DOI: https://doi.org/10.33103/uot.ijccce.21.4.6

Load Balancing and Detection of Distributed Denial of Service Attacks Using Entropy Detection

Hiba Salah Yaseen[1] , Ahmed Al-Saadi[2]
[1,2]Computer Engineering Department, University of Technology, Baghdad, Iraq
ce.19.01@grad.uotechnology.edu.iq[1], 120027@uotechnology.edu.iq[2]

Abstract— *Software Defined Network (SDN) is a modern network architecture that has a centralized controller. It is more flexible, and programmable due to the separation of the control plane from the data plane. However, Distributed Denial of Service (DDoS) attacks is one of the dangers that the SDN network is facing. It could attack and stop the controller from working, causing the whole system to be down. Moreover, DDoS attacks can target the hosts and the switches to stop the services for a long time as they could cause more damage to the network or datacenter. In this work, a proposed approach is utilized to protect datacenter networks and servers from DDoS attacks using entropy and real SDN-controller Python Network Operating system (POX) by redirect traffic to the edge of the datacenter to minimize the damage. The results of this experiment show how to detect abnormal traffics in an early stage and isolate them in a server outside the datacenter to distribute the huge amount of traffic in more than one server and avoid congestion on switches. Also, the throughput of the server was increased by about %16 during the suspected attack, this means maintaining the service until further analysis to be done on the traffic. These results are compared with the direct block mitigation method which was mostly used with the entropy detection method in previous researches. Moreover, this work is done to confirm whether the suspected traffic is an actual attack or not. Therefore, this method will decrease the false positives of detection.*

IndexTerms— *Software-Defined Networking (SDN), Distributed Denial of Service (DDoS) attacks, OpenFlow, Python Network Operating system (POX)*

I. INTRODUCTION

Networks are becoming more complex due to the introduction of many heterogeneous systems, devices, and software. Networks need to be capable of managing the continuous change of traffic patterns. The goals of next-generation networks are to build intelligent networking architecture for intellectualization, activation, and customization as much as possible [1]. Therefore, a new paradigm appeared which is known as Software Defined Network (SDN). SDN separates the control plane from the data plane to create centralized control which is more dynamic, flexible, manageable, and adaptable [2, 3]. The control plane contains the routing algorithm that determines the path from source to destination. It monitors all the networks in order to create routing tables or flow tables [4]. Data plane is responsible for reforwarding packets from switches or router's input to its appropriate output. In a traditional network, there is no centralized control and every element has its own control plane as shown in *Fig 1*. For that reason, every change or update in the configuration will take a long time to be done. Moreover, the increase in demand for online services like cloud computing, big data applications, and automated networking platforms for IoT was another reason for needing a network being more flexible, programmable, and lower cost than a conventional network [5].

Received 11/6/2021; Accepted 20l8l2021

DOI: https://doi.org/10.33103/uot.ijccce.21.4.6

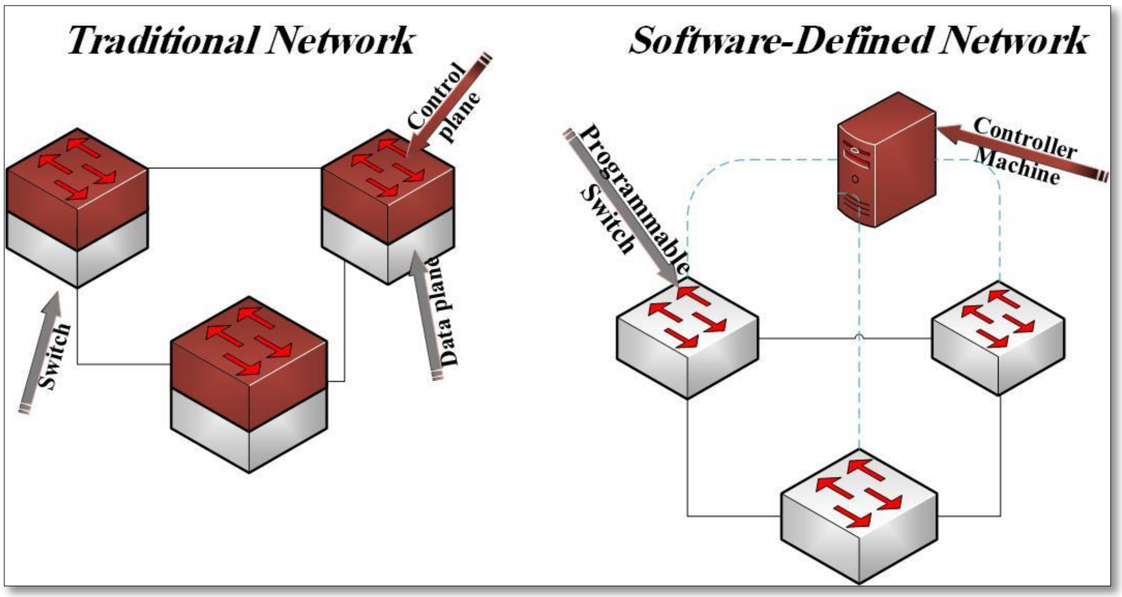

FIG. 1. THE DIFFERENCE BETWEEN THE TWO NETWORKS [6]

The SDN network is consists of 3 layers as shown in *Fig 2*:

I. **The application layer** contains network applications like security, load balancing…etc. These can be programmed to improve them by using any programming language such as python, java, and C++ depending on the type of controller being used in the network therefore the SDN is considered programmable [6].

II. **The Control layer** is considered the brain of the network because it is responsible for managing, creating routing tables or flow tables also it has a global view for all the network components [6].

III. **The infrastructure layer** contains the forwarding devices such as switches, routers that support OpenFlow protocol, and called OpenFlow switches [6].

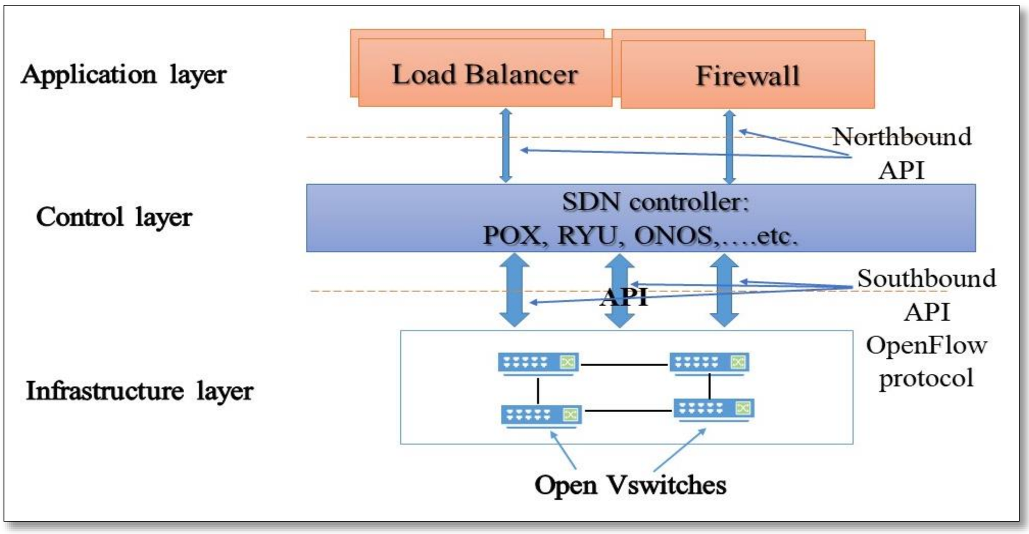

FIG. 2. SDN ARCHICTURE [6]

Received 11/6/2021; Accepted 20l8l2021

OpenFlow protocol is a southbound Application Program Interface (API) located between the control layer and data layer or infrastructure layer. It is responsible for the communication between the controller and OpenFlow switches by delivering control messages. Northbound API exists between the application layer and control layer. It works as an interface for programmers to develop new applications and deliver these commands for a controller. It helps the controller to understand and execute them on all network devices rather than updating them individually. Until now there is no popular standard for northbound API [6, 7].

However, despite SDN network benefits, there are new faults and attack points introduced due to the programmable and centralized. For instance, a successful **Distributed Denial of Service (DDoS) attack** in SDN controllers could stop the entire network. The attacker employs a huge amount of fake traffic packets that originated from multiple devices toward a victim device such as a web server. The objective of such attacks is to stop the services of some companies or governments as long as possible to prevent legitimate users from accessing services. The number of DDoS attacks on important websites is increasing every day and becoming more dangerous because they are causing huge commercial and economic loss [8]. Therefore, the security field in SDN networks is drawing the attention of many researchers more than a traditional network, especially DDoS attacks. DDoS attacks are considered today the most threatening challenge to internet security and traditional networks [9].

Also, there are many types of DDoS attacks such as SYN flooding attack, ping of death, IP address Spoofing attack…etc [10]. Therefore there are different types of DDoS detection methods and each one of them has its pros and cons. In another word, it's not possible to have a method with a superior solution to detect and mitigate DDoS attacks. The most popular detection techniques in SDN networks include entropy, machine learning, traffic pattern analysis, connection rate, and other techniques [11, 12]. This work proposes an entropy detection method based on the destination Internet Protocol (IP) address because it has a low calculation overload on the controller. This method is used with a real SDN controller to detect the unusual traffic in a datacenter and isolate it early before causing any loss.

The proposed approach utilizes entropy detection and threshold to detect the suspect traffic when it is lower than the threshold and redirect traffic to another server for further analysis rather than immediately blocking it. But when the traffic goes upper the entropy threshold the proposed algorithm returns the suspected traffic to the main server and keeps the service continuity without interruption. This method proved by results increasing the throughput for the servers during attacks and increases true positives while decreases false positives.

Beside section 1 explained above, this paper is organized as the following: section 2 highlights the related work; section 3 explains the proposed Entropy Detection; section 4 discusses experiment results; while section 5 consists of the conclusion and future work.

II. RELATED WORK

The first and foremost SDN standard is OpenFlow which was proposed by McKeown et al. in 2008 at Stanford University [13]. The OpenFlow protocol acts as a

Received 11/6/2021; Accepted 20l8l2021

DOI: https://doi.org/10.33103/uot.ijccce.21.4.6

fundamental role in SDN architecture and it allows separation control and data planes in modern networks. In order to forward data in switches, entities in flow tables are employed to select the forwarding port [14].

Each OpenFlow switch has one or more flow tables and each one of them has flow entries. The incoming packets can be manipulated and forwarded to the desired destination based on information of each entry in the flow table as shown in *Fig 3* [15].

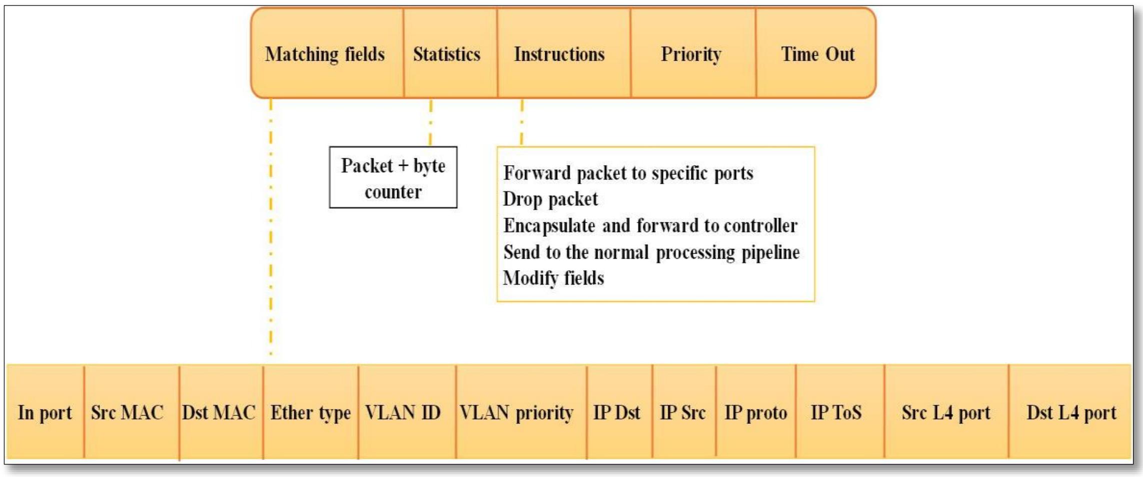

FIG. 3. FLOW ENTRY [15]

DDoS attacks are the most dangerous threat in traditional and SDN networks [16]. Therefore, there are many techniques used to detect these attacks based on the accepted rate of bandwidth consumption and pattern of network activities in normal circumstances. When a sudden increase happens in the traffic flow, delay, (Central Processing Unit) CPU utilization, or a sudden drop in performance of any of the network assets, these will be considered abnormal. These methods can be classified into statistical analysis and machine learning [16]. Statistical analysis such as entropy and chi-square depend on extracting independent information from packet header like packet type, destination IP address, source IP address…etc. Entropy is used to find the unexpected and unknown abnormal traffic while chi-square is used to find the abnormal traffic in a certain type of intrusion and known the type of packet header. For example, if TCP-SYN flood is the expected traffic, take a sample bin of data and measure the number of TCP-SYN headers. It will show a pattern of the average number of such headers. Any deviation beyond the recognized limits is abnormal [16].

Machine learning is another approach to detect DDoS attacks by using dynamic filters to detect abnormal traffic. An algorithm is trained to constantly update its filtering criteria based on the events in the network. An instance of it is the Self-Organizing Method (SOM) is trained by collecting the flow statistics from OpenFlow switches [17]. The parameter used to train SOM is average packet per flow, average bytes per flow, an average of duration per flow, percentage of pair flows, growth of single flow, and growth of single ports. SOM is improved as time passes and processes more information about these statistics [18]. However, there is a disadvantage in this method that in every time changes are needed in the algorithm, the algorithm needs to be trained again which consumes time and resources. In the proposed approach, the algorithm is based on

Received 11/6/2021; Accepted 20l8l2021

DOI: https://doi.org/10.33103/uot.ijccce.21.4.6

entropy and redirects suspected traffic to the edge of the network. Thus, no training is required to update the algorithm or add new parameters.

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. The window used here 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's size of 50 because it doesn’t impose a computational burden on the CPU and memory. Furthermore, it immediately blocks the suspect port. This approach has some limitations in which it doesn’t do further analysis to confirm whether this is an actual attack [19]. Another experiment is comparing the performance of different topologies (single, tree, and linear) and controllers (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. The result of this work was the POX controller because it has less delay, jitter, and high throughput. The entropy detection used in this approach with sixteen POX controllers to decrease the load in linear topology and get better detection for DDoS attack. In a single topology, the detection is much better but there is a problem with the failure point of switch and controller. Once the DDoS attack detects the suspect port it immediately blocks without further analysis to be sure of the attack [20].

Another work proposes an approach to monitor the network and protect it against UDP flooding and TCP-SYN flood DoS attacks. In this work, 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 detection fields that can be configured to match the values desired by the network operator. The result was detecting 250 packets during 100-165 seconds and the better window size is 60 because of its detection accuracy and doesn't impose an excessive processing load on the controller. In this approach, there are no mitigation mechanisms but only blocking the port of attack instantly after the detection [21]. Other research implemented in a small-scaled network test to detect UDP flood DDoS attacks by using simulation. This system uses a Raspberry Pi as Open V Switch (OVS) which is connected with at least two hosts and the used controller is POX. Although the detection method used is the entropy of a system that calculates the number of incoming packets for and a destination IP address in a specific window. The computed entropy is compared with a specific threshold to find out the DDoS attack. Then utilizes OpenFlow protocol 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. The result of detection was within three to ten seconds and blocked the detected port [22]. These researches can be summarized in Table 1.

This work handles the limitation in previous research by using the entropy method to detect the suspicious traffic and redirect it to more than one server on the edge of the network to reduce the harm of the attack and for further analysis. Blocking ports or dropping packets directly is a simple and fast approach to mitigate the effect of DDoS but it could result in false positives and block legitimate traffic too. Therefore, this research will isolate suspicious traffic away from the data center and make further analysis before doing any further action. While previous research uses a drop or block traffic which leads to shutting down that port on the switch and making it unconnected.

Received 11/6/2021; Accepted 20l8l2021

DOI: https://doi.org/10.33103/uot.ijccce.21.4.6

TABLE 1. RELATED WORK SUMMARIZED

No.	Author	Environment	Propose	Drawbacks
1	E. Mota, A. Passito R. Braga	NOX controller + virtual OpenFlow switch	using a machine learning to detect DDoS attack based on training Self-Organizing Method (SOM) by collecting the flow statistics from OpenFlow switches	In this method every time changes are needed in the algorithm, the algorithm needs to be trained again which consumes time and resources. Also the throughput is drop during attack [17].
2	Ankita Rai, Prakash D. Vyayahare, Anjiana,Jain	POX controller + virtual OpenFlow switch	Used Entropy-based detection algorithm to detect the DDoS attack at early stage by measuring the entropy of a system with different window size.	This approach has some limitations in which it doesn't do further analysis to confirm whether this is an actual attack but made immediate block which made lower throughput for a attacked server [19].
3	Mahmood Z. Abdullah, Nasir A.Al-awad, Fatima W.Hussein	POX, RYU, libfluid, Open Network Operating System (ONOS), OpenDaylight Controllers + virtual OpenFlow switch	Used Entropy-based detection algorithm on different types of controllers to test different network topologies and different number of controllers.	Once the DDoS attack detects the suspect port it immediately blocks without further analysis which lead to lower the throughput of server [20].
4	Tamer Omar, Anthony Ho, Brian Urbina	POX controller + Raspberry Pi as Open V Switch (OVS) which is connected with at least two hosts	Used Entropy-based detection algorithm to detect the DDoS attack and to mitigation the attack it is used non-existing a port.	There are no further analysis for the suspected traffic but only a direct block of the suspected port [21].
5	Ranyelson N. Carvalho, Jacir L. Bordim and Eduardo A. P.Alchieri	POX controller + virtual OpenFlow switch	Used Entropy-based detection algorithm on controller and the statistical information gathered from openflow switch to detect DoS attack.	there are no mitigation mechanisms but only blocking the port of attack instantly after the detection [22].

III. LOWER ENTROPY DETECTION (LED) FOR DDoS ATTACK

Entropy is a well-known and famous concept in information theory. It is measuring the uncertainty associated with a random variable which describes the degree of dispersion or concentration of distribution [21]. The entropy is increased when the randomness or disorder is high and vice versa. This was useful to be used in traditional networks to detect and categorize network anomalies, which are often very small and hidden in a network traffic volume. These anomalies are expressed by the number of flows, packets, or bytes, so their detection with popular solutions relies mostly on traffic volume changes [23].

Therefore, entropy can be used to detect DDoS attacks based on packet attributes randomness that enters the network. These attributes can be Source IP, Destination IP, Source Port, Destination Port, and Size [21]. This research relies on destination IP and two essential components which are window size and a threshold for detecting DDoS

Received 11/6/2021; Accepted 20l8l2021

DOI: https://doi.org/10.33103/uot.ijccce.21.4.6

attacks. Windows size is the number of packets in a period of time. Entropy is calculating the uncertainty of packets within a window in a system to detect DDoS attacks, it is required to limit the entropy by using a threshold. If the calculated entropy is below the threshold, then an attack occurs.

Entropy is defined as:

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

Let p_i is the probability of occurrence y_i which is the number of packets for each destination IP (Internet Protocol) address x in the window W while n is the number of different occurrences in the window [21].

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

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

When a new packet arrives at the SDN switch, its source address is unknown and will be sent to the controller. The controller will install a new flow entry inside the table in the switch so that any further incoming packets can reach their destination in the network without passing through the controller as shown in *Fig 4.*

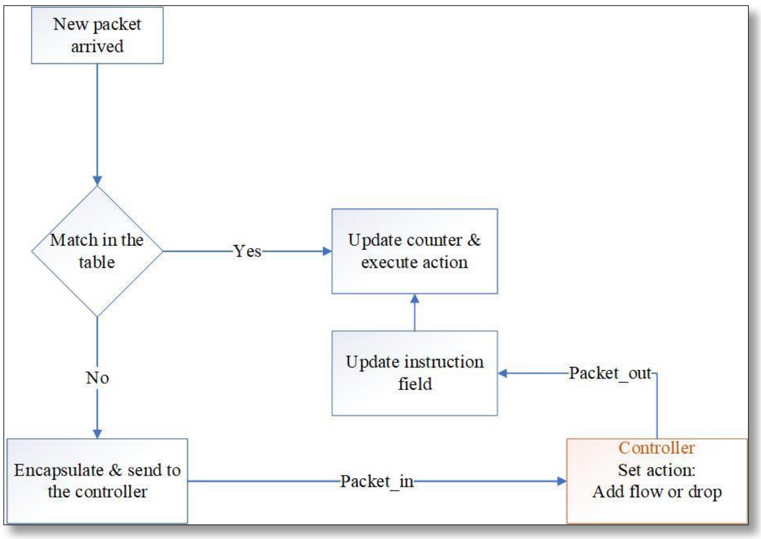

FIG. 4. FLOW ENTRY PROCESS [19]

In this approach, spoofing IP addresses are used to send new packets to the network and if they do not match the flow table, the switch will encapsulate the packet and send it to the controller as a Packet_In message. It will gather 50 packets for different destination IP addresses and store them in a hash table that consists of two columns. The first column for the destination IP address and the second one to count how many times this IP is repeated. Moreover, calculate the probability of occurrence for these IP addresses and the entropy for them in normal traffic and attack traffic to find the threshold. Furthermore, compare the entropy with the threshold if it was higher than a threshold or below. Then, two things must be done after comparison: Firstly, if the traffic is suspicious, then count the number of packets on the incoming port then append it, and the switch DPID to a black table which is DDoSlist. That table stored the number of times they were repeated

Received 11/6/2021; Accepted 20l8l2021

DOI: https://doi.org/10.33103/uot.ijccce.21.4.6

for normal and suspicious traffic. If it was normal, the number of times is increased by one. But if it was suspicious, then the number is decreased by one. Lastly, check the black table content with how many times the incoming port occurred in one hour after that clear the black table, as shown in *Fig 5*.

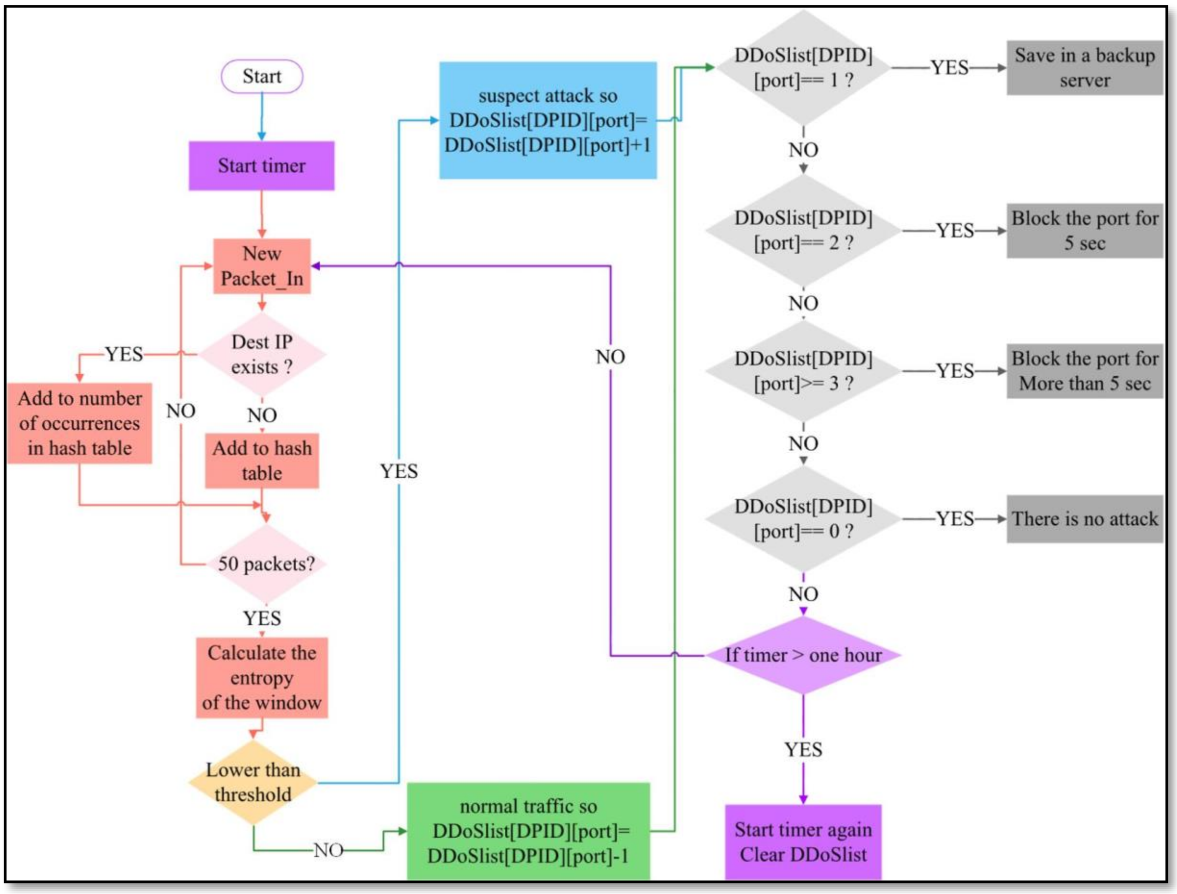
Start
Start timer
New Packet_In
Dest IP exists ?
YES
NO
Add to number of occurrences in hash table
NO
Add to hash table
50 packets?
YES
Calculate the entropy of the window
Lower than threshold
NO
normal traffic so DDoSlist[DPID][port]= DDoSlist[DPID][port]-1
YES
suspect attack so DDoSlist[DPID][port]= DDoSlist[DPID][port]+1
NO
DDoSlist[DPID] [port]== 1 ?
YES
Save in a backup server
NO
DDoSlist[DPID] [port]== 2 ?
YES
Block the port for 5 sec
NO
DDoSlist[DPID] [port]>= 3 ?
YES
Block the port for More than 5 sec
NO
DDoSlist[DPID] [port]== 0 ?
YES
There is no attack
NO
If timer > one hour
YES
Start timer again Clear DDoSlist

FIG. 5. LED DETECTION FLOWCHART

If the counted number is 1, the controller installs a flow rule in the switch's table using the OPFT-FLOW-MOD message to redirect the packets to a backup server. If the counted number is 2, then block the suspect port for 5 seconds, but if it was 3 or more, then block the port for more than 5 seconds , as shown in *Fig 5*.

IV. PERFORMANCE EVALUATION

In this part, LED is evaluated using Mininet [24] tool emulator which is very widely employed in SDN research and POX controller. This work shows the result of comparing the proposed approach of entropy detection and mitigation of DDoS attacks with the previous mitigation method in terms of throughput [19 – 22].

A. Simulation setup:

Mininet is used to emulate the Open V Switch [25], which supports OpenFlow [14] protocol in order to evaluate the proposed DDoS detection and mitigation method in the SDN networks. Mininet emulation is installed on oracle virtual box version 6.0.18 that is

Received 11/6/2021; Accepted 20l8l2021

DOI: https://doi.org/10.33103/uot.ijccce.21.4.6

used to create network topology utilized in this research. A real programmable SDN controller (POX) is used as a remote controller in which the proposed algorithm is deployed. The emulated network utilizes OpenFlow protocol using remote POX controller on port: 6633, one server, backup server, and the three sets of a client to implement three scenarios to test the performance of the proposed system, as shown in *Fig 6*. Internet Performance Working group (Iperf) [26] tool is utilized to generate TCP and UDP traffic from clients to servers and then calculate the throughput on each server. Scapy library is a tool used to generate TCP and UDP as normal and TCP flood attack traffic using python [27].

One host sends spoofing normal traffic through the network in order to calculate the entropy for every 50 packets which is the window size. Window size is the number of packets sent in a period of time and it sets 50 because the number of hosts is 64 and using a small number of window sizes like 5, 10, and 20 is too small to choose a threshold [19]. A suitable threshold is chosen by running several attacks on the hosts and controller and it is within the difference between the entropy of normal traffic and entropy of attack traffic. This experiment has been repeated 20 times to test the system under various loads.

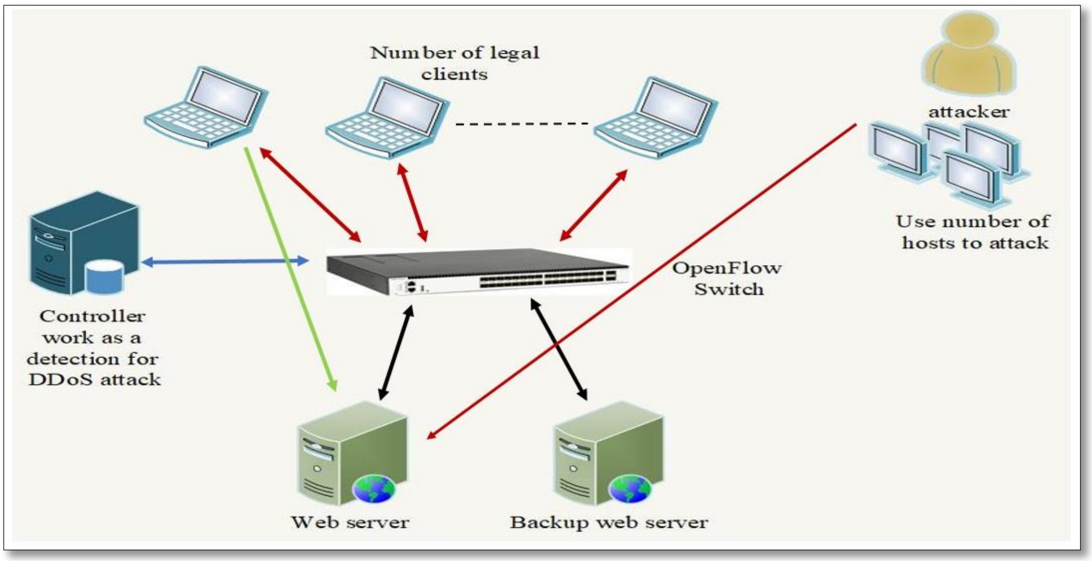
Number of legal clients
attacker
Use number of hosts to attack
OpenFlow Switch
Controller work as a detection for DDoS attack
Web server
Backup web server

FIG. 6. NETWORK TOPOLOGY

B. Experiments results:

The proposed approach in this project is evaluated and compared with DDoS attack mitigation that utilizes direct block by using OpenFlow protocol features and entropy detection [19]. The results are compared in terms of throughput for the webserver before and after the attack. This work is using a statistical tool which is an Analysis of Variance Statistical (ANOVA) to verify that the compared mitigations are statistically different using $F > F_{crit}$.

ANOVA has three parameters F, F_{crit}, and P; F is a comparison of the variation between sample means and the variation within sample means, F_{crit} is the value extracted from the analysis variance table, and P is the probability of the difference happening by

Received 11/6/2021; Accepted 20/8/2021

DOI: https://doi.org/10.33103/uot.ijccce.21.4.6

chance. The acceptable value for P is less than 0.05. If F is greater than F_{crit} then the null hypothesis is rejected at the 0.05 significance level and the mean of throughput samples is significantly different [29]. Fisher's least significant difference (LSD) is used to show whether the proposed approach (LED) achieves higher throughput or not. LSD value is calculated using Eq. (4).

$$LSD_{A,B} = t_{0.05/2DFW} \sqrt{MSW\left(\frac{1}{nA} + \frac{1}{nB}\right)} \qquad (4)$$

Where t is a critical value for the degree of freedom associated with mean square variance (MS_{within}) and n is the number of samples [28].

Tables 2 and 3 show the ANOVA and LSD results for the normal conditions and 2 approaches in terms of throughput.

TABLE 2. ANOVA RESULTS

Number of attack hosts	ANOVA Test		
	F	F_{crit}	$P < 0.05$
15Hosts_TCP	187.733925	3.1504	1.50E-26
30Hosts_TCP	925.5991	3.1504	8.04203E-46
35Hosts_TCP	512.78221	3.1504	1.8831E-38

TABLE 3. LSD RESULTS

Number of attack hosts	**Throughput Average for the Networks (Mbps)**			LSD
	LED	Entropy detection	Normal	
15Hosts_TCP	809.238	617.714	933.357	32.831
30Hosts_TCP	662.524	496.810	942.381	20.972
35Hosts_TCP	633.810	454.810	953.476	31.558

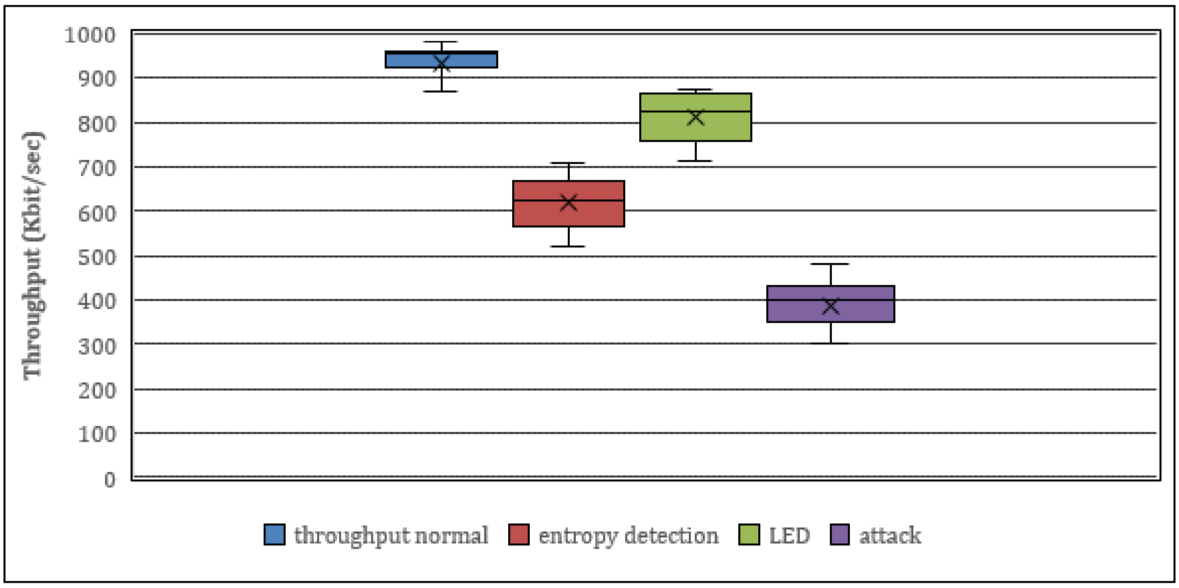

FIG. 7. TCP THROUGHPUT RESULTS ON THE DESTINATION SERVER WHEN 15 HOSTS ATTACK

Received 11/6/2021; Accepted 20l8l2021

DOI: https://doi.org/10.33103/uot.ijccce.21.4.6

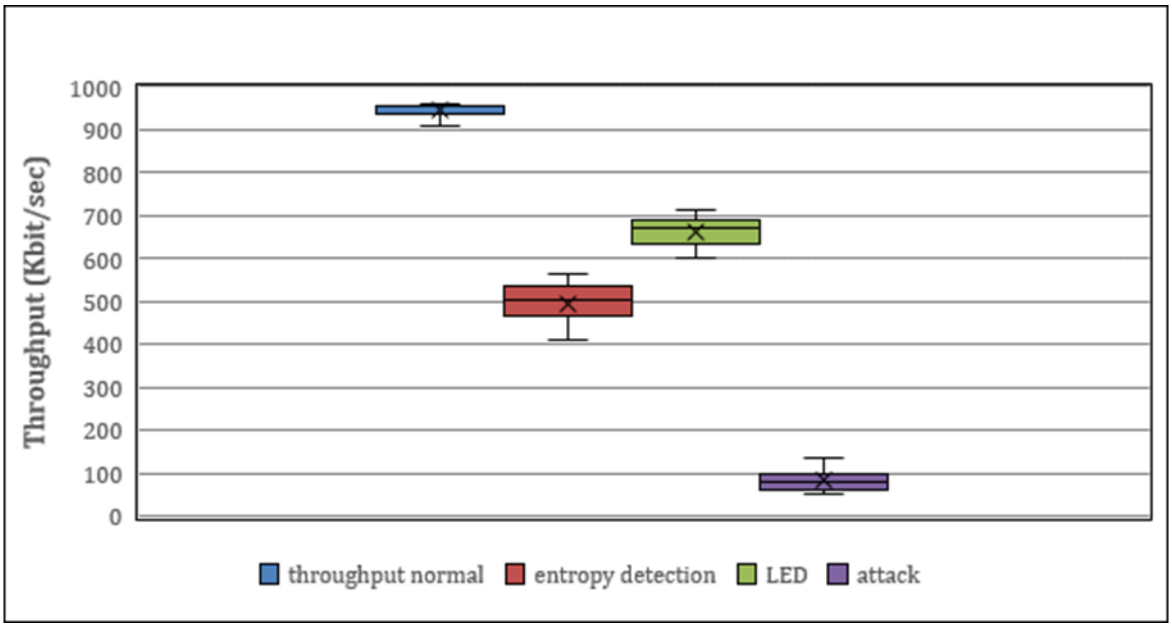

FIG. 8. TCP THROUGHPUT RESULTS ON THE DESTINATION SERVER WHEN 30 HOSTS ATTACK

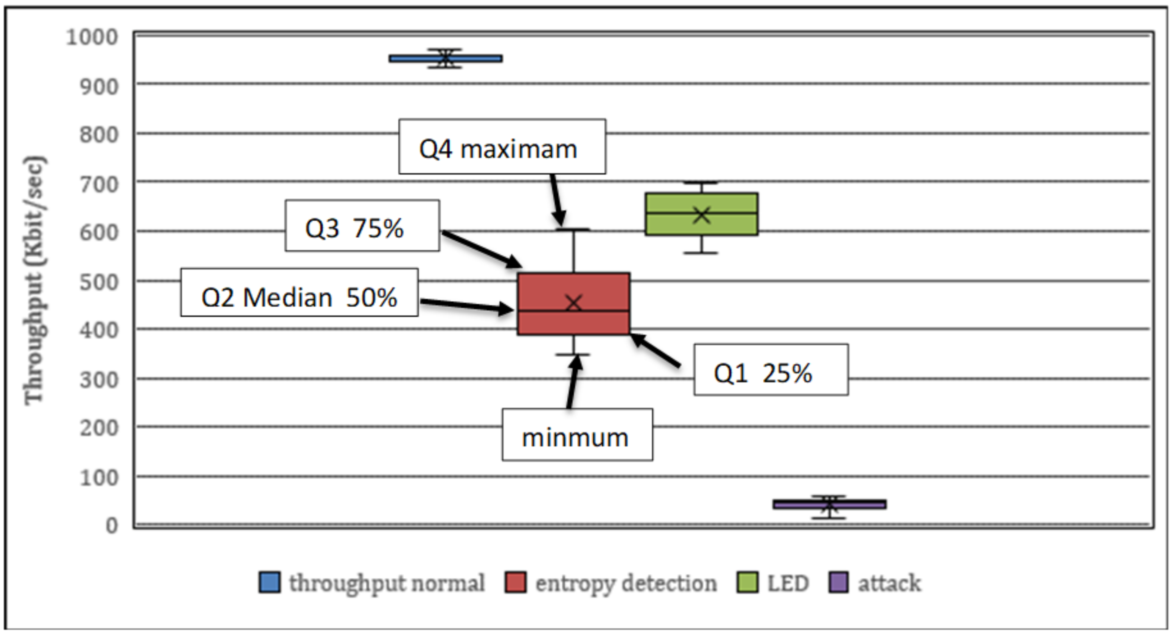

FIG. 9. TCP THROUGHPUT RESULTS ON THE DESTINATION SERVER WHEN 35 HOSTS ATTACK

Fig (7-8-9) illustrate the results of TCP throughput of a web server when different numbers of hosts attack; Box and Whisker graph is used to present the results. The Box is divided by median, so it can show the average throughputs higher and lower than the median, while Whisker represents the maximum and minimum values. For instance, in *Fig* 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 that 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

Received 11/6/2021; Accepted 20l8l2021

the highest quartile of data up to the maximum value. *Fig* 8 shows that the throughput in normal circumstances is between 932 and 957 Kbit/sec, while during DDoS attack the throughput drops to 62 or reaches 82 Kbit/sec maximums. This occurs when 30 hosts send 500 packets/sec as TCP flooding attacks at the same time to a server for 5 minutes to prevent other clients from reaching the webserver. Therefore, increasing throughput during an attack as long as possible is very important. According to the results, the LED method reaches 688 Kbit/sec and More than 50% of the median is greater than normal entropy detection. *Fig* 9 illustrates the range of throughput for entropy detection becomes lower when 35 hosts attack the web server, while the LED method is still constant between 592 and 678 Kbit/sec.

A number of different throughput results is generated in each scenario, as shown in figures. The average value of these results is calculated for (LEDavg, entropyavg, normalavg). If the absolute value of (LED avg – entropy avg) is greater than LSD, then the two averages are statistically different. Since the average LED is the nearest to the normal traffic and it is statistically different.

Then, it is considered a better mitigation solution due to the increase of throughput about 16% than using entropy detection and direct block for the webserver. Moreover, this work guarantees the continuity of accessing the services as long as possible until further analysis performed on the suspected traffic. Another benefit for this approach is to decrease false positives and increase true positives by redirecting suspected traffic to other servers on the edge of the network. In this work, the proposed algorithm is tested by generating normal traffic for one minute, then the traffic is increased and redirected to a backup server without blocking. Then, when the traffic goes upper the entropy threshold the proposed algorithm returns the suspected traffic to the main server and keeps the service continuity without interruption. The benchmark mitigation algorithm that employs entropy detection, the traffic will be blocked immediately and increase false positives results.

V. CONCLUSION AND FUTURE WORK

DDoS attack is considered as one of the most cybersecurity threat in traditional networks because it tries to stop some important websites for financial or economic reasons. Therefore, this subject has become very interesting for many researchers especially after the evolution in the network field and the increased use of SDN networks. These networks are considered more flexible and programmable than a traditional network in order to protect servers and switches from attacks. Previous researchers use entropy detection to find DDoS attacks and use the blocking mitigation method, which is considered not favorite because it increases false-positive results. This work utilizes the entropy detection method with OpenFlow protocol, and Open Vswitches, by redirecting the suspect traffic to the edge of the datacenter at an early stage to minimize damages and create load balancing. The throughput results prove that this mitigation method provides about %16 more throughputs than using direct blocking with entropy detection. Furthermore, this approach increases true positives and decreases false positives in detection attacks.

For future works, this approach can be used in big datacenters to protect web servers and networks from attacks, another future research is to implement a backup controller to

Received 11/6/2021; Accepted 20l8l2021

DOI: https://doi.org/10.33103/uot.ijccce.21.4.6

avoid single point failure in the network. Moreover, this approach can be combined with machine learning algorithms for future research.

REFERENCES

[1] 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.

[2] P. Zhai, Y. Song, X. Zhu, L. Cao, J. Zhang, and C. Yang, “Distributed denial of service defense in software defined network using OpenFlow,” *IEEE/CIC International Conference on Communications, China (ICCC)*, 2020.

[3] A. Akhunzada, E. Ahmed, A. Gani, M. K. Khan, M. Imran, and S. Guizani, “Securing software defined networks: taxonomy, requirements, and open issues,” *IEEE Commun. Mag.*, vol. 53, no. 4, pp. 36–44, 2015.

[4] B. B. Gupta, G. M. Perez, D. P. Agrawal, and D. Gupta, Eds., *Handbook of computer networks and cybersecurity: Principles and paradigms*. Cham: Springer International Publishing, 2020.

[5] Q. Zhang, L. Cheng, and R. Boutaba, “Cloud computing: state-of-the-art and research challenges,” *J. Intern. Serv. Appl*, vol. 1, no. 1, pp. 7–18, 2010.

[6] P. Goransson, C. Black, and T. Culver, *Software Defined Networks: A Comprehensive Approach"*, 2nd ed. Massachusetts, USA: Elsevier Inc, 2017.

[7] F. W. Hussein, M. Z. Abdullah, and N. A. A.-A. By, “A Hypothetical Design of Software Defined Network System Against Attacks of Distributed Denial of Service.”, pp. 28-155, 2019.

[8] N. Z. Bawany, J. A. Shamsi, and K. Salah, “DDoS attack detection and mitigation using SDN: Methods, practices, and solutions,” *Arab. J. Sci. Eng.*, vol. 42, no. 2, pp. 425–441, 2017.

[9] H. Wang, L. Si, X. Li, W. Deng, H. Yang, Y. Yang, Y. Fu, “DDoS Attack in Software Defined Networks: A Survey, Neural Regen,” Res. 7 (2017) 1095–1100, https://doi.org/10.3969/j.issn.1673.

[10] P. Alaa AL-Hamami and S. Hassan Hashem, “A proposed firewall security method against different types of attacks,” IRAQI JOURNAL OF COMPUTERS, COMMUNICATIONS, CONTROL AND SYSTEMS ENGINEERING, vol. 5, no. 1, pp. 65–74, 2005.

[11] K. Kalkan, G. Gur, and F. Alagoz, “Defense Mechanisms Against DDoS Attacks in SDN Environment",” *IEEE Communications Magazine*, vol. 55, no. 9, pp. 175–179, 2017.

[12] N. Z. Bawany, J. A. Shamsi, and K. Salah, “DDoS Attack Detection and Mitigation Using SDN: Methods, Practices, and Solutions",” *Arabian Journal for Science and Engineering*, vol. 42, no. 2, pp. 425– 441, 2017.

[13] N. McKeown, “OpenFlow: Enabling innovation in campus networks,” *Comput. Commun Rev*, vol. 38, no. 2, pp. 69–74, 2008.

[14] V.1.4.,“OpenFlow Switch Specification,” *Opennetworking.org*. [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.4.0.pdf. [Accessed: 24-May-2021].

[15] 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 UBICNET, India, 2019.

[16] S. M. S. Mousavi and M. St-Hilaire, “Early Detection of DDoS Attacks in Software Defined Networks Controller, Thesis.” pp. 77–81, 2014.

[17] E. Mota and A. P. R. Braga, “Lightweight DDoS flooding attack detection using NOX/Openflow,” 2010, pp. 408–415.

[18] S. Ostermann and B. T. M. Ramadas, “Detecting anomalous network traffic with self-organizing maps,” 2003, pp. 36–54.

[19] A. Rai, P. D Vyavahare, and A. Jain, “Distributed DoS attack detection and mitigation in software defined network (SDN),” *SSRN Electron. J.*, 2019, pp.2-5.

[20] M. Abdullah, N. Al-awad, and F. Hussein, “Implementation of entropy-based distributed denial of service attack detection method in multiple POX controllers,” *Rev. Comput. Eng. Stud.*, vol. 6, no. 2, pp. 29–38, 2019.

[21] R. Neres Carvalho, J. Luiz Bordim, and E. Adilio Pelinson Alchieri, “Entropy-based DoS attack identification in SDN,” *IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, Brazil, 2019.

[22] “Detection of DDoS in SDN environment using entropy-based detection,” *IEEE International Symposium on Technologies for Homeland Security (HST)*, California, 2019.

[23] P. Bereziński, B. Jasiul, and M. Szpyrka, “An entropy-based network anomaly detection method,” *Entropy (Basel)*,

Received 11/6/2021; Accepted 20l8l2021

DOI: https://doi.org/10.33103/uot.ijccce.21.4.6

vol. 17, no. 4, pp. 2367–2408, 2015.

[24] Mininet Project Contributors, "Mininet: An Instant Virtual Network on Your Laptop (or Other PC) - Mininet," *Mininet.org*. [Online]. Available: http://mininet.org/. [Accessed: 24-May-2021].

[25] "Open vSwitch," *Openvswitch.org*. [Online]. Available: https://www.openvswitch.org/. [Accessed: 24-May-2021].

[26] V. Gueant, "iPerf - iPerf3 and iPerf2 user documentation," *Iperf.fr*. [Online]. Available: https://iperf.fr/iperf-doc.php. [Accessed: 24-May-2021].

[27] P. Biondi and the Scapy community, "Scapy," Scapy.net. [Online]. Available: https://scapy.net/. [Accessed: 24-May-2021].

[28] M. K. Faraj, Dr. A. Al-Saadi, Dr. R. J. Albahadili, "An investigation of using traffic load in SDN based load balancing," *Iraqi Journal of Computer, Communication, Control and System Engineering*, pp. 65–74, 2020.

Received 11/6/2021; Accepted 20l8l2021