Journal of Network and Computer Applications 68 (2016) 65–79

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

SD-Anti-DDoS: Fast and efficient DDoS defense in software-defined networks

Yunhe Cui [a], Lianshan Yan [a,*], Saifei Li [a], Huanlai Xing [a,*], Wei Pan [a], Jian Zhu [b], Xiaoyang Zheng [b]

[a] School of Information Science and Technology, Southwest Jiaotong University, Chengdu, China
[b] Centec Networks, Suzhou, China

ARTICLE INFO

Article history:
Received 21 July 2015
Received in revised form
1 February 2016
Accepted 6 April 2016
Available online 7 April 2016

Keywords:
Software Defined Network (SDN)
Distributed Denial of Service (DDoS)
OpenFlow
Security
Detection
Traceback

ABSTRACT

In order to overcome Distributed Denial of Service (DDoS) in Software Defined Networking (SDN), this paper proposes a mechanism consisting of four modules, namely attack detection trigger, attack detection, attack traceback and attack mitigation. The trigger of attack detection mechanism is introduced for the first time to respond more quickly against DDoS attack and reduce the workload of controllers and switches. In the meantime, the DDoS attack detection method based on neural network is implemented to detect attack. Furthermore, an attack traceback method taking advantages of the characteristics of SDN is also proposed. Meanwhile, a DDoS mitigation mechanism including attack blocking and flow table cleaning is presented. The proposed mechanism is evaluated on SDN testbed. Experimental results show that the proposed mechanism can quickly initiate the attack detection with less than one second and accurately trace the attack source. More importantly, it can block the attack in source and release the occupied resources of switches.

© 2016 Elsevier Ltd. All rights reserved.

1. Introduction

Software-defined-networking (SDN) offers flexible network management by decoupling forwarding and control planes Software-defined networking. 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. Due to its flexibility, programmability and maintainability, SDN has been widely studied for its applications in backbone networks, data centers, enterprise networks, access networks, wireless networks, and etc. (Jain et al., 2013; Luo et al., 2012; Wang et al., 2016). As an emerging architecture, SDN is facing with security issues that seem to be quite an obstacle to overcome (Collings and Liu, 2014; Sezer et al., 2013; Shin and Gu, 2013; Yan and Yu, 2015). So far, seven main potential security issues have been presented in Kreutz et al. (2013), including 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 and lack of trusted resources for forensics and remediation. However, using easily 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 (Sezer et al., 2013). Among the current security problems, one of the most urgent and hardest security issues is Distributed Denial of Service (DDoS) (Mirkovic and Reiher, 2014). DDoS can easily cause serious damage because it is easy to start, hard to defend and trace. For example, a DDoS attack against Spamhaus has caused huge network congestion in Europe in March 2013 (Answers about recent ddos attack on spamhaus, 2013). Therefore, effectively detecting and resisting DDoS attack in SDN are crucial for future network architecture deployments in SDN.

So far, a number of mechanisms including DDoS attack detection (Braga et al., 2010; Giotis et al., 2014; Mehdi et al., 2011; Miao et al., 2014; Shin et al., 2013; Peng Xiao et al., 2015; Wang et al., 2015; Li et al., 2014), DDoS attack traceback (Francois and Festor, 2015; Zhang et al., 2015) and DDoS attack mitigation (Giotis et al., 2014; Miao et al., 2014; Shin et al., 2013; Wang et al., 2015; Kampanakis et al., 2014) in SDN have already been proposed. Previous studies mainly focus on the detection methods and the mitigation mechanisms. At current, most of the existing detection methods start periodically. 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 switches handle an extremely large

* Corresponding authors.
E-mail address: lsyan@home.swjtu.edu.cn (L. Yan).

http://dx.doi.org/10.1016/j.jnca.2016.04.005
1084-8045/© 2016 Elsevier Ltd. All rights reserved.

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. However, as an important factor that affects the detection efficiency and the system performance, the trigger mechanism for detection has not yet attracted much attentions from both academia and industry. Meanwhile, the traceback and mitigation of DDoS attack methods can also be improved to fully use the characteristics of SDN, which may be more valuable for the network security.

In order to solve the problems above, we propose a novel mechanism for resisting DDoS attack in SDN. The mechanism is called Software Defined Anti-DDoS (SD-Anti-DDoS), which is composed of four major modules: Detection Trigger Module, Detection Module, Traceback Module and Mitigation Module. The contribution of the paper is summarized below:

- A SD-Anti-DDoS mechanism is proposed, which consists of DDoS attack detection trigger, DDoS attack detection, DDoS traceback and DDoS attack mitigation in SDN.
- The DDoS attack detection trigger method is presented for the first time to achieve the rapid response of detection module and cope with the limitations of the fixed detection loop approach.
- A DDoS traceback method is put forward to find out the attack path where the attack traffic passed by using the characteristics of SDN (the ability of querying the total topology and the information of each switch).
- A DDoS mitigation mechanism is proposed for attack blocking and flow table cleaning.

The paper is organized as follows. Sections 2 and 3 describe the background and related works in detail respectively. Then the proposed mechanism is illustrated in Section 4. Experimental results are presented in Section 5 followed by the conclusions in Section 6.

2. Background

2.1. SDN

As shown in Fig. 1, SDN is composed of three layers and two interfaces, including Infrastructure Layer, Control Layer, Application Layer, Southbound Interface and Northbound Interface. The main difference between the traditional Internet architecture and SDN is that the latter decouples the control and forwarding planes (Software-defined networking). In SDN, the control function is decoupled from forwarding and centralized in the software-based SDN controllers. The infrastructures only need to accept the instructions from controllers and forward the coming packets under the instructions. It is easy to configure, manage, maintain and protect the entire network for managers through intelligent orchestration systems by decoupling the network control function and forwarding.

OpenFlow is the first communication protocol for connecting the control layer and infrastructure layer in SDN (McKeown et al., 2008; The openflow specification version 1.0.0). Now OpenFlow protocol has already become the de-facto standard in SDN. So far six versions of OpenFlow (from version 1.0 to 1.5) have been released. Among them, version 1.0 and version 1.3 are widely used in OpenFlow-enabled switches. In this paper the OpenFlow v1.0 is chosen as the communication protocol for connecting the controller and switches. Slight performance difference may exist between different versions of OpenFlow.

A switch supporting OpenFlow v1.0 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. 2 shows the architecture of a flow entry. Header fields can be composed of some or all of the following items: ingress port, Ethernet source address, Ethernet destination address, Ethernet type, VLAN id, VLAN priority, IP source address, IP destination address, IP protocol, IP ToS bits, transport source port and transport destination port. 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. When a new packet arrives through the switch's port, it will be compared with all flow entries' header fields of the flow table one by one until the matched flow entry is found or all flow entries have been compared. Once the relevant fields of the packet match a flow entry, the counters of this flow entry will be updated and the associated actions will be executed. If the packet does not match all existing flow entries in the flow table, the relevant information of that packet will be sent to the controller.

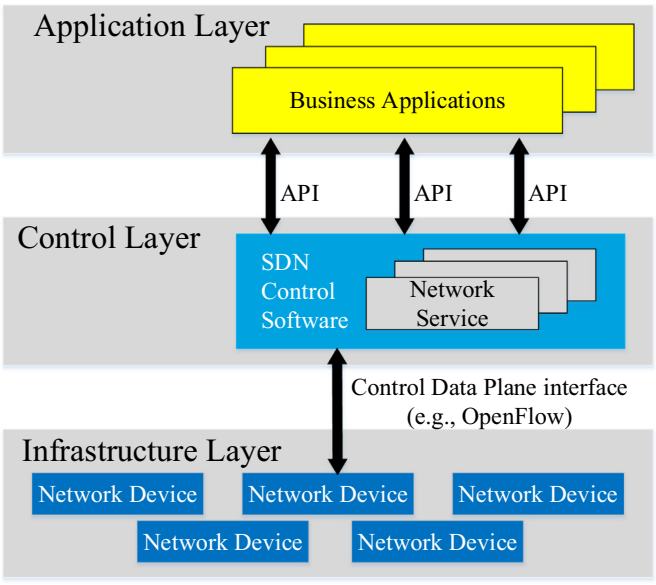

Fig. 1. The architecture of SDN.

2.2. DDoS in SDN

As discussed above, taking advantages of centralized controlling, high programmability and improved automation of SDN, technologies including real-time monitoring, accurate analyzing and rapid response will be supported in SDN. More specifically, SDN can bring the following benefits to DDoS attack protection:

- *Flexible monitoring mechanism:* SDN has ability to implement different kinds of monitoring mechanism. Such as traffic statistics and monitoring about ports or switches, traffic mirroring of special traffic, flow rate monitoring of special traffic and so on. In a word, it is possible to detect DDoS attack using different detection mechanisms by using the flexible monitoring mechanism.
- *Software based detection mechanism:* SDN provides a common programming environment for network managers to control

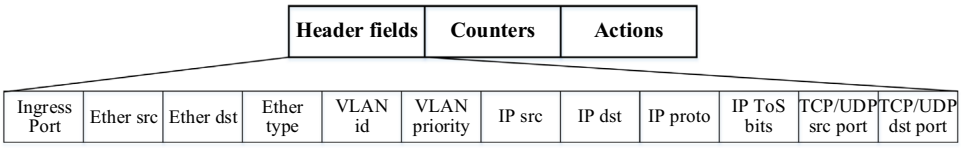

Fig. 2. The architecture of the flow entry.

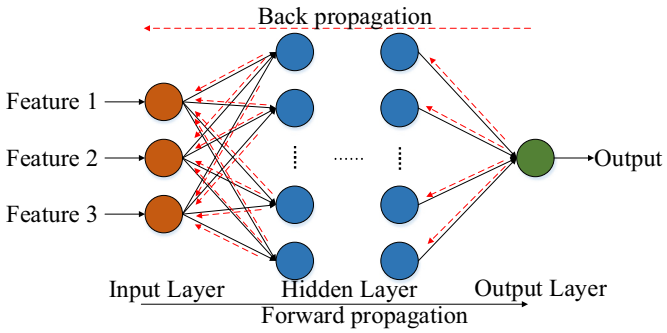

Fig. 3. The architecture of BPNN.

network via software. Therefore, the detection mechanism can also be easily implemented. On one hand, this kind of implementation can be achieved without using hardware. On the other hand, software based detection can be implemented faster than the one based on hardware. At the same time, many new technologies (clustering algorithm, classification algorithm, machine learning algorithm and neural network algorithm) can be easily introduced to DDoS attack detection based on the software implementation.

- *Increased network visualization:* In SDN, the controller has ability to know the real-time information of the entire network. Meanwhile, it can also query the detailed information about each network device. Using the visualization of the entire network in SDN, once the DDoS attack occurs, the attack path and attack source switch can be rapidly found out.
- *More granular network control:* SDN gives network managers the ability to implement different policies more subtly and control every network device more rapidly. For example, the network manager can drop or rate-limit the traffic between special user and server. More subtly, they can even just drop or rate-limit the TCP traffic between special user and server. Thus, once the DDoS attack source switch is found out, SDN enables network managers to mitigate the DDoS attack using blocking or rate-limiting method.

SDN is a flexible, open and programmable network architecture, which makes the innovation about preventing DDoS attack more easily. For example, the Moving Target Defense (MTD) can be implemented in SDN to against DDoS attack. Along with the deeper understanding of SDN and DDoS attack, the DDoS in SDN will be more easily to overcome.

2.3. Back propagation neural network

Back Propagation Neural Network (BPNN), also known as error back propagation algorithm, consists of two processes: computed information forward propagation and error information back propagation (Rumelhart et al., 1986). As shown in Fig. 3, a BPNN model contains three layers: input layer, hidden layer and output layer. One or more neurons could be used in each layer. In the input layer, each neuron is responsible for receiving input information, and passing the information to each neuron of the middle layer. The middle layer is the internal information processing layer, which is responsible for information calculation. The middle layer can be designed as a single hidden layer or multiple hidden layers based on the demand of sensitivity. The computed information is forwarded to each neuron of the output layer from neurons in the last hidden layer. If the actual output matches the expected output or the number of training reaches the upper limit, it will be output. Otherwise, the error back propagation will be started. During this process, each layer's weight will be adjusted according to the gradient descent algorithm. Such process will continue until the network output error is reduced to an acceptable level, or the number of training reaches the predefined upper limit.

3. Related work

3.1. Triggering of DDoS in SDN

As an important factor that affects the detection efficiency and system performance, the trigger mechanism of detection methods has not yet attracted much attentions. At current, most of the Anti-DDoS mechanisms use the periodic trigger to start the detection of DDoS attack based on inspecting flow entries. In Braga et al. (2010), the collection of flow entries is performed at predetermined time intervals by the controller. The authors know that it is hard to choose that time intervals. Therefore, each switch's flow entries are firstly collected at predetermined time intervals, while the system also gives the network manager right to change that time interval. However, choosing a proper period of detection is still hard. In Giotis et al. (2014), the periodic trigger of DDoS attack (called time-window) is used to collect flow information of each switch. The time interval of that window is set to 30 s. At the same time, in order to gather all flow entries, the soft-timeout of each flow entry is set to 31 s.

For the DDoS attack detection based on inspecting packets, there are mainly three kinds of ways to collect packets. The first one is collecting every packets by the controller, then the DDoS attack detection will be applied. The second one is collecting new packets of the session by the controller. The last one is adding sampling components in the switch to collect packets.

In Mehdi et al. (2011), four representative traffic abnormal detection algorithms including Threshold Random Walk with Credit Based (TRW-CB) (Jung et al., 2004), Rate-Limiting (Twycross and Williamson, 2003), Maximum Entropy Detector (Gu et al., 2005) and Network Traffic Anomaly Detector (NETAD) (Mahoney, 2003) were used to detect abnormal traffic in SDN. Among that, the TRW-CB algorithm and the Rate-Limiting algorithm collect the first packet of the new session. The collected packet will be inspected by the controller to judge whether this is an attack packet. The Maximum Entropy Detector will firstly collect the first packet and then it will collect the number of that packets every t seconds to build the packet class distributions. In Miao et al. (2014), packets are sampled at different sampling rates based on its features. In Peng Xiao et al. (2015), all packets are captured from the data center network and the Tcpdump is used to capture and display all packets.

For the DDoS attack detection based on inspecting flow entries, when there are no attacks, the detection will not start if the trigger

of that DDoS attack detection is designed wonderfully. Thus, the trigger of the DDoS attack detection based on inspecting flow entries is quiet important. Even there are three kinds of ways to collect packets in the DDoS attack detection based on inspecting packets, according to the different detected object (one is packet while another is flow entry) used in detection, those three kinds of ways are not suitable for DDoS attack detection based on inspecting flow entries. Therefore, nowadays the DDoS attack detection based on inspecting flow entries still uses the periodic trigger. How to trigger the DDoS attack detection based on inspecting flow entries is still an important issue to overcome.

3.2. Detection of DDoS in SDN

At present, most of the exiting researches focus on the DDoS attack detection. The mechanisms of DDoS attack detection in SDN can be classified into two types, including the one based on inspecting packets (Mehdi et al., 2011; Miao et al., 2014; Shin et al., 2013; Peng Xiao et al., 2015; Wang et al., 2015; Li et al., 2014) and the one based on inspecting flow entries (Braga et al., 2010; Giotis et al., 2014).

In Braga et al. (2010), a flow-based method for detecting DDoS flooding attack was presented. Braga et al. used Self Organizing Maps (SOM) as the kernel module of detection method to distinguish benign traffic and malicious traffic based on traffic features of flow entries in 2010. Compared with traditional approaches, this method only needs to collect flow entries of the flow table instead of checking heavy flow that pass through the switch.

By employing sFlow to gather flow-based data, Giotis et al. (2014) proposed an anomaly detection and mitigation mechanism used in SDN in 2014. The proposed system contains a collector, the anomaly detection and the anomaly mitigation. In the collector module, the native OpenFlow approach and the sFlow were chosen to collect flow-entries or flow packets. Then an entropy-based algorithm was utilized to analyze the collected information to classify abnormal traffic and normal traffic. Once the abnormal traffic was detected, the IP address and port number were provided to the anomaly mitigation module to mitigate the abnormal traffic.

In Mehdi et al. (2011), four traffic abnormal detection algorithms including TRW-CB, Rate-Limiting, Maximum Entropy Detector and NETAD were utilized to detect abnormal traffic in SDN. The performance of those four algorithms was tested and compared. In Miao et al. (2014), NIMBUS was proposed to detect and mitigate DDoS attack using commodity virtual machines (VMs). A flexible sampling strategy was used to capture the network traffic. Then the captured traffic was used for attack detection and mitigation. In Shin et al. (2013), Shin et al. proposed the AVANT-GUARD that aims to mitigate attack traffic which is sent to controller by extending functions of the OpenFlow-enable switch. In Peng Xiao et al. (2015), an effective detection approach based on K-nearest neighbor traffic classification with correlation analysis (CKNN) was proposed to detect the attack. Experimental results show that the proposed approach is good at detecting abnormal traffic. In Wang et al. (2015), a DDoS attack protection mechanism using cloud computing and SDN was proposed to detect and mitigate the DDoS attack. In Li et al. (2014), a method that enables the end hosts to use their knowledge of desired traffic to improve traffic engineering during DDoS attack was proposed. In Kampanakis et al. (2014), Moving Target Defense (MTD) was implemented in SDN. The introduction of MTD into SDN significantly increases the attacker's overheads and enhances the security of SDN.

Compared with the DDoS attack detection based on inspecting packets, the DDoS attack detection based on inspecting flow entries only need to inspect flow entries in switch instead of inspecting all packets. Even there is no attack in the network, the DDoS attack detection based on inspecting packets still need to check the packet to judge whether this is an attack one. This process will make the controller spend more time and resource to handle the normal messages. In contrast, for the DDoS attack detection based on inspecting flow entries, if the trigger mechanism of the DDoS attack detection is designed wonderfully, when there is no attack in the network, the detection will not start. Therefore, the DDoS attack detection based on inspecting flow entries is lightweight and efficient. So the DDoS attack detection based on inspecting flow entries is used to detect DDoS attack in this paper.

3.3. Traceback of DDoS in SDN

Although the traceback plays an important role in resisting DDoS attack, its fundamental mechanism still has not attracted much attention. So far as we know, only few traceback mechanisms used in SDN have been proposed.

The main mechanism used in packet traceback in SDN is mainly based on matching the head fields of the packet with the correlative flow entries. In Francois and Festor (2015), in order to implement the traceback in SDN, flow entries of each switch were periodically captured by the controller. Taking advantages of the entire topology known by the controller, once the attack is found out and the features of the attack packets are identified, the traceback will be carried out on the switch where the attack has been detected. The features of the attack packets (i.e., source Ethernet address, destination Ethernet address, source IP address, destination IP address, source port number, destination port number and protocol type) and the flow entries of the neighbor switches (the switch which is connect to a switch is called its neighbor switch) will be used to find out the possible path where the attack flows come from.

The basic method used in Zhang et al. (2015) is similar to that in Francois and Festor (2015). However, in Zhang et al. (2015), the NetCore policy was utilized to describe the processing of packets among switches. The back policy was made to map a located packet to a set of packets. The traceback mechanism contains two steps. At first, when the forwarding policy or the topology changes, the forwarding policy P and its back policy P' are computed. Secondly, the back policy P' will be applied iteratively on the located packet and the set of packets computed by the located packet. This process will continue until a point where the back policy will drop the packet is found out or the tracing reaches the ingress link at the edge.

When DDoS attack occurs in the network, a huge number of flow entries will be generated in the switches which are on the attack path. For the current traceback methods, a comparison between the header field of the packets and the match field of the flow entries will be executed, therefore, a huge number of comparisons will be made to find the attack source switch. In order to find a lightweight traceback mechanism which is suitable for the DDoS attack detection based on inspecting flow entries, a lightweight traceback mechanism using the detection result to rapidly find out the attack source switch is proposed in this paper.

3.4. Mitigation of DDoS in SDN

The main method currently used to mitigate the DDoS attack is to modify the flow entry of the switch or add a new flow entry in the switch. After the characteristics (e.g. destination IP address, source IP address, destination port address, source port address, ingress port number and protocol type) of the attack traffic are identified, the new flow entry will match the attack packets based on one or more of these characteristics. The new flow entry can modify the header of the attack packet to mark that as an attack packet. It can also forward the matched attack packets to the traffic cleaning center. Meanwhile, it can also directly drop the

attack traffic to block attack.

In Giotis et al. (2014), the IP address and port number are used to match the attack packets while the relevant action of that flow entry is set as drop. In Miao et al. (2014), once the characteristics of the attack packets have been identified, a blacklist or the rate-limit strategies will be made to block or mitigate the attack traffic flow. In Wang et al. (2015), when the attack alert is received by the mitigation module, one of the three basic operations including forwarding, dropping and modifying will be applied to the attack flow. After that, the attack packets and its relevant information will be recorded in the database.

Forwarding, modifying or blocking the attack traffic is easy yet efficient. Thus, it is commonly used. However, even after successfully blocking the attack, huge amounts of malicious flow entries, which are generated by attack flows, will still exist in switches. Therefore, the malicious flow entries need to be deleted to release the occupied storage resources of switches after mitigating the attack. It is a vital issue that need to be solved. Nevertheless, this issue has received little consideration. Thus, in this paper, we try to delete the malicious flow entries which are generated by the attack packets to release the occupied resources of switches after mitigating the attack traffic.

3.5. Main differences among existing research and the proposed technique

Currently, the DDoS attack detection based on inspecting flow entries often uses periodic trigger to schedule detection. Unfortunately, as an important factor that affects the detection efficiency and system performance, the trigger mechanism of detection methods has not yet attracted much attentions. Meanwhile, the attack traceback and mitigation mechanism that make full use of characteristics of SDN are of great importance for the overall network security.

In order to reduce the response time against attack and decrease the load (CPU load and network load) of the controller and the switches, an attack detection trigger used to schedule the attack detection is proposed for the first time. The mechanism of attack detection trigger is the main research content in this paper. However, the mechanisms of attack detection, attack traceback and attack mitigation are also studied. These mechanisms used in this paper have some differences with the existing ones. Therefore, we propose an integrated system to achieve better performance of resisting DDoS attack in SDN in this paper. The detailed descriptions of each module are given in the below subsections.

Main differences between the proposed mechanism in this paper and the exiting mechanisms are listed in Table 1.

4. Design principles and overall architecture

4.1. Design principles

So far, most of current studies mainly focus on the DDoS attack detection, attack traceback and attack mitigation in SDN. As an important part of anti-DDoS system, the trigger mechanism of attack detection has not attached much attention. The existing attack detection is usually scheduled by the periodic trigger. However, the periodic trigger has some deficiencies. When using periodic trigger, if the period is too large, the response time will be long, which makes the controller handle a lot of attack packets and even breaks down the controller and the switches. Otherwise, if the period is too small, considering that most attack detection methods need to collect and analyze the information from the local network, the periodic trigger will also cause extremely large load on the computer and the network. Consequently, a new attack detection trigger called *packet_in* trigger is proposed in this paper.

That trigger mechanism is proposed by using the special message (*packet_in* message) which only exits in SDN. When a switch receives a packet that does not match any flow entries in the flow table or the action of its matched flow entry explicitly stipulates that this packet should be output to controller, a *packet_in* message will be generated and sent to the controller. After receiving the *packet_in* message, the controller will handle this message. The possible actions made by the controller include adding some flow entries in the switch, sending a *packet_out* message (this message is used to ask the switch do something for once) to the switch or even do nothing. If the controller does not add new flow entries to handle this kind of packet, the switch will continually send *packet_in* messages to the controller.

In the meantime, some improvements are implemented to the attack detection, traceback and mitigation, respectively. The detailed descriptions of each module are given in the below subsections. In order to explain the design principle of the system, a state transfer diagram that contains four states and six events is shown in Fig. 4. The above four states are Init State, Detection State, Traceback State and Mitigation State. The explanation of each state is as follows.

Init State: Init State is the initial state of the SD-Anti-DDoS. A *packet_in* trigger is used to schedule attack detection in this state. If the system does not find any abnormal events, it will stay in the Init State. Otherwise, the system will go to Detection State.

Detection State: This state is responsible for detecting the DDoS attack. Once the system enters into this state, the attack detection will start to find whether there is a DDoS attack in the network.

Traceback State: After the system has found out that the network is under attack, the Traceback State will start. In this state, the system will try to trace the attack path and the attack source switch.

Mitigation State: Once in Mitigation State, the system will try to block the attack form source and then clean the malicious flow entries (here, we call the flow entry generated by attack flows as the malicious flow entry, otherwise, it is called benign flow entry) of affected switches.

The explanation of each event is shown below.

Event(1): In the Init State, the system finds that a DDoS attack may occur.

Event(2): DDoS attack is not found by the system in Detection State.

Event(3): The DDoS attack has been found by the system in Detection State.

Event(4): DDoS attack traceback is failed.

Event(5): The attack path and the attack source switch have been found out.

Event(6): Mitigation strategy that aims to block the attack and clean the affected flow tables has been executed successfully.

In the Init State, a DDoS detection trigger module which is responsible for controlling the starting of detection module is running in the system. This module holds the *packet_in* abnormal detection mechanism to find the abnormal behavior of *packet_in* message which may be the indication of DDoS attack. The detailed description of that mechanism will be carried out in Section 4.4. If the *packet_in* abnormal detection module determines that the attack detection should start, the system will be shifted to the Detection State from the Init State.

The Detection State maintains a detection module. This module is used to detect DDoS attack in the network. When the system enters into the Detection State, the attack detection based on BPNN will be carried out. If a DDoS attack is found, the system will be turned to the Traceback State. Otherwise, it will return to the Init State.

In the Traceback State, based on the destination address of

Table 1
Main differences between the proposed mechanism and the exiting mechanisms.

References	Type	Technique	Comments
Braga et al. (2010)	Attack detection	Inspecting flow entries; Periodic trigger of detection; Neural network (SOM); OpenFlow	There is no need to add hardware. It is able to detect DDoS attack based on inspecting flow entries. It is a lightweight way to detect DDoS attack. Periodic collection and detection will increase the load of the controller and the response time.
Giotis et al. (2014)	Attack detection and mitigation	Inspecting flow entries and packets; Periodic trigger of detection; Entropy-based algorithm; OpenFlow	The combination of OpenFlow and sFlow is used to collect information. It can detect anomaly using entropy-based algorithm. It can block attack traffic. Periodic collection and detection will increase the load of the controller and the response time. Anomaly traceback is not achieved. After mitigation, the flow entries generated by attack still exit.
Mehdi et al. (2011)	Attack detection	Inspecting new packets; TRW-CB; Rate-Limiting; Maximum entropy detector; NETAD; OpenFlow	Different ways to collect packets in SDN are proposed. Four different algorithms are implemented in SDN to detect anomaly. The detection algorithms are accurate.
Miao et al. (2014)	Attack detection and mitigation	Inspecting packets; Combining detection tools and algorithms; OpenFlow	Commodity virtual machines are used to analyze traffic for attack detection. Flexible sampling configuration on different traffic flows is achieved. Mitigation including blacklist and rate-limiting is achieved.
Peng Xiao et al. (2015)	Attack detection	Inspecting packets; K-nearest neighbors traffic classification with correlation analysis (CKNN); OpenFlow	The correlation information of traffic in data center is analyzed. The detection approach based on CKNN is used to detect DDoS attack. This detection is lack of scheduling mechanism. It will collect all packets in data center, which are very huge.
Wang et al. (2015)	Attack detection and mitigation	Inspecting packets; Chow–Liu algorithm; Maximum a posterior; OpenFlow	It collects the new arrive packets and inspects those to detect attack. The features of packets are extracted by Chow-Liu algorithm. The mitigation can drop, forward or modify the attack packets and record information of the packets. The traceback will be carried out on the switch where the attack has been detected.
Francois and Festor (2015)	Attack traceback	Graph based modeling; OpenFlow	A Graph-based model is proposed to represent the OpenFlow enabled network. The features of the attack packets and the flow entries of the neighbor switches are used to trace the attack.
Zhang et al. (2015)	Attack traceback	NetCore policy based modeling; OpenFlow	The SDN network is represented as the NetCore policies, then the back policies of the NetCore policies are computed to trace the attack. It is generalizable to any SDN implementation.
This paper	Attack detection trigger, detection, traceback and mitigation	Data stream abnormal detection algorithm; Neural network (BPNN); OpenFlow	An anti-DDoS mechanism contains attack detection trigger,detection, traceback and mitigation is proposed in SDN. A detection scheduling mechanism is proposed to manage attack detection. There is no need to add hardware. A DDoS traceback method using characteristics of SDN is put forward. A DDoS mitigation mechanism is proposed for attack blocking and flow table cleaning.

DDoS attack extracted by the Detection State, a traceback mechanism will be initiated to analyze the switches' flow tables to find out whether those switches contain malicious flow entries generated by attack traffic. Subsequently, the proportion of the malicious flow entry will be calculated to judge whether the switch is in the attack path. If the attack path is successfully found out, the traceback mechanism will trace the attack source switch using the global network topology. Then the system will turn to the Mitigation State. If the attack traceback is failed, the system will turn back to the Detection State to start another detection.

In the Mitigation State, the appropriate mitigation strategy will be made based on the information getting from the Traceback State and then performed at the source switch to block the attack traffic. Subsequently, the corresponding flow tables will be cleared to release the space occupied by attack traffic. After that, the system will go to the Init State.

4.2. Overall architecture

The proposed SD-Anti-DDoS is composed of four modules

which are shown in Fig. 5. As a new portion in Anti-DDoS systems, the Attack Detection Trigger module is marked in light yellow, while Attack Detection module, Attack Traceback module and Attack Mitigation module are marked in light green.

The Attack Detection Trigger module is responsible for controlling the starting of the Attack Detection module. More specifically, the Attack Detection Trigger module determines when the Attack Detection module should start. The *packet_in* message abnormal detection algorithm is the main part of the Attack Detection Trigger module. It is proposed to find out the abnormal representation of *packet_in* messages (asynchronous messages sent from switches to controller when the switches cannot handle the arrived traffic in SDN) which may indicate DDoS attack. The detailed description of the Attack Detection Trigger module is given in Section 4.4.

The Attack Detection module should be scheduled by the Attack Detection Trigger module. The Attack Detection module is responsible for judging whether a DDoS attack occurs. It firstly collects information of all flow entries from the switches. After capturing those information, associated features used in identifying whether the flow entry is benign or malicious will be extracted. Subsequently, the extracted features of the flow entry will be sent to the trained BPNN to find out the malicious flow entry. Based on the identification results of all flow entries, the Attack Detection module will start to judge whether a DDoS attack occurs.

Once the Attack Detection module finds that the network is under attack, the Attack Traceback module will start. This module is responsible for tracing the attack path and the attack source switch. From analyzing the flow entries, the attack destination and the affected switches will be found out. Taking advantage of the analyze results and the topology maintained by the controller, the path where the attack traffic passed will be worked out. Meanwhile, the attack source switch will be found out.

The attack path information produced by the Attack Traceback module will be sent to the Attack Mitigation module. Subsequently, the mitigation strategies will be firstly made and implemented by the Attack Mitigation module. Then it will start to delete the malicious flow entries to release the space occupied by attack traffic. Detailed explanations of each module can be found in the following subsections.

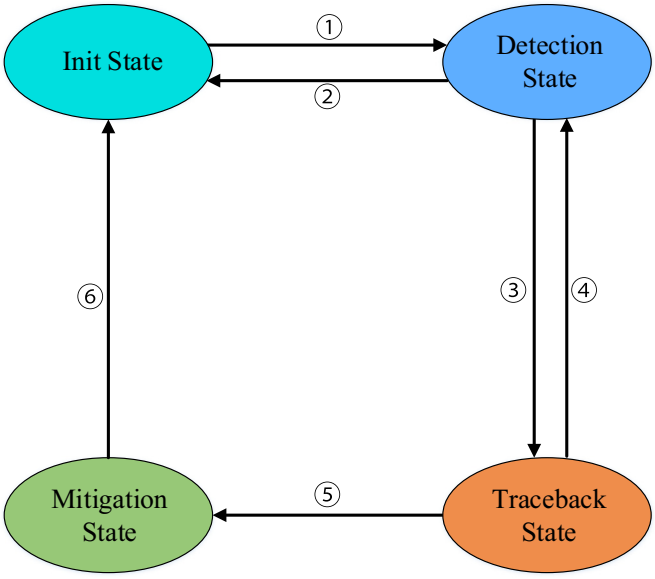

Fig. 4. The state diagram of SD-Anti-DDoS.

4.3. Attack detection trigger

As an important factor that affects the detection efficiency and the performance of system, the trigger mechanism of detection methods has not yet received much research attentions. Many of the current anti-DDoS systems used periodic trigger to start detection. However, the periodic trigger cannot identify and resist the DDoS attack rapidly. Therefore, the workload of controllers and switches will be quite heavy when the network is under attack. In order to respond more quickly against DDoS attack and reduce the workload of controllers and switches, a detection trigger mechanism (here, we called it *packet_in* trigger) which is implemented as the *packet_in* messages abnormal detection module in the Detection Trigger Module is proposed. The Detection Trigger Module is responsible for scheduling of attack detection in the SD-Anti-DDoS.

The *packet_in* message is a kind of special message which only exists in SDN. This kind of message contains the switch's id, the reason of why this packet being sent to the controller, and other

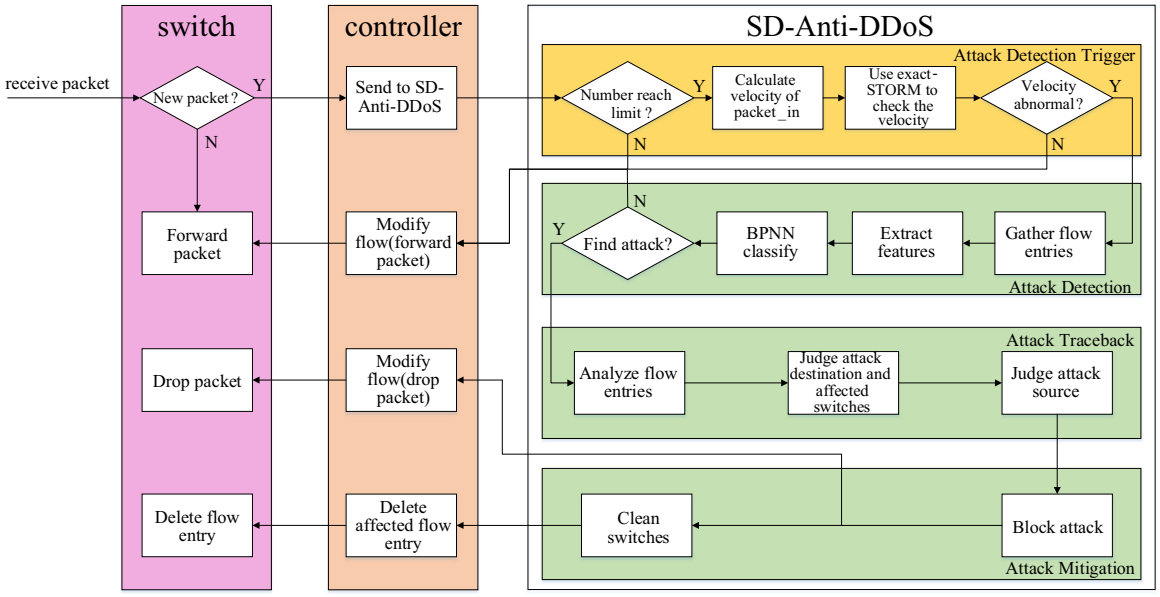

Fig. 5. The components and workflow of SD-Anti-DDoS.

relevant information. If the DDoS attack is launched by using IP Spoofing, a huge number of *packet_in* messages will be generated and sent to the controller. Even the DDoS attack is not launched by using IP Spoofing, considering that the switch in SDN cannot maintain all flow entries that used to match every host, massive new *packet_in* messages will also be generated and sent to the controller.

Based on the characteristic of *packet_in* messages, we propose the *packet_in* trigger which is utilized to detect the abnormal burst of *packet_in* messages. The *packet_in* trigger is implemented as the *packet_in* messages abnormal detection module. That module employs a *packet_in* message velocity calculation module and a *packet_in* message velocity abnormal detection module. The *packet_in* message velocity calculation module is used to calculate the arrival velocity of *packet_in* message. This module holds a *packet_in* message number counter to record the number of *packet_in* message, a predefined modulus and a timer to record the arrival time interval of *packet_in* message. The *packet_in* message velocity abnormal detection module uses the data stream abnormal detection algorithm to detect the outlier of *packet_in* messages (Bulut and Singh, 2005; Vlachos et al., 2004; Zhu and Shasha, 2003; Cormode and Muthukrishnan, 2005). In SD-Anti-DDoS, we choose exact-STORM as the outlier detection algorithm (Angiulli and Fassetti, 2003). However, other data stream abnormal detection algorithms can also be used in this module. The exact-STORM algorithm proposes a data structure called Indexed Stream Buffer (ISB) to maintain the summary of current window. ISB stores the node that contains the information of velocity of *packet_in* messages. The relevant information in the node is:

n.obj: the velocity of *packet_in* messages.
n.id: the identifier of n.obj.
n.count_after: the number of succeeding neighbors of n.obj.
n.nn_before: a list that contains the identifiers of the most recent preceding neighbors of n.obj.

The detailed process of the *packet_in* trigger is shown in Algorithms 1, 2 and Fig. 6. When the *packet_in* message arrives, the number of *packet_in* message will be increased by one. Then the number of *packet_in* message will modulo the predefined modulus. If the remainder is not zero, the *packet_in* trigger will notify the network controller and pass the *packet_in* message to it. The *packet_in* message will be handled by the network controller. Otherwise, the timer module will record the current time. The time interval between the current time and the last one will be calculated and the velocity of *packet_in* message will be obtained using the predefined modulus dividing the time interval. After that, the *packet_in* message velocity will be sent to the *packet_in* messages burst detection module. This module uses exact-STORM algorithm to find whether this velocity is an outlier. If the velocity is judged to be abnormal, the *packet_in* trigger will determine to start detection. Otherwise, the *packet_in* trigger will notify the network controller to handle the *packet_in* message.

Algorithm 1. *packet_in* message abnormal detection.

Input: *packet_in* message
Output: whether the *packet_in* message is abnormal
1: modulus=number interval of *packet_in* message
2: MAX_VELOCITY=a larger enough number
3: increase the number of *packet_in* message by one
4: **if** the remainder of the number of *packet_in* message modulo the modulus equals to zero **then**
5: record the current time
6: calculate the time interval between the current time and the last time
7: the velocity of *packet_in* message is calculated by the modulus dividing the time interval
8: **else**
9: notify the network controller to handle the *packet_in* message
10: **end**
11: /* send the velocity of the *packet_in* message to exact-STORM algorithm to find whether there is a burst*/
12: send the velocity of the *packet_in* message to exact-STORM algorithm
13: use exact-STORM to find whether there is a burst
14: return whether there is a burst

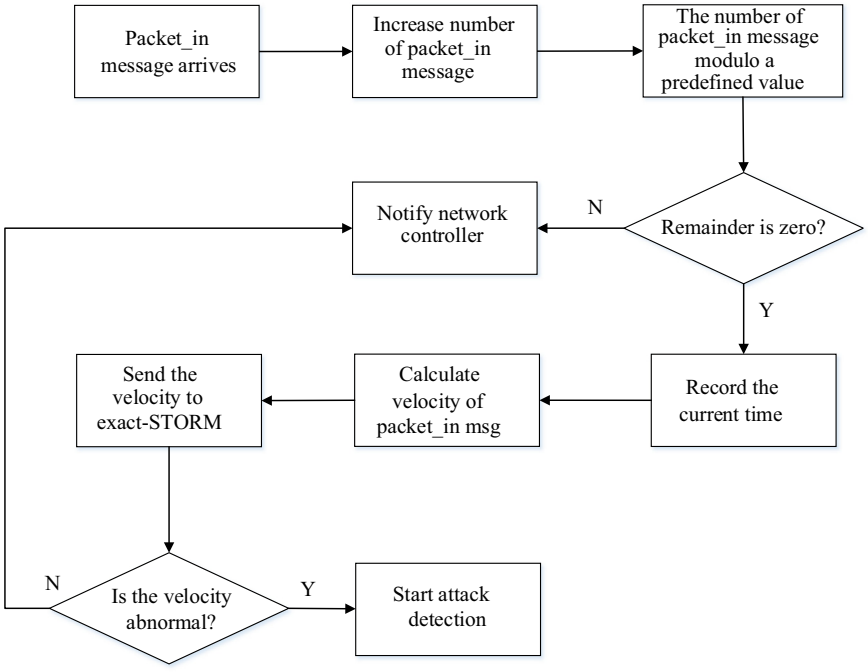

Fig. 6. The workflow of *packet_in* trigger.

Algorithm 2. exact-STORM.

Method 1: update_data_stream(velocity)
Input: the velocity of *packet_in* message
Output: nothing
1: R=the predefined radius
2: **if** the oldest node n_{oldest} in ISB expires then
3: remove that node from ISB
4: **end**
5: create a new node n_{curr}, with $n_{curr}.obj = velocity$, $n_{curr}.id = id$, $n_{curr}.nn\_before = []$, $n_{curr}.count\_after = 1$
6: **for** each node n_{each} in ISB do
7: **if** $\text{abs}(n_{each}.obj - velocity) < R$ then
8: increase $n_{each}.count\_after$ by one
9: append $n_{each}.id$ to $n_{curr}.nn\_before$
10: **end**
11: **end**
12: insert n_{curr} to ISB

Method 2: *find_packet_in_burst* ()
Input: nothing
Output: whether there is an outlier in data stream
1: **for** each node n in ISB do
2: prec_neighs=number of identifiers in n.nn_before
3: succ_neighs=number of identifiers in n.count_after
4: **if** $(prec\_neighs + succ\_neighs) < k$ then
5: return True
6: **end**
7: **end**

The exact-STORM algorithm employs ISB to store the velocity of *packet_in* messages. Here, we define the neighbor whose distance from the new coming velocity is less than the predefined value R as the legal neighbor. In exact-STORM, if the number of *legal neighbor* is higher than the predefined value k, then the new coming velocity will be recognized as an inlier, otherwise it is an outlier.

Now all the DDoS attack detection mechanisms need analyze plenty of information to find out the attack. However, the analyzing often makes huge load to the controller. Therefore, the attack detection mechanism needs to be well scheduled. Unfortunately, as an important factor that affects the detection efficiency and the system performance, the trigger mechanism of detection methods has not yet attracted much attentions. On the contrary, the current attack detection is usually started in the fixed cycle way (Braga et al., 2010; Giotis et al., 2014), which makes the controller spend more resource to implement the attack detection. Meanwhile, the response time of that will be long. As a new mechanism to manage the start of attack detection, the *packet_in* trigger proposed in this paper can catch the symptom of DDoS attack and start the attack detection when the symptom occurs. Compared with the periodic trigger, the *packet_in* trigger can obviously reduce the response time to against attack. Meanwhile, it can also decrease the load of controller.

4.4. Attack detection

At present, most of the exiting researches focus on the DDoS attack detection. The mechanisms of attack detection can be classified into several types, including the one based on neural network (Braga et al., 2010), entropy algorithm (Giotis et al., 2014), k-nearest neighbor algorithm (Mehdi et al., 2011; Peng Xiao et al., 2015), threshold random walk with credit (Mehdi et al., 2011), rate-limiting (Mehdi et al., 2011), graph model (Wang et al., 2015), etc.

Taking advantages of neural network, the attack detection can classify benign flow entry generated by normal traffic and malicious flow entry generated by DDoS attack traffic. In the attack detection, the information of a flow entry will be obtained by the controller, then the eigenvalues of this flow entry will be extracted and sent to the trained neural network to classify whether it is malicious. Consequently, the attack detection can be broadly divided into two steps: the neural network model training stage and the real-time detection stage. Once the system starts, the neural network will be trained firstly. The dataset is made in advance for training. At the beginning, a set of features of malicious attack traffic and benign traffic is collected. The corresponding target value of malicious attack and benign traffic is then assigned with different values (i.e. zero represents for benign traffic and one represents for malicious attack). Subsequently, the extracted flow features and the corresponding target value are combined to form the training dataset. When the system starts, the training dataset is read from the txt file and used to train the neural network. Thus, the available neural network model can be built by using the extracted feature parameters as the input parameters of neural network, and taking the target values as the output parameters.

The attack detection mechanism used in SD-Anti-DDoS is similar to the one proposed in Braga et al. (2010). However, due to the simple implementation, rapid operation and reliable running, the BPNN is used in SD-Anti-DDoS as the classify algorithm. In order to overcome the detection error caused by various packets in network, a detection method based on threshold is also introduced in the detection. The detailed description of the attack detection is given as follows.

Similar to the algorithm in Braga et al. (2010), the following characteristic values are used as the input parameters of neural network: number of packets matched by each flow entry, number of bytes matched by each flow entry, survival time of each flow entry, packet rate of each flow entry and byte rate of each flow entry. These features are the eigenvalues of a flow entry. These can be extracted from the flow statistics message received by the controller. Using these features, BPNN is able to classify the network traffic. The BPNN model used in this paper is built by using the follow parameter: one input layer, one hidden layer and one output layer. The number of neurons in input layer is five, and the number of neurons in hidden layer is ten, while the number of neurons in output layer is one.

As shown in Fig. 7, in real-time detection stage, all flow entries of each switch will be acquired firstly. After the controller receives the flow statistics message about flow entries send by the switch, the flow stats message will be parsed. The flow entry in that message will be processed one by one. At first, the eigenvalues of a flow entry including number of packets matched by each flow entry, number of bytes matched by each flow entry, survival time of each flow entry, packet rate of each flow entry and byte rate of each flow entry will be extracted. Then the extracted eigenvalues are transferred to the BPNN to determine whether the flow entry is benign or malicious. Meanwhile, the classify result will be stored in an array to avoid the repeated processing in traceback state. If the flow entry is considered as a malicious one, its destination IP address will be parsed and appended to an array (*attack_dst_store_list*). After that, if the number of malicious flow entry reaches the predefined upper limit, a DDoS attack alert will be generated firstly. Subsequently, the processing of flow statistics message will be stopped. At last, the attack destination will be found out by searching the one which has the max occurrence number in *attack_dst_store_list*. However, if the flow entry is considered as a benign one or the number of malicious flow entry is less than the predefined upper limit, another flow entry will be processed.

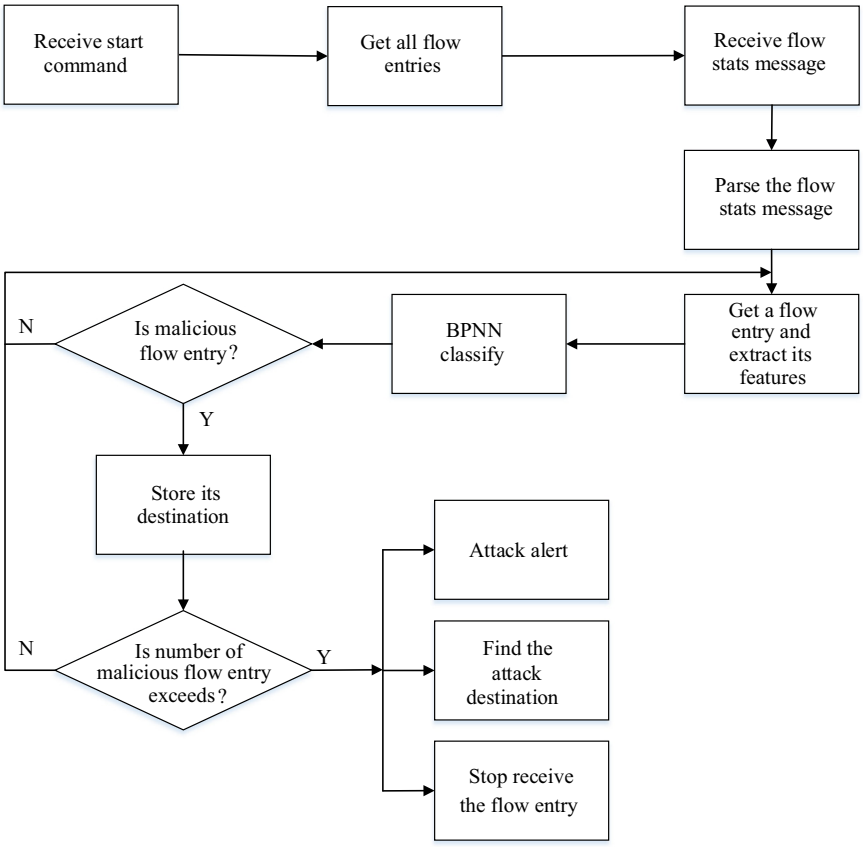

Fig. 7. The workflow of attack detection.

4.5. Attack traceback

So far as we know, the mechanisms of DDoS attack traceback in SDN only have few achievement. However, the traceback of attack source plays a very significant role in resisting DDoS attack. Therefore, in SD-Anti-DDoS, in order to react more quickly to attack, an attack traceback method correlated with the attack detection method is proposed to trace the DDoS attack source switch. In the detection stage, the analyzing results of flow statistics messages will be stored in an array and sent to the traceback module. Based on the results analyzed by the attack detection module and the topology known by the controller, the attack traceback module will continue to analyze the flow statistics message to find out the attack source switch. The attack traceback method is described in detail as follows.

Once the attack detection module detects the existence of DDoS attack, the system will step into the Traceback State. The Traceback module is the main component of the Traceback State. As a main module for determining whether a switch is in the attack path, it utilizes the same trained BPNN model used in attack detection to get the result. After the switches in attack path are successfully found out, the whole attack path in the order that the attack traffic passes will be located using the combination of the network topology, attack destination and the marked switches. As shown in Fig. 8, the main steps of the traceback mechanism used in this paper are as follows. The number of malicious flow entries and benign flow entries of each switch is recorded. Then the number of total flow entries and the proportion of malicious flow entries are calculated. Based on the number and the proportion of malicious flow entries in each switch, whether the switch is located on the attack path will be determined. If the number of malicious flow entries is over a predefined upper limit or the proportion of malicious flow entries is higher than the predefined value, the switch will be marked as a malicious switch which is on the attack path. Otherwise, it will be marked as a benign switch. Using the combination of global network topology, attack destination and marked malicious switches, the accurate attack path in the order that attack traffic passes will be found out.

4.6. Attack mitigation

After the attack path and the attack source switch are identified, the attack mitigation module will start. The most important task of the attack mitigation module is blocking the attack traffic. Now, most of current research has already achieved this function in SDN (Giotis et al., 2014; Miao et al., 2014; Shin et al., 2013; Wang et al., 2015; Kampanakis et al., 2014). However, after successfully blocking the attack traffic, huge amounts of malicious flow entries, which are generated by attack, will still exist in switches. Those flow entries are useless. Meanwhile, those can be a waste of storage space of switches. Therefore, after blocking the attack traffic, the malicious flow entries are deleted to release the occupied storage space in the proposed method.

Once the attack source switch is found out, the attack mitigation module will start. Using controller-to-switch messages to insert flow entries with highest priority (65,535) into the flow table of the switch which is marked as the attack source switch, the Attack block module tries to stop the attack traffic. Those flow entries are called blocking flow entries in this paper. The structure of blocking flow entry is shown in Fig. 9. In the blocking flow entry, the attack destination address is assigned as the IP destination address and the ingress port of attack traffic is assigned as the ingress port of flow entry's Header fields. Meanwhile, The Drop action is used to block the malicious attack traffic. In the meantime, this flow entry has the highest priority (65,535) than any other flow entries in this switch.

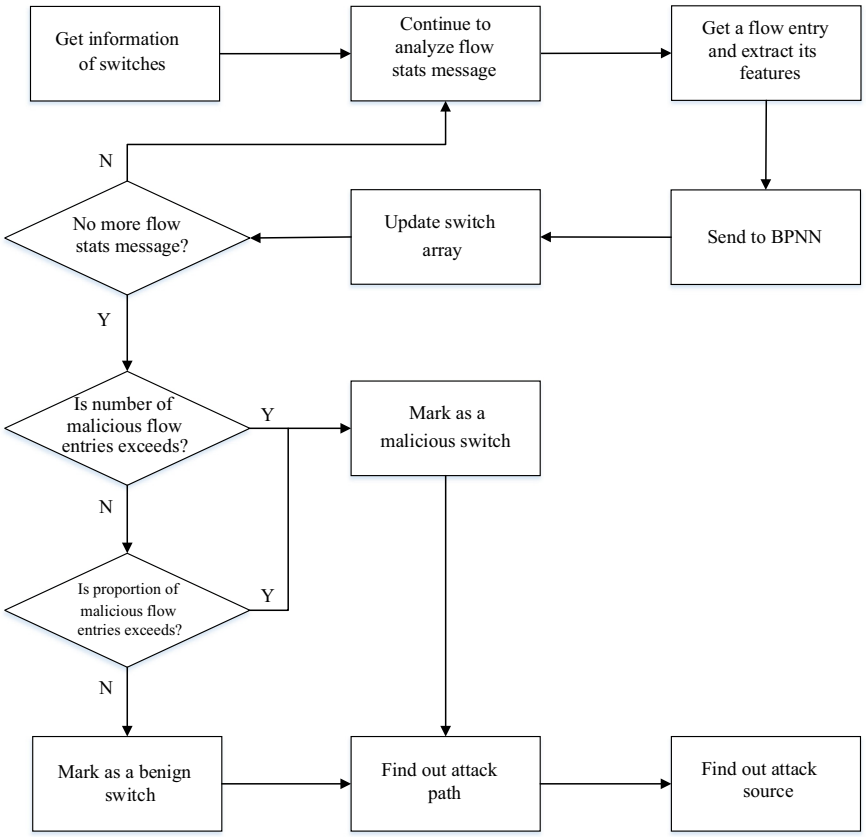

Fig. 8. The workflow of attack traceback.

After the blocking flow entry has been inserted into the attack source switch, the attack traffic arrives through the ingress port will be matched against the blocking flow entry, thus it will be directly discarded. Therefore, the attack mitigation can be achieved. Once the block strategy is executed successfully, the attack traffic will be blocked. But a huge number of flow entries still exist in switches that are in the attack path. Therefore, the flow table modification message (*OFPFC_DELETE* message) is proposed to delete the flow entries which are marked as malicious flow entry.

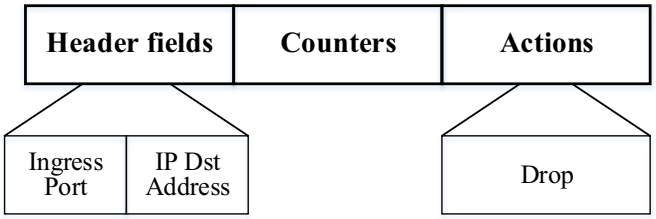

Fig. 9. The architecture of blocking flow entry.

5. SD-Anti-DDoS evaluation

In this section, we mainly focus on the evaluation of the SD-Anti-DDoS system. The performance comparison of *packet_in* trigger and periodic trigger in SD-Anti-DDoS is evaluated. The overall performance of SD-Anti-DDoS and each module is also tested.

5.1. Evaluation setting

The proposed mechanism is implemented on the RYU controller (Ryu sdn framework). The benign traffic transferred in the test was composed of several protocols: 85% was TCP, 10% was UDP and 5% was ICMP. It is generated based on the analysis of traffic which was transferred between Japan and USA in seven years (Borgnat et al., 2009). Distributed Internet Traffic Generator (D-ITG) is the benign traffic generator used in the evaluation (Botta et al., 2012). D-ITG is a platform that supports to generate TCP, UDP, SCTP, DCCP and ICMP traffic at packet level. Meanwhile, the packet size and the rate of generated traffic can obey different distributions, such as uniform distribution, exponential distribution, normal distribution, Poisson distribution, Pareto distribution, Cauchy distribution, gamma distribution, Weibull distribution and even customized distribution.

DDoS attack traffic in this paper is generated by TFN2K. TFN2K is a well-known DDoS attack tool, which has been widely used to attack several large famous sites. IP Spoofing technique is used in TFN2K. At the same time, TFN2K can produce most types of attacks: UDP flood attack, TCP/SYN flood attack, ICMP flood attack and mixed attack contains various kinds of attack.

Mininet is an experimental platform which is used to simulate the proposed mechanism (Lantz et al., 2010; Mininet). Taking advantages of Mininet, a network that contains twenty five switches including the core switch (c1), the aggregation switch (a2–a5) and the edge switch (e6–e25) and two hundred hosts (h1–h200) was created. In the experiment, twenty hosts tried to attack the host whose IP is 10.0.0.1 using TFN2K with IP Spoofing. Mininet was running on a computer. Meanwhile, the RYU controller was running in another computer that connected to the one running Mininet. Important parameters are set as Table 2.

5.2. Evaluating overall performance

Experiment 1 was implemented to show the effectiveness of mitigating DDoS attack of SD-Anti-DDoS system (here, we used *packet_in* trigger as the detection trigger). Mixed DDoS flood

Table 2
Important parameters.

Usage	Symbol	Definition	Value
packet_in trigger	*modulus_value*	The modulus used in calculating the velocity of *packet_in* message	2000
	STORM_window_size	The window's size of extra-STORM algorithm	20
	STORM_radius	The radius of extra-STORM algorithm	150
	STORM_neighbor_limit	The minimize number of neighbors in extra-STORM algorithm	10
Periodic trigger	*periodic_interval*	The cycle length of periodic trigger	300
Attack detection module	*BP_input_neuron_num*	The number of neuron in input layer	5
	BP_hidden_neuron_num	The number of neuron in hidden layer	10
	BP_output_neuron_num	The number of neuron in output layer	1
	upper_limit	The upper limit of suspected attacks	50
Attack traceback module	*malicious_flow_entries_max_rate*	The maximum proportion of malicious flow entries	0.3
	malicious_flow_entries_max_num	The maximum number of malicious flow entries	1000

Table 3
Actions of each module over time in simulation.

State	Event	Time
Init State	Application start	19:40:01
	BPNN model training start	19:40:02
	BPNN model training end	19:40:08
	Detect *packet_in* message abnormal	19:40:43
Detection State	Detection module start	19:40:43
	Detect attack	19:40:43
Traceback State	Traceback module start	19:40:43
	Successful traceback	19:40:58
Mitigation State	Attack block module start	19:40:58
	Flow table clean module start	19:41:03

attacks contain TCP/SYN flood, UDP flood and ICMP flood were launched when the system was running.

Table 3 is the detailed description of the starting time of each module in experiment 1. At the beginning of Init State, the BPNN model was firstly trained by using the training dataset which is made in advance by the above way in Section 4.4. This process was started at 19:40:02 and lasted 6 s. At 19:40:43, the *packet_in* message abnormal detection module found that the velocity of *packet_in* messages was abnormal. Therefore, it generated a startup command of the attack detection module. The detection module started at 19:40:43. This module found the attack at 19:40:43. Subsequently, the Traceback Module started at 19:40:43 and the path of attack was successfully traced at 19:40:58. After the Traceback Module was done, the Mitigation module was started at 19:40:58 to block the attack from the source and clean the switches.

Fig. 10 describes the utilization ratio of CPU of the controller. It is observed that once the application was started, the BPNN model was firstly trained. This process worked on CPU0 at 19:40:02 and continued to 19:40:08. During the training of BPNN model, the usage of total CPU was about 50%. At 19:40:12 *packet_in* messages generated by benign switch were received. Thus the usage of CPU was a little higher than usual from 19:40:12 to 19:40:17. At 19:40:42, the attack began. Therefore, the usage of CPU0 was increased rapidly to 100% after 19:40:43. From then on, the usage of CPU0 was all 100% due to the application required most CPU resources to deal with attacks. Subsequently, the blocking message was sent to the relevant switches at 19:40:58. So the CPU usage started to decrease to normal level after that moment.

Fig. 11 shows the sending rate and the receiving rate of relevant Ethernet port of the controller. It is observed that after the switches connected to the controller, the benign traffic caused a set of *packet_in* messages from 19:40:12 to 19:40:17. During that, the application sent a set of flow-mod messages to handle that event. At 19:40:43, because of the starting of attack, the receiving rate increased rapidly. Meanwhile, the application was still

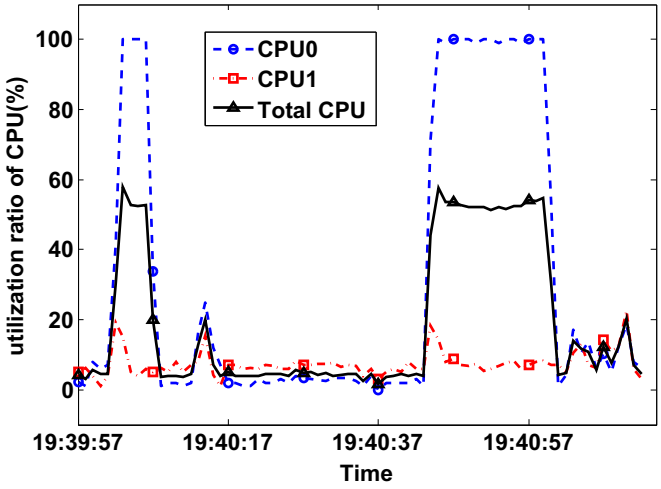

Fig. 10. Utilization ratio of CPU of controller over time.

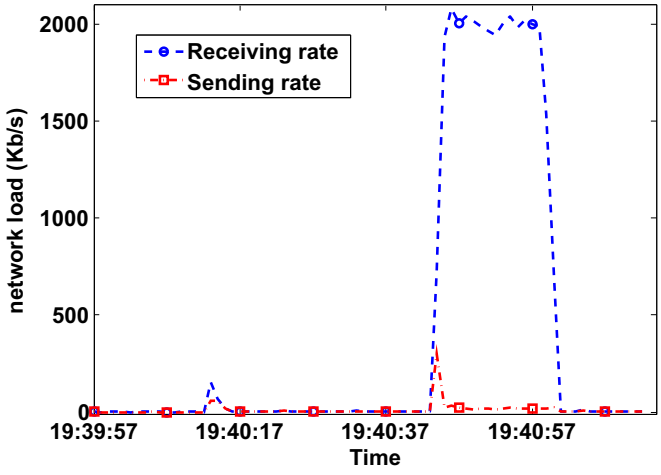

Fig. 11. Network load of controller over time.

sending a set of messages to handle the *packet_in* messages. But after the Traceback state was started, the *packet_in* messages was not processed any more. Therefore, C0 stopped sending traffic immediately after that time. At 19:40:45, the receiving rate was about 2000 KB/s. However, when the application was moved into mitigation state and began to block the attack at 19:40:58, the receiving rate and sending rate of controller decreased to normal level. Table 4 shows the results of traceback in experiment 1, where, the traceback was started at 19:40:43 and ended at 19:40:58. After successfully tracing the attack source switch and attack path, the Traceback Module has found that the attack

destination was h1 while the source switch was e11 and e16, while the attack path were $e11 \rightarrow a3 \rightarrow c1 \rightarrow a2 \rightarrow e6$ and $e16 \rightarrow a4 \rightarrow c1 \rightarrow a2 \rightarrow e6$. Experiment 1 indicated that the Traceback Module could successfully trace the attack source switch and attack path, which provided the prerequisite for the implementation of mitigation module.

Table 5 shows the comparison between the receiving rates of the controller before and after blocking. It indicates that once the blocking flow entry was added into the source switch, the attack traffic will be successfully blocked.

In Experiment 1, we randomly started DDoS attacks to evaluate the SD-Anti-DDoS. The result shows that when a DDoS attack was launched, the *packet_in* trigger in SD-Anti-DDoS would quickly find the abnormality of the *packet_in* message. Subsequently, the Detection module started to find out whether there is a DDoS attack in the network. Then the whole attack path in the order that the attack traffic passes would be found out by the Traceback module. After the attack path and the attack source switch were detected, the Mitigation module would start to block the DDoS attack and release the space occupied by malicious traffic.

5.3. Performance testing of attack detection trigger

Table 6 lists the performance comparison results of our method to Braga et al. (2010) and Giotis et al. (2014). In Table 6, the algorithm in Braga et al. (2010) and Giotis et al. (2014) used periodic trigger to schedule the attack detection. The evaluation of that trigger and the *packet_in* trigger is shown in the remainder of this session. All of the three mechanisms can detect the DDoS attack in SDN. As an algorithm firstly proposed to detect attack by inspecting flow entries in SDN, Braga et al. (2010) focus on the DDoS detection, therefore, it cannot achieve the DDoS traceback and mitigation. The mechanism in Giotis et al. (2014) is proposed to detect and mitigate DDoS attack. However, the traceback of DDoS attack is not achieved in that mechanism. Meanwhile, the mitigation of that mechanism only contains blocking of attack, which leaves a huge number of flow entries still exit in switches. Taking advantage of neural network, the mechanism proposed in this paper inspects flow entries to detect DDoS attack. At the same time, the traceback of attack is also achieved. What's more, besides blocking the attack flows, the mitigation method can also delete the flow entries generated by attack traffic.

In order to compare the performance of the *packet_in* trigger and the periodic trigger, experiment 2 has been implemented. In this experiment, the periodic trigger and the *packet_in* trigger are tested respectively. In each test, the DDoS attack was launched for ten times in two hours. Therefore, the periodic trigger and *packet_in* trigger was run respectively in two hours. The trigger mechanism is the only different part in that two methods, which means that the other modules such as detection module, traceback module and block module used in the two methods are same. Here, the periodic interval of periodic trigger is five minutes.

Fig. 12 is the result of the comparison between the periodic trigger and the *packet_in* trigger. In Fig. 12(a), the mean response time is the mean time interval from the attack's starting time to the detection module's starting time. In Fig. 12(a), we can see that the mean response time of the periodic trigger is up to 139 s while the *packet_in* trigger can start the detection in 1 s. Taking advantages of *packet_in* trigger, the response time will be reduced obviously into less than 1 s. Therefore, the *packet_in* trigger can react more quickly to DDoS attack.

The number of *packet_in* messages which are sent to the controller and the number of flow modification messages which are generated by the controller and sent to the switch are also used to evaluate the performance of the periodic trigger and the *packet_in* trigger. As shown in Fig. 12(b), in the periodic trigger, the number of *packet_in* message and flow modification messages can reach up to 405,630 and 362,274, which is quite a big data to handle. However, in *packet_in* trigger, the number of *packet_in* message and flow modification messages is only about 2000 and 1810. Compare with that of the periodic trigger, the number of data to handle is reduced significantly.

The load of controller that contains the utilization ratio of CPU and the network load is also used to evaluate the performance of that two triggers. By using Dstat (a Linux monitoring tool), we found that when the controller was running normally, the utilization ratio of CPU is about 2 percent. However, when it the DDoS attack occurs, the utilization ratio of CPU can be up to 58 percent. In experiment 2, the mean utilization ratio of CPU is shown in Fig. 12(c). The mean utilization ratio of CPU of the f periodic trigger is about 20.4 percent while that of the *packet_in* trigger is only about 5.5 percent.

The network load is also recorded by using Dstat. The mean network load is shown in Fig. 12(d), where the mean network load of the periodic trigger is up to about 156.74 Kb/s, while the mean network load of the *packet_in* trigger is only about 35.25 Kb/s.

In experiment 2, two tests are performed to evaluate the performance of the periodic trigger and the *packet_in* trigger. In each tests, we randomly started DDoS attacks for ten times during two hours. When the SD-Anti-DDoS system uses the periodic trigger (the cycle length is five minutes), it is easy to find that the mean response time will be near to half of the cycle length (139 s in this experiment). However, when using the *packet_in* trigger as the detection trigger mechanism, once the DDoS attack occurs, the exact-STORM will find the data stream of *packet_in* message is abnormal, thus the attack detection module will start quickly with less than 1 s.

Table 4
Results of attack traceback in simulation.

	Victim	h1
Results of attack traceback	Attack destination	h1
	Source switch	e11,e16
	Attack path	a3,c1,a2,e6;a4
	Traceback start	19:40:43
	Traceback end	19:40:58

Table 5
Results of blocking in simulation.

Number	Receiving rate before blocking	Receiving rete after blocking	Time of block starting
1	1925.1 KB/s	0.8 KB/s	19:24:27

Table 6
Results of performance comparison.

Reference	Trigger	Detection	Traceback	Mitigation	
				Block attack flows	Clear attack flow entries
Braga et al. (2010)	Periodic trigger of detection	Y	N	N	N
Giotis et al. (2014)	Periodic trigger of detection	Y	N	Y	N
This paper	*packet_in* trigger	Y	Y	Y	Y

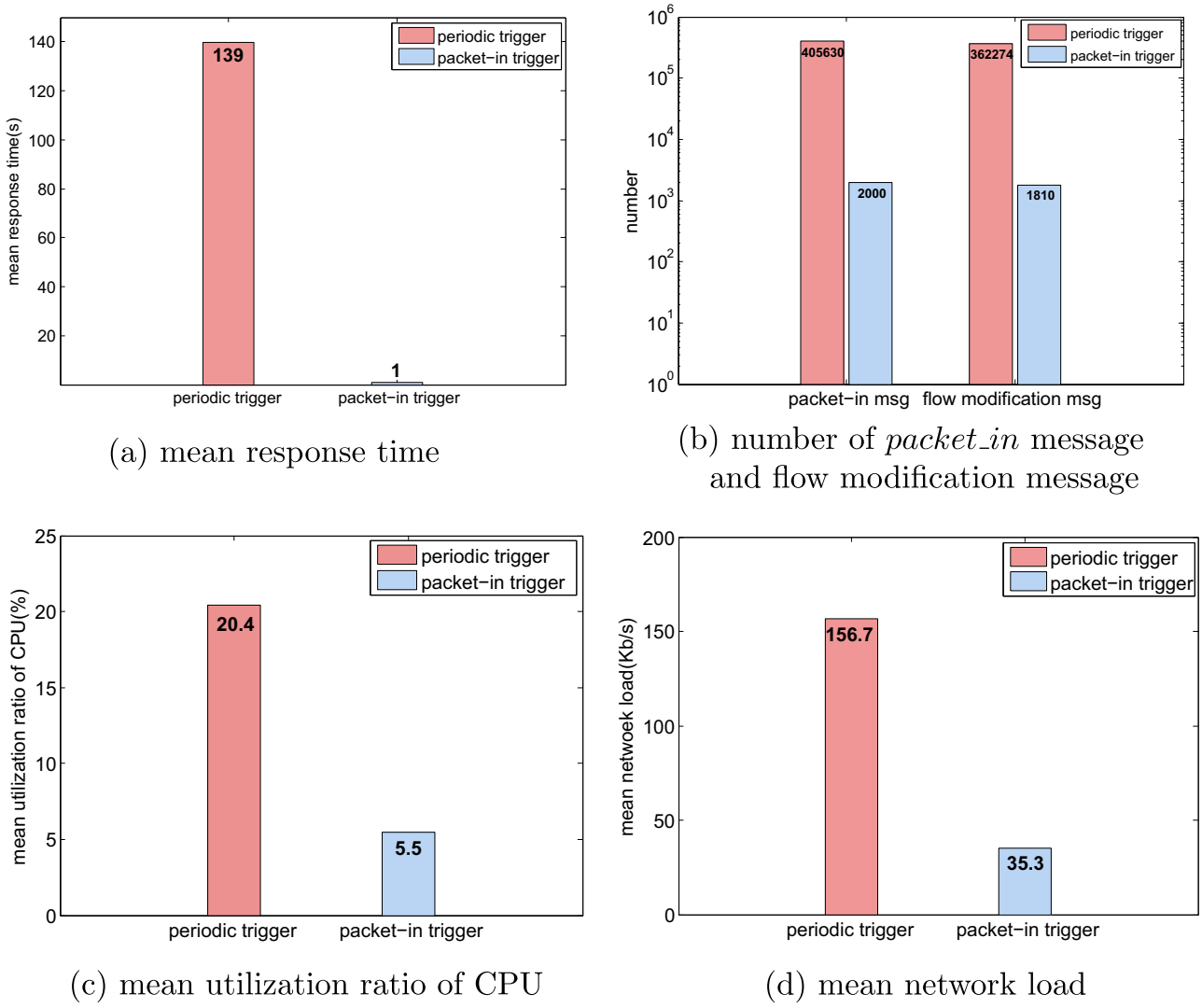

Fig. 12. The comparison of performance of the periodic trigger and the *packet_in* trigger.

From Fig. 12, it can be seen that the *packet_in* trigger can greatly reduce the response time to start the attack detection. When DDoS attack occurs, plenty of *packet_in* messages will be sent to the controller. Thus, if the periodic trigger is used as the trigger mechanism, the response time will be large, which makes the controller receive a huge number of *packet_in* messages and generate flow modification messages (this kind of message is generated by the controller to handle the *packet_in* message). However, if the *packet_in* trigger is used, the response time will be shorter, therefore, the number of *packet_in* messages and flow modification messages will be less. Hence, by reducing the response time to start the attack detection, the CPU load and network load of the controller will be reduced appreciably.

Therefore, the *packet_in* trigger can appreciably reduce the response time to against the attack, while it can also reduce the load of the controller. The experimental results show that the *packet_in* trigger can achieve better performance than the periodic trigger.

6. Conclusion and future work

The SD-Anti-DDoS mechanism used in SDN was proposed and demonstrated. An attack detection trigger method was for the first time presented to schedule the starting of attack detection. An attack detection method was used to detect the DDoS attack. A traceback method was introduced to trace the attack source using the characteristics of SDN. For blocking attack traffic and releasing the space occupied by attack, a mitigation method was also introduced. A system called SD-Anti-DDoS to realize the trigger mechanism of attack detection, the attack detection, the special traceback and mitigation in SDN was implemented on RYU controller.

The availability of that system was evaluated. The results show that the proposed mechanism can quickly start the attack detection with less than 1 s and accurately trace the attack source switch. More importantly, it can block the attack in source and release the occupied space of switches. It is demonstrated that the proposed attack detection trigger mechanism can appreciably reduce the response time against the attack compared to the currently used periodic trigger. Meanwhile, it can also reduce the CPU load and the network load of the controller.

As one of the most commonly used protocol in SDN, OpenFlow v1.0 was used in this paper. However, SD-Anti-DDoS can also be implemented by using other versions of OpenFlow. Slight performance difference may exist between different versions of OpenFlow. In the near future, we will continue to study new anomaly detection techniques in SDN, as well as mitigation methods. With the development and deployment of SDN, security will be of outmost importance for researchers.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (Grant no. 61325023 and 61401374), and the Key Project of China Railway (Grant no. 2014X008-A and 2016-ZB14).

References

Angiulli, F., Fassetti, F., 2003. Detecting distance-based outliers in streams of data. In: Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management, ACM, pp. 811–820.

Answers about recent ddos attack on spamhaus, 2013. Website. ⟨http://www.spamhaus.org/news/article/695⟩.

Borgnat, P., Dewaele, G., Fukuda, K., Abry, P., Cho, K., 2009. Seven years and one day: sketching the evolution of internet traffic. In: Proceedings of INFOCOM, pp. 711–719.

Botta, A., Dainotti, A., Pescap, A., 2012. A tool for the generation of realistic network workload for emerging networking scenarios. Comput. Netw. 56 (October (15)), 3531–3547.

Braga, R., Mota, E., Passito, A., 2010. A lightweight ddos flooding attack detection using nox/openflow. In: Proceedings of LCN, IEEE, pp. 408–415.

Bulut, A., Singh, A., 2005. A unified framework for monitoring data streams in real time. In: Proceedings of ICDE, IEEE, pp. 44–55.

Collings, J., Liu, J., 2014. An openflow-based prototype of sdn-oriented stateful hardware firewalls. In: Proceedings of ICNP, Raleigh, USA, IEEE, pp. 525–528.

Cormode, G., Muthukrishnan, S., 2005. What's new: finding significant differences in network data streams. IEEE/ACM Trans. Netw. 13 (6), 1219–1232.

Francois, J., Festor, O., 2015. Anomaly traceback using software defined networking. In: 2015 National Conference on Parallel Computing Technologies (PARCOMPTECH), IEEE, pp. 203–208.

Giotis, K., Argyropoulos, C., Androulidakis, G., Kalogeras, D., Maglaris, V., 2014. Combining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on sdn environments. Comput. Netw. 62 (April (7)), 122–136.

Gu, Y., McCallum, A., Towsley, D., 2005. Detecting anomalies in network traffic using maximum entropy estimation. In: Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement, USENIX, pp. 32–32.

Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A., Venkata, S., Wanderer, J., Zhou, J., Zhu, M., Zolla, M., Holzle, U., Stuart, S., Vahda, A., 2013. B4: experience with a globally-deployed software defined wan. In: Proceedings of SIGCOMM, Hong Kong, China, ACM, pp. 3–14.

Jung, J., Schechter, S., Berger, A., 2004. Fast detection of scanning worm infections. In: Proceedings of RAID, Sophia Antipolis, France, Springer, pp. 59–81.

Kampanakis, P., Perros, H., Beyene, T., 2014. Sdn-based solutions for moving target defense network protection. In: Proceedings of WoWMoM, IEEE, 2014, pp. 1–6.

Kreutz, D., Ramos, F., Verissimo, P., 2013. Towards secure and dependable software-defined networks. In: Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking, ACM, pp. 55–60.

Lantz, B., Heller, B., McKeown, N., 2010. A network in a laptop: rapid prototyping for software-defined networks. In: Proc. Hotnets, Monterey, CA, USA, ACM, page 19.

Li, J., Berg, S., Zhang, M., Reiher, P., Wei, T., 2014. Drawbridge: software-defined ddos-resistant traffic engineering. In: Proceedings of SIGCOMM, Chicago, USA, ACM, pp. 591–592.

Luo, T., Tan, H.P., Quek, T.Q.S., 2012. Sensor openflow: enabling software-defined wireless sensor networks. IEEE Commun. Lett. 16 (November (11)), 1896–1899.

Mahoney, M., 2003. Network traffic anomaly detection based on packet bytes. In: Proceedings of the 2003 ACM Symposium on Applied Computing, ACM, pp. 346–350.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J., 2008. Openflow: enabling innovation in campus networks. ACM SIGCOMM Comput. Commun. Rev. 35 (April (2)), 69–74.

Mehdi, S., Khalid, J., Khayam, S., 2011. Revisiting traffic anomaly detection using software defined networking. In: Proceedings of RAID, Menlo Park, USA, Springer, pp. 165–180.

Miao, R., Yu, M., Jain, N., 2014. Nimbus: cloud-scale attack detection and mitigation. In: Proceedings of SIGCOMM, Chicago, IL, USA, ACM, pp. 121–122.

Mininet: an instant virtual network on your laptop. Website. ⟨http://mininet.org/⟩.

Mirkovic, J., Reiher, P., 2014. A taxonomy of ddos attack and ddos defense mechanisms. ACM SIGCOMM Comput. Commun. Rev. 34 (April (2)), 39–53.

Xiao, Peng, Qu, Wenyu, Qu, Heng, Li, Zhiyang, 2015. Detecting ddos attacks against data center with correlation analysis. Comput. Commun. 67 (August (1)), 66–74.

Rumelhart, D., Hinton, G., Williams, R., 1986. Learning representations by back-propagating errors. Nature 323 (October (2)), 533–536.

Ryu sdn framework. Website. ⟨http://osrg.github.io/ryu/⟩.

Sezer, S., Scott-Hayward, S., Chouhan, P., Fraser, B., Lake, D., Finnegan, J., Viljoen, N., Miller, M., Rao, N., 2013. Are we ready for sdn? Implementation challenges for software-defined networks. IEEE Commun. Mag. 51 (July (7)), 36–43.

Shin, S., Gu, G., 2013. Attacking software defined networks: a first feasibility study. In: Proceedings of HotSDN, Hong Kong, China, ACM, pp. 165–166.

Shin, S., Yegneswaran, V., Porras, P., Gu, G., 2013. Avant-guard: scalable and vigilant switch flow management in software-defined networks. In: Proceedings of CCS, Berlin, Germany, ACM, pp. 413–424.

Software-defined networking: the new norm for networks. Website. ⟨https://www.opennetworking.org/⟩.

The openflow specification version 1.0.0. Website. ⟨https://www.opennetworking.org/⟩.

Twycross, J., Williamson, M., 2003. Implementing and testing a virus throttle. In: Proceedings of the 12th Conference on USENIX Security Symposium, Washington, DC, USA, USENIX, pp. 20–20.

Vlachos, M., Meek, C., Vagena, Z., Gunopulos, D., 2004. Identifying similarities, periodicities and bursts for online search queries. In: Proceedings of SIGMOD, ACM, pp. 131–142.

Wang, B., Zheng, Y., Lou, W., 2015. Ddos attack protection in the era of cloud computing and software-defined networking. Comput. Netw. 81 (April (22)), 308–319.

Wang, Y., Chen, H., Wu, X., Shu, L., 2016. An energy-efficient sdn based sleep scheduling algorithm for wsns. J. Netw. Comput. Appl. 59 (January), 39–45.

Yan, Q., Yu, F., 2015. Distributed denial of service attacks in software-defined networking with cloud computing. IEEE Commun. Mag. 53 (April (4)), 52–59.

Zhang, H., Reich, J., Rexford, J., 2015. Packet traceback for software defined networks (Master's thesis). Princeton University.

Zhu, Y., Shasha, D., 2003. Efficient elastic burst detection in data streams. In: Proceedings of SIGKDD, ACM, pp. 336–345.