Dynamic Threshold for DDoS Mitigation in SDN Environment

Guo-Chih Hong†, Chung-Nan Lee* and Ming-Feng Lee+
Department of Computer Science and Engineering
National Sun Yat-sen University, Kaohsiung, Taiwan
†E-mail: shps950418@gmail.com Tel: + 886-7-5254335
*E-mail: cnlee@mail.cse.nsysu.edu.tw Tel: + 886-7-5252000 ext. 4313
+ E-mail: mflee@mail.nsysu.edu.tw Tel: + 886-7-5252000 ext. 4335

***Abstract*—**Software-Defined Networking (SDN) is one of the key technologies of 5th generation mobile networks (5G). However, like the traditional network architecture, SDN is also vulnerable to the Distributed Denial of Service (DDoS) attack. This paper explores the dynamic threshold for DDoS attack in the SDN environment. Through the characteristics of SDN, we propose a feasible DDoS detection and defense mechanism. The proposed mechanism calculates the entropy of the network environment by the collected traffic status, and derives a dynamic threshold according to the network conditions to determine whether the environment is subject to DDoS attacks. In the event of a DDoS attack, the proposed mechanism discards the traffic from the malicious nodes to the victim nodes with a flow entry. In addition, if no DDoS attacks occur in the environment, the proposed system can disperse the traffic of the SDN switch, thereby balance the traffic load in the environment.

A. INTRODUCTION

The fifth generation (5G) mobile communication network will be the foundation for communications and computing in next few years. According to the requirements defined by the Next Generation Mobile Networks Alliance (NGMN) for the 5G mobile communication network, it will have significant improvements in all aspects compared to the fourth generation (4G) mobile communication network. The application of 5G is bound to become a major boost to the development of technology in the future. Software-Defined Networking (SDN) is one of the key technologies for 5G networks. SDN uses the remote controller to control the SDN switches to forward packets in a specific or optimal route and reduces the redundant packet forwarding process to improve transmission efficiency.

At present, there are some security issues in SDN. After entering the 5G era, if these security issues have not been resolved, these problems will continue in the 5G environment and endanger its usage. The impact of these security issues can range from small IOT devices to cloud servers or data centers, and even the entire network can cause irreparable damage.

SDN is a relatively new communication architecture compared to traditional networks and uses newer communication protocols and relies entirely on the central controller. Once the controller is compromised, the entire network where the controller is located may be subject to considerable risk. Although some organizations or researches have proposed the expected policy of security vision and security issues for the SDN network architecture, there is no more specific and complete treatment for Distributed Denial of Service (DDoS) attacks.

DDoS attacks often use many zombie computers (or called botnets), which are computers that are controlled by malicious attackers after being infected by malicious software. Zombie computers launch attacks in accordance with the control of the attackers, and usually the owners of these computers are not aware that their computers have been implanted with malicious programs. The types of DDoS attacks can be mainly divided into two categories, namely bandwidth consumption and resource consumption attacks. The purpose of bandwidth consumption attack is to consume the network bandwidth of the attacked target, so that the network bandwidth of the target is filled, resulting in the inability to provide services. On the other hand, the purpose of resource consumption attack is to exhaust the hardware or system resource of the attacked target and make it difficult for the system to handle other tasks, resulting in its inability to provide services stably. Common DDoS attacks include UDP flood, TCP SYN flood, ICMP flood, Http Flood and so on.

This study focuses on DDoS detection and defense in SDN environment. In this work, we use OpenDaylight, an open source software, as the central controller for the SDN environment, and conduct research on DDoS treatment through OpenDaylight. The characteristics of this study are as follows.

1. In the related research similar to this work, the thresholds of traffic and entropy used to determine whether the network environment is subject to DDoS attacks are mostly fixed values. The disadvantage of using fixed thresholds is that it cannot cope with a changing network environment. Network traffic has the characteristics of high usage period and low usage period with time. If a single and fixed threshold is used to judge whether or not to suffer DDoS, there will be a misjudgment. Therefore, based on statistics and observing the status of the network environment, we propose a mechanism for defining dynamic thresholds to adapt to the changing network environment. In addition, compared to some existing

methods that use dynamic thresholds [6, 7, 8], the proposed mechanism has the ability to prevent misjudgment if the entropy of the system slowly approaches steady state over time.

2. For peak network traffic or heavy traffic that is not subject to DDoS attacks, the proposed system balances the load on the network devices in the SDN environment, so that hardware resources can be used more efficiently and the transmission delay in the environment is reduced.
3. In the related research similar to this work, most DDoS defense schemes only deal with some specific SDN switches, whereas the proposed mechanism can monitor and manage all switches in the whole SDN environment. In addition to being able to find the source of the corrupted switch, the proposed mechanism can optimize the entire SDN environment.
4. The load balancing proposed here considers both the number of hops passing through the packet transmission path and the traffic utilization of the transmission route. Compared to considering only one of the above elements, the proposed scheme can choose a more suitable route and avoid inappropriate path switching.

II. RELATED WORK

Arbettu et al. [1] analyzed several SDN controllers that are currently used more frequently, and each controller provides different security mechanisms. Because of the different security mechanisms, the security problems faced by various controllers are also different. However, the common security problem of these SDN controllers is that the countermeasures against DDoS attacks still need to be strengthened. In addition, from some researches [2] [3], it can be known that DDoS is still a difficult security problem in the SDN environment. At present, some studies have focused on DDoS detection and defense in SDN environment.

The main detection method in [4] is to use the network traffic monitoring tool iftop as a tool for traffic detection. Their method directly collects traffic in the environment with iftop and compares the end-to-end transmission throughput. If the value of *req/resp* exceeds a specified threshold, the transmission request is determined to be a malicious attack, and the defense mechanism for the attack is to directly drop the OpenFlow flow entry of the traffic to achieve the effect like using a firewall. Lawal and Nuray [5] used the traffic analysis tool sFlow to collect traffic on the entire network in the SDN environment. When the traffic of the transmission is greater than the pre-defined threshold, their defense mechanism determines that the traffic is an attack and then issues an OpenFlow flow rule to block the attack.

Note that the traffic thresholds of the defense methods mentioned above [4, 5] are pre-defined static thresholds. Although these methods using static thresholds can work in situations that meet certain network conditions, if they are to be applied to different application scenarios, then their defense system needs to define a new threshold again to adapt to the new scenario. In contrast, the method of dynamically defining the threshold can be changed according to the status of the environment. Recently, dynamic thresholds have been used in some studies. Then we discuss the studies using dynamic thresholds for DDoS detection.

The DDoS detection method proposed by Pande et al. [6] is to check the packet characteristics (packet length, type, etc.) when the packet is forwarded to the controller to determine how to arrange the transmission route. If it is a specific type of packet that is pre-defined as suspicious, it will be discarded directly. For example, the packet with TTL (Time To Live) error is discarded directly. If the packet is an ARP (Address Resolution Protocol) packet, it is judged whether the MAC address is already in the record table, and if not, the packet is discarded. After passing the above filtering mechanism, the traffic of the entire environment is finally judged. If the traffic is higher than the pre-defined threshold, different defenses are taken depending on whether it is on the same network segment as the victim. The method proposed by Gkountis et al. [7] is to use the SDN switch to send the packet to the controller in the form of OpneFlow *packet_in*, check the traffic and size of the packet sent to the controller, and compare it with a threshold to determine if there is a possibility of suffering a DDoS attack. The method then issues and adjusts the *hard_timeout* and *idle_timeout* parameters of flow entry according to the situation to reduce the packet flow sent to the controller for query. The method proposed by You et al. [8] is also to observe the characteristics of the environment traffic by transmitting the OpneFlow *packet_in* packet to the controller through the switches. The method collects the source IP, the destination IP, and the destination port of each *packet_in* packet to calculate entropy, and uses the rule of thumb of normal distribution as the benchmark for the threshold to detect whether or not the DDoS attack is suffered. However, if the entropy of the system slowly approaches the steady state over time, there may be a risk of misjudgment.

The load balancing is also considered here. Attarha et al. [9] proposed a load balance method to determine whether the load in the transmission route exceeds the predefined maximum tolerance value. The tolerance value of their method is defined as 0.7 times of the capacity. When the load exceeds 0.7 times of the capacity, it means that the path is blocked and it is necessary to find a more suitable path. The path search method is to find all the paths that can be from the source to the target, then find the path that can also load the traffic flow, and finally direct flow to the found path. Zakia and Yedder [10] first find the shortest paths from the source to the target by Dijkstra's shortest path search method [12], and calculate the utilization of each shortest path. Finally, their method selects the path with the lowest utilization of these shortest paths as the best path and direct traffic flow to it.

Nkosi et al. [11] also use Dijkstra's shortest path search method to find the shortest paths from the source to the target and compare the load in these shortest paths. The method uses the path with the lowest load as the best path. Finally, traffic is directed to this best path.

III. THE PROPOSED SYSTEM

A. Architecture of Proposed Mechanism

The traffic analysis tool in this study is sFlow. We use the functions provided by OpenDaylight and sFlow to collect the traffic status of the network environment that is used the proposed mechanism to process DDoS detection and defense. As shown in Figure 1, the architecture of the proposed system contains three modules.

The DDoS detection and defense mechanism communicates with sFlow and OpenDaylight through the REST (Representational State Transfer) API (Application Programming Interface). The system obtains the topology and traffic information, and then manages the SDN network environment through OpenDaylight. The architecture can be divided into three modules, namely Collector, Manager and Executor. The Collector continuously collects the packets received and delivered by each SDN switch, and transmits the collected results to the Manager periodically. In addition, in order to make the maintenance of the administrator conveniently, Collector keeps the records of traffic collected within a time window, which is used for Manager to observe and make security strategy. The Manager is an important decision-making module. The main task is to detect whether the entire environment has been attacked by DDoS according to the collected data provided by the Collector. The Executor is a module that controls the OpenDaylgiht controller according to the instructions of the Manager, so that it can manage the OpenFlow switches to protect the entire SDN environment.

The main operation process of the Manager is to use entropy with the package transfer data sent by Collector to determine whether a DDoS attack is likely to occur. If it is judged that the DDoS attack is happening, the Manager requests the Collector to collect the traffic status of the switches that may be under attack and transmits the corresponding policy to the Executor. Then the Executor controls and setups the switches according to the policy. If high traffic occurs and no attacks are found, the Manager balances the load of all IP flows within the network environment controlled by the Controller. The system finds all the routes from each source IP to the target IP by calculating the scores of all the routes by (9) which will be detailed in Section III-D, and searches the lowest score as the most suitable route. The system then compares whether the new route is more suitable than the original route, and if so, transfers the traffic to the new route, otherwise the original route is maintained.

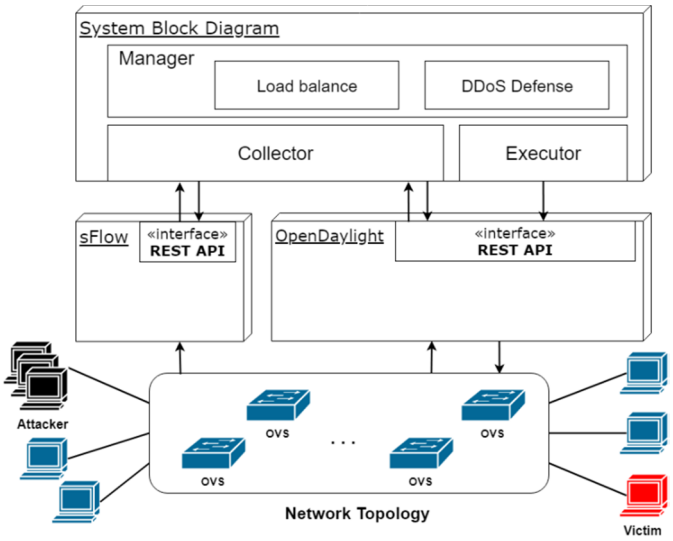

Figure 1. Architecture of the proposed system

B. Process of the Proposed Mechanism

Before entering the details of the proposed mechanism, we give the definition of some notations in Table 1.

Table 1: Notations used in the proposed mechanism

Notation	Description
H	Entropy
n	Total number of traffic sources
i	The *i*-th traffic source
P_i	The probability of traffic flowing from the *i*-th source to the detected target
f_i	Traffic from the *i*-th source received by the detected target (in bytes)
f_t	Total traffic received by the detected target (in bytes)
μ	The average traffic in a traffic time window
σ	The standard deviation of traffic in a traffic time window
$T_{traffic}$	The threshold of traffic
$TL_{traffic}$	The lowest traffic threshold
T_{Entropy}	Threshold for judging whether entropy is abnormal
H_{avg}	The average value of entropy in a traffic time window
σ_{H}	The standard deviation of entropy in a traffic time window

The proposed mechanism first collects the transmission traffic of the SDN switches as a preliminary detection, and then observes the traffic changes of the entire network environment by using the traffic analysis tool sFlow, and captures the total traffic of each switch from OpenDaylight to calculate the traffic flow. Through sFlow, one can observe many contents, such as protocols, traffic sources, traffic destinations, etc. The proposed mechanism calculates a dynamic threshold from a traffic time window and then compares with the current collected transmission traffic. If the transmission traffic is lower than the threshold, it means the network environment is in normal state, there is no heavy load or DDoS attack occurs. If the traffic is higher than the threshold, it may represent an abnormality in the environment. Excessive load may mean that the network environment is being attacked by DDoS.

Entropy can be used as a basis for judging whether the overall system environment is subject to DDoS attacks. The basis for calculating entropy can be source port, destination port, source IP, destination IP and so on. In the proposed mechanism, entropy can be calculated by the traffic status collected by sFlow and OpenDaylight to determine whether the DDoS attack is affected in the whole environment. Equation (1) is the common formula for calculating entropy.

$$H = -\sum_{i=1}^{n} P_i log_2 P_i \quad (1)$$

$$P_i = \frac{f_i}{f_t} \quad (2)$$

In (1), the meaning of the notation H is the degree of uncertainty, that is, the randomness of the traffic distribution in the whole system. Notation n is the number of connected targets. Notation P_i represents the probability that traffic flows from source i to the detected target, as defined by (2), where f_i represents the traffic (in bytes) received by the detected target from source i, and f_t represents the total traffic (in bytes) received by the detected target.

In (1), the closer H is to zero, the most traffic in the system comes from some specific sources. In other words, some specific targets in the system may be suffering from DDoS attacks; otherwise, if H is larger, it represents the traffic distribution in the system is relatively average, which means that the probability of suffering DDoS attacks is low.

This study uses OpenDaylight controller with sFlow to continuously collect the traffic and its direction of each SDN switch, and periodically calculates the entropy of the entire network environment to determine whether DDoS is happening in the system controlled by OpenDaylight controller. If it is determined that there is a DDoS attack, the proposed mechanism traces the source and blocks the attack traffic.

C. Thresholds of Traffic and Entropy

The threshold of the traffic in this study is not a fixed value, but is automatically adjusted according to the environment, which is a dynamic threshold. In order to calculate the traffic threshold, the data of traffic flow is stored for a traffic time window and continuously updated, and then the data is used to calculate the average value of traffic μ and the traffic standard deviation σ. Parameters μ and σ are calculated according to (3) and (4), where n is the number of traffic sources and f_i is the traffic flow from the i-th source.

$$\mu = \sum_{i=1}^{n} \frac{f_i}{n} \quad (3)$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} f_i^2 - \mu^2}{n}} \quad (4)$$

By the rule of thumb of normal distribution, "typically, 95% of the value is within the mean plus or minus twice standard deviations" and one can neglect the low flow rate (flow value below μ-2σ) because flow rate below μ-2σ has less impact on the network environment, it is not threatening. Hence, if the traffic flow is normal, 97.7% of the traffic flow will be maintained within two standard deviations, so the threshold of traffic $T_{traffic}$ can be derived from (5).

$$T_{traffic} = \mu + 2 * \sigma \quad (5)$$

If the traffic exceeds $T_{traffic}$, it means that abnormal traffic changes may occur, and further calculation of entropy is used to judge. If it is judged that the network environment is not under DDoS attacks, load balancing will be performed, and at the same time, the system will continue to store the traffic data to calculate the next traffic threshold. In other words, after confirming that the high traffic is not a malicious flow, the traffic threshold $T_{traffic}$ will be revised due to the change of the traffic, resulting in a new traffic threshold.

However, if the traffic threshold is completely dependent on the condition of traffic flow, the first reception of higher normal traffic will result in a misjudgment when the initial network traffic is extremely low or no traffic. If no additional corrections are made, it will cause false alarm frequently and reduce the system performance. Therefore, we define a minimum traffic threshold $TL_{traffic}$ as shown in (6).

$$T_{traffic} = \max(TL_{traffic}, \mu + 2 * \sigma) \quad (6)$$

In this study, we define this minimum traffic threshold $TL_{traffic}$ is 100K bps. The reason for this is that the traffic below this minimum threshold does not have a significant

impact on the performance of the entire network environment. In addition, since the mechanism needs to collect the traffic data for a traffic time window, after the system is initialized, there will be a time window of initial collection of environmental traffic, and no traffic judgment will be performed in the initialization phase.

In addition, in this study, the entropy threshold T_{Entropy} used for determining entropy is also dynamically defined. Similarly, the rule of thumb for the normal distribution is used, and T_{Entropy} is obtained as shown in (7).

$$T_{\text{Entropy}} = H_{\text{avg}} + 2 * \sigma_{\text{H}} \quad (7)$$

Since T_{Entropy} changes with the status of the environmental traffic, when the traffic in the entire network topology gradually enters a steady state, T_{Entropy} may approach the average etropy H_{avg} because of the standard deviation of entropy σ_{H} gradually approaches zero. If the etropy changes slightly in a steady state, it may lead to misjudgment. Therefore, in the initialization phase, in addition to collecting the traffic status of the environment to obtain the trafic and entropy thresholds, it is also necessary to obtain a lowest entropy difference to help determine the entropy change in steady state. Empirically, we set this lowest entropy difference to 0.005.

D. Load Balancing

When the proposed mechanism judges that no attack occurs, load balancing is performed to effectively balance the load of network devices in the environment, thereby achieving better hardware resource utilization efficiency. The proposed load balance scheme balances the load of all the switches controlled by the SDN controller. The scheme first detects all IP flows currently in the network topology. For all IP flows, we first find out all the routes from the source IP to the destination IP through a Python package NetworkX, and then estimate the score $Score_{route}$ .of each route. The lower the score, the better this route is. After finding the route with the lowest score, we then calculate whether the score of the new route is lower than the score of the original route if the IP flow is moved from the original route to the new one, and is lower than the score threshold T_{Score} . If so, the traffic is directed to the new route, otherwise the IP flow is maintained in the original route. In this way, it is possible to find the most suitable route and avoid the situation that the load is more unbalanced due to the flow transfer for load balancing caused by the slight traffic difference.

The calculation of the route score $Score_{route}$ is given in (9), and the calculation of route utilization $Util_R$ is given in (8), where capacity C is the maximum information rate that a channel is able to transmit/receive, hop_R represents the number of hops passed by this route, and hop_L represents the number of hops of the longest route among all the routes. Furthermore, as shown in (8) when calculating the utilization of route $Util_R$, the lagest traffic value $Trfc_L$ observed in the switches passed by the route is selected for calculation. The utilization calculated by this value can truly reflect the load status of the route. The design of this calculation method mainly considers the number of hops (or switches) that the route passes through and the utilization of the route. The reason for considering the number of hops is that the more hops are passed, the more processing time is required for the packets. The value α in (10) is a variable parameter, which can be changed according to the use scenario and conditions. The range of α is between 0 and 1, and its role is to adjust the ratio of the main reference conditions. If the choice of available routes is less and the difference of route hops is large, it is recommended to use a smaller α value, otherwise use a larger α value.

$$Util_R = Trfc_L \,/\, C \quad (8)$$

$$Score_{route} = \alpha \times \frac{hop_R}{hop_L} + (1-\alpha) \times Util_R \quad (9)$$

$$0 \le \alpha \le 1 \quad (10)$$

It is necessary to compare whether the score obtained by the new route is higher than the score threshold T_{Score} to evaluate whether the traffic flow needs to be directed to the new route. The score threshold T_{Score} is given in (11), where hop_N represents the number of hops passed by the newly discovered best route. The calculation of $Util_{new}$ is formulated in (13), where $Trfc_{sp}$ is the traffic flow expected to be transferred, capacity C is the maximum information rate that a channel is able to transmit/receive. $1/\beta$ of the score increase caused by $Trfc_{sp}$ is used as the score threshold, where β is an adjustable parameter and the value is greater than 0. We set β to 10, which means that the new score must be 10% lower than the original score to judge the path is suitable. If the score of the original route minus the score of new route is less than this score threshold, it means that the new route will not be better than the original one, or the effect is similar, so no change is required.

$$T_{Score} = (\alpha \times \frac{hop_N}{hop_L} + (1-\alpha) \times Util_{new})/\beta \quad (11)$$

$$\beta > 0 \quad (12)$$

$$Util_{new} = Trfc_{sp} \,/\, C \quad (13)$$

IV. SIMULATION

A. Simulation Environment

The OpenDaylgiht controller and Mininet v3.3 are used as the tools for simulating the SDN environment. Since the Open vSwitch (OVS) that can operate the OpenFlow protocol has been supported on the Mininet 3.2 version, Mininet is used as a tool for simulating software-defined networking environment. We adopt multiple OVS and virtual machines (VMs) for network topology. Several virtual machines are connected under the OVS to perform packet transmission and reception, and the entire topology is controlled and monitored by the OpenDaylight controller.

Figure 2 shows the topology we establish using Mininet, where h_i represents host i ($i \in \{1,\ldots,8\}$), s_j represents OpenFlow switch j ($j \in \{1,\ldots,10\}$), and the IP addresses of h_1~h_8 are 10.0.0.1~10.0.0.8. It can be seen from Figure 2 that the topology used in this study is roughly a tree topology.

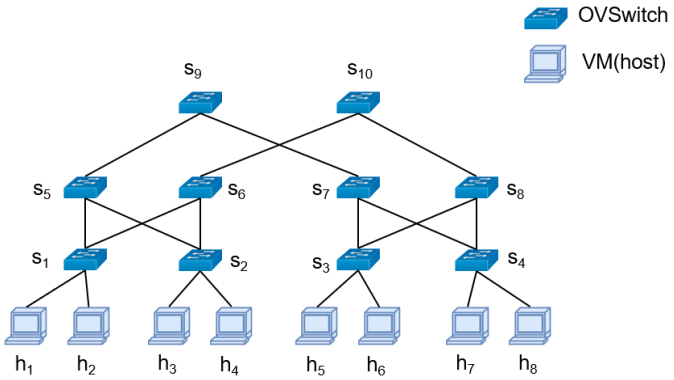

Figure 2. Network topology of the simulation environment

B. *Simulation Results*

The simulation is divided into normal state and the state of simulated malicious attack. In addition, the system detects traffic flow and performs the appropriate mechanism every three seconds. Under normal state, we tested whether the system would be misjudged, and whether load balancing could be performed when there was high traffic but no attack occurs. In a malicious attack state, the underlying normal traffic was first set, and malicious traffic was injected into the environment after a period of time to simulate the DDoS attack. In this way, we tested whether the system could correctly determine the DDoS attack and successfully retain the original normal traffic to verify the feasibility of the proposed mechanism.

First of all, we used the packet processing tool Scapy as the traffic generation tool, and transmitted in packets of 100 KB per second in the manner of host h_1 to h_8, h_2 to h_3, h_5 to h_4 and h_6 to h_7, respectively, and captured the traffic load of each switch in the system.

Figure 3 shows the simulation results under normal network traffic. In Figure 3, the Y axis represents the flow load, and its unit is KB/s; X-axis shows in which round the system captures traffic flow. The first 10 rounds belong to the initialization phase and the work in this phase was to collect the environment status. Therefore, no detection and defense mechanism was started. After the 10th round, the proposed mechanism started to operate, and the load balancing was started around the 12th round and finally been completed at the 20th round. Part of the traffic was transferred from s_9 to s_{10}, so the load of s_9 was reduced, and the load of s_{10} was relatively increased. After that, the system also had a situation that triggered the load balancing. However, under the load balance evaluation of the system, the load of each switch was balanced, so no transmission path was changed.

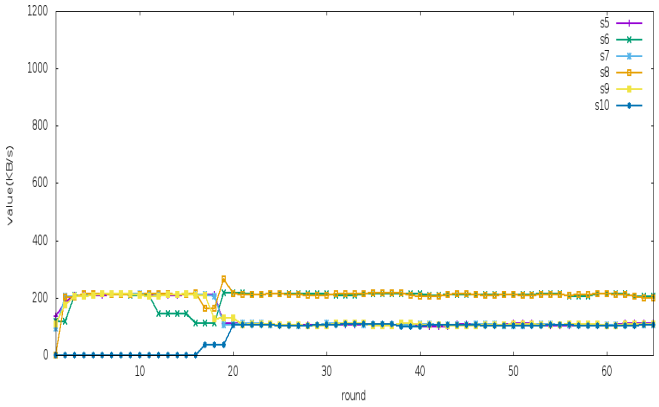

Figure 3. Simulation result under normal network traffic.

Then we used Scapy as a tool to simulate DDoS attacks. In the network topology of Figure 2, we simulated maliciously attack from two VMs (h_7 and h_8) to a victim VM (h_3). Figure 4 shows the test results of the simulated DDoS attack. It shows only the traffic load scenarios of switches s_5, s_6, s_7, s_8, s_9 and s_{10}. Because s_1, s_2, s_3 and s_4 are switches directly connected to VMs, even if the flow entry lets s_1, s_2, s_3 and s_4 discard the packets, they continue to receive the traffic from hosts, so there is no change in traffic load of these switches. We do not specifically show the load status of s_1, s_2, s_3 and s_4.

Before the 40th round, the traffic of each switch is the normal traffic set by Scapy. The normal traffic is packet transmissions from h_1 to h_8, h_2 to h_3, h_5 to h_4, and h_6 to h_7 respectively for 100KB/s. Then after 40 round, we started to inject 400KB/s attack traffic from h_7 to h_3 and h_8 to h_3, respectively. From the results in Figure 4, it can be seen that before the 10th round, the traffic load was not even, because the system is in the initialization phase of collecting the current environmental status, and starts load balancing around the 12th round. The environment load is balanced after the 20th round and remains stable until the 40th round. At around the 43 round, there is spikes in some the switches due to the injection of attack traffic. The proposed mechanism works at about the 47 rounds, and then successfully blocks the malicious traffic at 50 rounds, and the regular traffic in this environment was continuously transmitted. The delay between 43 and 50 rounds is due to the sampling delay of sFlow. This shows that the proposed mechanism can

effectively detect and defend against malicious traffic from DDoS attack and maintain normal traffic.

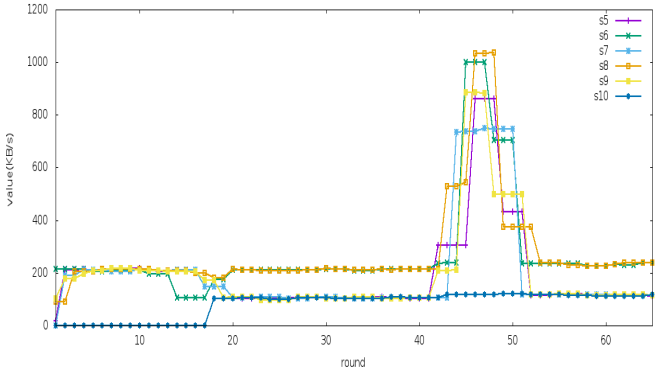

Figure 4. Simulation result under DDoS attack

V. CONCLUSIONS

In this paper, we proposed a mechanism that can cope with DDoS attack in SDN environment. The detection mechanism of DDoS is to use adaptive thresholds of entropy and traffic to continuously analyze and evaluate the communication of the whole SDN environment. Further, our mechanism makes use of the flow entry of OpenFlow to deal with DDoS attack. From the simulation results, it can be seen that our mechanism can properly cope with DDoS, so it can preserve the normal traffic in the environment and block malicious traffic. It can also process load balance to reduce the transmission delay of the environment when the environment is under heavy traffic load.

ACKNOWLEDGEMENT

This research was supported in part by the Ministry of Science and Technology of Taiwan under contract No. MOST 107-2221-E-036-MY3.

REFERENCE

[1] R. K. Arbettu, R. Khondoker, K. Bayarou and F. Weber, “Security analysis of OpenDaylight, ONOS, Rosemary and Ryu SDN controllers,” *2016 17th International Telecommunications Network Strategy and Planning Symposium (Networks)*, pp. 37–44, 2016.

[2] I. Ahmad, T. Kumar, M.Liyanage, J. Okwuibe and M. Ylianttila, “5G security: analysis of threats and solutions,” *2017 IEEE Conference on Standards for Communications and Networking (CSCN)*, pp. 193–199, 2017.

[3] K. K. Karmakar, V. Varadharajan and U. Tupakula, “Mitigating attacks in Software Defined Network (SDN),”*2017 Fourth International Conference on Software Defined Systems (SDS)*, pp. 112–117, 2017.

[4] R. M. Thomas and D. James, “DDOS detection and denial using third party Application in SDN,” *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, pp. 3892–3897, 2017.

[5] B. H. Lawal and A. T. Nuray, “Real-time detection and mitigation of Distributed Denial of Service (DDoS) attacks in Software Defined Networking (SDN),” *2018 26th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, 2018.

[6] B. Pande, G. Bhagat, S. Priya and H. Agrawal, “Detection and mitigation of DDoS in SDN,” *2018 11th International Conference on Contemporary Computing (IC3)*, pp. 1–3, 2018.

[7] C. Gkountis, M. Tahab, J. Lloret and G. Kambourakis, “Lightweight algorithm for protecting SDN controller against DDoS attacks,” *2017 10th IFIP Wireless and Mobile Networking Conference (WMNC)*, pp. 1–6, 2017.

[8] X. You, Y. Feng and K. Sakurai, “Packet_In message based DDoS attack detection in SDN network using OpenFlow,” *2017 Fifth International Symposium on Computing and Networking (CANDAR)*, pp. 522–528, 2017.

[9] S. Attarha, K. H. Hosseiny, G. Mirjalily and K. Mizanian, “A load balanced congestion aware routing mechanism for Software Defined Networks,” *2017 Iranian Conference on Electrical Engineering (ICEE)*, pp. 2206–2210, 2017.

[10] U. Zakia and H. B. Yedder, “Dynamic load balancing in SDN-based data center networks,” *2017 8th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pp. 242–247, 2017.

[11] M. C. Nkosi, A. A. Lysko and S. Dlamini, “Multi-path load balancing for SDN data plane,” *2018 International Conference on Intelligent and Innovative Computing Applications (ICONIC)*, pp. 229–234, 2018.

[12] D. E. Knuth, “A Generalization of Dijkstra's Algorithm,” *Information Processing Letters*, vol. 6, no. 1. pp. 1–5, 1977.