REV Journal on Electronics and Communications, Vol. 6, No. 3–4, July–December, 2016

Regular Article

A Multi-Criteria based Software Defined Networking System Architecture for DDoS-Attack Mitigation

Tuyen Dang-Van, Huong Truong-Thu

Hanoi University of Science and Technology, Hanoi, Vietnam

Correspondence: Tuyen Dang-Van, tuyen.dangvan-set@hust.edu.vn

Communication: received 5 July 2016, revised 27 November 2016, accepted 2 December 2016

Online publication: 1 February 2017, Digital Object Identifier: 10.21553/rev-jec.123

The associate editor coordinating the review of this article and recommending it for publication was Prof. Tran Xuan Nam.

Abstract– **Nowadays, Software-Defined Networking (SDN) has become a promising network architecture in which network devices are controlled in a separate Control Plane (i.e., SDN controller). In a specific aspect, employing SDN in a network offers an attractive network security solution due to its flexibility in building and adding more new software security rules. From another perspective, attack prediction and mitigation, especially for Distributed Denial of Service (DDoS) attacks, are still challenges in SDN environments since a SDN control system works probably slower than a non-SDN one and the SDN controller can become a target of attacks. In this article, at first, we analyze a real traffic use case in order to derive DDoS indicators and thresholds. Secondly, we design an Openflow/SDN-based Attack Mitigation Architecture that is able to quickly mitigate DDoS attacks on the fly. The design solves the existing problems of the Openflow protocol, reducing the traffic volume traversing over the interface between the data plane (switch) and the control plane (SDN controller) and decreasing the buffer size at the Openflow switch. Applying our proposed Fuzzy Logic-based DDoS Mitigation algorithm that deploys multiple criteria for DDoS detection - FDDoM, the system demonstrates the ability to detect and filter** 97% **of attack flows and reach a False Positive Rate of** 5% **that are acceptable figures in real system management. The results also show that the network resource which is required to cope and maintain flow entries is** 50% **reduced during attack time.**

Keywords– **OpenFlow/SDN, DDoS attack, Fuzzy Logic.**

1 Introduction

Network security has become an international critical concern as we find more and more types of attacks that harass networks. Within the security landscape, Distributed Denial of Service (DDoS) has become the most common and major threat to the Internet at present. In this type of attacks, many hosts controlled by attackers combine to send extremely huge volume of seemingly legitimate network traffic to request services from victims. Therefore, it consumes computer resources of the victims. This causes bandwidth be congested, and the processing capability of network systems and servers be exhausted. There are many proposed solutions for detecting and mitigating DDoS attacks. These solutions are divided into two groups: signature based techniques [1–3] and anomaly based techniques [4–7]. The solutions in the former group detect attacks by comparing incoming traffic with stored attack samples; hence they are inappropriate for detecting new DDoS attack methods. The techniques in the latter group require an exclusive device for collecting and analyzing traffic data and applying statistical analysis or machine learning methods. Some solutions are used just in the offline mode to analyze in order to find out the original attack source after the attack has occurred.

In another aspect, Software-Defined Networking (SDN) [8] is a prominent future network model nowadays. In the SDN architecture, the control plane is separated from the data plane that can provide network operators with the ability to easily monitor, control, manage and configure network resources and network states through software executing in the controller. OpenFlow/SDN [9] has come to the aforementioned scene as a promising protocol used for communicating between the control plane (controller) and the data plane (OpenFlow switch). It allows the Controller to send configuration messages to and receive messages from OpenFlow switches. As a result, administrators can centrally monitor, collect network traffics and device statuses in an easy way. Besides, in the SDN architecture, OpenFlow switches can provide network traffic statistical parameters, which are useful for security applications such as DDoS attack mitigation. Another competitive advantage of SDN is that it can provide an ability to process packets dynamically and flexibly, differently from the configuration-fixed way in conventional networks.

Within the SDN context, most of recent solutions for DDoS detection and mitigation follow the method of collecting and analyzing traffic characteristics from flow entries existing in SDN switches, then creating control policies. The disadvantage of these solutions within the SDN system context is that the controller must frequently query the information of flow entries existing in switches, resulting in high bandwidth demand between the switch and the controller. Besides, the capacity of the controller is actually reduced since the controller must do the security job beside other jobs (e.g., to analyze traffic, to make security policies). The bandwidth interface between the controller and the switch can easily get saturated when DDoS appears in the network. Therefore, this bandwidth saturation is an important issue for SDN system deployment since attackers may make use of it to flood not application servers but the controller instead.

In this article, we propose a novel SDN-based network architecture with deployment of the FDDoM (Fuzzy-based DDoS Mitigation) algorithm at a separate device called Security Device (SD). The architecture has capability of monitoring and analyzing statistical parameters of incoming flows on the fly. FDDoM provides simple detection and fast mitigation since it does not rely on learning from a huge database but only on given thresholds. Our design architecture also outperforms the current Openflow/SDN system by reducing traffic volumes traversing through the interface between the SDN controller and the Openflow switch (OFS), and decreasing the demanded buffer size at OFS, by which we can mitigate the aforementioned bandwidth saturation problem.

The rest of the article is structured as follows. Section 2 describes related works. Section 3 focuses on analysis of traffic collected from NetNam, which is one of the largest ISPs in Vietnam, to derive DDoS indicators for the use case. In Section 4, we propose a multi-criteria based SDN architecture for mitigating DDoS attacks. The designed mitigation algorithm, FDDoM, is elaborated in Section 5. The performance of such a system is analyzed in Section 6. Finally, conclusion and future work are given in Section 7.

2 Related Work

The common approach for DDoS-attack detection and mitigation is collecting traffic characteristics before applying different algorithms in order to confirm whether an attack is taking place or not. Each time the system detects an attack, throughout the policies setting at firewalls or prevention devices, the attack traffics are identified and dropped [10]. Prevention plays a key role in fighting against DDoS attacks. Currently, the majority of the existing DDoS Prevention solutions, such as entropy prevention presented in work [11] and [12], source address distribution described in [13] and [14], and activity profiling in [15], depends on the characteristics of particular DDoS attacks. Consequently, attackers can fool them in an easy way. For instance, attackers may spoof source IP addresses by using some common tools. But if distributed source addresses created by attackers are the same as in reality, the source address distribution method will not be helpful for detecting DDoS attacks.

The works in [16, 17] present other solutions called Self-Organizing Maps (SOM), which is a machine learning technique for DDoS Prevention. In these works, six statistical parameters of a flow collected from OpenFlow switches are the inputs to the SOM training process. However, SOM training requires hours to be completed, and huge matrix computations and training table for the machine learning technique.

Several other recent works dealt with the application-layer DDoS attacks, such as [18–20]. The work in [18] presents a solution for DDoS in the application layer wherein the authors proposed a solution based on Adaptive Selective Verification to defend an attack that targets a particular application of a server. The work in [19] copes with a set of algorithms to block attacks while allowing legitimate user traffic, including flash traffic which can only be differentiated by application-layer methods. The authors in [20] also developed a solution based on the machine learning based random-tree method and traffic authentication in order to mitigate undetected malicious traffic that mimicks legitimate traffic.

Within the SDN context, there have been multiple proposed DDoS detection and mitigation solutions. Most of them rely on the process of collecting traffic characteristics from flow entries existing in SDN switches in order to issue control policies such as drop or forward packets by creating corresponding flow entries [17, 21, 22]. A disadvantage of these solutions is that the controller must frequently query the information of flow entries existing in the switches, resulting in high bandwidth between the switches and the controller. Besides, the process of analyzing traffic and issuing security policies implemented at the controller actually reduces the controller capacity. When a DDoS attack appears, the controller capacity and the bandwidth between the controller and the switches can easily get saturated.

In [23], the authors proposed a solution called AVANT-Guard which detects and mitigates TCP SYN based on SYN Cookie right in the OF switch. With this mechanism, only complete TCP connection is forwarded to the controller and allowed to connect to a server. A disadvantage of AVANT-Guard is that it mitigates only the TCP SYN attack. Moreover, for each TCP packet being migrated to the switch, they need to process a step of translating the sequence and the ACK number. Therefore, the switch has to maintain states of each flow in order to do this translation step, resulting in more resource consumption and reducing performance of the OF switch.

The work in [24] proposed the solution called LineSwitch which is an extension of AVANT-Guard in which the OF switch maintains 2 lists of legitimate IP addresses and recently suspected IP addresses. Once there is a new connection, if the IP address exists in the list of the legitimate IP addresses, the packet will be forwarded immediately or controlled by the SYN mechanism like AVANT-Guard with a given probability p. If existing in the suspecting IP list, the packet will be dropped and its IP address continues to be monitored in the following duration T. A strong point of this solution is that it can be applied for other types of attacks other than TCP SYN Flood. But a weak point is that the system must maintain 2 lists, which increase time for packet processing. When a DDoS attack takes place, the 2 lists will quickly occupy the switch memory, resulting in switch saturation.

Overall, common weak points of current SDN DDoS detection and prevention methods are: (i) the process takes place in either the switch or controller, resulting in degradation of the whole system performance and high latency, (ii) processed traffic is transmitted via the encrypted interface between the OF switch and the controller, leading to high bandwidth occupation at this interface. This can be the Achilles's heel for such an SDN-based system since the interface can be saturated by an attack if there is no approach to decrease the traffic volume traversing over this interface.

Our proposed solution solves this issue by sending data that are unmatched with existing flow entries in OF switch to another network component called Security Device (SD) instead of to the controller in case the corresponding server is set in the suspected mode. In addition, the fact that only a useful part of packet headers is sent from the SD to the controller makes the bandwidth demand over the SD–controller interface remarkably reduced.

3 Case Study: Netnam's Traffic Analysis

In our previous work [25], we presented the analysis for traffic logged from Netnam – one of the main ISPs in Vietnam. The traffic was captured in both *Normal* and *Under-DDoS-attacked* states upstreaming to a web server and stored as log files. After analyzing the traffic characteristics from the log files, our findings in the characteristics of the legitimate and anomaly traffic are as follows:

- 70.38% of IATs (Inter Arrival Times) in the legitimate state and 97.77% of IATs in the attack state have values in the range of (0–0.2 ms],
- 9.88% of IATs in the legitimate state and 1.56% of IATs in the attack state have values in the range of (0.2–0.4 ms] and so on as shown in Figure 1.
- Flows containing only 1 packet occupy 13.18% in the normal state while hitting the peak 88.76% in the attack case.
- Flows consisting [2–10] packets account for 30.71% in the normal case while 7.8% in the attack scenario as described in Figure 2.
- The study also points out when a server is under a DDoS attack, the number of incoming flows increases very much, up to 27.5 times higher compared to the flows in case the server is in normal state.

Therefore, we can conclude the following:

- The *rate IATs in the range of* (0–0.2 *ms]* can be a DDoS attack indicator since the figures of 0.977 and 0.7038 are significantly superior to other rates of the same legitimate and attack state accordingly. The figure below 0.7 may indicate a guarantee of normal traffic while above 0.9 may indicate an absolute attack. However, the range from 0.7 to 0.9 may present a certain degree of belonging to an attack or non-attack state and depend on the other attributes.
- The *rate of* 1*-packet flows* can be another indicator for detecting DDoS attacks where the figure of equal or more than 0.9 may indicate an absolute attack and the figure of equal or smaller than 0.15 may indicate the normal traffic case. But the range in between the two figures remains uncertain to what degree the traffic belongs to the attack or legitimate state.

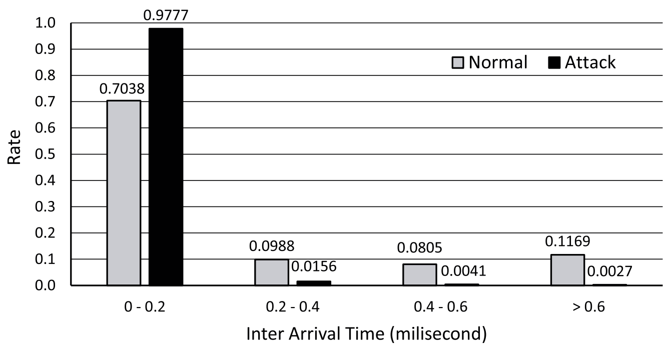

Figure 1. Histogram of packets grouped in the same IAT categories.

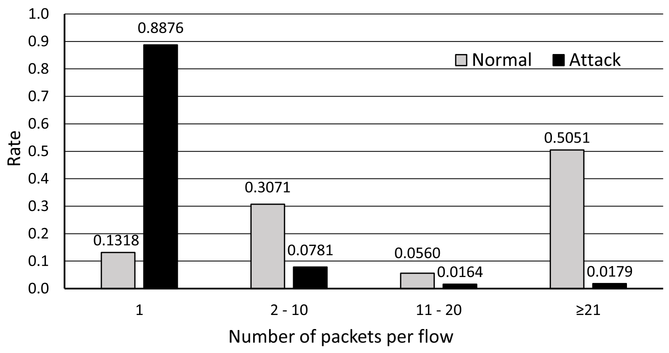

Figure 2. Histogram of flows having different packet quantity.

At the bottom line of this analysis, the knowledge of how legitimate and attack traffic looks like is a lodestar to build thresholds to detect and mitigate DDoS attacks with the FDDoM algorithm elaborated in Section 5. The creation rate of new flows connecting to a server may also be used as a signature of attack presence and therefore we use this parameter as a prerequisite for detecting a DDoS attack in Section 4.

4 Proposed System Architecture

4.1 The SDN based Network Security System Architecture for DDoS Attack Mitigation

In this section, we propose an SDN/Openflow-based network security system architecture that can detect and mitigate DDoS attacks as demonstrated in Figure 3. This architecture is designed to protect internet servers at access networks in data centers or servers inside small networks of enterprises/campuses which are connected directly to the Internet via ISPs' edge routers.

The architecture consists of an SDN Openflow switch (OFS) connecting network servers to the Internet through a core network or directly via a gateway.

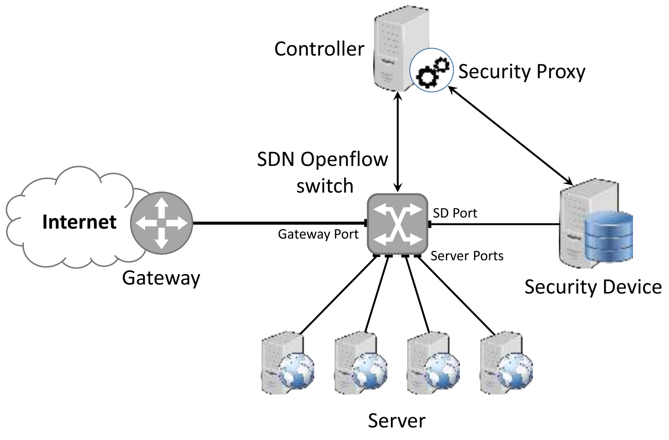

Figure 3. The proposed SDN based network security architecture.

Algorithm 1: Determining the state of server

```
IF ($nf >= S_2$) THEN
    Server_state = "Attack Mitigating"
ELSE
    IF ($nf >= S_1$ in consecutive $N$ times) THEN
        Server_state = "Suspected to Be Attacked"
    ELSE
        Server_state = "Normal"
    ENDIF
ENDIF
```

The OFS is controlled and managed by an Openflow Controller. In this architecture, a Security Device (SD) is connected to the OFS via the data port so that the SD can copy some ingress packets of coming flows for security analysis.

4.1.1 Security Device: Security Device (SD) is the main component of the SDN/Openflow security architecture, responsible for analyzing, detecting attacks and issuing packet processing policies to prevent or reduce attack harm. In order to perform these functions, SD includes a database storing traffic properties of the current flows connecting to internal servers. Depending on the security mechanism scheme, the system defines states of servers, rules to determine server states, signatures to identify and policies to mitigate attack traffic. With the scheme based on FDDoM (described in Section 5), states of each internal server may be *Normal*, *Suspected to be Attacked* (Under monitoring), or *Attack Mitigating*. The system monitors and changes states of the servers after each period T. For each server, SD maintains counters for the number of current incoming flows cf, the number of 1-packet flows opf, and the number of flows recently created within the current time period T - nf. Depending on the statistical characteristics of incoming normal traffic, administrators can set three thresholds: S_1, S_2 and N for the server. The values of cf and opf are used for calculating rate of 1-packet flows, and the nf for determining server states in next period based on the Algorithm 1.

4.1.2 Security Proxy: The Security Proxy (SP) is actually an application carried out in the Controller and responsible for creating and modifying flow entries at the OFS based on the packets-processing policies defined by the SD. In a reverse way, the SP updates information of flow characteristics in the database of the SD whenever the controller receives an asynchronous message from the OFS. For example, when the controller receives event FLOW TIMEOUT from the OFS, the SP analyzes the message and reduces, via the SD, the number of current flows connecting to the corresponding server.

4.2 Principles of DDoS Detection and Mitigation

The system monitors incoming packets from the Internet to internal servers via flow entries installed in flow tables in the OFS. Each TCP/UDP flow is determined with five parameters corresponding to five matched fields: source IP address, destination IP address, protocol number, source port and destination port. For ICMP flows, six parameters are used: source IP, destination IP, protocol number, ICMP type, ICMP code and ICMP number. Each flow entry is set with timeout values (idle timeout and hard timeout) that can be automatically removed from the flow table for saving memory when its flow no longer exists.

4.2.1 In the "Normal" state: When a server is in the *Normal* state, incoming packets from the Internet are processed and forwarded as the way of the conventional SDN/Openflow mechanism. The incoming packets are matched with existing current flow entries (CF_FEs) in the OFS, then being processed and forwarded to servers through corresponding matched flows (Steps 2 and 3 in Figure 4). If a packet can not be matched with any CF_FE, it will be sent to the controller for processing request. Consequently, the controller installs a new corresponding CF_FE to the OFS for directing the packet and following ones to the destination server (Steps 4, 5, 6 and 9 in Figure 4).

During the progress, the SD counts for the current flow number to each server cf, the new flow number nf, and the number of flow which has only one packet opf. Once the SP creates a new CF_FE, it also notifies the SD to increase cf and nf. When the controller receives a FLOW TIMEOUT message, the SP sends the information to the SD to decrease the corresponding nf counter (Steps 7, 8, 10, 11, 12 and 13 in Figure 4). If the timeout flow has only 1 packet, the SD also increases the opf counter.

As mentioned above, after each period T, the SD determines if the server should be set in the *Under Monitoring*, or *Attack Mitigating* state by comparing the numbers of new flows created within the recent period nf with thresholds as described in Algorithm 1.

4.2.2 In the "Suspected to be Attacked" state: When a server is switched to the state *Suspected to Be Attacked (Under Monitoring)*, the SD requires the SP to install and modify related flow entries to forward or copy all incoming packets to the SD for traffic analysis. In detail, the SD requires the controller via the SP to:

1) Install a flow entry NF_FE (New flow): This flow has the lowest priority in the flow table in the OFS and is set with wild card for all matched fields excepted for the destination IP address. The action of the flow is to send matched packets to the SD. When a packet incoming to a server

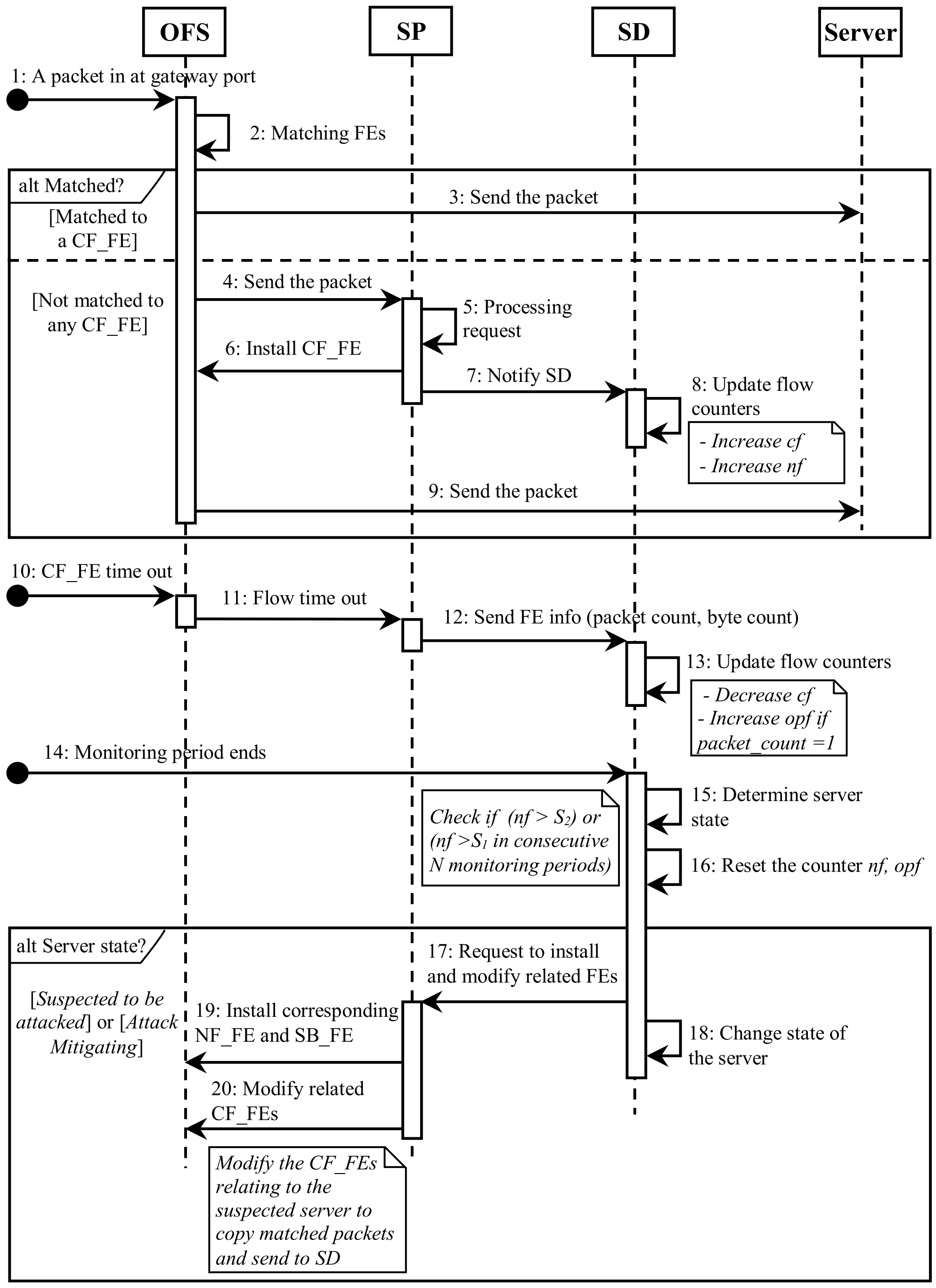
OFS
SP
SD
Server
1: A packet in at gateway port
2: Matching FEs
alt Matched?
3: Send the packet
[Matched to a CF_FE]
4: Send the packet
[Not matched to any CF_FE]
5: Processing request
6: Install CF_FE
7: Notify SD
8: Update flow counters
- Increase cf
- Increase nf
9: Send the packet
10: CF_FE time out
11: Flow time out
12: Send FE info (packet count, byte count)
13: Update flow counters
- Decrease cf
- Increase opf if packet_count =1
14: Monitoring period ends
15: Determine server state
Check if (nf > S2) or (nf >S1 in consecutive N monitoring periods)
16: Reset the counter nf, opf
alt Server state?
17: Request to install and modify related FEs
[Suspected to be attacked] or [Attack Mitigating]
19: Install corresponding NF_FE and SB_FE
18: Change state of the server
20: Modify related CF_FEs
Modify the CF_FEs relating to the suspected server to copy matched packets and send to SD

Figure 4. Sequence diagram of the system in *Normal* state.

from the Internet reaches the OFS and it is not matched with any existing CF_FE, the packet is then matched to NF_FE and sent to the SD instead of being forwarded to the controller.

2) Install a flow entry SB_FE (Send back from the SD): After security analysis, if the SD decides to send back the packet to the OFS, an SB_FE is installed for matching and forwarding the packet directly to an internal server or to other functionality flow tables in pipeline processing. To reduce matching time with other CF_FEs, the priority level of SB_FE is set to be highest in the flow tables. Differ from other CF_FEs, NF_FE and SB_FE are not set with timeout values and therefore they can exist until the server state is changed to *Normal*, and the OFS receives the message to remove these flow entries.

3) Modify related existing CF_FE flow entries (Steps 17 and 20 in Figure 4) so that matched packets are not only sent to the corresponding server but also copied and sent to the SD (Step 4 in Figure 5).

With the packets received from the OFS, after each period T, the SD analyzes the suspected traffic by calculating the two parameters: rate of 1-packet flows appearing in the period, and rate of IATs having values smaller than a given threshold (Step 5 in Figure 5). Taking these parameters as an input, the SD can issue appropriate policies by using Fuzzy Logic-based DDoS Attack Mitigation (FDDoM) algorithm deployed at the

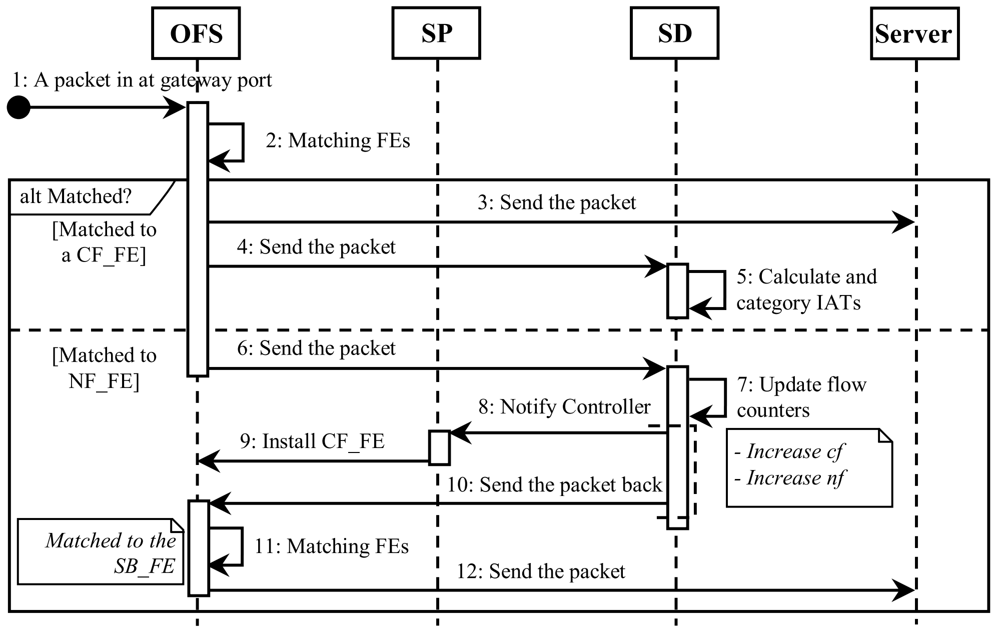

Figure 5. Sequence diagram of the *Packet in* event when a server in *Suspected to be attacked* state.

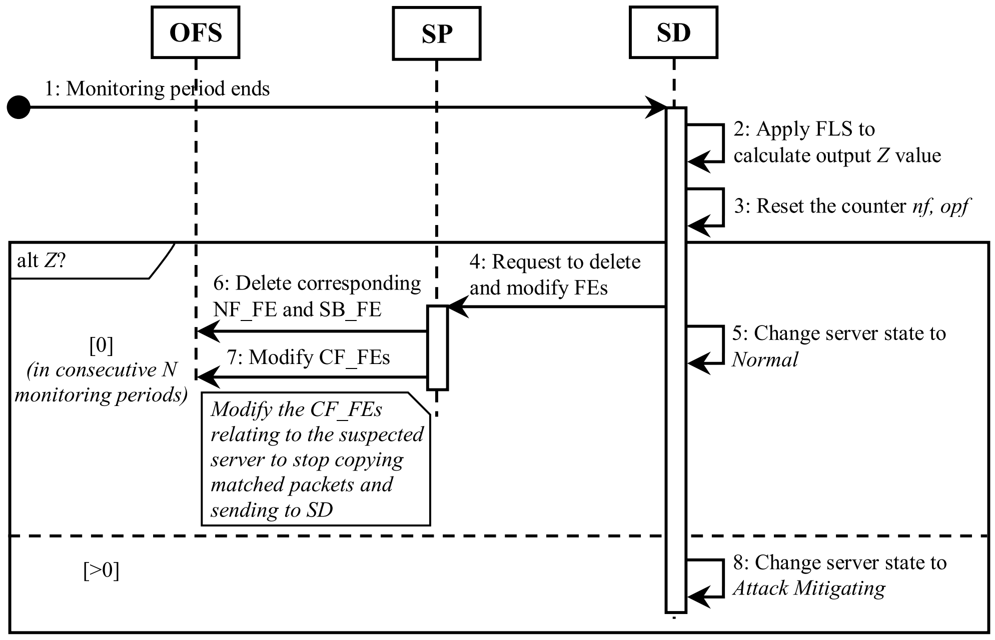
OFS
SP
SD
1: Monitoring period ends
2: Apply FLS to calculate output Z value
3: Reset the counter nf, opf
alt Z?
4: Request to delete and modify FEs
6: Delete corresponding NF_FE and SB_FE
5: Change server state to Normal
[0] (in consecutive N monitoring periods)
7: Modify CF_FEs
Modify the CF_FEs relating to the suspected server to stop copying matched packets and sending to SD
[>0]
8: Change server state to Attack Mitigating

Figure 6. Sequence diagram of an event of *Monitoring period ends* when a server in *Suspected to be attacked* or *Attack Mitigating* state.

SD. FDDoM outputs decision Z (Step 2 in Figure 6) according to the degree that a server could reach when being under attack.

- If $Z = 0$ in consecutive N monitoring periods, the system determines the server in *Normal* state. In responding to the determination, the SD requires the SP to delete NF_FE, SB_FE and to modify CF_FEs in the OFS to stop copying packets being sent to the SD (Steps 4, 6 and 7 in Figure 6).
- If $Z > 0$, the system determines server under a DDoS attack and switches to the *Attack Mitigating* state (Step 8 in Figure 6).

4.2.3 In "Attack Mitigating" state: When a server is under the state *Attack Mitigating*, based on the Z value returned by FDDoM, the SD drops rate Z of the flows going in the corresponding server. The dropping process is implemented in sequence as follows:

1) First, drop 1-packet flows up to rate Z of all flows; flows which are first created are dropped first.
2) Besides, newly created flows are checked if they can be forwarded to the server (Step 8 in Figure 7).
3) If the number of current flows going in a server excesses the server capacity, the corresponding new flow entry is not created and the correspon-ding packets are discarded.
4) If the number of current flows to a server is under the acceptable level, the new flow entry corresponding to the packet is created; and the packet is forwarded to the server (Steps 9 to 13 in Figure 7).

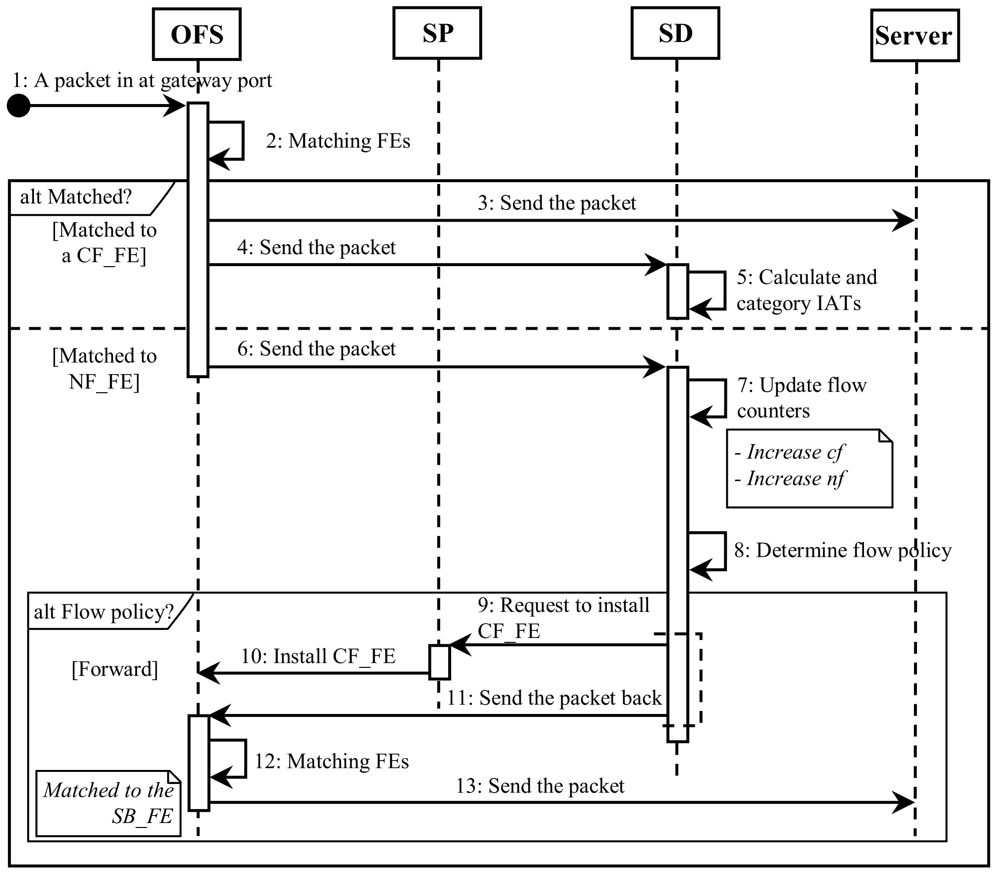

Figure 7. Sequence diagram of the *Packet in* event when a server in *Attacked Mitigating* state.

5 Fuzzy logic-based DDOS Mitigation Algorithm (FDDoM)

As stated at the beginning of this article, to detect possible attacks, network administrators may have to establish a very long look-up table, which is an entangled matrix of various detection and prevention rules. This issue can be more problematic when the number of inputs (rules) increases or values of inputs vary in a continuous domain. In that situation, an administrator may have to pay a lot of effort in determining mitigation rules from all combinations of the inputs in the look up table. From this intangibility perspective, fuzzy control may be a right solution to fix that problem since fuzzy control can make the detection implementation much simpler [26, 27] by just adjusting thresholds other than creating a look-up table of rules. In this article, we propose the Fuzzy Logic-based DDoS Mitigation (FDDoM) algorithm to detect attacks and decide the policy of dropping incoming traffic to mitigate the attack effects. Based on multiple criteria (e.g., the tw criteria defined in Section 3), FDDoM determines the degree at which the traffic belongs to an attack. FDDoM is implemented in the SD of the proposed SDN architecture.

FDDoM is built based on the well-known Sugeno Fuzzy Inference System method [27] to form rules as well as to de-fuzzy the outputs. Our contribution is that we define the shape of memberships in the fuzzification step (clause B) and the rules (clause C) that results in our detection and mitigation scheme and performance.

5.1 Choose the Inputs as Detection Indicators

Based on the traffic analysis of Netnam in Section 3, the two appropriate detection indicators can be:

- IAT: rate of packets having IATs in range (0–0.2 ms],
- PpF: rate of flows having only one packet per flow.

The FDDoM mitigation rules can be linguistically summarized as follows:

1) If traffic has the two indicators (IAT and PpF) which fall below our predefined normal thresholds, then the traffic is defined as absolute normal traffic and will be 100% forwarded (or dropping rate $Z = 0$).
2) When the two indicator values go beyond the predefined attack thresholds, traffic is detected as absolute attacks. Traffic within this monitoring window will be dropped totally ($Z = 1$).
3) However, when both indicators or one of them fall in the range between absolute normal and absolute attack thresholds, then the traffic has a certain degree of belonging to attack and another degree of belonging to the normal traffic. The system will then drop rate Z of the incoming flows to reduce the risk of being attacked. The figure Z (range from 0 to 1) will be calculated by FDDoM in three main steps: Fuzzification, Rule Evaluation and Defuzzification.

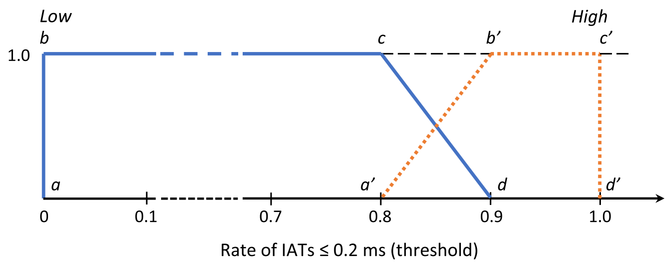

Figure 8. Fuzzification of IAT.

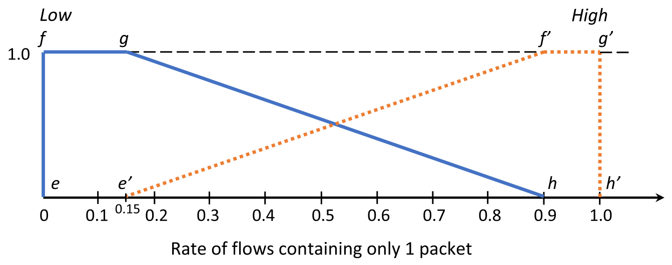

Figure 9. Fuzzification of PpF.

5.2 Fuzzification of the Selected Inputs

The SD fuzzifies the two crisp inputs (IAT, PpF) into two fuzzy numbers. In order to realize this step, we have to define the fuzzy shapes which reflect fuzzy memberships to which we map non-fuzzy input values (IAT, PpF).

In this case, IAT and PpF can be qualified in linguistic terms of *Low* and *High*. It means that based on the two attributes, traffic can be classified in either a low or high volume, or some state in between that shows the degree of traffic belonging to the attack or normal category. We define membership functions to quantify a linguistic term of *Low* or *High* which are the set of decomposition for the linguistic variables IAT and PpF. And the two inputs IAT and PpF are fuzzified as discribed in Figures 8 and 9.

As illustrated in these figures, fuzzification membership *Low* is presented in blue while *High* in red. We define the two memberships in the trapezoidal shapes since the shape can appropriately presents the membership characteristics such as:

- When the IAT and PpF values are below 0.8 and 0.15 respectively, they are considered to absolutely belong to the *Low* membership, therefore incoming traffic shall be detected as legitimate traffic and being 100% forwarded through the switch.
- When the IAT and PpF values are above 0.9, they are considered as to absolutely belong to the *High* membership and therefore traffic shall be detected as malicious traffic and being 100% dropped.
- The cross-region of the *Low* and *High* membership presents the ranges in which we are not sure if traffic is normal or attack. In other words, traffic has a certain degree of belonging to the *Low* membership and has another degree of belonging to the *High* membership, where the thresholds such as 0.8, 0.9 and 0.15 (the vertices of the trapezoids) shown in Figures 8 and 9 are derived from the traffic analysis of Netnam elaborated in Section 3. That range is where the Fuzzy Logic comes into play as it will decide a degree of the traffic belonging to a membership and decide how many percent of flows should be dropped (Rate Z).

Based on the defined fuzzification of the inputs (i.e., the two trapezoids), the degrees of the inputs belonging to either the *Low* or the *High* membership are expressed as follows:

$$F_{\text{Low}}(IAT) = \max[\min(\frac{IAT-a}{b-a}, 1, \frac{d-IAT}{d-c}), 0], \quad (1)$$

$$F_{\text{High}}(IAT) = \max[\min(\frac{IAT-a'}{b'-a'}, 1, \frac{d'-IAT}{d'-c'}), 0], \quad (2)$$

$$F_{\text{Low}}(PpF) = \max[\min(\frac{PpF-e}{f-e}, 1, \frac{h-PpF}{h-g}), 0], \quad (3)$$

$$F_{\text{High}}(PpF) = \max[\min(\frac{PpF-e'}{f'-e'}, 1, \frac{h'-PpF}{h'-g'}), 0], \quad (4)$$

where a, b, c, d, e, f, g, h and $a', b', c', d', e', f', g', h'$ are the vertices of the 2 trapezoids for 2 inputs; IAT and PpF are any considered crisp input values in the X axis.

5.3 Rule Evaluation

In this stage, the control rules in the SP upon receiving information of crisp inputs (IAT and PpF) from the Security Device, referred to as *"actions"*:

- $Action = Fw$ (i.e., Forwarding all flows).
- $Action = Dr$ (i.e., Dropping rate of Z of flows).

Then the rules are formed as follows:
Rule 1: If $IAT = Low$ AND $PpF = Low$, then $Z = Fw$.
Rule 2: If $IAT = High$ AND $PpF = High$, then $Z = Dr$.
Rule 3: If $IAT = High$ AND $PpF = Low$, then $Z = Dr$.
Rule 4: If $IAT = Low$ AND $PpF = High$, then $Z = Dr$.

Each rule weighs its output level by the firing strength of the rule, W_i, which are computed as follows:

$$W_1 = \min[F_{\text{Low}}(IAT), F_{\text{Low}}(PpF)], \quad (5)$$

$$W_2 = \min[F_{\text{Low}}(IAT), F_{\text{High}}(PpF)], \quad (6)$$

$$W_3 = \min[F_{\text{High}}(IAT), F_{\text{Low}}(PpF)], \quad (7)$$

$$W_4 = \min[F_{\text{High}}(IAT), F_{\text{High}}(PpF)], \quad (8)$$

where F_{Low} and F_{High} are the membership function of input IAT and PpF.

Classes of the two output actions can be assigned as:

$$Dr = 1; \qquad Fw = 0. \quad (9)$$

5.4 Defuzzification

Using the Sugeno FIS, the resulting output (action) is a mathematical combination of the rule strengths (degrees of applicability) and the outputs (actions). The final crisp output of the system can be defined as the weighted average of all rule outputs, computed as:

$$Z = \frac{\sum_{i=1}^{N} W_i Z_i}{\sum_{i=1}^{N} W_i} = \frac{W_1.Fw + W_2.Dr + W_3.Dr + W_4.Dr}{W_1 + W_2 + W_3 + W_4}, \quad (10)$$

where N is the number of rules ($N = 4$).

This step helps result in a crisp output Z that ranges from 0.0 to 1.0. It tells the system to drop a rate of Z of incoming flows based on each monitored specific set of crisp inputs.

6 Discussion and Performance Analysis

6.1 Discussion

With the architecture and principle operation described in Section 4 and 5, our solution has the following three advantages.

6.1.1 Reduction of traffic over the OFS-Controller interface: According to the current SDN principle, unmatched (table-missed) packets are forwarded to the controller. But when an attack happens, the traffic volume over this interface gets much higher, making this interface saturated. Besides, since the traffic over this interface is encrypted, the increased traffic volume also consumes more resources of the OFS and controller for encryption and decryption. Attackers may make use of this weak point to attack the system by generating a huge volume of packets belonging to different flows that then creates plenty of *Packet-in* events in the system. When our solution is applied, the whole traffic is only transmitted over this interface when a server is considered in the normal state. When a server is switched to the state of *Suspected to Be Attacked* or *Attack Mitigating*, the whole table-missed traffic is forwarded to the SD. At the SD, a part of table-missed packets are discarded (i.e., not being forwarded to the controller) if the server is determined under attack. Consequently, traffic destined to the server is reduced. In addition, the SD does not send the whole header of a packet to the controller, but send the useful part of the header, such as source IP address, destination IP address, source port, and destination port. This makes the encrypted traffic traversing between the SD and the controller significantly reduced in comparison with data in the system without this solution.

6.1.2 Fast Response Capacity: In the other approaches like [21, 22], the process of analyzing traffic and issuing packet-processing policies is implemented at the controller. After each monitoring duration T, the controller must query flow tables to retrieve data parameters, analyzing and detecting an attack. In responding to detecting an attack, flow entries are created in the OFS with the corresponding action: drop ingress packets. This mechanism not only increases traffic over the OFS-Controller interface since the system needs to query/create flow entries but also makes the system to respond slow during the attack. With our proposed solution, the policy of processing packets is implemented right at the SD. When packets are forwarded to the SD, if a server is determined to be under attack, the packets can be discarded immediately without any policy of processing the packets at the OFS. This approach also overcomes another common issue: process 1-packet flows. A common DDoS attack technique is IP spoofing when an attacker generates a large number of 1-packet flows to a victim server. After receiving the first and also the only one packet of a flow, a corresponding flow entry will be created in the OFS and last until the idle timeout is over. If the controller continues creating flow entries, these flow entries keep existing in the OFS during another idle timeout without any processing policy since no 2nd packet of the flow arrives. Our solution overcomes this issue. When there is a sign of only 1 packet, the SD requires the controller to discard the corresponding flow entry in the switch (OFS).

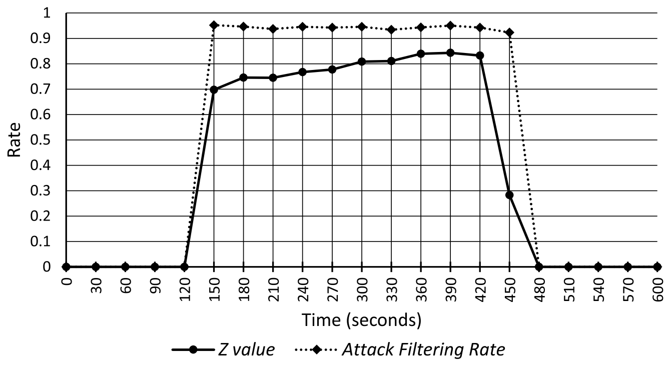

Figure 10. Z values output of FuzzyLogic vs. Attack Filtering Rate.

6.1.3 Buffer reduction in OFS: With the existing conventional SDN operation, once an arriving packet does not match with existing flow entries in the OFS, a part of the packet is forwarded to the controller while its whole content is buffered in the OFS. In case the resource of the OFS becomes exhausted, the whole payload of the packet is forwarded to the controller. During DDoS attacks, the number of table-missed packets dramatically increases, resulting in fast OFS buffer exhaustion. From this perspective, our proposal has advantage that it requires no buffer usage during the *Suspect to Be Attacked* or *Attack Mitigating* state since packets are forwarded directly to the SD.

6.2 False Positive Rate and Filtering Rate Performance

In this subsection, we verify the performance of FDDoM by checking the False Positive Rate. False Positive Rate is popularly defined as any normal or expected behavior that is identified as anomalous or malicious. The false positives evaluation is carried by computing the rate of legitimate flows misdropped by FDDoM. In order to realize the purpose, we apply the FDDoM scheme in the system to analyze Netnam traffic dataset for 10 minutes with monitoring period T of 30 seconds. The traffic scenario is set up as follows: during the first stage of 120 seconds (4 window samples), there was only legitimate traffic. In the next stage from second 120th to 420th, we sent both normal and attack traffic to the server. Beyond the second 420th up to second 600th, we got only normal traffic back again.

During each monitored window, based on the traffic statistical characteristics, FDDoM will calculate Z values to kill rate Z of incoming flows in order to reduce the attack risk.

As Figures 10 and 11 illustrate, during the period of having only normal traffic (i.e., from second 0th to 120th, and from second 480th to 600th), FDDoM decides to drop a rate of 0 of the incoming flows. However, during the period of having both normal and attack (i.e., from second 120th to 450th), FDDoM always return dropping rates (Z values) approximately to 1. Figure 11 shows that false positive killing appears only within the attack

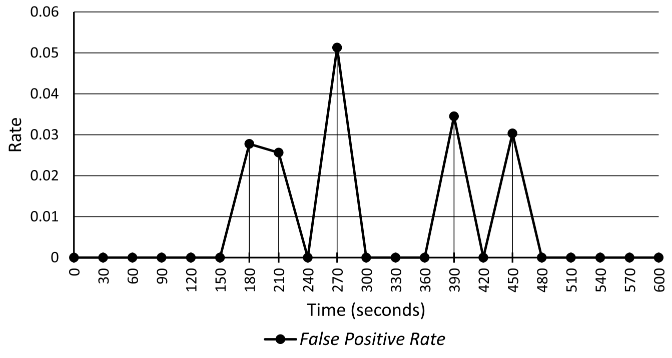

Figure 11. False Positive Rate performance.

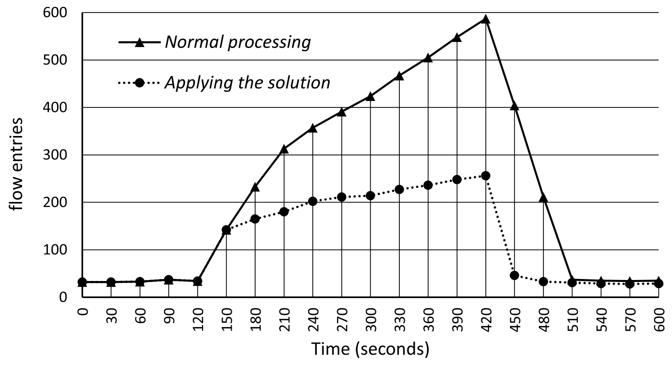

Figure 12. Comparison of existing flow entries in OFS.

duration with the peak rate of below 0.05 outside the attack duration, FDDoM can recognize normal traffic exactly by returning Z value of 0 (i.e., the false positive rate is equal to 0).

In conclusion, if we consider the performance in each period:

- The Attack filtering Rate reach 92% to 95%;
- The False Positive Rate is limited by 5.1%.

If we consider the performance in the whole course:

- The Attack filtering Rate reaches 97%;
- The False Positive Rate is bounded by 4%.

To the best of our knowledge, those performance figures can be accepted in a real system.

6.3 Reduction of attack flow entries in the OFS

Since most of attack flows have only one packet, these flow entries occupy the memory of a server once created in the OFS, leading to waste of the resource of the OFS. Moreover, the processing of matching a packet takes longer, resulting in lower efficiency of the switch. When our solution is applied, the flow entries of attack flows are removed, mitigating the attack impact. As Figure 12 shows, the number of flows entries in the OFS is reduced up to 50% during attack.

7 Conclusion

In this article, we have presented our design of an Openflow/SDN system architecture that is capable of dealing with DDoS attacks. The architecture owns both the control flexibility of an SDN system and the efficiency of mitigating DDoS malicious traffic. With our designed FDDoM algorithm based on our findings on the Netnam traffic characteristics, the mitigation performance is proved to work efficiently with high filtering rate while the false positive rate is kept reasonably low (approximately 5%). This design can be used for other use cases by finding other thresholds for other traffic scenarios and adjusting the vertices of the trapezoids in FDDoM accordingly to achieve the same performance results. In future, we vision to extend the Openflow protocol more to achieve robuster and faster performance of the OFS and the controller since the SDN system is actually slower than a non-SDN system.

Acknowledgment

This study was funded by the National Project No. DTDLCN.38/15 on "Study and develop a system for traffic monitoring and DDoS detection based on Software Defined Networking".

References

[1] C. Jin, H. Wang, and K. G. Shin, "Hop-count filtering: an effective defense against spoofed DDoS traffic," in *10th ACM conference on Computer and communications security*, 2003, pp. 30–41.

[2] J. Ashraf and S. Latif, "Handling intrusion and DDoS attacks in Software Defined Networks using machine learning techniques," in *IEEE National Software Engineering Conference (NSEC)*, 2014, pp. 55–60.

[3] S. X. Wu and W. Banzhaf, "The use of computational intelligence in intrusion detection systems: A review," *Applied Soft Computing*, vol. 10, no. 1, pp. 1–35, 2010.

[4] Y. Kim, J. Y. Jo, and K. K. Suh, "Baseline profile stability for network anomaly detection," *International Journal of Network Security*, vol. 6, no. 1, pp. 60–66, 2008.

[5] O. Salem, S. Vaton, and A. Gravey, "An efficient online anomalies detection mechanism for high-speed networks," in *IEEE/IST Workshop on Monitoring, Attack Detection and Migitation (MonAM)*, 2007.

[6] M. Thottan and C. Ji, "Anomaly detection in IP networks," *IEEE Transactions on signal processing*, vol. 51, no. 8, pp. 2191–2204, 2003.

[7] R. Saurabh and B. Anup, "DDOS Attacks on Network; Anomaly Prevention using Statistical Algorithm," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 2, December, 2012.

[8] *Software Defined Networking Definition*. [Online]. Available: https://opennetworking.org/sdn-resources/sdn-definition. Accessed: 11 May 2016.

[9] *OpenFlow: Enabling Innovation in Campus Networks*. [Online]. Available at: http://archive.openflow.org/documents/openflow-wp-latest.pdf. Accessed: 2 May 2016.

[10] K. G. et al. Dileep, "A Survey on Defense Mechanisms countering DDoS Attacks in the Network," *International Journal of Advanced Research in Computer and Communication Engineering, ISSN: 2319-5940*, vol. 2, no. 7, pp. 2599–2606, July, 2013.

[11] X. Ma and Y. Chen, "DDoS detection method based on chaos analysis of network traffic entropy," *IEEE Communications Letters*, vol. 18, no. 1, pp. 114–117, 2014.

[12] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred, "Statistical approaches to DDoS attack detection and response," in *DARPA Information Survivability Conference and Exposition*, vol. 1, 2003, pp. 303–314.

[13] Z. Duan, X. Yuan, and J. Chandrashekar, "Controlling IP spoofing through interdomain packet filters," *IEEE*

Transactions on Dependable and Secure Computing, vol. 5, no. 1, pp. 22–36, 2008.

[14] F. Yi, S. Yu, W. Zhou, J. Hai, and A. Bonti, "Source-Based Filtering Scheme against DDOS Attacks," *International Journal of Database Theory and Application*, vol. 1, pp. 9–20, 2008.

[15] D. Moore, C. Shannon, D. J. Brown, G. M. Voelker, and S. Savage, "Inferring internet denial-of-service activity," *ACM Transactions on Computer Systems (TOCS)*, vol. 24, no. 2, pp. 115–139, 2006.

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

[17] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in *35th IEEE Conference on Local Computer Networks (LCN)*, 2010, pp. 408–415.

[18] Y. G. Dantas, V. Nigam, and I. E. Fonseca, "A selective defense for application layer DDoS attacks," in *IEEE Joint Intelligence and Security Informatics Conference (JISIC)*, 2014, pp. 75–82.

[19] S. Sivabalan and P. Radcliffe, "A novel framework to detect and block DDoS attack at the application layer," in *IEEE TENCON Spring Conference*, 2013, pp. 578–582.

[20] J. D. Ndibwile, A. Govardhan, K. Okada, and Y. Kadobayashi, "Web Server protection against application layer DDoS attacks using machine learning and traffic authentication," in *39th IEEE Annual Computer Software and Applications Conference (COMPSAC)*, vol. 3, 2015, pp. 261–267.

[21] S. A. Mehdi, J. Khalid, and S. A. Khayam, "Revisiting traffic anomaly detection using software defined networking," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2011, pp. 161–180.

[22] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, "Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments," *Computer Networks*, vol. 62, pp. 122–136, 2014.

[23] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "AVANT-GUARD: Scalable and vigilant switch flow management in software-defined networks," in *2013 ACM SIGSAC Conference on Computer & Communications Security*, 2013, pp. 413–424.

[24] M. Ambrosin, M. Conti, F. De Gaspari, and R. Poovendran, "Lineswitch: Efficiently managing switch flow in software-defined networking while effectively tackling DoS attacks," in *10th ACM Symposium on Information, Computer and Communications Security*, 2015, pp. 639–644.

[25] P. Van Trung, T. T. Huong, D. Van Tuyen, D. M. Duc, N. H. Thanh, and A. Marshall, "A multi-criteria-based DDoS-attack prevention solution using software defined networking," in *International Conference on Advanced Technologies for Communications (ATC)*, 2015, pp. 308–313.

[26] Z. Kovacic and S. Bogdan, *Fuzzy Controller Design: Theory and Applications*. CRC press, 2005.

[27] M. Sugeno, *Industrial Applications of Fuzzy Control*. Elsevier, 1985.

Tuyen Dang-Van received his B.Eng. in Electronics and Telecommunications in 1999 and M.Eng. in Telecommunication Engineering in 2008 from Hanoi University of Science and Technology (HUST), Vietnam. He is currently a lecturer at the Faculty of Electronics and Telecommunications, University of Technology and Logistics, Ministry of Public Security, Bacninh, Vietnam. From 2012, he has enrolled in the PhD program in the field of Telecommunication Engineering at HUST. His research interest includes Network security, SDN networks, Quality of service, Wireless sensor networks.

Huong Truong-Thu received her B.Eng. in Electronics and Telecommunications in 2001 from Hanoi University of Science and Technology (HUST), where she is currently giving lecture and doing research. Being funded in by DAAD-the German Government Fund, she then achieved her M.Eng. in Information and Communication Systems from the Technical University of Hamburg-Harburg (TUHH), Germany, in 2004. From 2004 till 2007, she pursued her Ph.D. career at the University of Trento, Italy. From 2009 to now, her educational, research, and development work is oriented toward next generation networks, protocols and mechanism, traffic analysis, QoE/QoS measuring, green networking, SDN-based security solutions, development of Internet of Things ecosystems and applications, and deployment of new integrated multimedia services into fixed and mobile networks.