# Packet in Message Based DDoS Attack Detection in SDN Network Using OpenFlow

**Conference Paper** · November 2017

DOI: 10.1109/CANDAR.2017.93

Yaokai Feng
Kyushu University
**83** PUBLICATIONS **1,319** CITATIONS

Kouichi Sakurai
Kyushu University
**556** PUBLICATIONS **4,086** CITATIONS

Fifth International Symposium on Computing and Networking, CANDAR 2017, . IEEE Computer Society 2017, ISBN 978-1-5386-2087-8, Pages 522-528.

# Packet_In message based DDoS attack detection in SDN network using OpenFlow

Xiang You
*Department of Informatics*
*Kyushu University*
Fukuoka, Japan
Email: 2IE16031E@s.kyushu-u.ac.jp

Yaokai Feng
*Department of Advanced Information Technology*
*Kyushu University*
Fukuoka, Japan
Email: fengyk@ait.kyushu-u.ac.jp

Kouichi Sakurai
*Department of Informatics*
*Kyushu University*
Fukuoka, Japan
Email: sakurai@csce.kyushu-u.ac.jp

***Abstract*—Using the OpenFlow protocol, the virtual network technology SDN (Software Defined Network) is now widely used. In recent years, the number of DDoS attacks has been increasing year by year. To detect DDoS attacks in SDN, data recorded in the flow table in OpenFlow switch is analyzed and various detection methods are submitted. However, SDN centrally manages communication within the network, when detecting DDoS (Distributed Denial of Service) attacks. This creates a heavy processing load, and the processing load of the OpenFlow controller must be considered. In this paper, in order to reduce the processing load of the controller, we do not collect data of the flow table, extract three features from the Packet In message for communication between the controller and the switch, and perform real-time attack detection. Furthermore, to avoid stringent detection time intervals, triggers will be added before detection to realize light and dynamic DDoS attacks detection.**

***Index Terms*—SDN, DDoS Attack, OpenFlow protocol, OpenFlow controller, OpenFlow Switch**

## I. Introduction

Software-Defined Networking (SDN) is an emerging architecture that is dynamic, manageable, cost-effective, and adaptable, making it ideal for the high-bandwidth, dynamic nature of today's applications.

It offers flexible network management by decoupling forwarding and control planes, in the SDN architecture, network management is logically centralized at the control plane, while the forwarding plane only needs to forward packets under the manipulation of the control plane. enabling the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services. Due to its flexibility, programmability and maintainability, SDN has been widely studied for its applications in backbone networks, data centers, enterprise net- works, access networks, wireless networks, and etc[1].

As an emerging network technology, SDN is faced with security issues that seem to be quite an obstacle to overcome. Seven main potential security issues have been presented presented In Kreutz's research[2].

- forged or faked traffic flows
- attacks on vulnerabilities in switches
- attacks on control plane communications
- attacks on the vulnerabilities in controllers
- lack of mechanisms to guarantee trust between the controller and management applications
- attacks on vulnerabilities in administrative station
- lack of trusted resources for forensics and remediation

However, using simple network programing, network monitoring and dynamic flow policies implementation provided by SDN, network forensics, security policy alteration and security service insertion can be achieved in SDN. Among the current security problems, one of the most urgent and hardest security issues is Distributed Denial of Service (DDoS). It is easy to start, hard to defend and trace[15].For example, Domain Name System(DNS) provider Dyn was attacked by DDoS on October 21, 2016. This caused major Internet platforms and services to be unavailable to large swathes of users in Europe and North America[3]. SDN network management is centralized at the controller. DDoS can easily cause serious damage to controllers. Therefore, effectively and realtime detection of DDoS attacks in SDN is crucial for future network SDN architecture deployments.

A number of DDoS attack detection methods in SDN have already been proposed. Most of the previous studies start DDoS detection periodically[4][5][6][7]. However, choosing the proper period of detection loop is hard.If the selected period is too large, the response time(from launching an attack to starting attack detection) will be long, which makes the controller and the switch handle an extremely large amount of attack packets and even destroys the controller and the switches. In contrast, if the period is too small, the attack detection will start more frequently, which makes the controller waste a lot of resources(i.e. CPU and the network bandwidth) and affect the efficiency of the controller.

To avoid periodical detection, a mechanism called Software Defined Ant-DDoS(SD-Anti-DDoS) was proposed[1]. Different from previous studies which start detection of DDoS periodically,SD-Anti-DDoS starts detection dynamically by using a trigger mechanism. So, it can reduce the response time against attacks and decrease the CPU and network load of the controller and switches. However, this research needs to collect all the flow entries in switches when DDoS detection happened, which will increase the load of controller, and may even cause overload[16].

In order to solve the problems above, we propose a novel mechanism for detecting DDoS attack in SDN called packet in message based detection. Different from previous studies about DDoS attack detection in SDN, our mechanism doesn't need to inspect all the packets or collect all the flow entries in switches. Instead, we propose a novel mechanism based on packet in packets.These special packets are forwarded between controller and switches by using OpenFlow protocol. According to the packet in packets, controller will set up flow entires into switches, which means we can collect information of flow entries through packet in message. Compared with previous research, our mechanism distributes processing load of detection and realized a lightweight DDoS attack detection in SDN.

## II. Background

### A. SDN

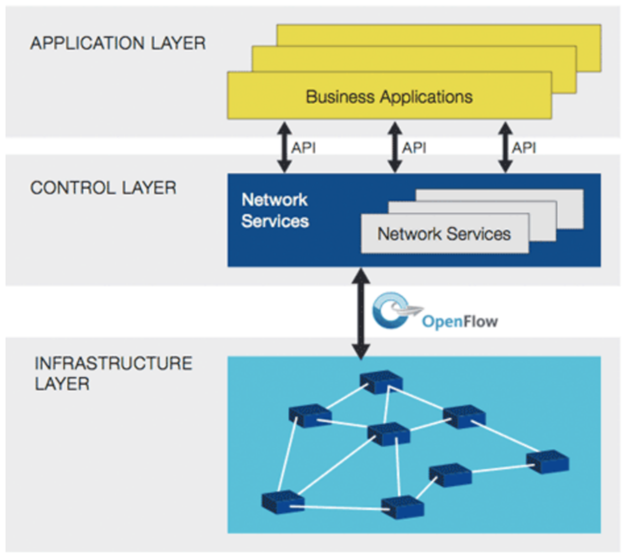

Fig. 1. The architecture of SDN[1].

As shown in Fig1, SDN is composed of three layers(Application Layer, Control Layer, Infrastructure Layer). Unlike traditional internet architecture, the control function is decoupled from forwarding and centralized in the software-based SDN controller[17].

### B. OpenFlow

OpenFlow is the first standard communications interface defined between the control and forwarding layers of an SDN architecture, which enables network controllers to determine the path of network packets across a network of switches[9].As shown in Fig2, SDN network is composed of controller and switches. OpenFlow is a communications protocol between controller and switches that gives access to the forwarding plane of a network switch or router over the network[18].

So far six versions of OpenFlow (from version 1.0 to 1.5) have been re- leased. Among them, version 1.0 and version 1.3 are widely used in OpenFlow-enabled switches. In this paper the OpenFlow v1.3 is chosen as the communication protocol for connecting the controller and switches.

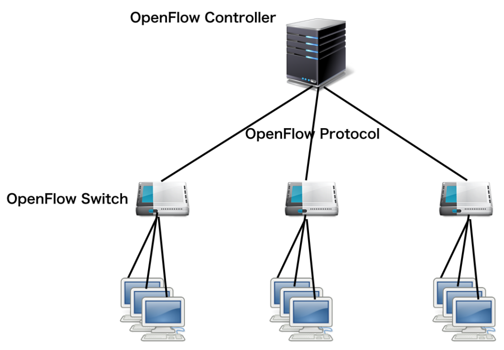

Fig. 2. OpenFlow controller and switch.

### C. OpenFlow switch

As shown in fig3, OpenFlow switch consists of a flow table which performs packet lookup and forwarding and a secure channel used to connect the switch to the controller. The flow table is composed of a set of flow entries. Each flow entry contains header fields, counters and actions. Header fields are used to match against the incoming traffic packets. Counters are applied to count packets matched by a certain flow entry and the actions define related actions that will be applied to the matching packets.

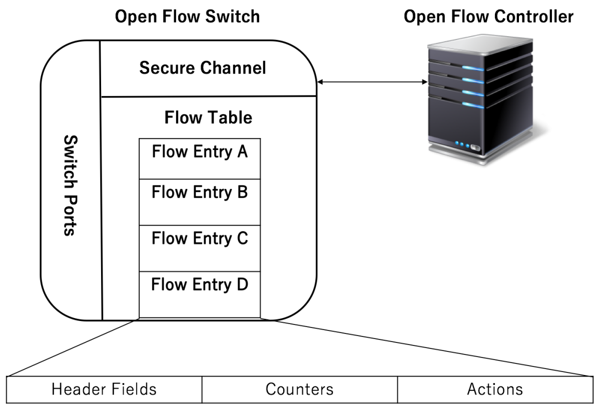

Fig. 3. The architecture of OpenFlow switch.

In this configuration, when a new packet arrives through some switch port, it is compared against all flow entries using header the fields. If this packet matches an existing entry in the switch's Flow Table, the switch updates its counters and executes the associated actions. Otherwise, the packet is sent to the controller through a secure channel.

### D. Flow Entry

Flow Entry is composed of three parts shown below:

- Header Fields: TABLE I shows the main items of a flow entry using OpenFlow1.3.Header fields can be composed of some or all of these items: ingress port, Ethernet source address, Ethernet destination address, Ethernet type, VLAN id, VLAN priority, IP source address and etc.Each item of the header fields may have a special value or ANY (a wildcard used to match all value). All items in the header fields will be used to compare with the relevant information of the coming packet.For example, we can set flow entries like ”the in port is 1, and the mac destination address ” or ”the in port is 2, and the IP destination address is 192.168.1.1”.

TABLE I
HEADER FIELDS

| Fields | Description |
|---|---|
| Ingress port | Switch input port |
| Ethernet src address | Ethernet source address |
| Ethernet dst address | Ethernet destination address |
| Ethernet type | Ethernet frame type |
| VLAN id | VLAN id |
| VLAN priority | VLAN priority |
| IP src address | IPv4 source address |
| IP dst address | IPv4 destination address |
| IP protocol number | IP protocol |
| IP ToS bits | the value of ToS |
| TCP src port | TCP source port |
| TCP dst port | TCP destination port |
| UDP src port | UDP source port |
| UDP dst port | UDP destination port |

- Action: TABLE II shows four standard categories of actions. These actions specify the processing of packets matching the head field. For one flow, multiple actions can be used to process packets. For example, it is possible to designate ”rewrite the IP destination address and forward the packet to port 1”.

TABLE II
ACTIONS

| Actions | Description |
|---|---|
| Forward | forward packets |
| Enqueue | put packets into the special queue |
| Drop | discard the packet |
| Modify-Field | rewrite fields |

- Counter: OpenFlow has a counter for each flow entry, and some statistical information(Number of packets received, Number of bytes received, Interval time since flow was created) can be obtained.For example, ”the number of packets processed according to this flow entry is 80”.

## III. Related Work

In this section, we will introduce previous research on DDoS attack detection using OpenFlow in SDN. Among existing researches, the mechanisms of DDoS attack detection in SDN are classified into two types, which includes mechanisms based on packet inspection [6] [7] [10] [11] and mechanisms based on flow entry inspection [1] [4] [5].

For mechanisms based on packet inspection, it is necessary to check all packets for detection, even when the network is not attacked. With this processing, the controller spends more time and resources handling regular messages which will cause a waste. Unlike mechanisms based on packet inspection, since flow entry records statistical information of packets matching this flow, in mechanisms based on flow entry inspection. We don’t need to inspect all packets, but inspect the statistical information stored in flow entries through collecting them by controller. Therefore, DDoS attack detection based on flow entry inspection is lighter and more efficient than DDoS attack detection based on packet inspection.

Braga proposed a DDoS flooding attack detection system based on flow entry using OpenFlow [4]. As shown in FIG. 4, this system is divided into three modules. The detection loop of the three modules is described below.

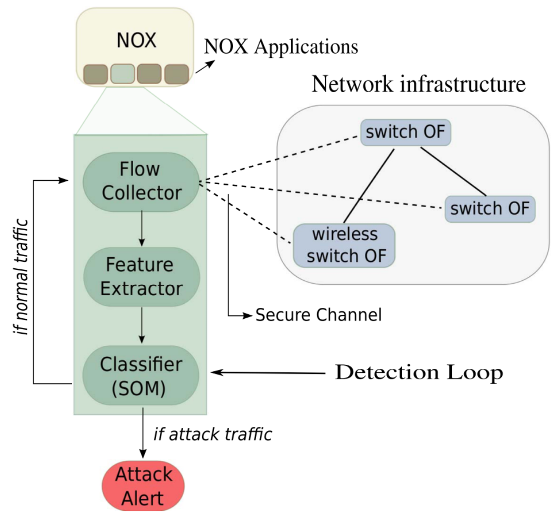

Fig. 4. Detection Loop Operation.

- Flow Collector module: flow entries are collected from the OpenFlow switch registered in the controller at pre-determined time intervals.
- Feature Extractor module: process the collected flow entries, features relating to DDoS flooding attack detection are extracted and passed to the classifier.
- Classifier module: extracted features will be analyzed by SOM (Self Organizing Maps) to judges whether or not a DDoS flooding attack has occurred.

Since using statistical information in flow entries, Braga’s method is lighter than those methods based on packet inspection[19]. However, it is difficult to select the proper interval time during the detection loop. As stated in section 1, if it is too long or too short, efficiency will decrease.

In order to solve this problem, Cui et al. proposed Packet In trigger which can start a detection dynamically[1]. For a switch, if the received packet doesn’t match flow entries, switch will inquire controller by sending a packet in message to it. This trigger judges whether there is abnormality increase of packet in message before starting a DDoS attack detection. As a principle, when a DDoS attack using IP spoofing occurs, switches generates a huge number of packet in packets. On the other hand, since the OpenFlow switch can’t maintain flow entries matching all hosts, DDoS attacks generate a large number new packet in messages even IP spoofing is not used.

Based on the above characteristics of Packet In message, Cui et al. proposed a packet In trigger to detect abnormal increase of packet in packets. Using the Packet In trigger reduces the load on the controller and the switch while responding more quickly to DDoS attacks.

However, the work of controller is network management, collecting statistical data of all flow entries, especially when DDoS attack is happening, will become a huge load and even overloads the controller. In particular, when a high-speed DDoS attack occurs, thousands of Packet In messages generated from the OpenFlow switch are sent to the controller, which may further increase the load and cause a DoS attack on the controller.

To solve overload problem of controller, Giotis et al. proposed a method combining OpenFlow and sFlow[5], they use packet samples instead of real flow entries for decreasing load of controller. However, their method based on packet samples will decrease detection accuracy, and it starts a detection every 30s, which means that it has the same problem with existing researches.

## IV. Packet-In Message Based Detection of DDoS Attack

In this paper, we propose a DDoS attack detection method based on Packet In message, without collecting statistical information or packet samples from switches. To avoid periodically detection, we also use dynamical trigger[1] to realize real time detection.

### A. Design Principle

So far, most previous research has focused on flow entries inspection to detect DDoS attack in SDN. However, all these DDoS attack detection methods have two problems. First, in many detection methods, DDoS attack detection is performed periodically (predetermined time interval). If this cycle is too long, DDoS attacks can not be detected promptly. On the other hand, if it is too short, detection will start frequently, which causes a waste of resources such as CPU and bandwidth. It is also difficult to select an proper interval time. In order to solve this problem, we adopt Packet In trigger in this paper, which makes it become possible to detect DDoS attack at appropriate timing.

Another problem is that, in those methods based on the flow entries inspection, it is necessary to collect statistical information from all the switches registered in the controller. However, when a DDoS attack occurs, a large amount of Packet In messages are sent, and as the processing load of controller increases, a large amount of statistical data collection processing may overload controller.In order to solve this problem, this paper proposes a novel DDoS attack detection method based on Packet In message, which reduces the load of controller by distributing processing load of collecting statistical data.

As a principle, a flow entry is set by controller according to a received Packet-In packet received by the controller, which means that, one Packet In message matches one flow entry. Therefore, we can collect statistical information (IP address, port number, etc.) of header fields through Packet-In message without collecting flow entries. Previous studies collect these at the timing of detection, while our mechanism collect data all the time when packet-in packets is received, which means that processing load of collecting data is distributed. Therefore, our mechanism reduces the load of controller at detection time and avoids overloading.

### B. Architecture

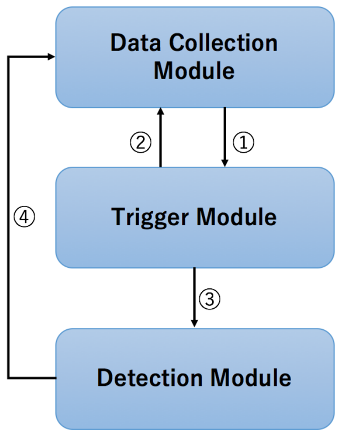

Fig. 5. Detection Loop Operation.

As shown in Fig. 5, this method can be divided into three modules. Specific module contents and transition conditions between modules will be described below.

- Data Collection Module:
  As shown in Fig. 7, data collection module stores the header fields and number of Packet-In packets that have been received, and for each m (predefined) Packet-In packets, controller records the interval time and calculates the arrival rate of Packet-In packets by using the following formula.

$$V_{packet-in} = \frac{m}{Time_{current} - Time_{last}} \tag{1}$$

  This module extracts the following three features for DDoS attack detection.
  - Entropy of IP destination address:

$$h_{dstIP} = \sum_{i=1}^{n} P_i{*}log_2 P_i (\text{n is the number of dstIP types}) \tag{2}$$

  - Entropy of IP destination port:

$$h_{dstPort} = \sum_{i=1}^{n} P_i{*}log_2 P_i (\text{n is the number of dstPort types}) \tag{3}$$

– Entropy of IP source address:

$$h_{srcIP} = \sum_{i=1}^{n} P_i * log_2 P_i (\text{n is the number of srcIP types}) \tag{4}$$

As shown in Fig. 6, In this module, node is created for storing data from Packet-In packets. Put the node into node list and examines all the nodes in the list. Delete expired nodes to update the list. After processing, state transition ① occurs and the system goes to the trigger module.

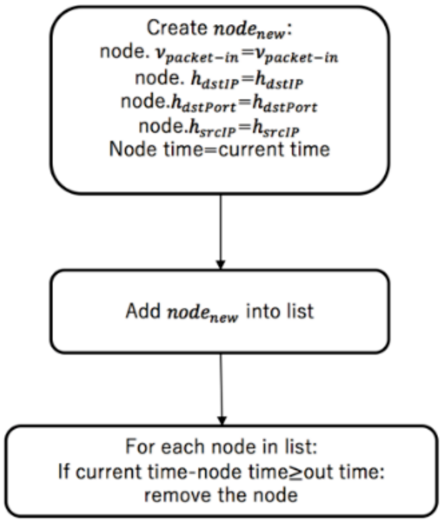
Create $node_{new}$:
node. $v_{packet-in}$=$v_{packet-in}$
node. $h_{dstIP}$=$h_{dstIP}$
node.$h_{dstPort}$=$h_{dstPort}$
node.$h_{srcIP}$=$h_{srcIP}$
Node time=current time
Add $node_{new}$ into list
For each node in list:
If current time-node time≥out time:
remove the node

Fig. 6. Node generation and updation.

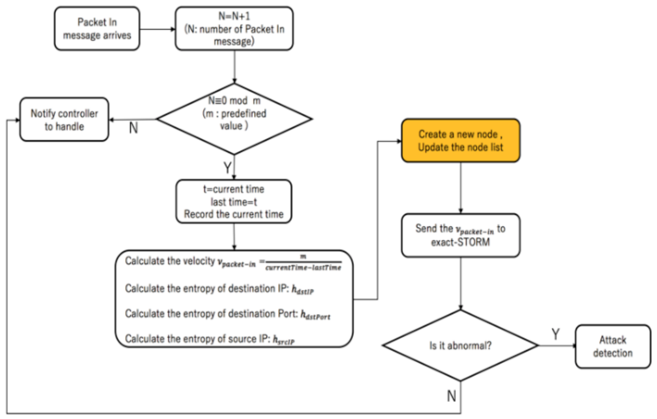
Packet In message arrives
N=N+1 (N: number of Packet In message)
N≡0 mod m (m : predefined value )
N
Notify controller to handle
Y
t=current time
last time=t
Record the current time
Calculate the velocity $v_{packet-in} = \frac{m}{currentTime-lastTime}$
Calculate the entropy of destination IP: $h_{dstIP}$
Calculate the entropy of destination Port: $h_{dstPort}$
Calculate the entropy of source IP: $h_{srcIP}$
Create a new node , Update the node list
Send the $v_{packet-in}$ to exact-STORM
Is it abnormal?
Y
Attack detection
N

Fig. 7. Data collection and workflow of trigger.

- Trigger Module:
The trigger module is used to detect an abnormal burst of Packet-In packets. It uses the algorithm exact-STORM [12] to judged whether $V_{packet-in}$ is abnormal which decides the transition of the system state. If there is no abnormality, the system returns to the data collection module (transition ②). Oterwise, the system goes to the attack detection module (transition ③).
- Detection Module:
In a normal network, for a host or server, there are some parties (hosts, servers, etc.) that often connects with. On the other hand, there are also some parties (hosts, servers, etc) that almost doesn't connect with. Therefore, we suppose that there is a normal state for normal network communication.
Based on that hypothesis, we can see that the three features (entropy of dstIP, entropy of dstPort, entropy of srcIP) extracted from Packet In message follow the normal distribution.As a detection method, this module adopted a method based on entropy. This method is independent of the network topology and traffic characteristics and can be applied to various kinds of network monitoring.
Entropy can estimate the randomness of collected data, high entropy means more distributed probability distribution, low entropy means more focused probability distribution.When a DDoS attack occurs, since target host or server will receive a large number of requests from a lot of machines. Entropy of IP destination address and entropy of destination Port will obviously reduced, while entropy of srcIP will increase than normal.

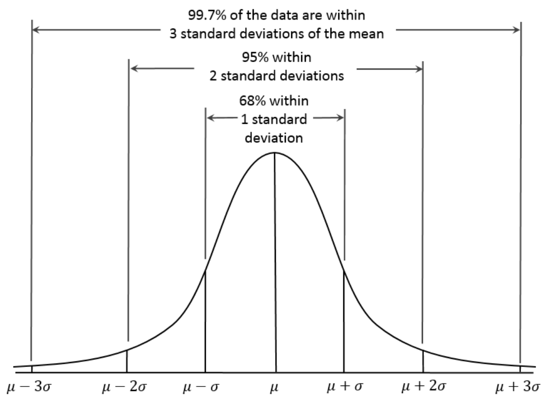

Fig. 8. Normal distribution[21].

– $\mu$: average
– $\sigma$: standard deviation

As shown in Fig. 8, the normal distribution has the following characteristics.

– 68% within 1 standard deviation
– 95% within 2 standard deviation
– 99.7% within 3 standard deviation

Based on above characteristics, we get related data (node.$h_{dstIP}$ , node.$h_{dstPort}$ , node.$h_{srcIP}$ ) from the node list, and if they satisfy one of the following three conditions, detection module judges that a DDoS attack is happening and notify the system.

– Entropy of IP destination address($h_{dstIP}$):

$$h_{dstIP} < \mu_{node.h_{dstIP}} - 2 * \sigma_{node.h_{dstIP}} \quad (5)$$

– Entropy of IP destination Port($h_{dstPort}$):

$$h_{dstPort} < \mu_{node.h_{dstPort}} - 2 * \sigma_{node.h_{dstPort}} \quad (6)$$

– Entropy of IP source address($h_{srcIP}$):

$$h_{srcIP} < \mu_{node.h_{srcIP}} - 2 * \sigma_{node.h_{srcIP}} \quad (7)$$

Otherwise, the system returns to the data collection module by judging that no DDoS attack has occurred (transition ④ ).

## V. Experiments

In this section, we created a controller including DDoS attack detection system based on Packet-In message, and tested attack detection. By comparative experiments, we compared the performance of DDoS attack detection based on flow entries inspection.

### A. Experiment Design

Proposed DDoS attack detection system based on Packet-In message was realized by using the Ryu controller [13]. In order to simulate this mechanism, an experimental platform called Mininet [14] was used. Mininet is a network emulator that can virtually build a network by combining multiple hosts, switches, and routers on one Linux kernel. In this experiment, Mininet created a network including 25 switches, with core switch ($c_1$), aggregation switch ($a_2$ -$a_5$), edge switch ($e_6$ - $e_{25}$), 100 hosts ($h_1$ - $h_{100}$) .

In order to simulate the normal communication, we add a custom command into mininet, what makes it possible for $host_q$(each host) has probability $P_t$ to connect with $host_{q+i}$, probability $P_a$ to connect with $host_{q+j}$, probability $P_c$ to connect with $host_{q+k}$.

The DDoS attack traffic of this experiment was generated by TFN 2K[20], which is a well-known DDoS attack tool. We created a botnet composed of 20 hosts, and performed a DDoS attack called UDP flooding to the target host using IP Spoofing by TFN 2K. Important parameters are set in Table 4.

TABLE III
Parameters used in UDP flooding detection experiment

| Module | Parameter | Definition | Value |
|---|---|---|---|
| Data Collection | m | extract features every m packet-in packets | 1000 |
| Trigger | R | The radius of extra-STORM algorithm | 120 |
| Trigger | s | window's size of extra-STORM | 40 |
| Trigger | k | minimize number of neighbors in extra-STORM | 20 |

The results of DDoS attack detection based on Packet-In message is shown in Fig9, Fig10, Fig11. DDoS attack happened 20 minutes after the start point of experiment. In

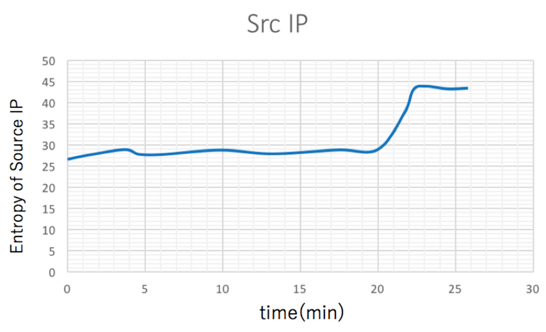

Fig. 9. Entropy of srcIP.

the case of normal communication, entropy of ip-destination-address is about 29 bits, destination-port is about 28 bits, and ip-source-address is about 28 bits. After UDP flooding attack started at 20, ip-destination-address entropy and destination-port entropy obviously decreased, and ip-source-address entropy increased, an attack was detected.

### B. Results

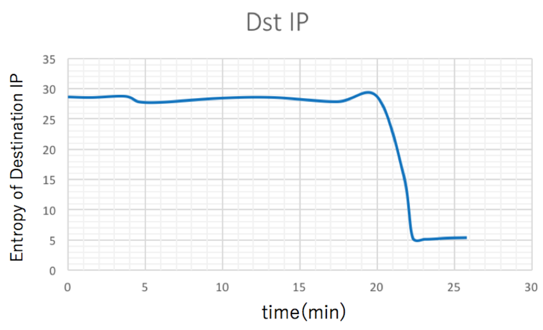

Fig. 10. Entropy of dstIP.

As shown in Table IV, in normal network, for DDoS attack detection based on flow entry, the CPU utilization of controller is lower than attack detection based on Packet-In message ($5.8\% < 11.3\%$). As a cause, traditional method collects statistical data only when an attack detection starts, while our method distributes the processing load by collecting data from packet in packets.

However, when DDoS attack occurs, since the detection based on flow entries requires collecting all flow entries from the switches, the CPU utilization of controller is obviously higher than detection based on Packet-In message ($63.4\% > 32.6\%$).

TABLE IV
CPU utilization of controller

| Detection method | Normal | DDoS attack |
|---|---|---|
| Flow entry based | 5.8% | 63.4% |
| Packet-In message based | 11.3% | 32.6% |

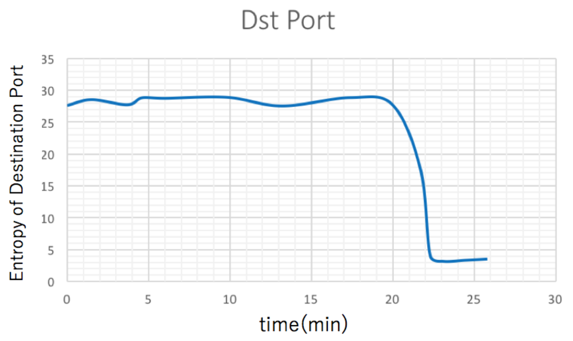

Fig. 11. Entropy of dstPort.

## VI. Conclusion and future work

In the experiment, we showed that entropy of ip-source-address increases, ip-destination-address entropy and destination-port entropy decrease when DDoS attack occurs. Compared with previous DDoS attack detection method based on flow entry inspection, Since the load of collecting data is distributed, the processing load of controller is obviously reduced. Furthermore, by adopting a dynamic Packet In trigger, the response time is faster than many existing studies.

However, in this paper, it is difficult to confirm the precision of experimental data because the experiment was based on a simulated environment. It is necessary to obtain accurate data by using real network traffic. Future challenges are to trace the attack source, reduce the damage from DDoS attacks, and create a complete DDoS attack defense system.

## Acknowledgment

We would like to thank the authors of the previous research, which was indispensable for our work.Thank Prof. Mike DAVID for his English improving comments.

## References

[1] Y. Cui, L. Yan, S. Li, H. Xing, W. Pan, J. Zhu, et al, ”SD-Anti-DDoS: fast and efficient DDoS defense in software-defined networks”, J. Netw. Comput. Appl, pp.65-79, 2016
[2] D.Kreutz, F.M.V.Ramos, P.Verissimo, ”Towards Secure and Dependable Software-Defined Net- works”, Proc. of the second workshop on Hot topics in software defined networking(HotSDN’12), pp.55-60, August 2013.
[3] http://www.itmedia.co.jp/news/articles/1610/22/news024.html.
[4] R. Braga, E. Mota, A. Passito, ”Lightweight DDoS flooding attack detection using NOX/OpenFlow”. Proc. IEEE 35th Conf. Local Comput. Netw., pp.408-415, 2010.
[5] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, V. Maglaris, ”Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments”, Comput. Netw., pp. 122-136, 2014.
[6] S.A. Mehdi, J. Khalid, S.A. Khayam, ”Revisiting traffic anomaly detection using software defined networking”. Recent Adv. Intrusion Detect. pp. 161-180, 2011.
[7] Wang, B., Zheng, Y., Lou, W., ”DDos attack protection in the era of cloud computing and software-defined networking”, Comput. Netw. pp. 308-319. April 2015.
[8] Ryosuke Miyazaki, Junpei Kawamoto, Shinichi Matsumoto, Koichi Sakurai, ”Implementation and Evaluation of Host Independent Network Detection System on OpenFlow Controller”, hinokuni information symposium, 2015.
[9] Nick McKeown et al., ”OpenFlow: enabling innovation in campus networks”. ACM SIGCOMM Computer Communication Review, vol. 38, no. 2, pp. 69-74 , April 2008.
[10] Xiao, Peng, Qu, Wenyu, Qu, Heng, Li, Zhiyang, ”Detecting ddos attacks against data center with correlation analysis”, Comput. Commun. 67 (August (1)), 66-74, 2015.
[11] Miao, R., Yu, M., Jain, N., ”Nimbus: cloud-scale attack detection and mitigation”, In: Proceedings of SIGCOMM, Chicago, IL, USA, ACM, pp.121-122, 2014.
[12] Angiulli, F., Fassetti, F., ”Detecting distance-based outliers in streams of data”, In: Proceedings of the Sixteenth ACM Conference on Information and Knowl- edge Management, ACM, pp. 811-820, 2003.
[13] http://osrg.github.io/ryu/.
[14] http://mininet.org/.
[15] Sunny Behal, Krishan Kumar , Monika Sachdeva, ”Discriminating Flash Events from DDoS Attacks: A Comprehensive Review”, International Journal of Network Security, Vol.19, No.5, PP.734-741, 2017
[16] K.Giotis, C.Argyropoulos, G.Androulidakis, D.Kalogeras, V.Maglaris, ”Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments”, Computer Networks Volume 62, pp.122-136, April 2014
[17] Paul Goransson, Chuck Black, Timothy Culver, ”How SDN Works”, Software Defined Networks, pp.61-88, 2017
[18] Ahmed Abdelaziz, Tan Fong Ang, Mehdi Sookhak, Adnan Akhunzada, ”Survey on Network Virtualization Using OpenFlow: Taxonomy, Opportunities, and Open Issues”, KSII Trans. Internet Inf. Syst., vol. 10, no. 10, pp. 4902-4932, 2016.
[19] Y. Feng, R. Guo, D.Wang, and B. Zhang, ”Research on the Active DDoS Filtering Algorithm Based on IP Flow”, in 2009 Fifth International Conference on Natural Computation. IEEE, 2009, pp. 628-632, 2009.
[20] Neumann, Peter G, ”Denial-of-Service Attacks”, Communications of the ACM, p. 136. Academic OneFile, Aug 2017.
[21] https://en.wikipedia.org/wiki/Normal_distribution.