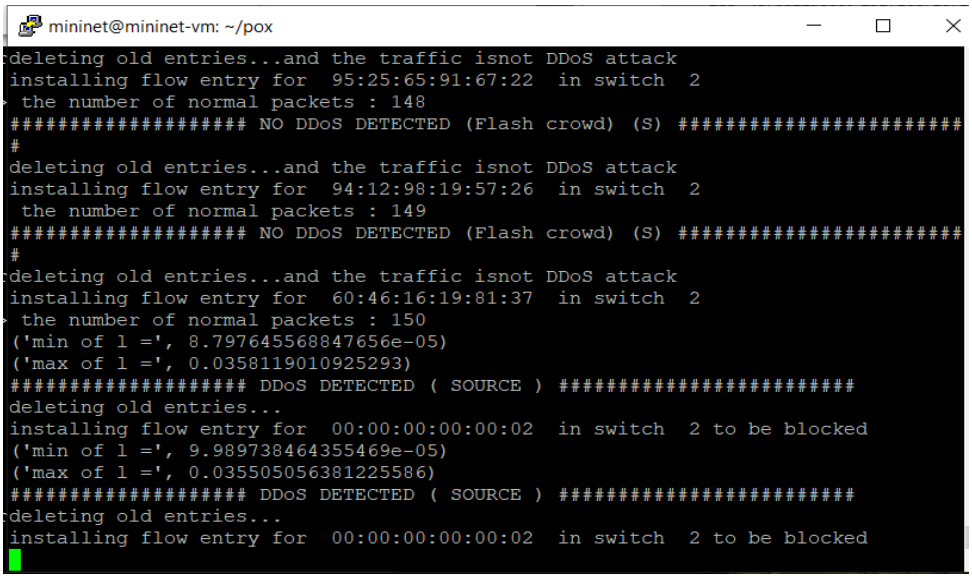

Figure 13. The second controller more analysis

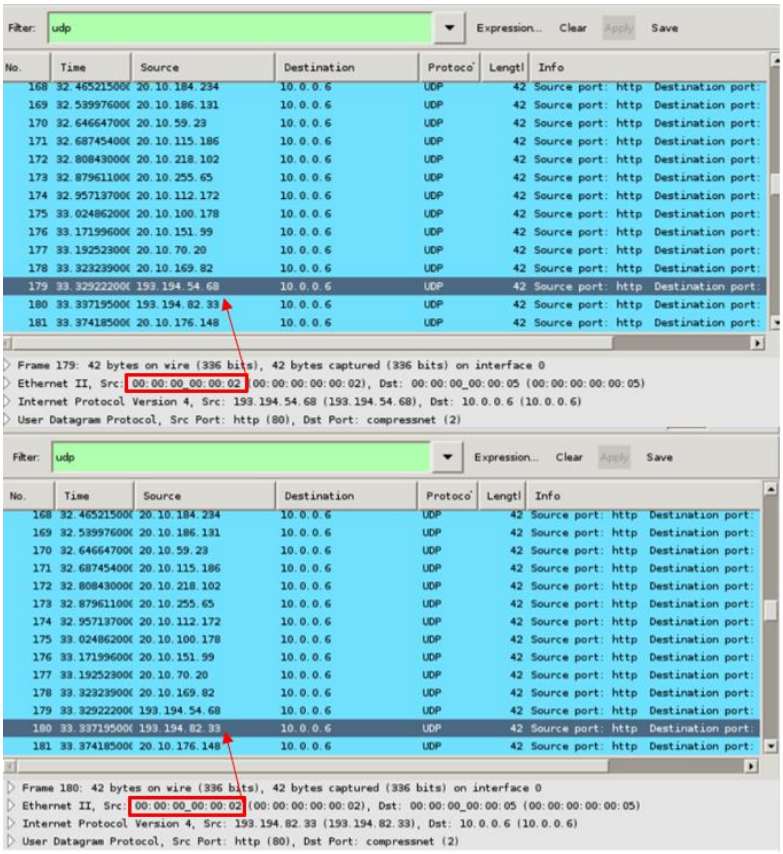

Figure 14. Wireshark showing the spoofing UDP flood atta

4. CONCLUSION

DDoS attacks are one of the most challenging cybersecurity risks in traditional networks because they attempt to shut down significant services for economic reasons. Therefore, many scholars have become interested in this subject, particularly when the network industry has evolved and the use of SDN networks has expanded. SDN networks are considered more flexible and customizable than traditional networks.

Previous research has used entropy detection to detect DDoS attacks with a direct blocking mitigation method, which is not recommended due to the increased number of false-positive results. This study combines the entropy detection and q-learning method using the OpenFlow protocol, and Open vSwitches to redirect suspected traffic to the edge of the data center. A second controller is used at an early stage to minimize damage and load balancing. The results of this work show how to distinguish between DDoS attacks and flash crowds based on MAC address and time delay. Moreover, the throughput of the server remains stable during DDoS attacks so that legal users can use the server without interrupting. Furthermore, the results indicate an increase of up to 50% in the throughput compared to other approaches and better performance to detect the attacks and give more than one solution to mitigate the attacks. This strategy increases true positives while decreasing false positives as shown by using the CICIDS2017 dataset in the experiment.

This strategy could be used in the future to protect web servers and networks from attacks in large data centers; another area for future research is the implementation of a backup controller to avoid single point failure in the network. Furthermore, this approach can be combined with other types of machine learning algorithms. Moreover, the captured traffic in the honeypot server can do further analysis in the future to distinguish the attacks more specifically.

REFERENCES

[1] M. Vizvary, "Mitigation of DDoS attacks in software defined networks," Ph.D. dissertation, Faculty of Informatics, Masaryk University, 2015.

[2] N. Z. Bawany, J. A. Shamsi, and K. Salah, "DDoS attack detection and mitigation using SDN: methods, practices, and solution," *Arabian Journal for Science and Engineering*, vol. 42, no. 2, pp. 425–441, Feb. 2017, doi: 10.1007/s13369-017-2414-5.

[3] P. Zhai, Y. Song, X. Zhu, L. Cao, J. Zhang, and C. Yang, "Distributed denial of service defense in software defined network using OpenFlow," in *IEEE/CIC International Conference on Communications in China (ICCC)*, Aug. 2020, pp. 1274–1279, doi: 10.1109/ICCC49849.2020.9238872.

[4] A. Vij and S. Pasha, "Azure DDoS protection-2021 Q3 and Q4 DDoS attack trends," *Microsoft*, 2022. https://azure.microsoft.com/en-us/blog/azure-ddos-protection-2021-q3-and-q4-ddos-attack-trends/ (accessed Mar. 30, 2022).

[5] ONF, "Software-defined networking (SDN) definition," *Open Networking Foundation*, 2017. https://opennetworking.org/sdn-definition/ (accessed Sep. 03, 2021).

[6] J. H. Cox *et al.*, "Advancing software-defined networks: a survey," *IEEE Access*, vol. 5, pp. 25487–25526, 2017, doi: 10.1109/ACCESS.2017.2762291.

[7] C. Dharmadhikari, S. Kulkarni, S. Temkar, S. Bendale, and B. E. Student, "A study of DDoS attacks in software defined networks," *International Research Journal of Engineering and Technology*, vol. 06, no. 12, pp. 448–453, 2019.

[8] F. W. Hussein, M. Abdullah, and N. Al-awad, "Design and implementation of secured software defined network system against attacks of distributed denial of service," Master Thesis, Al-Mustansiriyah University, 2019.

[9] A. H. B. Alghuraibawi, R. Abdullah, S. Manickam, and Z. A. Alkareem Alyasseri, "Detection of ICMPv6-based DDoS attacks using anomaly based intrusion detection system: A comprehensive review," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 6, pp. 5216–5228, Dec. 2021, doi: 10.11591/ijece.v11i6.pp5216-5228.

[10] D. Kreutz, F. M. V. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: a comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015, doi: 10.1109/JPROC.2014.2371999.

[11] K. Kalkan, G. Gur, and F. Alagoz, "Defense mechanisms against DDoS attacks in SDN environment," *IEEE Symposium on Computers and Communications*, pp. 669–675, 2017, doi: 10.1109/ISCC.2017.8024605.

[12] ONF, "OpenFlow 1.4 specifications," *Open Networking Foundation*, pp. 1–36, 2013.

[13] N. Joshi and D. Gupta, "A comparative study on load balancing algorithms in software defined networking," in *International Conference on Ubiquitous Communications and Network Computing*, 2019, pp. 142–150, doi: 10.1007/978-3-030-20615-4_11.

[14] S. M. S. Mousavi and M. St-Hilaire, "Early detection of DDoS attacks in software defined networks controller," Master Thesis, Carleton University, Ottawa, Canada, pp. 77–81, 2014.

[15] S. Ostermann, B. Tjaden, and M. Ramadas, "Detecting anomalous network traffic with self-organizing maps," *Recent Advances in Intrusion Detection*, pp. 36–54, 2003.

[16] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in *IEEE Local Computer Network Conference*, Oct. 2010, pp. 408–415, doi: 10.1109/LCN.2010.5735752.

[17] A. Tootoonchian "noxrepo/nox," *GitHub*. https://github.com/noxrepo/nox (accessed Nov. 02, 2021).

[18] Y. Jiang, X. Zhang, Q. Zhou, and Z. Cheng, "An entropy-based DDoS defense mechanism in software defined networks," in *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering*, vol. 209, 2018, pp. 169–178.

[19] A. Rai, P. D Vyavahare, and A. Jain, "Distributed DoS attack detection and mitigation in software defined network (SDN)," *SSRN Electronic Journal*, pp. 2–5, 2019, doi: 10.2139/ssrn.3363568.

[20] M. A. Mohsin and A. H. Hamad, "Implementation of entropy-based DDoS attack detection method in different SDN topologies," *American Academic Scientific Research Journal for Engineering*, vol. 86, no. 1, pp. 63–76, 2022.

[21] K. Bavani, M. P. Ramkumar and E. Selvan G.S.R., "Statistical approach based detection of distributed denial of service attack in a software defined network," in *6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Mar. 2020, pp. 380–385, doi: 10.1109/ICACCS48705.2020.9074231.

[22] A. Banitalebi Dehkordi, M. Soltanaghaei, and F. Z. Boroujeni, "The DDoS attacks detection through machine learning and statistical methods in SDN," *The Journal of Supercomputing*, vol. 77, no. 3, pp. 2383–2415, Mar. 2021, doi: 10.1007/s11227-020-03323-w.

[23] M. Abdullah, N. Al-awad, and F. Hussein, "Implementation of entropy-based distributed denial of service attack detection method in multiple POX controllers," *Review of Computer Engineering Studies*, vol. 6, no. 2, pp. 29–38, 2019, doi: 10.18280/rces.060201.

[24] R. N. Carvalho, J. L. Bordim, and E. A. P. Alchieri, "Entropy-based DoS attack identification in SDN," in *IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, May 2019, pp. 627–634, doi: 10.1109/IPDPSW.2019.00108.
[25] T. Omar, A. Ho, and B. Urbina, "Detection of DDoS in SDN environment using entropy-based detection," in *IEEE International Symposium on Technologies for Homeland Security (HST)*, Nov. 2019, pp. 1–4, doi: 10.1109/HST47167.2019.9032893.
[26] G. Al-Sharif, A. Barnawi, and M. Boulares, "SDN multi-controller based framework to detect and mitigate DDoS in large-scale network," *International Research Journal of Engineering and Technology*, pp. 1134–1139, 2019.
[27] D. Kavitha, R. Ramalakshmi, and R. Murugeswari, "The detection and mitigation of distributed denial-of-service (DDoS) attacks in software defined networks using distributed controllers," in *2019 IEEE International Conference on Clean Energy and Energy Efficient Electronics Circuit for Sustainable Development (INCCES)*, 2019, pp. 1–5, doi: 10.1109/INCCES47820.2019.9167698.
[28] T. Pandikumar, F. Atkilt, C. A. Hassen, and M. Tech Student, "Early detection of DDoS attacks in a multi-controller based SDN," *International Journal of Engineering Science and Computing*, vol. 7, no. 6, pp. 13422–13429, 2017.
[29] M. Badrinath, P. Parab, R. Mishra, V. Gupta, V. Chavan, and V. Vasudev, "Custom network security implementation for detection and prevention of DDoS attacks in SDN," vol. 6. 2016.
[30] P. Bereziński, B. Jasiul, and M. Szpyrka, "An entropy-based network anomaly detection method," *Entropy*, vol. 17, no. 4, pp. 2367–2408, Apr. 2015, doi: 10.3390/e17042367.
[31] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, Jul. 1948, doi: 10.1002/j.1538-7305.1948.tb01338.x.
[32] Y. Zhao, Y. Li, X. Zhang, G. Geng, W. Zhang, and Y. Sun, "A survey of networking applications applying the software defined networking concept based on machine learning," *IEEE Access*, vol. 7, pp. 95397–95417, 2019, doi: 10.1109/ACCESS.2019.2928564.
[33] Y. Cui, X. Huang, D. Wu, and H. Zheng, "Q-learning," *Machine Learning*, vol. 8, pp. 279–292, 1992, doi: 10.1109/ICCC49849.2020.9238991.
[34] A. Al-Saadi, R. Setchi, Y. Hicks, and S. M. Allen, "Routing protocol for heterogeneous wireless mesh networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 12, pp. 9773–9786, 2016, doi: 10.1109/TVT.2016.2518931.
[35] Mininet Project Contributors, "Mininet: an instant virtual network on your laptop (or other PC) - Mininet," *Mininet.org*. http://mininet.org/ (accessed Sep. 17, 2021).
[36] OvS, "Production quality, multilayer open virtual switch," *Open vSwitch?*, 2016. https://www.openvswitch.org/ (accessed Sep. 17, 2021).
[37] Murphy *et al.*, "noxrepo/pox," *Noxrepo.org*. https://github.com/noxrepo/pox (accessed Nov. 02, 2021).
[38] Wireshark, "Wireshark · go deep," *Wireshark.org*. https://www.wireshark.org (accessed Oct. 10, 2021).
[39] Scapy, "Scapy," *Scapy.net*. https://scapy.net/ (accessed Oct. 10, 2021).
[40] Hping, "active network security tool," *Hping.org*. http://www.hping.org/ (accessed Nov. 13, 2021).
[41] R. L. Ott and M. T. Longnecker, *An introduction to statistical methods and data analysis*, 6th ed. USA: Thomson Learning Academic Resource Center, 2015.
[42] A. A. Abdulrahman and M. K. Ibrahem, "Evaluation of DDoS attacks detection in a new intrusion dataset based on classification algorithms," *Iraqi Journal of Information and Communications Technology*, vol. 1, no. 3, pp. 49–55, Feb. 2019, doi: 10.31987/ijict.1.3.40.
[43] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy*, 2018, pp. 108–116, doi: 10.5220/0006639801080116.
[44] W. McKinney and P. D. Team, "Pandas: powerful Python data analysis toolkit release 1.4.3," *Wes McKinney and the Pandas Development Team*. 2022.

BIOGRAPHIES OF AUTHORS

Hiba Salah Yaseen received a B.Sc. degree in computer engineering from the University of Technology, Baghdad, Iraq in 2009, and works as technical support at the Information technology Department in the Ministry of Planning. She has an experience in the network field and the operating system of servers besides computer maintenance. Currently, she is studying to get an M.S degree in computer networks from the University of Technology. Her research is interested in software defined networks and security. She can be contacted at email: ce.19.01@grad.uotechnology.edu.iq.

Ahmed Al-Saadi received the B.Sc. and M.Sc. degrees in computer and software engineering from the University of Technology, Baghdad, Iraq, in 2005 and 2008, respectively, and the Ph.D. degree from Cardiff University, Cardiff, United Kingdom (UK), in 2015. He is a Researcher/Lecturer with the Computer Engineering Department, University of Technology, Baghdad, Iraq. His current research interests include wireless communication, heterogeneous networks, semantic computing, and routing algorithms. He can be contacted at email: 120027@uotechnology.edu.iq.