FLOODING ATTACK DETECTION AND MITIGATION IN SOFTWARE-DEFINED NETWORKING

NAN HAYMARN OO

UNIVERSITY OF COMPUTER STUDIES, YANGON

OCTOBER, 2019

Flooding Attack Detection and Mitigation in Software-Defined Networking

Nan Haymarn Oo

University of Computer Studies, Yangon

A thesis submitted to the University of Computer Studies, Yangon in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

October, 2019

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research and has not been submitted for a higher degree to any other University or Institution.

…..……………………………	…………..........……………………
Date	Nan Haymarn Oo

ACKNOWLEDGEMENTS

First and foremost, I would like to thank His Excellency, the Minister for the Ministry of Education, for providing full facilities during the Ph.D. Course at the University of Computer Studies, Yangon.

Secondly, a very special gratitude goes to Dr. Mie Mie Thet Thwin, Rector of the University of Computer Studies, Yangon, for allowing me to develop this research and giving me general guidance during the period of my study.

I sincerely would like to express my greatest pleasure and the deepest appreciation to my supervisor, Dr. Aung Htein Maw, Professor, the University of Information Technology, Yangon. Without his excellent ideas, guidance, caring, patience, and persistent help, this dissertation would not have been possible.

I would like to express my deepest gratitude to Dr. Ling Teck Chaw, Associate Professor, the University of Malaya, Malaysia, for his kindly guidance, invaluable practical advices, and fruitful ideas for the whole of my dissertation.

I would also like to extend my special appreciation and thanks to the external examiner, Professor Dr. Thandar Phyu, Director of Technology Group, ATG Company Ltd., for her patience in critical reading the thesis, the useful comments, advices and insight which are invaluable to me.

I am also very grateful to Dr. Khine Moe Nwe, Professor, and Course-coordinator of Ph.D. 9th Batch, University of Computer Studies, Yangon, for her valuable advice, moral and emotional support in my research work.

I deeply would like to express my respectful gratitude to Daw Aye Aye Khine, Associate Professor, Head of English Department for her valuable supports from the language point of view and pointed out the correct usage not only through the Ph.D. course work but also in my dissertation.

I am also very grateful to Dr. Zin May Aye, Professor, and Dr. Kyi Kyi Maw, Lecturer, from the Cisco Network Lab, for their warmly supports, kindly careness, and insightful discussions for doing the dissertation without any hardening.

My sincere thanks also go to all my respectful Professors for giving me valuable lectures and knowledge during the Ph.D. course work.

I also thank my friends from Ph.D. 9th Batch for providing support, care, and true friendship along the way.

Last but by no means least, I must express my very profound gratitude to my family for always believing in me, for providing me with unfailing support and continuous encouragement, for their endless love throughout my years of Ph.D. study and through the process of researching and writing this dissertation. This accomplishment would not have been without them.

ABSTRACT

Flooding attack is a network attack that sends a large amount of traffic to the victim networks or services with the aim of causing denial-of-service. In Software-Defined Networking (SDN) environment, this attack might not only breach the hosts and services but also the SDN controller. Besides, it will also cause disconnection of links between the controller and the switches. Thus, an effective detection and mitigation technique of flooding attack is required. Statistical analysis techniques are widely used for detection and mitigation of flooding attack. However, the effectiveness of these techniques strongly depends on the defined threshold. Defining the static threshold is a tedious job and most of the time produces a high false positive alarm. In this system, we proposed the dynamic threshold which is calculated using Modified Adaptive Threshold Algorithm (MATA). The original Adaptive Threshold Algorithm (ATA) is based on the Exponential Weighted Moving Average (EWMA) formula which produces high number of false alarms. To reduce the false alarms, the alarm signal will only be generated after a minimum number of consecutive violations of the threshold. This however has increased the false negative rate when the network is under attack. In order to reduce this false negative rate, MATA adapted the baseline traffic information of the network infrastructure. The comparative analysis of MATA and ATA is performed through the measurement of false negative rate, and accuracy of detection rate. The experimental results show that MATA is able to reduce false negative rate up to 17.74% and increase the detection accuracy of 16.11% over the various types of flooding attacks at the transport layer.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

CHAPTER 1

INTRODUCTION

Flooding attack sends an extremely amount of network traffic intending to overwhelm the target network or a particular victim server and prevent the normal connection requests from benign users. Thus, it is a common type of Distributed Denial of Service attack (DDoS). The impact of this flooding attack can bring down the target victim within a very short time. The target can be the network or servers running with traditional or advanced techniques, software-defined networking (SDN) [8, 26].

The software-defined network is a new modern network architecture decoupling logically control functions with data forwarding devices by using OpenFlow protocol [77]. As the whole SDN network is controlled by the controller, the main feature of SDN is a logically centralized control of the network. By using this feature, the DoS attack can be detected and mitigated straightforwardly [24]. Oppositely, the attackers might use the advantage of this feature by breaching the centralized controller to control the whole SDN network [79].

There are two different types of flooding attacks: protocol exploited attack, and amplification or reflection attack. The common protocol exploited attacks are SYN flooding, UDP flooding, and ICMP flooding attack. DNS amplification and NTP amplification attacks are belonging to the group of amplification or reflection attacks [6]. Among them, this system is implemented especially for the detection and mitigation of SYN flooding and UDP flooding that exploits the TCP and UDP protocol, respectively.

SYN flooding attack is one of the most common types of DoS attacks. It exploits the TCP three-way handshake procedure for interrupting and repudiating the normal network services. The attackers launch the SYN flooding attack by sending a very large number of SYN packets continuously to the victim server without waiting for any acknowledgments.

In the UDP flooding attack, the attackers send a large stream of UDP packet with the specific or random port number to the target server using a spoofed source IP address. The victim server responds to ICMP packets for the port which does not listen. As a result, the attack consumes all the network bandwidth and overloads the server to be able to disturb normal operations. In the SDN network, the UDP flooding attacks significantly increase both bandwidth consumption and the controller's CPU consumption [62].

Hence, even a few seconds of flooding attacks might breach the server. If their target victim is the SDN controller, the whole SDN network will break down due to the impacts of the attacks. Thus, SDN networks are needed to protect the flooding attack by using the effective detection and mitigation technique of the attacks.

The SDN-based DoS attacks can be detected by using many different techniques such as entropy, change point detection, machine learning, traffic pattern analysis, connection rate analysis, and integration of traffic monitoring tools and OpenFlow. Moreover, these attacks can be mitigated by dropping packets or blocking port, redirection, control bandwidth, and change network topology [6]. Effective mitigation of the attack strongly depends on the result produced from the detection techniques.

Each detection technique has its advantages and disadvantages. For instance, the entropy-based detection technique can be used in measuring the randomness of the network traffic within a given period. But it has some limitations such as only a single value that can be used to calculate the probability distribution of a feature. Similarly, although machine learning-based techniques are useful for detecting malicious activity based on the abnormal behavior of the network, the performance of these techniques is depending on the training dataset [6]. Statistical analysis-based change point detection technique is widely used among the techniques. As any statistical analysis techniques define abnormal by comparing the number of traffic statistics with a threshold value, the effectiveness of these techniques is depending on the definition of a threshold value.

The threshold can be defined statically or dynamically. The static threshold needs to find the baseline of the network traffic before setting up the threshold value. Then the value is needed to change manually by the network administrator when the network situation is changing. Hence, it is a tedious job for network administrators. Moreover, it might produce a high number of false alarms. Thus, they used the dynamic threshold instead of the static threshold.

The dynamic threshold can be calculated simply by using an adaptive threshold algorithm (ATA) [52]. The ATA calculates dynamic and adaptive threshold based on the Exponential Weighted Moving Average (EWMA) formula. According to the formula, it calculates the threshold value over the factor of both current network traffic and previous average network traffic. However, this formula often produces a high number of false alarms. To avoid raising these false alarms, the ATA algorithm raises alarm after the number of consecutive violations of the threshold. Consequently, the mitigation process will be started late for dropping abnormal traffic and the false-negative rate will be increased when the network is under the real attack.

To reduce the false-negative rate, this dissertation describes a detection technique that removes the false alarm significantly by modifying ATA based on the baseline traffic. Thus, as soon as the threshold value is violated, the alarm will be raised and the mitigation process will install drop flow rule into the source switch of the attack instantaneously.

1.1 Problem Statement

The detection and mitigation of SDN-based DDoS attacks have been proposed by using the various mechanisms such as statistical analysis (change point detection, and entropy), machine learning, traffic pattern analysis, connection rate analysis, and integration of traffic monitoring tool and OpenFlow. Although each technique has pros and cons, the widely used technique among them is the statistical analysis technique. The function of this technique is comparing the number of incoming traffic with the threshold, and defining the attack traffic when the threshold is violated by the incoming traffic.

There are two types of threshold: static and dynamic. But the weakness of the static threshold is raising a high number of false alarms and a tedious job for the network administrators. Thus, the dynamic threshold is commonly used in statistical analysis techniques. The dynamic threshold for the flooding attack can be calculated by using the adaptive threshold algorithm (ATA). This algorithm produces a high rate of detection but it also raises a high number of false alarms. Consequently, it produces a high number of false negative rates avoiding false alarms. Thus, this algorithm is modified by taking into account the baseline of the network traffic to reduce the false negative rate avoiding false alarms. The main objective of the modified adaptive threshold algorithm (MATA) is to produce the dynamic threshold value that is adaptable over the incoming traffic based on the baseline.

1.2 Motivation of the Research

The legacy defense mechanisms are not completely effective as the rising occurrence of Distributed Denial of Service (DDoS) attacks. The recent DDoS attacks on various recognized organizations are listed in Table 1.1. These attacks are targeting nearly every organization especially in financial institutions and government organizations relying on IT infrastructure and resources. As shown in Table 1.1, even such great organizations are unable to countermeasure the DDoS attacks. A new modern design of the network is needed to explore needed to explore for detecting and mitigating the attacks successfully. Software-Defined Networking (SDN) has developed for solving the growing problems of DDoS attacks.

Table 1.1 Recent DDoS Attack

Target	Date	Impacts
Spain's central bank	August 2018	Bank’s website was intermittently offline for one day [50].
Americas Cardroom (ACR)	April 2018	Poker tournaments cancelled after DDoS attacks. A series of massive DDoS attacks forcing the company to pause all running tournaments from April 24 until May 1 [21].
Sucuri (Website security firm)	April 2018	A series of massive DDoS attacks causing service outage in West Europe, South America and parts of Eastern United States [61].
GitHub’s code hosting website	February 2018	The traffic peaked at 1.3 terabytes per-second. The attack last for about 20 minutes [42].
Luxembourg government servers	February 2017	Over a hundred servers had been affected by the attack and that the attack impacted servers for more than 24 hours [27].
Dyn (Internet performance management and web application security company)	October 2016	Dyn came under attack by two large and complex Distributed Denial of Service (DDoS) attacks against their Managed DNS infrastructure for about four hours [25].

1.3 Objectives of the Research

The general objective of this system is to detect and mitigate flooding attacks at the transport layer over the SDN network most simply and effectively. The specific objectives of this system have two folds: detection and mitigation.

The main objective of the detection is to reduce the false negative rate caused by false alarms. Thus, this system intends to reduce the false alarms significantly by considering the baseline of the network infrastructure while the calculation of dynamic and adaptive threshold value. The consequent objectives are to increase the detection rate and accuracy of this system.

As this system is detecting and mitigating the flooding attacks with a non-spoof IP address, the objective of its mitigation is to prevent the reduction of the network speed during the attack by decreasing the bandwidth consumption. Since flooding attack is one type of DoS attack, the main objective is to maintain the availability of the network infected by the attack.

1.4 Focus of the Research

This research focuses on developing an intrusion detection system for monitoring the network to detect and mitigate the DDoS attacks. These focus works include the following:

- create a simple firewall application in SDN for filtering one of the types of DoS attack (i.e, Ping of Death attack)
- create a stateful forwarding application based on the existing fwd application in ONOS controller by adding the connection state inspection
- create a stateful firewall application by the combination of stateful forwarding application with the acl application (i.e. stateless firewall) in ONOS controller
- detect the SDN network by sFlow-RT analyzer with static threshold and mitigate the detected attack with the drop flow rule
- find the algorithm to calculate dynamic threshold and implement the ATA algorithm in the sFlow analyzer for detecting the SYN flooding attack
- monitor the rate of the incoming frame for differentiating the normal and attack
- modify the adaptive threshold algorithm and implement it in the sFlow analyzer

- find baseline traffic of the network infrastructure
- mitigate the detected attack by using two types of defense mechanism
- measure the percentage of performance parameters and compare the results of the two algorithms (i.e. existing algorithm and modified algorithm) with the two types of defense mechanisms.

1.5 Contributions of the Research

The combination of the operations of sFlow-RT analyzer and OpenFlow for the flooding attack detection and mitigation are contributed to reducing the overload of traffic statistics analysis in the SDN controller.

The default function of statistical analysis in sFlow-RT analyzer using static threshold might produce many false alarms. Thus, the dynamic and adaptive threshold is contributed to the traffic comparison of the analyzer.

The existing algorithm for the calculation of the dynamic and adaptive threshold, adaptive threshold algorithm, consequently increased the false negative rate while avoiding false alarms. Thus, the main contribution of this system is modifying the adaptive threshold algorithm by taking into account the baseline of the network traffic while calculating the adaptive and dynamic threshold.

Another contribution is applying the source-based defense mechanism when the mitigation system is installing the drop flow rule for the attack as this system is detecting and mitigating the non-spoofing flooding attack especially.

Moreover, depending on the attacking condition, two different types of drop flow rule installation is contributed to effectively mitigate the flooding attack. The new incoming attack is temporarily dropped by flow rule with a timeout value and the attack which has been income in this system is permanently discarded.

For practically experiment this system with the nearly like real traffic in the virtualized environment, the traffic generation is contributed to the combination of the traffic generation models in the $D-ITG$ (Distributed Internet Traffic Generator) tool. The various options in $hping3$ tool is also used for launching the flooding attack on both types of transport layer protocols: TCP and UDP.

Besides, for effectively evaluating this system, the various experimental results are produced with a different point of view by using both types of traffic capturing tool: $tcpdump$ and $wireshark$. This system also checks the number of packets passing the drop flow rule by using the Open Vswitch commands and used it in the calculation of the security performance parameters with the standard formulas.

1.6 Organization of the Research

This dissertation is organized with six chapters.

Chapter 1 describes the introduction of the dissertation with the motivation of why this proposed system is needed to implement, the problems in the area of detection and mitigation of flooding attack, and the solution to the problems, by means of the proposed system. Moreover, the contribution of this dissertation and the focus works along the dissertation are described in this chapter.

Chapter 2 describes the DDoS attack as the preliminary of the flooding attacks, the limitation of the traditional network for defending the attack, and the brief description of SDN and its benefits over the limitations of existing DDoS defense mechanisms. Besides, this chapter also presents the detection and mitigation of the DDoS attack based on the SDN architecture over the various types of mechanisms in the previous work.

Chapter 3 presents the detailed description of SDN with its architecture and the main function of SDN, flow rule installation. Furthermore, it has described the two essential theories needed in the detection of the traffic such as sampling and polling. It also presents the dynamic threshold calculation algorithms used in this dissertation. Apart from the detection theory, it has described the mitigation mechanism of the DDoS attack and the evaluation method for the network security performance.

Chapter 4 describes not only the overall system architecture of flooding attack detection and mitigation but also the detailed system architecture of it with the detection and mitigation phase separately. It also presents each phase with the two dynamic threshold algorithms.

Chapter 5 demonstrates the design and experimental implementation of the flooding attack detection and mitigation by using two scenarios representing the SYN flooding attack specifically and flooding attacks at the transport layer (i.e. SYN flood and UDP flood). Moreover, the experimental results of them are evaluated with the detection results and mitigation results.

The final chapter 6 concludes this dissertation with a description of the advantages and limitations of this system. It finally lists the works that can be furthered continue to do in the future.

CHAPTER 2

LITERATURE REVIEW

Although many techniques are proposed and deployed in the detection and mitigation of the DDoS attacks, some famous and recognized organizations were infected by the high-speed DoS attacks recently. This might be the weakness of the detection and mitigation mechanism for the DoS attacks in the traditional network. The advanced features of the SDN network such as centralized control and separate the control functions and data forwarding devices are useful in detecting and mitigating the attacks. However, SDN has some weaknesses related to the DoS attacks such as breaking down the controller, hosts connected to the SDN network, and links between the control plane and the data plane.

The purpose of this chapter is to enlighten why SDN is important in the detection and mitigation of DDoS attacks. First, the DDoS attacks are briefly described. Then, the limitations of traditional networking and the advantages of SDN for defending them are presented. The chapter also explains the reviews of the techniques used in SDN-based DDoS detection and mitigation.

2.1 DDoS Attack

A significant attack targeting the availability of victim network resources or services is known as Denial of Service (DoS). DDoS attacks are one type of DoS attacks that are launched by more than one attacker who is being distributed all over the internet and sending a large volume of malformed or legitimate packets.

The aim of the attack is repudiating the normal users from using the resources of the victim network or services. The attack traffic consumes the bandwidth of the victim network or computing resources of the victim host. Moreover, to conceal their location and identity, the attackers are using spoofed IP addresses.

Some popular examples of DDoS attacks are flooding based attacks that send an extreme amount of network traffic to overwhelm the victim. The type of flooding attacks can be divided into two categories: protocol exploited attack, and amplification or reflection attack. The common protocol exploited attacks are SYN flood, UDP flood, and ICMP flood attack. DNS amplification and NTP amplification attacks are belonging to the group of amplification or reflection attacks.

In the SYN flood attack, the attackers exploit the TCP's three-way handshaking mechanism of connection establishment process and send a large number of SYN packets continuously. The server's memory is filled with the connection requests and rejects connection requests from legitimate users eventually.

In the UDP flood attack, the attackers send a large stream of UDP packet with the specific or random port number to the target server using a spoofed source IP address. The victim server responds to ICMP packets for the port it does not listen. As a result, the attack consumes all the network bandwidth and overloads the server to be able to disturb normal operations.

In the ICMP flood attack, the attackers exploit the ICMP's ICMP_ECHO_ REQUEST packet. They send the packets with a broadcast address to the victim network with a spoofed IP address of the victim host as the source address of the packet. The attack might use an intermediate network to flood the victim. The network could be congested with a huge number of ICMP_ECHO_REPLY packets. Consequently, the server rejects the legitimate user's requests.

In the DNS amplification attack, the attackers exploit the vulnerability of the Domain Name System (DNS). They send a large number of UDP packets by using publicly available open recursive DNS servers intending to flood the victim. Thus, the attack is also known as a type of reflection attack. The attacker can increase the volume of traffic through different amplification techniques. As a result, the victim network could be affected seriously, which hinders legitimate users to get into the server. The NTP amplification attack is similar to a DNS amplification attack. It exploits the NTP server instead of the DNS server.

2.2 Limitation of Traditional Networking for Defending of DDoS Attack

The main limitations of DDoS attack defending mechanisms in the traditional network are administration cost and flexibility. Since most of the network configurations are performed manually, it is not easy to change configuration dynamically in network devices to control the suspicious network traffic of the DDoS attacks.

It is very difficult, error-prone, and time-consuming to manage the traditional network such as configuring the network security equipment like a firewall, Intrusion Detection System (IDS), Intrusion Prevention System (IPS), MiddleBoxes, and

Access Control List (ACL). Moreover, an individual network configuration in every device needs to be reconfigured when any network policy is changing. As a result, steering network traffic is not flexible in the traditional network.

In addition to the software configuration of these devices, the physical location is also important in considering network security. For example, a firewall must be located at the main gate of the network as a border of the entire network. Another huge problem deals with the traditional network is the cost of the investment and maintenance for each network device. As the future network system is changing rapidly, the traditional way of deploying network cannot meet the requirement, especially in security, manageability, and adaptability.

Moreover, the current trends of DDoS attacks indicate the limitation of traditional networking. Attackers have developed the sophisticated mechanisms for bypassing the traditional protection not only by increasing the attack in size but also by sophisticating them in methods.

The introduction of SDN has taken the solution for the above-mentioned problems of security in traditional networking. The two main features of SDN, centralized control of the whole network architecture, and decoupling the data forwarding layer and control layer are very useful to develop the SDN-based Flooding attack detection and mitigation mechanisms. Before the discussion of these mechanisms, the SDN is briefly explained.

2.3 Software-Defined Network (SDN)

Unlike traditional networking, Software-Defined Networking decouples the network control operations from the data forwarding devices by using the OpenFlow protocol. Thus, all control functions are logically moved to the centralized controller and the devices at the data forwarding layer are controlled by the controller. The main objective to do so is to overcome the limitations of a traditional network. The network architecture can be rearranged with the software program by giving the authority of network administration to the controller on the control plane. Thus, the programmability and the centralized control approaches can improve the flexibility of network operations.

OpenFlow protocol is used as the communication between the controller at the control plane and switch at the data plane. The controller installs forwarding rules into the flow table of OpenFlow enabled switches. The switch matches the incoming traffic with the flow rule and then relays the traffic according to the action defined in the flow rule such as forwarding, modifying and dropping. OpenFlow switch becomes a layer 2 forwarding switch, layer 3 switch or router, load balancer, or firewall according to the rules installed by the application running in the controller. Therefore, DDoS attack detection and mitigation application could also be developed in SDN architecture.

The attacks can be easily detected by using the features of SDN such as the concept of flow-based traffic, centralized control function, and data and control plane separation. Moreover, these features are helpful to mitigate the DDoS attack rapidly and flexibly. Although SDN architecture provides the capabilities to develop an effective countermeasure for DDoS attack, it also has some weaknesses in each layer of it, such as flow-table overflow attack to the OpenFlow switches in the data plane and DDoS attack to the controller in control plane. But this research focuses on the features for improvement of security using SDN. The next section lists the effective features of SDN for releasing the limitation of existing DDoS defense mechanisms over the traditional network.

2.4 Advantages of SDN over the Existing DDoS Defense Mechanism

The DDoS attack can be easily detected and mitigated in software-defined networking because of its new available features such as:

1) **Logically centralized control of the network:** The controller provides effective security policies for the entire network to protect the DDoS attacks because it has the visibility of the network from the global point of view.
2) **Network programmability by external applications:** Any type of algorithm for DDoS attack detection and mitigation can be programmed as an application of SDN. Thus, this feature supports the highest flexibility of network management.
3) **Software-based traffic analysis:** OpenFlow enabled switch used in SDN architecture can enhance its capabilities by using a software-based mechanism. It means that the traffic can be analyzed in real-time by any software or machine learning techniques [32].

4) **Dynamic updating of forwarding rules:** The forwarding rule for dropping, redirecting the attack traffic can be installed dynamically into the switch as soon as the attack is detected [24].

2.5 SDN-based DDoS Attack Detection

The distinctive features of SDN in the above-mentioned session can be used as the main function in DDoS attacks' detection and mitigation. In this session, the common SDN-based DDoS attack detection techniques are described with their literature reviews.

2.5.1 Statistical Analysis

An Nguyen Viet et al. [58] proposed a hardware-based defense system in SDN architecture to protect the HTTP GET Flooding attacks by using the per-URL counting mechanism. Their defense system was implemented on FPGA as an extension of the 1G NetFPGA-based OpenFlow switch.

Ping Dong et al. [17] proposed an effective detection method for detecting the DDoS attack and locating the compromised interfaces connected with the attackers. The approach firstly classifies the flow events associated with an interface, and then make a decision using the Sequential Probability Ratio Test (SPRT), which has ranged false positive and false negative error rates. Their method produces higher promptness, versatility, and accuracy when compared with the other three detection methods such as percentage, count, and entropy of the flows.

Hung-Chuan Wei et al. [62] proposed a lightweight countermeasure for the UDP flooding attack in SDN by monitoring the number of incoming or outgoing packets in a period by the controller. In other words, the controller differentiates the normal and abnormal conditions by using the number of packets. Hence, as soon as an abnormal number of packets appear, the controller can detect it instantly.

Laura Mutu et al. [40] proposed a responsive technique to UDP attacks in SDN using activating triggers at the switch level. This technique designed a traffic percentage trigger based on the proportionality of outgoing UDP traffic to overall network traffic. Moreover, they also proposed an amplification trigger based on incoming UDP traffic relative to outgoing.

2.5.1.1 Entropy

Entropy has been successfully used in measuring the randomness of the network traffic within a given time. It has also been recognized as an effective mechanism in computing the randomness of a dataset. Depending on the entropy value: high or low, the probability distribution is different. The former is represented for more dispersed probability distribution and the later signifies the concentration of a distribution. Thus, these mechanisms are used broadly for attack detection in traditional networking. These techniques are more useful in the fine-grained pattern that incapable to get the volume-based traffic. Various types of features such as the number of packets, IP addresses, and flow of network can be used for computing entropy. These mechanisms can provide low overhead in computing.

Entropy has successfully been used in the detection of DDoS attacks over traditional networking. For this reason, it has become an effective method for the detection in SDN. Giotis et al. [22] proposed an effective and scalable anomaly detection and mitigation mechanism for DDoS, worm propagation, and portscan attacks using maximum entropy value with the flow-based traffic features such as source and destination IP address and port. They also evaluated the data collection methods: native OpenFlow and sFlow on the real network traffic data by comparing CPU and flow table size usage.

The advantages of their approach were reducing data gathering with sampled via sFlow, less communication between switches and OF controllers and minimized the false-positive rates. The attacks are mitigated by installing bidirectional flow rule with drop action and higher priority than other forwarding rules. But they did not emphasize the deployment location for mitigation as their experimental topology had only one switch. The experiment was conducted with one of the two types of switches: Open vSwitch and NEC hardware switch, and NOX controller. And they used a predefined threshold value for deciding which traffic is normal or malicious. Moreover, the sampling flow rate may reduce the accuracy of their method.

To reduce communication overhead, Wang et al. [60] ran the entropy-based DDoS attack mechanism in each local edge switch locally by experimenting on the modified version of Open vSwitch in mininet emulator and FloodLight controller. They used the CAIDA dataset for testing the system and evaluated with different monitoring intervals comparing the detection rate and false-positive rate. They dropped the attacked packets by installing flow rule with drop action at the source switch of the attack. They observed that the smaller monitoring interval, the quicker detection time. Since they run the detection mechanisms in each local edge switch, their system can only detect DDoS attacks on its local network connected to the switch.

Mehdi et al. [38] implemented anomaly detection with four prominent algorithms including maximum entropy, NETAD, rate-limiting, and TRW-CB on both SOHO network and ISP network datasets. And they compared and described the accuracy, efficiency of NOX implementation on the two networks. They also evaluated the CPU usage on the NOX BOX for the home dataset. They observed that the accuracy of home networks is higher than ISP networks. Although they did not implement any mitigation mechanism for the system, they assumed the result of the detection could be used not only in the home network but also in the global network.

J. Mao et al. [37] proposed a novel joint-entropy-based DDoS detection solution with multiple features of packets. They chose the flow duration, packet length, source address, and destination port as the key features for the detection of different types of DDoS flooding attacks. They also carried out experiments with simulated campus networks based on SDN architecture.

Dingwen Hu et al. [26] proposed FADM, an efficient and lightweight framework for the detection and mitigation of DDoS attacks in SDN. They firstly collected the network traffic information through the SDN controller and sFlow agents. Then, they used an entropy-based method to measure network features. They also applied the SVM classifier to identify network anomalies. Their combination methods effectively improved the timeliness and accuracy of attack detection. For keeping the major network functionality working, they proposed an efficient attack mitigation mechanism based on the white-list and traffic migration. They also introduced the mitigation agent for blocking attack traffic in a timely fashion while forwarding the benign traffic as usual. In this way, they prevented the resource of the controller from being exhausted and allowed the legitimate users can access the network normally.

Prashant Kumar et al. [31] proposed SAFETY, a novel solution for the early detection and mitigation of TCP SYN flooding. The proposed system harnessed the programming and wide visibility approach of SDN with the entropy method for determining the randomness of the flow data. The information on entropy contains destination IP and a few attributes of TCP flags.

Houda Guesmi et al. [24] proposed an approach for the detection of DDoS attacks by the SDN controller using Fast Entropy algorithms. They used SDN capabilities and the Fast Entropy method to maintain the security of the cloud system from DDoS attacks in real-time. Fast Entropy is modified information entropy for reducing the computational time than the conventional entropy in detecting the attack. By using the SDN and fast Entropy method, the approach is efficient in the collection and analysis of the traffic, detection of DDoS in real-time, blocking attack packets and forwarding legitimate flows to the cloud provider.

Phan, The Duy et al. [20] presented an approach of DDoS attack detection in the SDN environment by using the entropy metric and considering the differences in the host role profile for suspecting under-attack state. Moreover, they dealt with the time factor in the process of collecting information. Then, a statistical method was used to investigate the flow information sent from OpenFlow switches for confirming the previous suspicion. This method could detect DDoS attacks promptly at its early stage, where the role profile of the host is taken into consideration.

Entropy-based mechanisms can be used for the detection of DDoS attack, however, they had some limitations such as only a single value can be used to calculate the probability distribution of a feature. This mechanism is very effective for analysis. But the other information related to the distribution of the analyzed feature is lost. As a result, the anomaly effects can be concealed in some cases. In the same way, this mechanism could not differentiate the different distributions with equal uncertainty. Therefore, malicious traffic without randomness will not be detected.

2.5.1.2 Change Point Detection

The most common detection technique for identifying DDoS attacks in a particular network is statistical analysis. The values of parameters representing the network traffic are either remain constant or change slowly over time for the duration of normal operation. In contrast, these parameters are not remaining constant and changing abruptly when abnormal traffic is occurred by DDoS attack. Hence, the problem of DDoS detection can be formulated as a change point detection problem.

The basic function of change point detection techniques is to detect varying in the statistical properties of the network traffic with fast detection time and a low false-positive rate. Statistical Process Control (SPC) techniques and Quality Control (QC) are a statistical mechanism for anomaly detection. These mechanisms use the statistical properties of network traffic for the estimation based on the type of original distribution. They can be divided into two types: parametric and non-parametric. SPC techniques can be used in the detection of the variation of the mean and variance of a process [13].

A non-parametric CUSUM with an adaptive threshold algorithm was used for calculating the dynamic threshold in [13]. The authors, Conti et al. proposed a comprehensive, effective and lightweight mechanism for the detection of the various types of DDoS. They used CAIDA for defining initial threshold value as baseline traffic and DARPA dataset to assess effectiveness and versatility for attack types such as smurf, Neptune, IPsweep and Portscan. They evaluated their system by comparing the variation of threshold value with CUSUM value under both normal and attack traffic. Moreover, they measured the average and standard deviation of detection time, Detection Rate (DR), False Alarm Rate (FAR), and Accuracy (ACC) for each window size under every unique combination of different values for parameters. Although this system provided an adaptive threshold for attack detection effectively, it had a little overhead for exchanging FLOW_STAT messages to get real-time traffic statistics. And mitigation mechanism was not described in this system.

Moreover, Conti et al. [14] proposed another effective framework including a lightweight SDN assisted Moving Target Defense (MTD) for protection of network reconnaissance and an efficient approach for tackling DoS attacks using Software Defined-Internet Exchange Point (SD-IXP) with one of the widely used change point detection, Exponentially Weighted Moving Average (EWMA) control chart. As the system previously proposed, they tested their system with the same dataset, and type of attacks on the network with three ASs that are emulated by using MiniNext and SD-IXP, EXaBGP-based controller. It also had a few percentages of CPU overhead for collecting data. The controller overwrote the forwarding rule for violating flows with drop action temporarily.

Bawany et al. [6] also used a modified EWMA formula for application-specific detection and mitigation of DDoS attacks over the smart city data center. In their proposed system, they considered the tolerance level of each application for deciding the value of the factor variable and mitigation methods. But they did not implement and evaluate the proposed system.

Abimbola Sangodoyin et al. [49] proposed an effective DDoS detection mechanism in SDN using throughput as an impact metric. They polled the throughput within an interval of time for determining the normal distribution of benign network data and obtaining the Confidence Interval (CI) for the normal distribution. An attack was specified by a significant deviation in mean throughput value gained at subsequent intervals compared to the without attack mean throughput. Finding the value of confidence interval and mean throughput had low overhead and could be easily implemented in the SDN controller for detecting the anomaly.

2.5.2 Machine Learning

Traditional IDS applied machine learning-based mechanisms such as Bayesian networks, self-organizing map (SOM), Artificial neural networks, and fuzzy logic principles and concepts for detecting the anomalies on both wired and wireless network. These mechanisms are also widely applied in SDN-based DDoS attack detection. A machine learning-based mechanism distinguished the normal and attack traffic based on the features of the traffic.

Braga et al. [8] implemented a lightweight DDoS attack detection mechanism with SOM based on traffic flow features such as Average of Packets per flow (pdf), Average of Bytes pre flow (ABf), Average of Duration per flow (ADf), Percentage of Pair-flows (PPf), Growth of Single-flows (GSf), and Growth of Different Ports (GDP). The flow collector module in NOX controller collected the features, the feature extractor module extracted the important information for DDoS detection from the collected features, and then the classifier module analyzed the extracted information corresponding to the attack or legitimate traffic by using SOM. This mechanism provided a high rate of detection and a low rate of false alarm. Moreover, it also incurred the lower overhead of collection for a feature and more flexibility for adapting the detection with changing network topology than traditional approaches. But it did not discuss any mitigation mechanism for the detected attack.

Trung V. Phan et al. [47] proposed an optimized protection mechanism (OpenFlowSIA) for SDN network from DDoS flooding attacks based on SVM and Idle-timeout Adjustment (IA). Their methodology utilized high accuracy and little processing time from SVM advantages in classification. Moreover, it also effectively applied the IA algorithm and coherent policies for protecting the network from resource exhaustion caused by flooding attacks, especially for the OpenFlow switches and SDN controller.

Dotcenko et al. [18] proposed a fuzzy logic-based security management system by using the combination of the two anomaly detection algorithms with fuzzy inference carried out by Mamdani algorithm. These algorithms are TRW-CB and rate-limiting. This system avoided the excessive amount of computation by using fuzzy logic. Moreover, the proposed system provided more accurate decisions than the other systems using detection algorithms without fuzzy logic. However, this system did not describe any mechanism for attack mitigation.

Dillon et al. [16] proposed a DDoS mitigation system by using OpenFlow. The proposed system worked with three components. The first component is detecting anomalies by making the comparison of expected and real standard deviation over packet and byte rates. The deviation is calculated by utilizing the OpenFlow flow statistics collected by the Ryu controller for every second. The second component is identifying the source of the attack with two methods such as packet symmetry and temporarily blocking. The third component is blocking the identified attacker by installing OpenFlow rule with drop action.

R. Priyadarshini et al. [48] proposed an innovative Source-based DDoS defense mechanism that can be used in not only fog environment but also the cloud environment to mitigate DDoS attacks. It used the SDN to set up the DDoS defender module at the SDN controller to detect the unusual behavior of DDoS attacks in the Network/Transport level. They also provided deep learning (DL) based detection method which uses the network traffic analysis techniques to filter and forward the normal packets to the server and block the attacked packets.

Tran Manh Nam et al. [41] proposed two DDoS attack detection approaches which are based on the SOM. They implemented their proposed algorithms with the detection architecture in the SDN technology which provides flexibility and programmable abilities. They can quickly perform complex classification and detection algorithms because of the SDN controller.

Aapo Kalliola et al. [29] proposed a DDoS mitigation and traffic management system which is largely automated and can be implemented on the SDN technology. Their mechanism combines fixed or dynamic blacklist integration, automated hierarchical clustering-based normal traffic learning, and service distribution or additional server capacity invocation within the network. The mechanism is intended to be effective detection of packet and bandwidth flooding attacks and can be used to protect both network links and end hosts.

Machine learning-based mechanisms are more useful for detecting malicious activity based on the abnormal behavior of the network. However, the performance of these mechanisms is depending on the training dataset.

2.5.3 Traffic Pattern Analysis

Traffic pattern-based analysis mechanism differentiates the normal and abnormal traffic by assuming all abnormal hosts produce similar traffic patterns that are different from the traffic of benign hosts. When a network is infected by a botnet attack, all bot machines are controlled by a single botmaster. As a result, the bot machines will have a similar traffic pattern sent from the master host.

Shin et al. [51] implemented an OpenFlow security application development framework with the OF-enabled detection and mitigation module, FRESCO, especially for helping the security researchers in their contribution of different security detection and mitigation modules. One example application written in FRESCO script is the FRESCO version of Botminer application. BotMiner in this application also assumes that hosts infected with the same botnet exhibit similar patterns at the network level, and these patterns are different from benign hosts. Therefore, the hosts with similar packets per second and bytes per second are defined as bots.

Jin et al. [28] proposed a malware detection system using SDN that detects mobile malware by identifying suspicious network activities through real-time traffic analysis. As they assumed malware infrequently infect only one victim host, their detection techniques collect similar features of communications such as common detection, connection time, and common platform from multiple other hosts for identifying the attack.

Jing Zheng et al. [64] proposed RADAR for the detection and throttling of the DDoS attacks via adaptive correlation analysis built upon unmodified commercial off-the-shelf (COTS) SDN switches. It is a practical system for defending against the various type of DDoS flooding-based attacks such as SYN flooding, link flooding, and UDP-based amplification attacks. There is no need to modify the SDN switches or protocols and extra appliances. By identifying attack features in suspicious flows, and locates attackers (or victims) to control the attack traffic by adaptive correlation analysis, their proposed system could detect the attacks accurately.

I Gde Dharma N. et al. [15] proposed a method that considers the time duration of DDoS attack detection and attacks the time pattern of DDoS attack for preventing a future attack. They also presented the potential vulnerabilities in the SDN controller that can be exploited for DDoS attacks and discuss the methods to detect and mitigate DDoS attacks.

Ahmad Aleroud et al. [2] presented an inference-relation context-based technique for the detection of DoS attacks on SDNs. They proposed an inference-relation context-based technique for the detection of DoS attacks on SDNs. This technique utilized contextual similarity with existing attack patterns to identify DoS in an OpenFlow infrastructure. They developed a graph-based mechanism for the detection of DoS attacks over the SDN network. They also showed the benefit of using existing attack patterns to discover attacks that target the relation context-based SDNs. They created a flow aggregation mechanism for further improving the efficiency of detecting DoS attacks on SDNs.

2.5.4 Connection Rate

There are two types of connection rate-based anomaly detection mechanism: Number of connections established, and Connection success ratio.

The connection rate-based mechanism with the number of connections established considers the number of connections instantiated within a certain time. The number of connections attempts from the infected machine to a particular server is extremely higher than that from the normal machine. Generally, the normal host produces a lower number of connections and repeat access to a recent particular application during a certain time. Thus, this detection mechanism compares the number of connections established with a threshold to decide the attempting connections hosted from the normal machines or infected machines.

Similarly, another mechanism with the connection success ratio considers the number of successful connections ratio over all of the connections including success and failed connections. The probability of the number of successful connections from the normal host is much higher than that of the infected host. This mechanism monitors the new connection request such as the SYN connection request in TCP protocol without being received a response SYN-ACK or received an RST response when the timeout expires for unsuccessful connections and compared with a threshold for identifying the infected host. Threshold Random Walk with the credit-based algorithm is typically used in this mechanism. This algorithm has also been used in SDN based DDoS detection as in the proposed system for blocking the botnet-based attacks that are implemented by authors Lim et al [34].

Reza Mohammadid et al. [39] proposed SLICOTS, an effective and efficient countermeasure for mitigating the TCP SYN flooding attack in SDN. SLICOTS takes the benefits of the dynamic programmability nature of SDN for detecting and preventing attacks. SLICOTS was implemented in the controller for surveying TCP connection requests and blocking malicious hosts. It was implemented as a lightweight extension module of OpenDayLight controller. They evaluated SLICOTS with various types of attack scenarios. SLICOTs installed temporary forwarding rules during the TCP handshaking process. It also installs permanent forwarding rules after the validation of a request. Moreover, it blocked the attacker that generated a large number of half-open TCP connections. One of the main benefits of SLICOTS was that it did not interrupt other legitimate requests.

Tushar Ubale et al. [56] presented an SRL module for preventing the SYN Flooding DDoS attack in the SDN environment. SRL installed permanent forwarding for the requested connection and moved the user to the Whitelist as soon as the handshake was completed. It did not install temporary rule during the process of TCP handshakes like SLICOTS and OPERETTA. Moreover, SRL gave a chance to the user if there is any interference in connection establishment. It also detected a malicious user who firstly created the complete TCP handshake and then launches the attack. As it was composed of two modules, hashing, and flow aggregator, it had the specific advantages of each module. The first module could replace the flow rules from the flow table according to the hash value's priority. As well as the second one could block the malicious connection requests. Another advantage of SRL was that it could detect slow rate DDoS attacks by limiting the TCP connection requests.

The connection-rate based mechanisms have been successfully used in some of the experiment testing running on the various types of SDN controllers: NOX, Floodlight, Baecon, and POX controller.

2.5.5 Integration of Traffic Monitoring Tool and OpenFlow

There are various types of traffic monitoring tool and they are combined with OpenFlow for implementing a more lightweight DDoS detection and mitigation system by reducing the traffic monitoring overhead of the SDN controller. The widely used tools are Snort, Bro, sFlow and iftop.

One of the most popular and widely used tools for both the network intrusion detection system and network intrusion prevention system is SNORT. This tool has been successfully used in the SDN-based detection mechanism together with a standard OpenFlow protocol in SDN. Xing et al. [63] proposed an intrusion prevention system based on the combination of OpenFlow protocol and snort monitoring tool in the Xen-based cloud environment with the name of SnortFlow. This system was feasible in a cloud system because of using the combination of two useful functions such as the capability of intrusion detection and flexibility of network reconfiguration.

Similarly, Tommy et al. [11] proposed a collaborative attack detection and containment approach for defending the SYN flooding attack with the spoofed IP address. In this approach, any new traffic has sent to not only the controller but also the monitor. Snort was used for monitoring the traffic that has sent by the Open vSwitch, and generating an alert when the traffic exceeds the predefined threshold value. Correlator in controller processes for confirming the attack, finding the attack location, and dropping the flow coming from the attack when getting the alert message from the monitor.

Another famous traditional IDS, Bro has been used to monitor traffic as an assistant in the SDN-based DDoS detection mechanism. Lukaseder et al. [36] proposed a framework based on SDN and Bro monitoring tool for network security. They split their DDoS mitigation system into three steps: detection, observation, and mitigation. The first step determined whether the attack is ongoing or not. The second one identified perpetrators of the ongoing attack by observing every connection with the observer, Bro. The SDN controller blocks the attackers or redirects them to the CAPTCHA server in the final step. As a result, this mitigation system can detect attacks, identify the attackers, and mitigate the effects of the attack within minutes or even seconds without optimization for the specific network infrastructure and mitigate SYN, HTTP, and TLS flooding attacks with common off the shelf hardware, even in high-speed networks. However, this system might redirect falsely the legitimate client to the CAPTCHA server.

T. Lukaseder et al. [36] proposed a framework based on SDN and the Bro Security Monitor that can mitigate attacks simply within the network infrastructure. They provided a system that detects attacks, correctly identifies the attackers, and mitigates the effect of the attack. The objective of this system is to be independent of the affected services and to be used within general network infrastructures.

sFlow is a sampling technology embedded within the network equipment such as routers and switches. It meets the main requirements of network traffic monitoring solutions such as network-wide view, scalable, low-cost solution, and industry standard. It has been widely used in SDN-based DDoS detection mechanisms. Lu et al. [35] implemented an approach for defending the Botnet-based DDoS flooding attack by using the combination of the advantages of SDN and sample flow (sFlow) analyzer. It also used a detection algorithm based on a statistical inference model and a response scheme that drops the attack with a source-based defense mechanism. This method is simple, feasible, low-cost implementation and effective. Similarly, Aizuddin et al. [1] proposed a mechanism for detection and mitigation of the DNS amplification attack via sFlow with security-centric SDN. Giotis et al. [22] also demonstrated the solution of the scalability problem in the SDN-based network by using the combination of sFlow and OpenFlow.

Juan Wang [59] proposed LFADefender, a novel link-flooding attack (LFA) defense system that leverages some key features of SDN, such as programmability, network-wide view, and flow traceability, for the effective detection and mitigation of LFA. In LFADefender, they proposed an LFA target link selection approach and design an LFA congestion monitoring mechanism for detecting the LFA effectively. Moreover, they described multiple optional paths rerouting method for the temporal mitigation of links congestion caused by LFA. They also proposed a malicious traffic blocking approach to completely mitigate LFA.

Chaitanya Buragohain et al. [10] proposed FlowTrApp, an SDN framework for data centers that detects and mitigates the DDoS by using flow rate and flow duration of flow. By using an SDN engine containing sFlow based flow analytics engine sFlow-RT and an OpenFlow, it attempts for detecting attack traffic ranging from low rate to high rate and long-lived to short-lived attacks controller. Their proposed mechanism firstly matches an incoming flow with a legitimate sample of traffic and then installs mitigation actions if a flow is found not lying in the bounds of the legitimate traffic pattern. They used both of these technologies to enables the duty sharing between the OpenFlow controller and the sFlow-RT application and towards DDoS attack detection and mitigation which enhances the performance of FlowTrApp.

Another traffic monitoring tool especially finding the bandwidth of incoming packets and the address of the packet is iftop. It can be used in a DDoS detection mechanism by evaluating the bandwidth of the incoming packets. Thomas et al. [53] proposed a DDoS Detection mechanism by using third party applications in the SDN-based network. This mechanism used iftop for collecting the traffic data to differentiate the traffic is normal or malicious. If the bandwidth of the collected traffic exceeds the defined threshold, the traffic will be assumed as malicious traffic. The firewall application running in the SDN controller continued to handle the suspicious traffic and the application drops the real malicious.

To provide continuity of services in a particular network, DDoS attack mitigation mechanisms are needed as soon as the attack is detected. Without mitigation capabilities, the effective detection mechanism will not be sufficient for providing network service continuously. Therefore, DDoS mitigation mechanisms with the capabilities of SDN are discussed in the next session.

2.6 SDN-based DDoS Attack Mitigation

The main capabilities of SDN are improving the agility and flexibility of a network. One of the features of SDN, logical centralized controlling of the network, provides the visibility of the entire network from a global point of view by a centralized SDN controller. Thus, the controller can configure the network consistently and rapidly according to the requirements of network changing. As a result, the SDN network can be mitigated effectively from DDoS attacks by using the advantage of a centralized controller. As soon as the attacks are detected, the DDoS attack mitigation application blocks the attack traffic by installing new flow rules into the switches.

The common mitigation mechanisms in SDN networks are dropping packets, blocking ports and redirecting traffic. Moreover, deep packet inspection, isolating traffic and changing MAC and IP addresses are also used as the SDN-based DDoS attack mitigation mechanism.

2.6.1 Drop Packet or Block Port

The packet from network traffic can be dropped by installing flow rule with drop action into the switch. But blocking port means the attacking port of the switch are completely blocked. The two mitigation mechanisms are simple and fast in blocking the attack traffic. However, they can be blocked legitimate traffic in case of false attack detection or compromised legitimate hosts.

Depending on the deployment location, the defense mechanism for DDoS flooding can be divided into three mechanisms: source-based defense mechanism, destination-based defense mechanism, and hybrid defense mechanism.

2.6.1.1 Source-based Defense Mechanism

Source-based mechanisms are deployed near the sources of the attack to prevent network customers from generating DDoS flooding attacks. These mechanisms can take place at ingress switch connected to the attacker's host. The advantage of this mechanism is for the reduction of wasting resources existing along the path from source to destination.

2.6.1.2 Destination-based Defense Mechanism

In the destination-based defense mechanisms, detection and response are mostly done at the destination of the attack (i.e., victim). These mechanisms can closely observe the victim, model its behavior and detect any anomalies. Most of the destination-based mechanisms cannot accurately detect and respond to the attack before it reaches the victims and wastes resources on the paths to the victims; hence, they are not capable of detecting and responding to the DDoS attack traffic properly.

2.6.1.3 Hybrid Defense Mechanism

Hybrid defense mechanisms usually place the attack detection modules near the victims and execute packet filtering close to the attack sources. These mechanisms only limit the rate of malicious packets and do not harm legitimate flows.

2.6.2 Redirection

The redirection mechanism sends the detected attack packet to a new destination such as deep packet inspection, or captcha that analyze and check the packet in more detail. The main objective of this mechanism is to reduce the false alarm rate by checking again the detected packet to ensure that whether the suspicious packet is a real attack or not. Before redirecting to the new IP address, the mechanism needs to tear down all the connection to the existing server or destination address so that the bots cannot access the new destination directly. The drawback of this mechanism is that the processing time might increase because of deeply checking the packets.

L. Dridi et al. [19] proposed SDN-Guard, a novel mechanism that can protect DoS attacks on the SDN network efficiently by dynamically rerouting potential attack traffic, adjusting flow timeouts and aggregating flow rules. Their proposed system was intended to reduce the overloading of the controller processing and communication capacity and flood switch CAM tables, and upgrade the overall network performance.

2.6.3 Control Bandwidth

This mechanism controls bandwidth at each switch interface for a specific host or network. For example, it limits the bandwidth for the hosts that have been compromised, and gives the full bandwidth for the benign hosts that have never been infected. According to the proposed mechanism at [55], the authors divided the mitigation methods into three levels: the first level with highest data or packet rate for the detected attackers, the second level with a lower rate for the ongoing attackers, and the last one with blocking all packets from the attackers.

2.6.4 Change Network Topology

This mechanism is very effective in DDoS mitigation over the SDN network. The connection of the attack is torn down after the attack is detected. However, the normal packets can be forwarded by using an alternate path.

2.7 Chapter Summary

The architecture of SDN and its two capabilities such as centralized control the network logically and reprogrammed the switches at the data plane dynamically make the natural choice for enhancing the network security, especially in DDoS attack detection and mitigation. The above-mentioned SDN-based DDoS detection and mitigation mechanisms clearly show that they are more flexible than traditional mechanisms. However, each detection and mitigation mechanism has its advantages and disadvantages. Among the mechanisms, the widely used one is a statistical analysis-based change point detection mechanism.

There are two algorithms based on the Exponentially Weighted Moving Average (EWMA) formula in this technique: adaptive threshold algorithm (ATA) and cumulative sum (CUSUM) algorithm. A non-parametric CUSUM with an adaptive threshold algorithm has been implemented to calculate the dynamic threshold for the detection of DDoS attacks in [13]. They used CAIDA to define the initial threshold value. Conti et al. [14] used the EWMA control chart to protect network reconnaissance. But the proposed systems in [13-14] had a little overhead for the collection and manipulation of traffic statistics in the SDN controller. Bawany et al. [6] proposed a framework for the application-specific detection and mitigation of DDoS attacks over the smart city data center by using a modified EWMA formula. For reducing the overhead of traffic statistic in the SDN controller, the sFlow-RT analyzer is used in this dissertation.

Siris et al. [52] investigated these two algorithms for SYN flooding attack detection. The ATA is a simple and naive algorithm for detecting the high-intensity attack. But this algorithm produced high false alarms. The false alarm's avoiding method used in this algorithm affects the false negative rate. In order not to raise such a false negative rate, the existing ATA is modified by encountering the baseline of the network in the calculation of the dynamic threshold in this dissertation.

A. Arins proposed firewall as a service in SDN for solving the two main problems of DDoS: distinguishing good packets from bad packets and dropping bad packets at the closet point to attacker networks [3]. The authors in [43] also drop the detected malicious packets as their mitigation mechanism is protecting the IoT-based DDoS attack in SDN. Lu et al. focused on the source-based defense mechanism against botnet-based DDoS flooding attacks through the combination of the power of

SDN and sFlow technology [35]. Conti et al. [14] also mitigated the flooding-based DoS attack by installing the temporarily Drop flow rules. The authors of [60] discussed the anomaly mitigation with the consideration of the centralized control feature of the SDN network for tracing back the source of the attackers and doing the source filtering at the source switch. All their proposed mitigation systems are discarding the detected attack with the installation of a simple drop flow rule according to the advantages of the centralized control feature of SDN.

In this dissertation, in order to mitigate the flooding attack effectively, drop flow rules are installed with two options at the ingress switch of the attack. Temporarily drop flow rules are installed for the attacks that come for the first time. Permanent drop flow rules are installed when the same attacks come again after their respective temporarily drop flow rule expires.

CHAPTER 3

BACKGROUND THEORY

This chapter will cover the theoretical descriptions of DDoS attack defensive mechanisms using SDN. Thus, the first section explains in detail about SDN including the architecture of SDN with its components and protocol, and the main function of SDN, flow rule installation. The statistical data collection and change point detection techniques for the detection mechanism and abnormal packets discarding for mitigation mechanisms are also described in the next sections.

3.1 Software-Defined Networking

Open Network Foundation (ONF) defines that software-defined networking (SDN) is the physical separation of the network control plane from the forwarding plane, and where a control plane controls several devices [5].

3.1.1 Architecture of SDN

The basic SDN architecture is shown in Figure 3.1 [73]. Generally, according to the main concept of SDN, decoupling the logical control function and physical data forwarding operation, it has two layers or planes: control plane, and infrastructure or data plane. The logically centralized controller lies in the control plane, manages and orchestrates the networking devices existing in the data plane depending on the type of activating applications hosted on it. Thus, the classical SDN architecture consists of an additional plane, application plane, occupied by an instance of applications. The applications take the exchanging information of data plane via the controller for their various types of processing. The networking devices are termed as switches in SDN, but the switches can operate different functions in OSI's seven layers: from layer 4 up to layer 7 according to the types of flow rules set up by the application.

The switches that can be used in SDN are responsible for matching the incoming packet header with existing flow rules stored in the flow tables of the switches, passing the header information of the first packet to the controller in case of non-existing flow rule for it and updating the flow rules installed by a particular application via the controller. The switch can be software switches such as Open vSwitch, Cisco Nexus 1000v, VMware vSphere, NEC Hyper-V, and so on or hardware switch such as Brocade, Cisco, HP, IBM, Juniper Networks, NEC, and so on [23]. Among them, OVS (Open vSwitch) is the most commonly used switch in the SDN network.

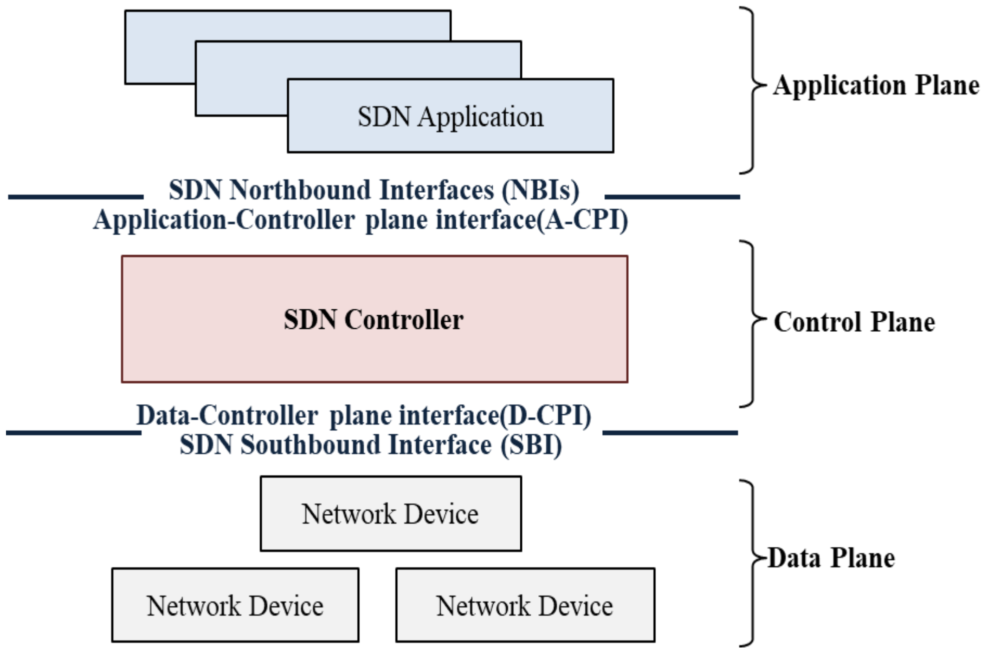

SDN Application
Application Plane
SDN Northbound Interfaces (NBIs)
Application-Controller plane interface(A-CPI)
SDN Controller
Control Plane
Data-Controller plane interface(D-CPI)
SDN Southbound Interface (SBI)
Network Device
Network Device
Network Device
Data Plane

Figure 3.1 Basic SDN Architecture

The controller is the heart of the architecture of SDN and manages the flow records in the flow tables of the switches by installing flow rules with different actions for the network traffic reactively or proactively. Depending on the use of different programming languages and environments, various types of SDN controllers have been developed. For example, ONOS, Beacon, Open Daylight, and Floodlight are based on Java, but ONOS is specially developed for a distributed architecture. Similarly, NOX is based on C++ and Python, and POX and Pyretic are based on Python [57].

There are two interfaces to interconnect between the planes of the SDN architecture. They are North Bound Interface (NBI) and South Bound Interface (SBI). The former is used to communicate the application plane and control plane, also called applications-control plane interface (A-CPI) or intent interface. The latter one interconnects between the control plane and data plane and is also referred to as a data-controller plane interface (D-CPI). OpenFlow is the most commonly used protocol in D-CPI or SBI. The other protocols such as sFlow and SNMP can also be used to communicate between controller and networking devices but OpenFlow is the standard protocol used in SDN.

3.1.1.1 OpenFlow Protocol

There are several standard protocols used in real applications of SDN. OpenFlow is one of the standard protocols that can implement the SDN concept in either software or hardware. It was proposed by Stanford as the standard of SDN protocol. It is a flow-oriented protocol. Since the protocol is mainly used for communication between the SDN switch and SDN controller, the presentation of this protocol is described with the operation of the main components of OpenFlow enabled switch as shown in Figure 3.2. An OpenFlow enabled switch has OpenFlow port, OpenFlow table, OpenFlow channel, and OpenFlow switch protocol [65-66].

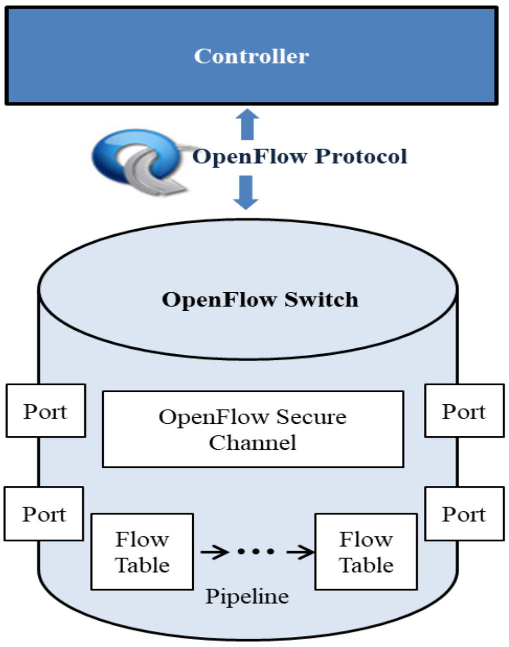


Figure 3.2 Main Components of OpenFlow Enabled Switch

OpenFlow ports are the network interfaces of the OpenFlow enabled switch that pass the packets between internal OpenFlow processing and their connected external network. Packets are received on an ingress port and forward them to the output port after pipeline processing. The ingress port can be used in matching the incoming packets with flow entries. The pipeline process can decide for sending the packet on an output port according to the action that defined how to send back the packet to the network.

An OpenFlow switch is needed to have at least one flow table or more. The OpenFlow pipeline processing defines how the packets are sending among the flow tables. The processing is simplest when the switch has only one flow table.

A flow table consists of one or more flow entries. The main components of a flow entry are as shown in Table 3.1.

Table 3.1 Main Components of a Flow Entry

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie

The matching field contains ingress port and packet headers. Besides, it may contain the optional metadata specified by a previous flow table. The function of the fields is for matching against packets. The matching precedence of the flow entry is defined in the priority field. The value of the counters field describes the up to date number of matching packets with the flow entry. The instructions of a flow entry define a set of instructions for the matched packets with the flow entry. The instructions are getting from the specified actions and/ or pipeline processing such as Goto-Table next-table-id. The most commonly used and required actions are:

1) **Output:** The action forwards a packet to a specified OpenFlow port connected to its destination.
2) **Drop:** The packets matched the flow entry with no output port action will be dropped. On the other hand, the flow entry can be specified with drop action. For example, the attack packet can be discarded with the flow rule or entry with drop action.
3) **Group:** The action processes the packet according to the specified group buckets. For example, the group action may be the combination of two output port actions: To Controller and To destination port to send the packet to both the controller and its destination concurrently.

The OpenFlow secure channel is used to make the interconnection between the OpenFlow switch and the OpenFlow controller. A typical OpenFlow controller manages OpenFlow switch by exchanging OpenFlow messages between them. Since it controls all of the OpenFlow switches over the whole network, it has many OpenFlow channels for every single connection between each switch and controller. However, each switch may have one OpenFlow channel to a single controller or multiple channels connected with different controllers for reliability purposes. The controller usually manages the switches remotely over one or more networks. The OpenFlow channel is usually created by initiating a single connection between OpenFlow switch and OpenFlow controller with Transport Layer Support (TLS) or plain TCP. However, the channel may be created with multiple connections for achieving parallelism. The channel must be initiated by OpenFlow switch to be able to get a connection with the OpenFlow controller. But, in some cases, the OpenFlow switch may allow the controller to initiate the connection. To prevent unauthorized connections, such an OpenFlow switch should control itself to have a secure connection.

The core of the OpenFlow switch specification is the structure of the OpenFlow Switch Protocol messages. The OpenFlow protocol is implemented by using OpenFlow messages transmitted over the OpenFlow secure channel. Each message has its structure starting with the common OpenFlow header and includes the other structures that may be common to multiple message types. Each structure defines the order in which information is included in the message and may contain other structures, values, enumerations or bitmasks [30,57].

3.1.1.2 Open vSwitch

Open vSwitch (OVS) is one of the most popular, software-driven OpenFlow switches. It is a multilayer software switch under the open-source Apache 2 license. Its kernel is written in Linux 3.3 and its firmware including Pica8 and Indigo. Open vSwitch is used in multiple products and runs in many large production environments. It is the default switch in XenServer 6.0, the Xen Cloud Platform and also supports Xen, KVM, Proxmox VE, and VirtualBox. It has also been integrated into many virtual management systems including OpenStack, openQRM, OpenNebula, and oVirt. The kernel datapath is distributed with Linux, and packages are available for Ubuntu, Debian, Fedora, and OpenSUSE. Open vSwitch is also supported on FreeBSD and NetBSD. The Open vSwitch release in development has been ported to DPDK. Open vSwitch can operate both as a soft switch running within the hypervisor, and as the control stack for switching silicon. It has been ported to multiple virtualization platforms and switching chipsets.

The main part of the code is written in platform-independent C and is easily ported to other environments. The current release of Open vSwitch supports the following features: standard 802.1Q VLAN model with trunk and access ports, NIC bonding with or without LACP on upstream switch, NetFlow, sFlow(R), and mirroring for increased visibility, QoS (Quality of Service) configuration, plus policing, Geneve, GRE, VXLAN, STT, and LISP tunneling, 802.1ag connectivity fault management, OpenFlow 1.0 plus numerous extensions, transactional configuration database with C and Python bindings, and high-performance forwarding using a Linux kernel module.

As shown in Figure 3.3, the above-mentioned Open vSwitch's features can be categorized into four main groups: security, monitoring, Quality of Service (QoS), and automated control. As one of the security features, VLAN isolation can be used to enforce VLAN membership of a VM without having the knowledge of the guest itself. Another security feature is tunneling that provides isolation and reduces dependencies on the physical network. As a monitor feature, visibility supports industry-standard technology to monitor the usage of a network by using one of the monitoring tools or traffic analyzers such as sFlow, NetFlow, and port mirroring. As a QoS feature, the traffic rate can be limited by not only the existing traffic control layer such as the policer for ingress rate limiter but also the Open Flow controller to select traffic class for each virtual machine. Open vSwitch supports a database for storing network state (OVSDB) that supports remote triggers. Therefore, a piece of orchestration software can "watch" various conditions of the network and respond if/when they change. Open vSwitch supports OpenFlow to be able to export remote access for controlling traffic. Open vSwitch is used in many areas including global network discovery through inspection of discovery or link-state traffic (e.g. LLDP, CDP, OSPF, etc.).

The main components and available tools provided in an Open vSwitch are ovs-vswitchd, ovsdb-server, ovs-dpctl, ovs-vsctl, ovs-appctl, ovs-ofctl, and ovs-pki. The ovs-ofctl tool is the most commonly used tool in the SDN network for querying and controlling OpenFlow switches and controllers [66].

3.1.1.3 Controller (Open Network Operating System – ONOS)

The controller is the main part of an SDN network since the whole operations of the entire network are managed by a single centralized controller or more than one distributed controller. There are various types of SDN controllers that have developed for the SDN network. They are implemented with different programming languages such as Java, and python.

The first open-source network operating system aiming for Service Provider and mission-critical networks is the Open Network Operating System (ONOS). It has been built to provide high availability, scalability, and performance according to the demands of these networks. Moreover, to develop application easily, and be able to control both OpenFlow-enabled and legacy devices, ONOS has created useful Northbound and Southbound abstractions and interfaces respectively. Therefore, ONOS provides carrier-grade features including scalability, availability, and performance via web-based control plane helps in migrating black boxes-based existing networks to white boxes-based SDN networks and reduces Capability expenditure (CapEx) and Operational Expenditure (OpEx) for service providers. To validate its architecture with real-world use cases, ONOS has been developed in concert with demanding network vendors, leading service providers, research and education network operators, collaborators, and ONF.

In addition, as ONOS is an operating system that manages network resources, provides APIs and abstractions for programming, monitoring, managing network devices used in the data plane, it immensely simplifies the creation of effective and innovative network applications operated over a variety of hardware. There are many open-source SDN controllers such as POX, Beacon, SNAC, and NOX that have been developed by collaborators at Stanford, Berkeley, and Nicira Networks. However, these controllers were not intending to design for the foundation of commercial products, providing scalability, high availability and performance features, and having general abstractions for every kind of device. Moreover, since they just exchanged OpenFlow messages between OpenFlow switches and controller directly, they are more like the drivers of the devices. Therefore, they are not developed as a complete SDN platform including key features of the ONOS controller like scalability, high availability, and performance.

Since the architecture of ONOS has been designed specifically for the service provider, it considered the key features needed by the service provider network: high availability, scalability, and performance together with two types of interfaces: southbound and northbound.

The key features provided by ONOS controller are:

Distributed Core is the key architectural feature that provides scalability, high availability, and performance as it brings carrier-grade features to the control plane of SDN. The obvious function of ONOS, clustering that brings the web style agility to the control plane of SDN and service provider networks.

Northbound abstraction/APIs that include two types of abstractions: intent framework and global network overview. The first abstraction allows the application for requesting service from the network without knowing how the service will be performed. The second abstraction provides the application with a view of the network as a network graph. By reviewing these abstractions, this northbound API insulates the application from the detail of the networks that are needed by the application to increase application development velocity and allowing network changes without application downtime.

Southbound abstraction/APIs allow plug-ins for various southbound protocols and devices. The devices may include both OpenFlow-enabled and legacy devices. The benefit of this abstraction is the insulation of the core of ONOS from the details of different devices and protocols, and migration of the existing legacy devices to white boxes supporting OpenFlow.

Software Modularity makes it easy to develop, debug, maintain, and extend, and customize ONOS as a software system by a community of developers and by the providers [7].

3.1.2 Flow Rule Installation in SDN

There are three modes of operation when using OpenFlow to populate TCAM in switches: reactive flow instantiation, proactive flow instantiation, and hybrid flow instantiation.

3.1.2.1 Reactive Flow Instantiation

When a switch receives a packet that cannot be matched to any installed flow rule, the switch normally first buffers the packet and then sends a request to get a new flow rule from the controller with an OFPT PACKET IN message. This message includes the header field of the data packet. The controller then replies with an OFPT FLOW MOD message, comprising the action to be performed on the data packet and the duration for which to maintain the flow rule in its flow table. This duration is called a timeout. Each flow rule has two associated timeout values, an idle timeout value, a soft timeout, which is initiated when the flow remains inactive, and a hard timeout, which is initiated at the timer expiry. When either of these timers expires, the switch eliminates the corresponding flow entry from its flow table and sends an OFPT FLOW REMOVED message to the controller. Moreover, the flow rule can be installed temporarily or permanently. Temporary flow rules expiration is depending on the predefined timeout value while the permanent ones are never eliminated from the flow table. Since rules are requested by the switch only upon receiving data packets, this mechanism of flow rule installation is "reactive" [9] as shown in Figure 3.3 [36].

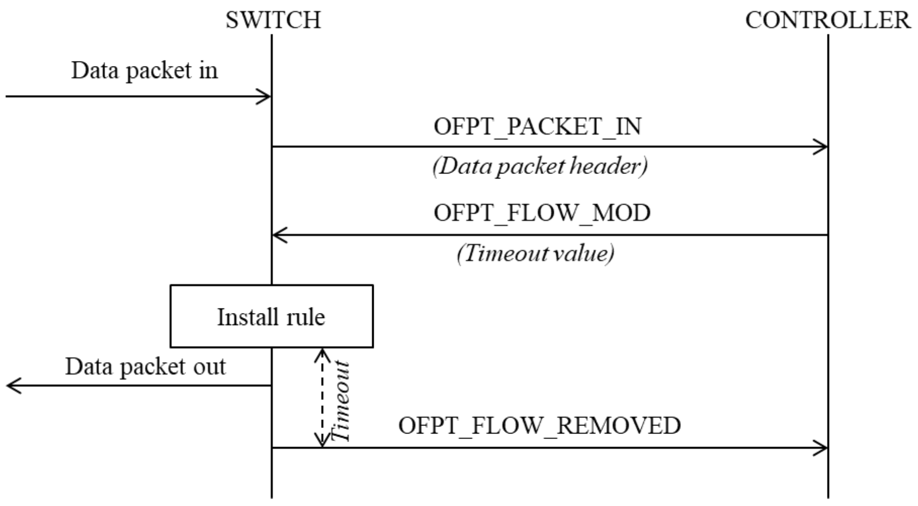


Figure 3.3 Installation of Reactive Flow Rule in OpenFlow

3.1.2.2 Proactive Flow Instantiation

OpenFlow controller could populate the flow tables before all traffic coming into the switch for matching with the flow rules. By pre-defining all of the flows and actions in the switch’s flow tables, the packet-in event never occurs. As a consequence, all packets are forwarded after looking up the flow rules in the switch’s flow tables. Thus, proactive OpenFlow flow tables eliminate any latency made by consulting a controller on every flow.

3.1.2.3 Hybrid Flow Instantiation

A combination of the reactive and proactive flow rule installation allows not only the flexibility of reactive forwarding the packets but also preserving low-latency forwarding [9].

3.2 SDN-based DDoS Attack Detection Scheme

Statistical schemes are used to solve the DDoS Detection problem. There are two main reasons for choosing to use a statistical methodology. The first reason is that it provides a sound statistical background that can be easily evaluated and also is simple to implement and understand. The second one is that although statistical approaches have been around for a long time, they still provide a safe starting point, and are both computationally efficient and computationally effective [12].

In general, the functions of the statistical analysis method are collecting the network traffic statistic, comparing the collected statistic with a threshold value, and raising alert for the threshold violation. In the SDN network, the statistics of traffic can be collected by the SDN controller itself or using the help of a particular third-party analysis tool such as a sFlow-RT analyzer. The former method might be overloaded by the SDN controller. Thus, the latter method is commonly used in the collection of traffic statistics.

After getting the statistic of the traffic, the next operation of statistical analysis is to compare the traffic statistic with a threshold value. Thus, a threshold value can be defined statically or dynamically. The static threshold produces a high number of false alarms and a tedious job for the network administrator. Hence, the dynamic threshold is the most commonly used in statistical analysis techniques. The change point detection technique is simple to implement for calculating the dynamic threshold.

3.2.1 Packet Statistic with sFlow-RT analyzer in SDN

Software-defined networking (SDN) separates the network Data Plane and Control Plane, permitting external software to monitor and control network resources. Open Southbound APIs like sFlow and OpenFlow are an essential part of this separation, connecting network devices to external controllers, which in turn present high-level, Open Northbound APIs to SDN applications.

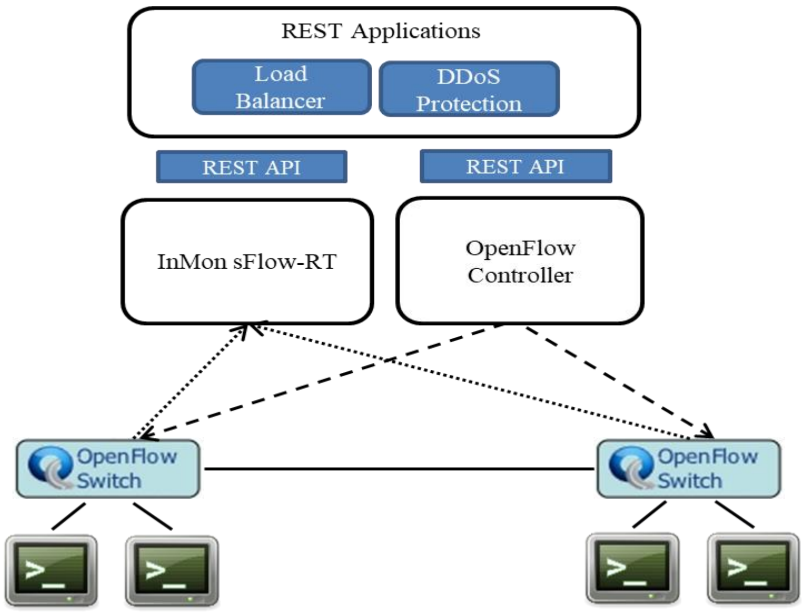


Figure 3.4 SDN Architecture with sFlow-RT Analyzer

One of the unique features of sFlow is its ability to monitor entire networks, not just selected devices or links. When configuring sFlow monitoring, enable sFlow on every switch port on every switch in the network. sFlow is implemented in hardware so it can operate at line rate without impacting switch performance [44].

3.2.1.1 Sampling

Packet-based sampling mechanisms are commonly used to characterize network traffic. To prevent synchronization with any periodic patterns in the traffic, packet sampling uses randomness in the sampling process. On average, it captures and analyzes 1 in every N packets.

Although this type of packet sampling does not provide a 100% accurate result, it does provide a result with quantifiable accuracy [46].

Table 3.2 Suggested Values for Sampling Rates

Link Speed	Sampling Rate
10Mb/s	1 in 200
100Mb/s	1 in 500
1Gb/s	1 in 1000
10Gb/s	1 in 2000

An important part of configuring sFlow on a switch is selecting a suitable packet sampling rate. The suggested values that should work well for general traffic monitoring in most networks are listed in Table 3.2. However, the sampling rate may be decreased if traffic levels are unusually high (e.g. use 1 in 5000 instead of 1 in 2000 for 10Gb/s links).

For getting the full visibility of the network, flow monitoring is needed to configure on all interfaces on the switch connected in the network. To monitor all the interfaces with very little overhead, packet sampling can be implemented in hardware [45].

3.2.1.2 Polling

Configure the interval (in seconds) that the device waits between port statistics update messages. Polling refers to the device gathering various statistics for the network interfaces configured for sFlow monitoring and exporting the statistics to the configured sFlow collector [44].

To track accurately link utilization, a suitable counter polling interval is needed to select. In general, the polling interval should be set to export counters at least twice as often as the data will be reported according to the Nyquist-Shannon sampling theory. For example, for trending the utilization with minute, the polling interval of between 20 and 30 seconds should be selected. Setting the relatively short polling intervals could not be a problem because the counter polling with sFlow is very efficient, allowing more frequent polling with less overhead than is possible with SNMP [74].

3.2.2 Change Point Detection Algorithms

One of the effective algorithms for calculating the dynamic threshold is ATA [52]. In this section, both algorithms (i.e. original ATA and modified ATA) are described with their respective false alarm avoiding method.

EWMA is commonly used in finding the dynamic threshold for the network traffic. The calculated threshold value provides not only high detection rate but also high false positive rate. Thus, ATA uses the twice of the EWMA result or more as its threshold value. However, it still raises some false alarms in some cases. In order to reduce false alarms, this algorithm only raises the alarm signal after a minimum number of consecutive violations of the threshold.

3.2.2.1 Adaptive Threshold Algorithm (ATA)

Let $\mathrm{CF_t}$ be the current number of incoming frames at time t, $\mathrm{PF_{t-1}}$ is the average number of frames estimated from the measurement prior to t. The initial value of PF is defined as 0. This initial value is used as the value of $\mathrm{PF_{t-1}}$ for the very first calculation.

EWMA formula is used to pre-calculate the average number of incoming frames that will be used in the comparison for the next second as shown in Equation 3.1:

$$\mathrm{PF_t} = \alpha \mathrm{PF_{t-1}} + (1-\alpha)\mathrm{CF_t} \tag{3. 1}$$

α is the factor parameter, $0 \le \alpha \le 1$, used in making the decision of the factors of current number of frames and average number of previous frames for calculating the average number of previous frames to be used at the next calculation.

If $\mathrm{CF_t} \ge (\mathrm{p}+1)\mathrm{PF_{t-1}}$ then the alarm signaled at time t. (3. 2)

The percentage parameter p is the percentage parameter, $\mathrm{p} > 0$. It is used to indicate the anomalous behavior when the defined percentage of the previous average number of frames is exceeded by the current number of incoming frames.

Applying the algorithm directly would yield a high number of false alarms. Thus, a simple modification is made to signal an alarm after a minimum number of consecutive violations of the threshold as shown in Equation 3.3:

If $\sum_{\mathrm{i=n-k+1}}^{\mathrm{n}} 1_{\{\mathrm{CF_i} \ge (\alpha+1)\mathrm{PF_{i-1}}\}} \ge \mathrm{k}$ then the alarm signaled at time t. (3. 3)

In this equation, k is the parameter that indicates the number of consecutive intervals the threshold must be violated for alarm to be raised, $\mathrm{k} > 1$.

3.2.2.2 Modified Adaptive Threshold Algorithm (MATA)

For reducing the false negative value while avoiding the false alarms, MATA is taken into account the baseline traffic, b, in the traffic comparison. Thus, the value for the baseline traffic is needed to define at the initial state.

The baseline of a network can be identified by analyzing the monitoring result of the network for a period of time. In this system, the sFlow analyzer is used as the monitoring tool and the monitoring result (i.e. the event information) is analyzed for defining the value of the baseline. The process of defining baseline can be divided into four steps:

Step 1: Collect all event information from the various types of service produced by the analyzer.

Step 2: Categorize the collected information according to their type of service (i.e. Web, FTP, Mail, NTP, DHCP and DNS).

Step 3: Find the maximum number of frames per second for each type of service from the collected event information.

Step 4: Define the maximum number of frames per second as the value of the baseline.

According to the step 3 and 4, the baseline traffic for web service is defined as the maximum number of SYN frames per second from the web event information produced by the sFlow analyzer as shown in Equation 3.4.

$$b_{\mathrm{web}} = \mathrm{Max}(\text{number of SYN frames per second for web service}) \qquad (3.\,4)$$

Similar to the web service, the baseline traffic of other TCP services such as FTP, and mail are identified. For the UDP services, the baseline traffic for the DNS service is identified as the maximum number of frames per second from the DNS event information as shown in Equation 3.5.

$$b_{\mathrm{DNS}} = \mathrm{Max}(\text{number of frames per second for DNS service}) \qquad (3.\,5)$$

The baseline traffic of the NTP and DHCP services are similarly identified as the baseline definition of DNS service.

Thus, in general, the baseline of a particular service, b_{SV} ,is identified as the maximum number of frames per second for the service, $\mathrm{Max}(n_{\mathrm{SV}})$,as shown in Equation 3.6.

$$b_{SV} = \mathrm{Max}(n_{SV}) \tag{3. 6}$$

In the Equation 3.6, b is the baseline traffic parameter, SV is represented for a particular service, n is the number of frames per second containing in the event information. The calculation of baseline traffic over an emulated SDN network environment is described in detail in section 5.2.2.

After getting the baseline value, the initial value of PF is defined as the value of b_{SV}. The initial threshold value is also identified as the value of b_{SV}. Thus, the initial number of incoming frames is compared with the initial threshold value. Then, the new threshold value is calculated and compared with the number of incoming traffic in the next second by using the Equations 3.7 and 3.8.

As the original ATA, the average number of incoming frames for the next second is pre-calculated by using the EWMA formula as shown in Equation 3.1 of session 3.2.2.1.

$$PF_t = \alpha PF_{t-1} + (1 - \alpha)CF_t \tag{3. 7}$$

Moreover, the Equation 3.2 of ATA is modified by adding the baseline traffic parameter, b_{SV} , for indicating the anomalous behaviour when the total number of the defined percentage of the average number of previous frames and the number of frames of the baseline traffic is exceeded by the current number of incoming frames. The modified equation is shown in Equation 3.8:

If $CF_t \geq (p + 1)PF_{t-1} + \mathbf{b_{SV}}$ then the alarm signaled at time t. (3. 8)

3.3 SDN-based DDoS Attack Mitigation Scheme

In the process of DDoS attacks mitigation, the first function is finding the source of the attack or the location of the attacker and then mitigating the attack which is the malicious traffic. The vast advantage of OpenFlow is obvious in attack mitigation because OpenFlow is tightly related to the forwarding function of any network component. New network traffic flows are passed on to the OpenFlow switch flow table, along with a specific action. The most commonly used actions are Drop, Modify-field, and Forward actions [66]. Forward action is used in benign flow entry for forwarding the flow to its destination. Drop action is attached to each attack flow to block the malicious traffic.

Moreover, each different action can use predefined priority values. A low priority value is assigned to flow-entries that are intended to forward packets matched to them. On the other hand, a higher priority value is assigned to flow-entries for dropping the packets. Thus, a drop flow rule will permanently take precedence against a forwarding one.

Another mitigation method is controlling the incoming flows with the priority of queuing or metering. The flow comes from the hosts that have a history of the attacker or showing suspicious behaviors are put into ‘warning queue' with a low priority value and are processed after handling all other requests in ‘normal queue' with a high priority [14]. By using this mechanism, the malicious traffic can be mitigated in the SDN network.

3.4 Performance Evaluation in Network Security

The confusion matrix is widely used as an assessment of the classification method in network security. The matrix is shown in Table 3.3 [13].

Table 3.3 Confusion Matrix

Confusion Matrix		**Predicted Label**	
		Normal	**Attack**
Actual Label	**Normal**	True Negative (TN)	False Positive (FP)
	Attack	False Negative (FN)	True Positive (TP)

In order to calculate False Negative Rate (FNR), i.e. the percentage of abnormal incorrectly identified as normal over the entire abnormal traffic, the Detection Rate (DR), i.e. the percentage of correctly identified abnormal traffic over the actual entire abnormal traffic, and Accuracy (ACC), i.e the percentage of correct detection over all traffic detection, we use the formulas as in Equations 3.9, 3.10, and 3.11, respectively.

$$\mathrm{FNR}(\%) = \frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TP}} * 100 \tag{3. 9}$$

$$\mathrm{DR}(\%) = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} * 100 \tag{3. 10}$$

$$\mathrm{ACC}(\%) = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} * 100 \tag{3. 11}$$

3.5. Chapter Summary

This chapter described the background theories related to the whole flooding attack detection and mitigation system over the SDN network infrastructure. Thus, it contained the detailed description of the SDN and SDN-based DDoS detection and mitigation schemes. Moreover, the performance evaluation for network security is also presented for the final evaluation of this system.

CHAPTER 4

FLOODING ATTACK DETECTION AND MITIGATION SYSTEM

The overall system architecture for the detection and mitigation of flooding attack is firstly described in this chapter. Then, the detailed architecture is separately described for each part of this system.

4.1 Architecture of Flooding Attack Detection and Mitigation System

The overall architecture of flooding attack detection and mitigation system is composed of two main phases: flooding attack detection, and mitigation of the detected attacks as shown in Figure 1. In the detection phase, the various types of frames from the SDN hosts incoming into the Open vSwitch are collected and detected by sFlow-RT analyser [75] in order to differentiate the normal frames and the malicious frames of the flooding attack. If the malicious flooding frames are incoming into the switches, then the analyser produces abnormal event information. The mitigation application running in ONOS controller [76] instantaneously discards the frames when it receives the event information from the analyser.

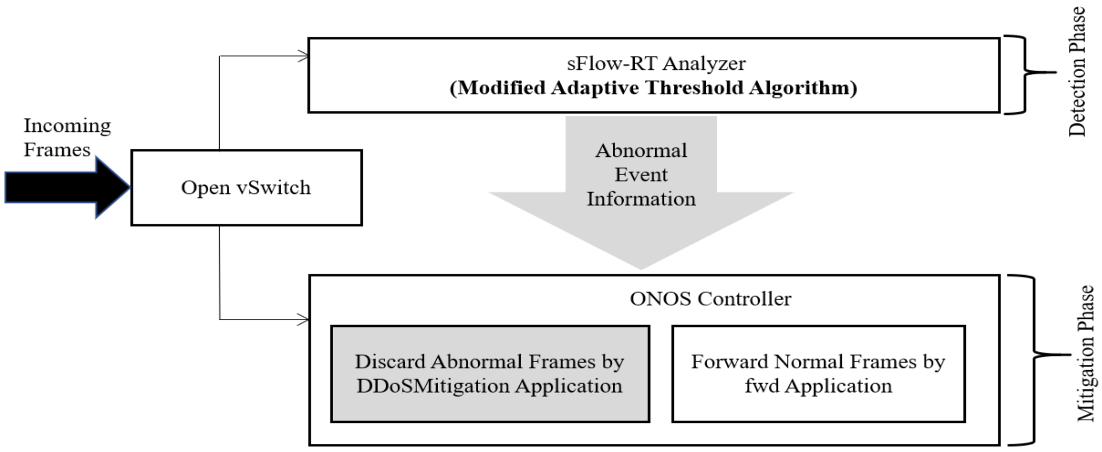

sFlow-RT Analyzer
(Modified Adaptive Threshold Algorithm)
Detection Phase
Incoming Frames
Open vSwitch
Abnormal Event Information
ONOS Controller
Discard Abnormal Frames by DDoSMitigation Application
Forward Normal Frames by fwd Application
Mitigation Phase

Figure 4.1 Overall Architecture of Flooding Attack Detection and Mitigation

4.1.1 Detection Phase

The flooding attack detection phase is implemented by using the sFlow-RT analyzer to reduce the load of traffic statistics in the SDN controller. It is composed of three parts: flow definition, flow handling, and event handling. The detailed architecture of the detection phase is as shown in Figure 4.2.

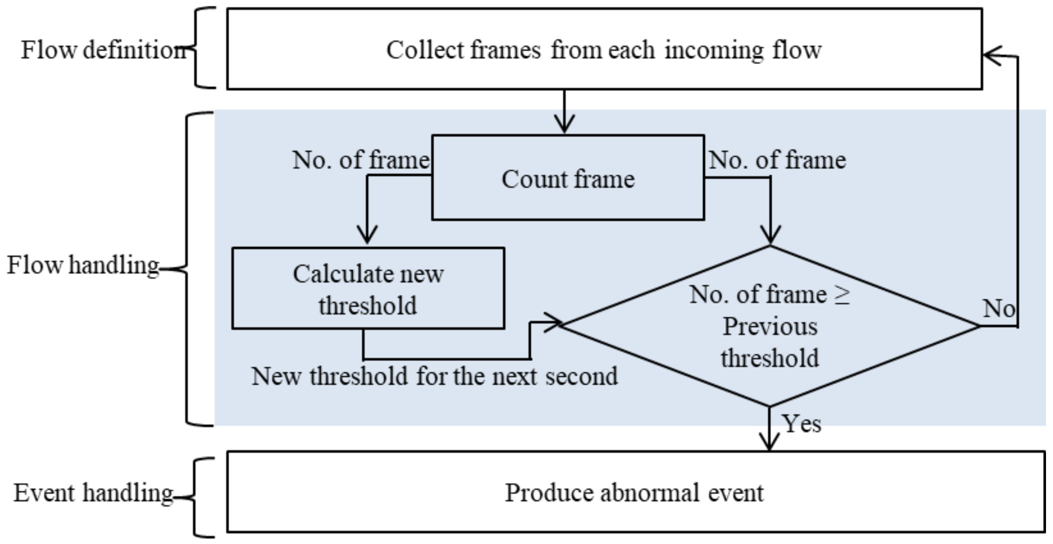


Figure 4.2 Detection Process in sFlow-RT Analyzer

4.1.1.1 Flow Definition

The analyzer collects the incoming flow of each service according to the predefined polling interval. As this system is detecting the flooding attack at the transport layer, it has two types of flow definitions for TCP and UDP protocols. For TCP protocol, the analyzer only collects SYN frames from the incoming TCP traffic of the WeB, FTP, and mail server by using *setFlow* function with the flow keys represented for the source MAC, destination MAC, source IP, destination IP, and destination port. Moreover, in order to obtain solely SYN frames for the individual service, the analyzer filters the flows with the respective destination port (i.e. 80 for Web, 20 for FTP, and 25 for mail) and the TCP flag for that frames, 000000010.

Similarly, for UDP protocol, the analyzer collects all frames from a particular incoming UDP traffic of DNS, DHCP, and NTP server with the flow keys for representing source MAC, destination MAC, source IP, destination IP, and destination port, and filtering with the respective destination port (i.e. 53 for DNS, 67 for DHCP, and 123 for NTP) for obtaining separated traffic flow for each service.

4.1.1.2 Flow Handling

The analyzer handles the various types of incoming flows in every second. It also controls the time of handling for each service to handle every flows incoming from the various types of service alternatively. The process of flow handling function as shown in the algorithm 4.1 can be sub-divided into three parts: initialization, frame comparing, and new threshold calculation.

1) **Initialization:** The initial value for baseline and threshold is predefined as the baseline traffic of the respective service for MATA as shown in the step 1 of the algorithm 4.1. For ATA, the initial value of threshold and the average number of previous frames is 0. Moreover, the factor parameter of the EWMA formula, alpha (i.e α), is defined as 0.1 in order to receive abnormal event information as quickly as the attacker is launching the attack. For doubling the average number of previous frames, the value of p is defined as 1 aiming for avoiding the false alarms.
2) **Frame comparing:** The sFlow analyzer compares the number of incoming frames with the respective dynamic and adaptive threshold value calculated in the previous second according to the step 2 of the algorithm 4.1. In the beginning of the frame comparing (i.e. t = 0s), the analyzer compares the number of frames with the predefined initial threshold.
3) **New threshold calculation:** The number of incoming SYN frames collected from the flow definition is counted by the analyzer for each TCP service as shown in the step 3 of the algorithm 4.1. Similarly, the analyzer counts the number of all incoming frames from the respective UDP flow definition for each UDP service. After counting the number of frames, the average number of previous frames for the next second is calculated by using Equation 3.7 of the MATA, EWMA formula, as described in the step 5 of the algorithm 4.1. Then the new threshold value is calculated by taking the combination of the twice of the EWMA result and the baseline by using Equation 3.8 of MATA according to the step 6 of the algorithm 4.1.

After calculating them, the old threshold is replaced by the new threshold as presented in the step 7 of the algorithm 4.1. Moreover, the old average number of previous frames is also replaced by the current average number of previous ones as shown in the step 8 of the algorithm 4.1.

Algorithm 4.1 MATA in sFlow Analyzer

```
Input: Collected sampled flow from the flow definition: $flow$,
        Observed the baseline traffic info of the current network infrastructure:
        $baseline\ frame$;
Output: Abnormal event information: $abnormal$
Step 1: Set the initialized value:
        $PF_{t-1} \leftarrow 0$; $threshold \leftarrow baseline\ frame$;
        $baseline \leftarrow baseline\ frame$; $\alpha \leftarrow 0.1$; $percentage \leftarrow 1$;
Step 2: Compare the number of frame with the threshold value:
        $setThreshold('abnormal', \{metric: 'flow, value: threshold,$
        $byFlow: true, timeout: 1\})$;
Step 3: Count the number of frame from the flows:
        $flowCount \leftarrow Count(flow)$;
Step 4: Calculate the number of frame per second:
        $CF_t \leftarrow CalFramePerSec(flowCount)$;
Step 5: Calculate the average number of frames:
        $PF_t \leftarrow \alpha\, PF_{t-1} + (1 - \alpha)CF_t$;
Step 6: Calculate the new threshold value:
        $threshold_{new} \leftarrow \big((percentage + 1) * \ PF_t\big) + baseline$;
Step 7: Storing the current EWMA result to use as an average no. of previous traffic
        in calculating another EWMA value for the next second:
        $PF_{t-1} \leftarrow PF_t$;
Step 8: Replace the new calculated threshold to the threshold value for the next
        second:
        $threshold \leftarrow threshold_{new}$;
Step 9: End
```

4.1.1.3 Event Handling

According to the frame comparing function, once the threshold is violated by the number of incoming frames, the *setEventHandler* function in sFlow analyser produces the alerts for indicating that the abnormal event is occurring in the network.

4.1.2 Mitigation Phase

The $ddosmitigation$ application running in the ONOS controller periodically takes the abnormal event information from the sFlow analyzer via REST API in every second. In this mitigation phase, MATA and ATA operate in a different way over the event information from the sFlow analyzer because the analyzer produced the different abnormal event information in the detection phase. The detailed architecture of the mitigation phase is as shown in Figure 4.3. There are four main parts in the mitigation phase: getting event information from sFlow, finding the source switch attached to the attacker host, checking the drop flow rule already installed for the current event, and installation of flow rule with drop action.

1) **Getting event information from sFlow:** The $\boldsymbol{ddosmitigation}$ application takes the event information from the sFlow analyzer periodically via the URL such as "$\boldsymbol{http://127.0.0.1:8008/events/json}$".
2) **Finding the source switch attached to the attacker host:** In order to get the attacker host, the source MAC address is extracted from the taken JSON file information. Then, find the source $\boldsymbol{HostID}$, source $\boldsymbol{Host}$, and source $\boldsymbol{DeviceId}$ via the source MAC address for getting the source switch ID attached to the attacker host.
3) **Checking the drop flow rule already installed for the current event:** This sub-session is to avoid installing drop flow frequently for the same event information as the $\boldsymbol{ddosmitigation}$ application receives the information in every second. Moreover, the application aims to install temporarily drop flow rule for the first-time abnormal events and a permanent one for the frequent abnormal events. Thus, it controls the incoming events and installed drop flow rules by using the timestamp value. If the same event information is received again by the application after one minute from the time of installing drop flow for it, the application will install the permanent drop flow rule for it.
4) **Installation of Flow Rule with Drop Action:** In order to install a flow rule in the SDN application, it needs first to define the source and destination point of the flow, the action for the flow, the switch ID, and the priority, and the type of the flow (i.e. temporary or permanent). Thus, the $\boldsymbol{ddosmitigation}$ application extracts the source IP and destination IP from the event information. Then, the application installs drop flow rule into the source

Device ID with the drop action and the higher priority than the other normal flow rules with a particular type of flow.

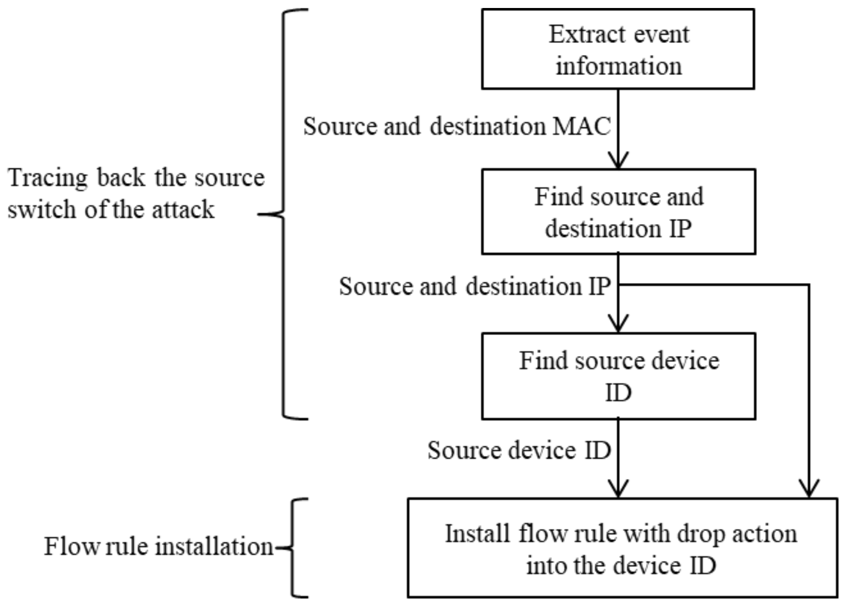

Tracing back the source switch of the attack
Flow rule installation
Extract event information
Source and destination MAC
Find source and destination IP
Source and destination IP
Find source device ID
Source device ID
Install flow rule with drop action into the device ID

Figure 4.3 Mitigation Process in *ddosmitigation* Application

4.1.2.1 MATA–based Mitigation

As soon as the *ddosmitigation* application receives the information, it first extracts the source and destination IP address from the information and finds the source switch connected with the attacker host by using the source IP address. Then, the application installs temporarily drop flow rule for 60 seconds into the source switch of the attack for discarding the flooding packets at the nearest point to the attacker host. If the application receives again the previous event information when the flow rule has been expired, it installs permanent drop flow rules for such event information.

4.1.2.2 ATA–based Mitigation

The ATA-based *ddosmitigation* application does not discard any frames as soon as it receives the event information from the analyzer. It firstly confirms whether the received event information is signaled the attacks or not because some information might be the false alarms. In order to define the event formation that is not false alarms, the algorithm predefines the number of consecutive threshold violation within a time for each service.

The application monitors and counts the number of consecutive event information within a predefined time and compares the number of information with the predefined value. If the predefined value is exceeded by the number of information, then it finds the source switch connected with the attacker host. Finally, the application installs the drop flow rule into the source switch of the attack.

4.2 Chapter Summary

This chapter described not only the overall system architecture of flooding attack detection and mitigation but also the sub-components of it in detail. Moreover, it presented the workflow of the algorithms for each service separately with the procedure and flowchart. This chapter also described specifically for the detection and mitigation system with MATA and ATA algorithms.

CHAPTER 5

IMPLEMENTATION AND EVALUATION OF FLOODING ATTACK DETECTION AND MITIGATION

In this chapter, the implementation and evaluating of flooding attack detection and mitigation system is described with the experimental testbed design, and experimental results produced from the two scenarios depending on the types of flooding attack: specific SYN flooding attack, and overall flooding attacks. Moreover, it also presents the design of experimental testbed with hardware and software specifications.

5.1 Experimental Testbed Design

For assessing the flooding attack detection and mitigation over the SDN network infrastructure, the experimental testbed is designed as shown in Figure 5.1. According to the three-layer architecture of the SDN network, this system is composed of a network topology emulated by mininet as a data forwarding layer, ONOS controller as a control layer, and *ddosmitigation* application activating at the application layer. In order to collect and analyze all the traffic passing through the whole SDN network, this system uses the sFlow-RT analyzer. sFlow datagram connection is for carrying the sample packets between sFlow analyzer and the network. The abnormal event information of the sFlow analyzer can be extracted via the REST API from the ddosmitigation application running in the ONOS controller. OpenFlow connection is interconnected between controller and mininet for discovering the topology and installing flow rules into the switches.

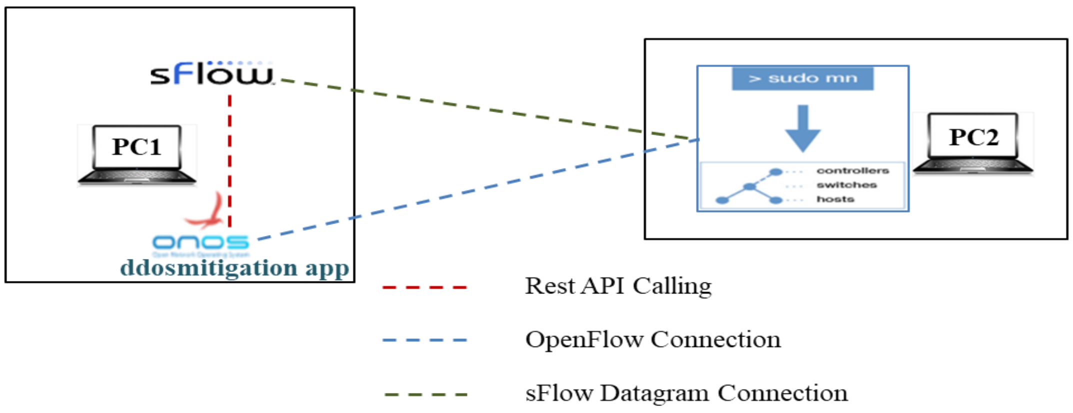


Figure 5.1 Experimental Testbed Design

Two laptop PCs are used in implementing the testbed. ONOS controller and sFlow-RT analyzer are running on PC1 and the mininet network is running on PC2. The two PCs are connected by an Ethernet cable. The hardware specifications are listed in Table 5.1. The software version information is also presented in Table 5.2.

Since the analyzer is directly connected to the mininet emulator by using peer to peer connection, the sampling rate is set to 1 in 1 packet and the polling interval is specified for one second in the sFlow analyzer.

Table 5.1 Hardware Information for Experimental Testbed

Parameters	**Descriptions for PC1**	**Descriptions for PC2**
CPU	Core i7- 4500U CPU @ 1.80GHz, 64 bits	Core i5-3210M CPU @2.50GHz x 4, 64 bits
RAM	8 GB	4 GB
OS	Linux Ubuntu Desktop 16.04LTS	Linux Ubuntu Desktop 16.04LTS

Table 5.2 Software Information for Experimental Testbed

Parameters	**Descriptions**
ONOS	1.8 (Ibis)
sFlow-rt	2.0-r1 121
Mininet	2.2.1
Open vSwitch	2.9.2
OpenFlow	1.3

5.2 Experimental Results

Two scenarios are used to produce the various types of experimental results for comparing the ATA-based with the MATA-based flooding attack detection and mitigation system. Firstly, this system is evaluated over the SYN flooding attack as the scenario 1. Then, the various types of flooding attack are assessed in the scenario 2.

5.2.1 Scenario 1: SYN Flooding Attack Detection and Mitigation

SYN flooding attack detection and mitigation are tested and implemented over the network running only web server. Thus, web traffic is solely generated and analyzed on the virtual environment in this part of the system. Since EWMA formula is included in both ATA and MATA, the factor parameter α value is defined as 0.5 for taking an equal factor of the current number of SYN frames and the previous average one, and the value for percentage parameter p is assigned as 1 in order to double the previous average number of SYN frames. The value of baseline parameter b in MATA is defined as 150 for the current network topology. In order to get the value for baseline of the network, we observed the network for three times of five minutes while all normal hosts are accessing the server, and chose the maximum number of SYN frames. The value of each parameter is listed in Table 5.3.

Table 5.3 Parameter Setting for MATA

Parameter	Value
Factor α	0.5
Percentage p	1
Baseline b	150 frames

For MATA, the $ddosmitigation$ application installs drop flow rule as soon as the event information is received because the algorithm avoids false alarms in the detection module.

For ATA, the $ddosmitigation$ application installs drop flow rule after a minimum number of consecutive event information is received. In this case, the application decides the receiving event as the abnormal event if the same events are received continuously for more than one second and the number of events is greater than five.

The main objective of this system is to forward the normal packets from the normal user hosts to the server and drop the abnormal packets from the attacker host at the nearest switch or the source switch of the host as the color traffic shown in Figure 5.3. According to the objective, the network topology constructed by using mininet emulator [67] is used. It consists of three OpenFlow switches, one victim web server, one attacker and three normal users as listed in Table 5.4. All links are configured with 100 Mbps.

In this scenario, the monitoring result of the SDN network has been evaluated for three minutes. During this time, baseline traffic was generated by concurrent access to the web server from normal hosts and an SYN flooding attack was launched for one minute from an attacker host. The duration of monitoring and attack time is demonstrated in Figure 5.2.

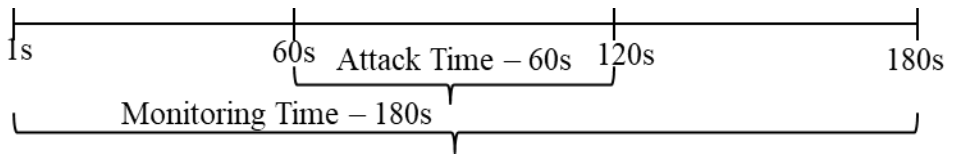


Figure 5.2 Duration for the Monitoring and Attacking

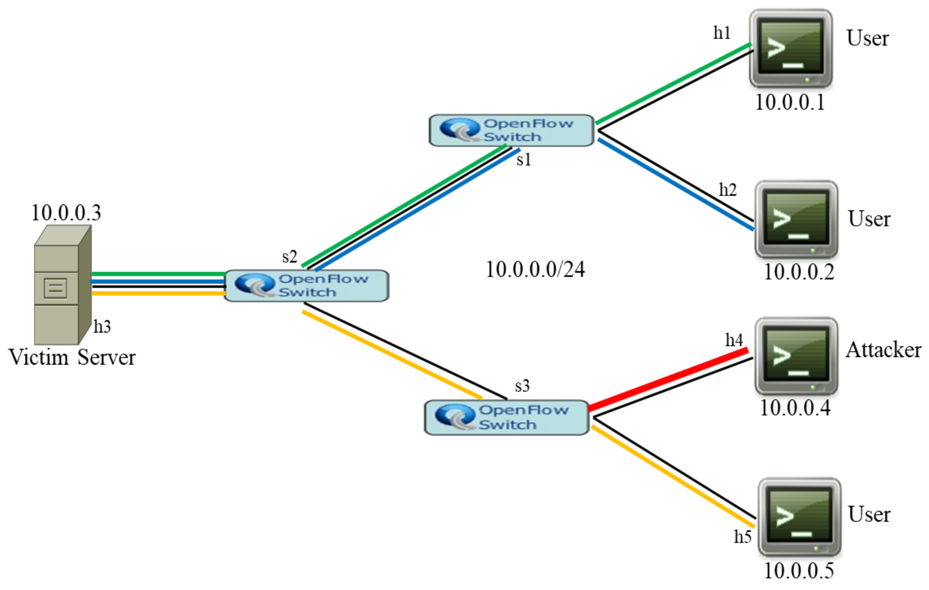


Figure 5.3 Detailed Network Topology of Experimental Testbed

Table 5.4 Experimental Testbed Information

Type	Host name	IP address
Web server (victim)	h3	10.0.0.3
Attacker	h4	10.0.0.4
Normal hosts	h1,h2,h5	10.0.0.1, 10.0.0.2, 10.0.0.5

This scenario includes four steps:

Step 1: A Simple HTTP web server was set up with port 80 at host h3 by using the command: $python\ \ -m\ \ SimpleHTTPServer\ \ 80$ &.

Step 2: The web server was accessed from other normal hosts h1, h2, and h5 continuously and concurrently by using $wget$ command [68] with 50000 loops: for NUM in 'seq 1 1 50000'; $do\ wget\ -O\ -$ 10.0.0.3; $done$. At the same time, the web traffic from server host h3 was started to monitor for three minutes by using a packet capturing tool, $tcpdump$ [69]: $timeout\ \ 180\ \ tcpdump\ \ -i\ \ any\ \ -w\ \ mata.pcap$.

Step 3: After one minute from the start of the monitoring, an SYN flooding attack was launched for one minute at the rate of 100 thousand SYN packets per second from the attacker host h4 by using $hping3$ command [70]: $timeout\ \ 60\ \ hping3\ \ -i\ \ u1\ \ -S\ \ -p\ \ 80\ \ 10.0.0.3$.

Step 4: After one minute attack, how many abnormal packets that the system can filter were defined by checking the drop flow rule in the source switch s3 of the attacked host with the command: $ovs\ \ -ofctl\ \ dump\ \ -flows\ \ s3$.

After performing the scenario, the value of each performance parameter is defined as follows:

1) **True Positive (TP):** The number of packets passing through the drop flow rule is considered as the value of TP.
2) **False Negative (FN):** The number of packets in filtering traffic (i.e. attacker host h4 to server host h3) from the captured result, $mata.pcap$, is defined as the value of FN.
3) **True Negative (TN):** The value for TN is calculated by subtracting the total number of packets in the captured result, $mata.pcap,$ to the value of FN.
4) **False Positive (FP):** Since the implementation of this system is already considered to avoid the occurrences of the false alarm, the value of FP is zero.

5.2.1.1 Experimental Results for Scenario 1

The experimental results for the comparison of ATA with MATA are described with three sections: adaptive threshold vs incoming SYN frames, filtering results by using $ddosmitigation$ application, and evaluation of performance parameters.

A. Adaptive Threshold vs Incoming SYN Frames

The comparative results of the adaptive threshold over the incoming SYN frames produced by ATA and MATA are shown in Figure 5.4 and 5.5, respectively. These results are provided by the sFlow analyzer in the detection module. Since ATA avoids false alarms in the mitigation module, the alarms can be seen at the detection module. Each result is divided into three states: initial state (before 10s), attack state (60s - 120s), and normal state ((20s - 59s) and (121s - 180s)).

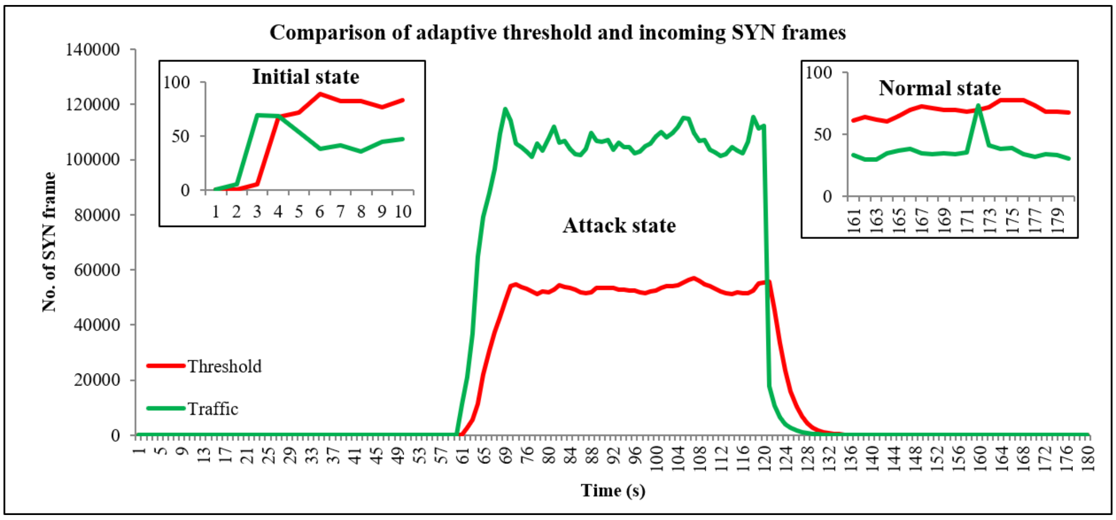


Figure 5.4 Adaptive Threshold Vs Incoming Traffic by ATA

1) **Initial state**:
    a. **ATA:** Since the initial threshold value is zero, the threshold is surpassed by the incoming SYN frames as shown in the initial state of Figure 5.4.
    b. **MATA:** Although the initial threshold value is baseline, the false alarms are not raised in this state.
2) **Attack state**: Both algorithms could detect the abnormal packets as soon as the attack is launching by the attacker.
3) **Normal state**:
    a. **ATA**: The false alarms are raised when the number of incoming SYN frames is strongly increased as shown in the normal state of Figure 5.4.

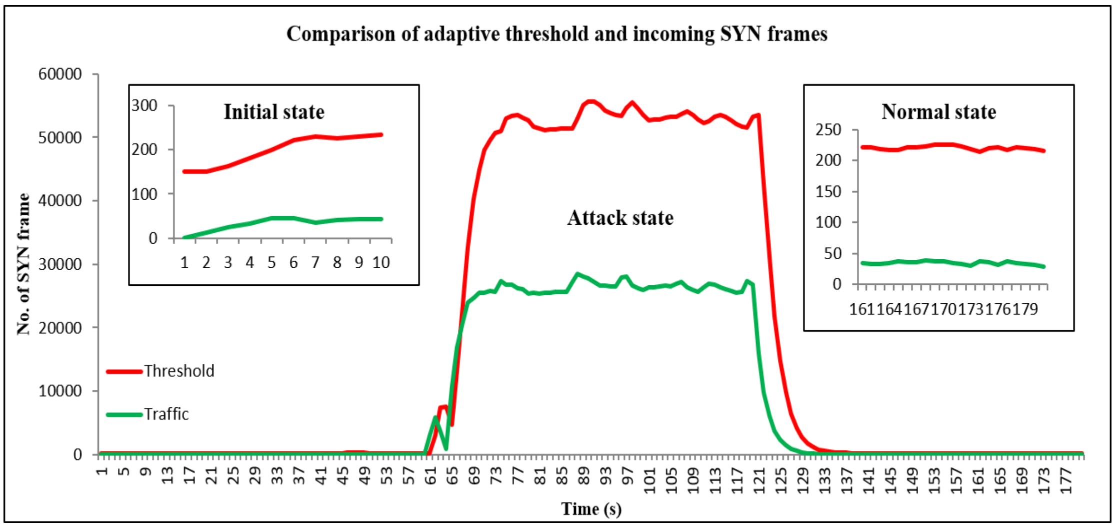


Figure 5.5 Adaptive Threshold Vs Incoming Traffic by MATA

b. **MATA:** According to the added baseline, the minimum threshold is the same as the baseline. As a result, false alarms are not raised in normal conditions as shown in the normal state of Figure 5.5.

B. Filtering Results

The filtering results using *ddosmitigation* applications based on the detection results of ATA and MATA are shown in Figure 5.6. The SYN flooding packets reach the victim web server about 46.6% when the network is filtering with ATA-based *ddosmitigation* application while the MATA-based application incorrectly allows the abnormal packets about 1.8%.

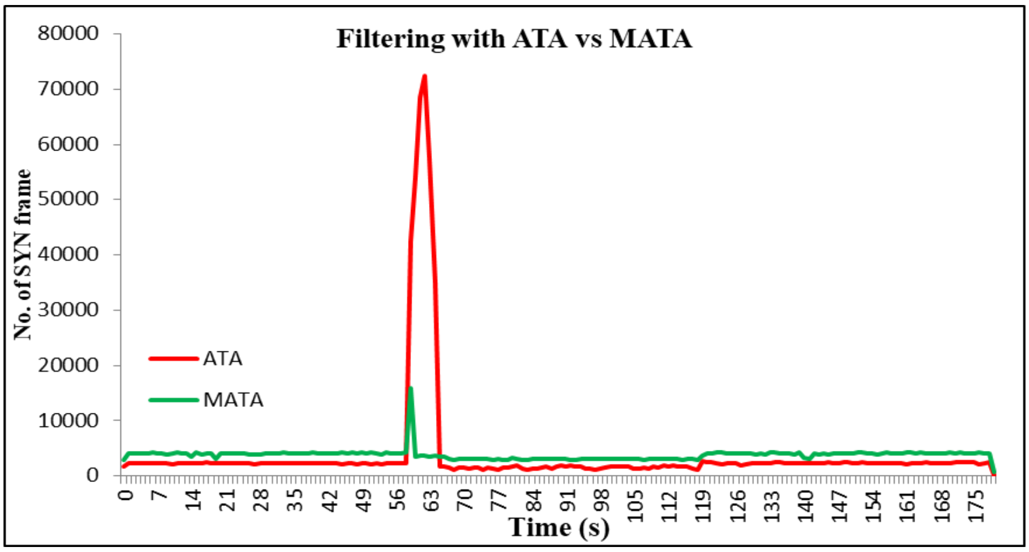


Figure 5.6 Comparison of Filtering Result of *ddosmitigation* Application with ATA and MATA.

The network not being filtered by any *ddosmitigation* application might allow the attack up to 92.9%. Thus, the reduction of the percentage of attack by using ATA-based application and MATA-based application is 46.3% and 91.1%, respectively.

C. Evaluation of Performance Parameters

The percentage of FNR, DR, and ACC of each algorithm describing the MATA is more effective than that of ATA as listed in Table 5.5. These results are the average of ten runs for each algorithm. The average FNR is reduced from 6.15 % to 0.59 %, and the average DR and ACC also increased from 93.85% to 99.41% and 94.3% to 99.47%, respectively.

Table 5.5 Comparison of Performance between ATA and MATA

Performance	Type of Algorithm	
	ATA	MATA
FNR(%)	6.15	0.59
DR(%)	93.85	99.41
ACC(%)	94.30	99.47

5.2.2 Scenario 2: Flooding Attacks Detection and Mitigation

In this section, the various types of flooding attacks are tested and implemented over the virtual network environment that is running various types of servers and generating the various types of network traffic as in the real network environment. For evaluating this system can detect and mitigate not only SYN flooding attack but also the other flooding attacks, this scenario presents the various types of flooding attack detection and mitigation.

The testbed for testing this scenario is composed of four OpenFlow switches, one controller, six servers, and twelve clients as shown in Figure 5.7. One switch is connected with all servers and the others are connected with the clients. One of the client hosts is treated as an attacker and the remaining clients are benign users. Each server is a target victim alternatively.

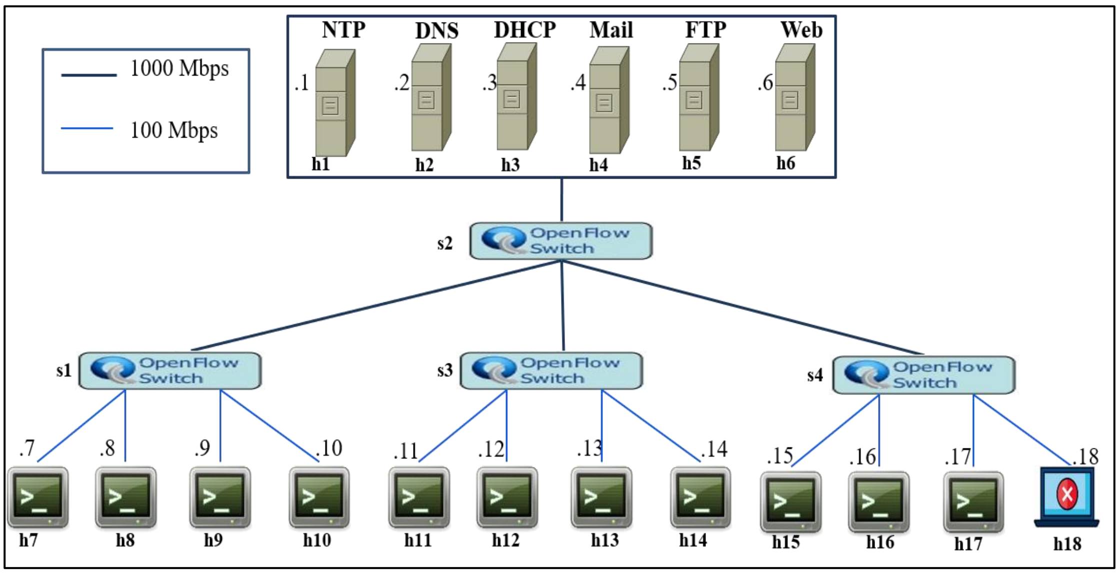


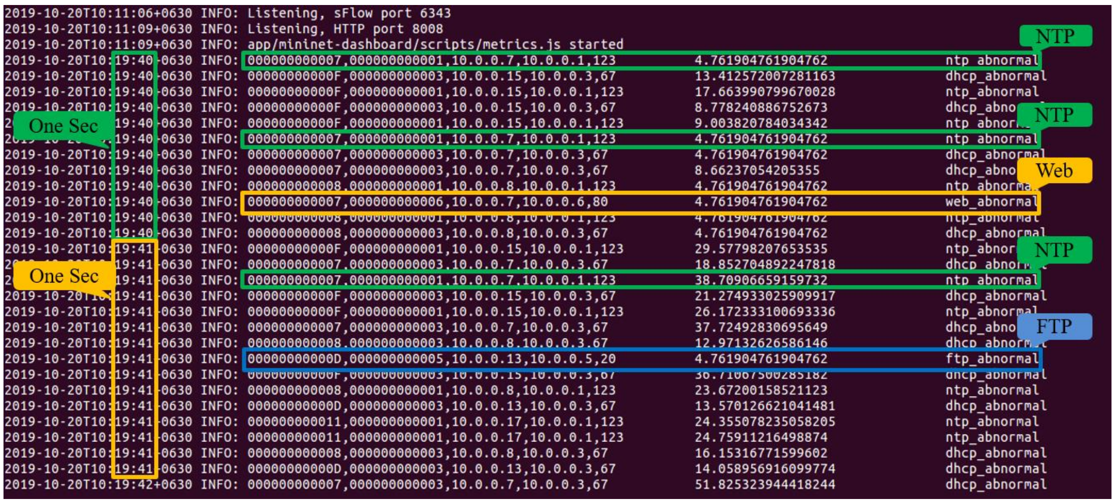
Figure 5.7 Network Topology for the Experimental Testbed

Table 5.6 Testbed Information

Type	Host name	IP address
Servers	h1 – h6	10.0.0.1 – 10.0.0.6
Clients	h7 – h17	10.0.0.7 – 10.0.0.17
Attacker	h18	10.0.0.18
Switches	s1 – s4	-

5.2.2.1 False Alarms Avoidance in ATA-based Detection

The ATA raises the alarm signal after a minimum number of consecutive threshold violations for avoiding false alarms. According to the result from the analysis of sFlow event information, the real abnormal event information and false alarms is differentiated by defining the number of same event information occurring continuously within a period of time. As shown in Figure 5.8, the sFlow analyzer produced the same NTP service's abnormal event information continuously during the two seconds for three times. Actually, these events information are false alarms occurring in normal conditions. Thus, more than three same event information occurred within the two seconds is defined as the real abnormal event information for NTP service. Similarly, the number of event information and its duration are analyzed and predefined for each service as shown in Table 5.7.



```
2019-10-20T10:11:06+0630 INFO: Listening, sFlow port 6343
2019-10-20T10:11:09+0630 INFO: Listening, HTTP port 8008
2019-10-20T10:11:09+0630 INFO: app/mininet-dashboard/scripts/metrics.js started
2019-10-20T10:19:40+0630 INFO: 000000000007,000000000001,10.0.0.7,10.0.0.1,123    4.761904761904762     ntp_abnormal
2019-10-20T10:19:40+0630 INFO: 00000000000F,000000000003,10.0.0.15,10.0.0.3,67    13.412572007281163    dhcp_abnormal
2019-10-20T10:19:40+0630 INFO: 00000000000F,000000000001,10.0.0.15,10.0.0.1,123   17.663990799670028    ntp_abnormal
2019-10-20T10:19:40+0630 INFO: 00000000000F,000000000003,10.0.0.15,10.0.0.3,67    8.778240886752673     dhcp_abnormal
2019-10-20T10:19:40+0630 INFO: 00000000000F,000000000001,10.0.0.15,10.0.0.1,123   9.003820784034342     ntp_abnormal
2019-10-20T10:19:40+0630 INFO: 000000000007,000000000001,10.0.0.7,10.0.0.1,123    4.761904761904762     ntp_abnormal
2019-10-20T10:19:40+0630 INFO: 000000000007,000000000003,10.0.0.7,10.0.0.3,67     4.761904761904762     dhcp_abnormal
2019-10-20T10:19:40+0630 INFO: 000000000007,000000000003,10.0.0.7,10.0.0.3,67     8.66237054205355      dhcp_abnormal
2019-10-20T10:19:40+0630 INFO: 000000000008,000000000001,10.0.0.8,10.0.0.1,123    4.761904761904762     ntp_abnormal
2019-10-20T10:19:40+0630 INFO: 000000000007,000000000006,10.0.0.7,10.0.0.6,80     4.761904761904762     web_abnormal
2019-10-20T10:19:40+0630 INFO: 000000000008,000000000001,10.0.0.8,10.0.0.1,123    4.761904761904762     ntp_abnormal
2019-10-20T10:19:40+0630 INFO: 000000000008,000000000003,10.0.0.8,10.0.0.3,67     4.761904761904762     dhcp_abnormal
2019-10-20T10:19:41+0630 INFO: 00000000000F,000000000001,10.0.0.15,10.0.0.1,123   29.57798207653535     ntp_abnormal
2019-10-20T10:19:41+0630 INFO: 000000000007,000000000003,10.0.0.7,10.0.0.3,67     18.852704892247818    dhcp_abnormal
2019-10-20T10:19:41+0630 INFO: 000000000007,000000000001,10.0.0.7,10.0.0.1,123    38.70906659159732     ntp_abnormal
2019-10-20T10:19:41+0630 INFO: 00000000000F,000000000003,10.0.0.15,10.0.0.3,67    21.274933025909917    dhcp_abnormal
2019-10-20T10:19:41+0630 INFO: 00000000000F,000000000001,10.0.0.15,10.0.0.1,123   26.172333100693336    ntp_abnormal
2019-10-20T10:19:41+0630 INFO: 000000000007,000000000003,10.0.0.7,10.0.0.3,67     37.72492830695649     dhcp_abnormal
2019-10-20T10:19:41+0630 INFO: 000000000008,000000000003,10.0.0.8,10.0.0.3,67     12.97132626586146     dhcp_abnormal
2019-10-20T10:19:41+0630 INFO: 00000000000D,000000000005,10.0.0.13,10.0.0.5,20    4.761904761904762     ftp_abnormal
2019-10-20T10:19:41+0630 INFO: 00000000000F,000000000003,10.0.0.15,10.0.0.3,67    36.71067500285182     dhcp_abnormal
2019-10-20T10:19:41+0630 INFO: 000000000008,000000000001,10.0.0.8,10.0.0.1,123    23.67200158521123     ntp_abnormal
2019-10-20T10:19:41+0630 INFO: 00000000000D,000000000003,10.0.0.13,10.0.0.3,67    13.570126621041481    dhcp_abnormal
2019-10-20T10:19:41+0630 INFO: 000000000011,000000000001,10.0.0.17,10.0.0.1,123   24.355078235058205    ntp_abnormal
2019-10-20T10:19:41+0630 INFO: 000000000011,000000000001,10.0.0.17,10.0.0.1,123   24.75911216498874     ntp_abnormal
2019-10-20T10:19:41+0630 INFO: 000000000008,000000000003,10.0.0.8,10.0.0.3,67     16.15316771599602     dhcp_abnormal
2019-10-20T10:19:41+0630 INFO: 00000000000D,000000000003,10.0.0.13,10.0.0.3,67    14.058956916099774    dhcp_abnormal
2019-10-20T10:19:42+0630 INFO: 000000000007,000000000003,10.0.0.7,10.0.0.3,67     51.825323944418244    dhcp_abnormal
```

Figure 5.8 Event Information from sFlow Analyzer

Table 5.7 Number of Consecutive Threshold Violation

Type of service	No. of event information (n)	Time (second)
Web	n > 1	2
FTP	n > 1	2
Mail	n > 1	2
DNS	n > 6	2
NTP	n > 3	2

By default, the function of the sFlow analyzer for comparing the incoming traffic with the threshold value (i.e. $setThreshold$) raises the alarms as soon as the threshold is violated. As a result, the avoiding function of false alarm in ATA could not implement in the detection phase of the sFlow analyser. Thus, it is implemented in the $ddosmitigation$ application of the mitigation phase.

5.2.2.2 False Alarms Avoidance in MATA-based Detection

For avoiding false alarms, this algorithm is taken into account the baseline of the network. By adding baseline into the comparison of incoming traffic and the calculated dynamic threshold, false alarms can be avoided significantly.

In order to define the value of baseline traffic for the current network topology, the network was observed for one minute while all normal users are accessing all available network services concurrently. As this system is experimenting in the virtual mininet network environment, in order to get the baseline similar to the actual baseline traffic of the real network environment, $D-ITG$ (Distributed Internet Traffic Generator) tool [80] was used for generating the network traffic in the virtual network. This tool generates the traffic with Inter Departure Time (IDT) and Packet Size (PS) using stochastic models such as uniform, constant, exponential, pareto, cauchy, normal, poisson, gamma, and weibull distribution. It can also generate the transport layer traffic (i.e. TCP, UDP) and application layer traffic (i.e. DNS, Telnet, VoIP).

The traffic generation model [72] described that poisson distribution can be used to generate the traffic with the number of incoming packets or calls per time unit (i.e. IDT). Moreover, the traffic including the length of each phone call (i.e. PS) can be generated by the exponential distribution.

The theoretical traffic model [4] can be summarized as shown in Table 5.8. The IDT for the telnet traffic and new network transfer protocol (NNTP) traffic can be generated by using poisson distribution and weibull distribution respectively. Moreover, both IDT and PS of VoIP traffic is generated by using exponential distribution. The PS of NNTP, SMTP, and FTP traffic during the whole session are generated by using log2-normal distribution. In addition, pareto distribution is used to produce the PS of web and FTP traffic during a burst session.

Table 5.8 Theoretical Traffic Model for Generating IDT and PS

Service	IDT	PS
Telent	Poisson distribution	-
NNTP	Weibull distribution	Log2-normal distribution
SMTP	-	Log2-normal distribution
FTP	-	Log2-normal distribution during the whole session Pareto distribution during a burst session
WWW	-	Pareto distribution
VoIP	Exponential distribution	Exponential distribution

The combination of the traffic generation model and theoretical traffic model was referred for generating the various types of virtual network traffic similar to the real network traffic as shown in Table 5.9. The IDT of all traffic is generated by using poisson distribution. Log2-normal distribution is used for the generation of PS for SMTP and FTP traffic. Moreover, the PS of WWW traffic is generated by pareto distribution. According to the description of the theoretical traffic model, each type of traffic with different percentages of the packet was generated as shown in Table 5.10.

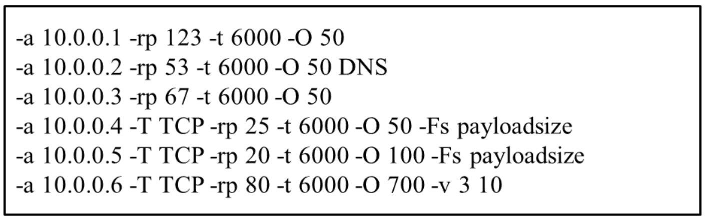
Table 5.9 Assumption for Generating of IDT and PS for Each Service

Service	IDT	PS
NTP, DHCP, DNS	Poisson distribution	-
SMTP	Poisson distribution	Log_2-normal distribution
FTP	Poisson distribution	Log_2-normal distribution
WWW	Poisson distribution	Pareto distribution

Table 5.10 Assumption for Packet Generation Rate

Service		Percentage of the packet (%)
TCP	WWW	70
	FTP	10
	SMTP	5
UDP	NTP	5
	DNS	5
	DHCP	5

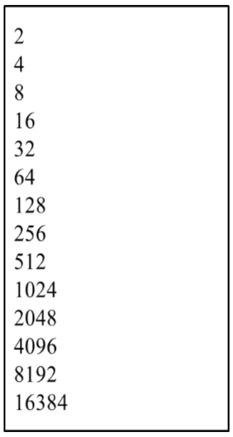
After defining the traffic generation format, firstly we setup the NTP, DHCP, DNS, SMTP, FTP, and Web server on the mininet host h1, h2, h3, h4, h5, and h6 respectively by running $./ITGRecv$ command. Then, all client hosts except the attacker host h18 access the server concurrently by using $./ITGSend\ script\_file$. The list of commands for accessing the servers is written in the $script\_file$ as shown in Figure 5.9.



```
-a 10.0.0.1 -rp 123 -t 6000 -O 50
-a 10.0.0.2 -rp 53 -t 6000 -O 50 DNS
-a 10.0.0.3 -rp 67 -t 6000 -O 50
-a 10.0.0.4 -T TCP -rp 25 -t 6000 -O 50 -Fs payloadsize
-a 10.0.0.5 -T TCP -rp 20 -t 6000 -O 100 -Fs payloadsize
-a 10.0.0.6 -T TCP -rp 80 -t 6000 -O 700 -v 3 10
```

Figure 5.9 List of Command in the $Script\_File$

To represent the log2-normal distribution for the payload size for the mail and ftp traffic is defined as shown in Figure 5.10.



```
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
```

Figure 5.10 List of Payload Size with Log2 in the $payloadsize$ File

In order to get all possible event information from the sFlow analyzer, the initial baseline and threshold value is defined as zero and the $ddosmitigation$ application is not activated in the ONOS controller. The final value for the baseline traffic of each service is defined according to the process and equation which previously described in section 3.2.2.2.

Firstly, some of the various event information produced by sFlow analyzer are listed in Table 5.11. This information includes the mixing abnormal information from the various types of services. Thus, in order to find the baseline of the specific service, this information is filtered by using some word containing in the Abnormal info field.

Table 5.11 List of Some Event Information from sFlow Analyzer

Flow key information	Frames per second	Abnormal information
2019-06-13T01:58:03+0630 INFO: 000000000011,000000000005,10.0.0.17,10.0.0.5,20	4.761904762	ftp_abnormal
2019-06-13T01:58:04+0630 INFO: 00000000000E,000000000003,10.0.0.14,10.0.0.3,67	9.259591724	dhcp_abnormal
2019-06-13T01:58:04+0630 INFO: 00000000000E,000000000003,10.0.0.14,10.0.0.3,67	28.27168373	dhcp_abnormal
2019-06-13T01:58:04+0630 INFO: 000000000011,000000000001,10.0.0.17,10.0.0.1,123	12.68794475	ntp_abnormal
2019-06-13T01:58:04+0630 INFO: 00000000000E,000000000001,10.0.0.14,10.0.0.1,123	18.55222873	ntp_abnormal
2019-06-13T01:58:04+0630 INFO: 000000000009,000000000002,10.0.0.9,10.0.0.2,53	4.761904762	dns_abnormal
2019-06-13T01:58:04+0630 INFO: 000000000009,000000000001,10.0.0.9,10.0.0.1,123	4.761904762	ntp_abnormal
2019-06-13T01:58:04+0630 INFO: 000000000009,000000000003,10.0.0.9,10.0.0.3,67	4.761904762	dhcp_abnormal
2019-06-13T01:58:04+0630 INFO: 000000000009,000000000003,10.0.0.9,10.0.0.3,67	4.761904762	dhcp_abnormal
2019-06-13T01:58:04+0630 INFO: 000000000011,000000000001,10.0.0.17,10.0.0.1,123	15.6037493	ntp_abnormal
2019-06-13T01:58:04+0630 INFO: 000000000011,000000000003,10.0.0.17,10.0.0.3,67	16.60809384	dhcp_abnormal
2019-06-13T01:58:04+0630 INFO: 000000000009,000000000006,10.0.0.9,10.0.0.6,80	4.761904762	web_abnormal

1) **NTP Service:** For getting specific abnormal information for NTP service, Abnormal info field is filtered by using the $Text\ Filter$ function and searching the field with the keyword ntp to extract any records containing the word ntp. Some of the extracted NTP's records are shown in Table 5.12.

Table 5.12 List of Some Abnormal Information for NTP Service

Flow key information	Frames per second	Abnormal information
2019-06-13T01:58:04+0630 INFO: 000000000011,000000000001,10.0.0.17,10.0.0.1,123	12.68794475	ntp_abnormal
2019-06-13T01:58:04+0630 INFO: 00000000000E,000000000001,10.0.0.14,10.0.0.1,123	18.55222873	ntp_abnormal
2019-06-13T01:58:04+0630 INFO: 000000000009,000000000001,10.0.0.9,10.0.0.1,123	4.761904762	ntp_abnormal
2019-06-13T01:58:04+0630 INFO: 000000000011,000000000001,10.0.0.17,10.0.0.1,123	15.6037493	ntp_abnormal
2019-06-13T01:58:04+0630 INFO: 000000000009,000000000001,10.0.0.9,10.0.0.1,123	4.761904762	ntp_abnormal

2) **DNS Service:** For getting specific abnormal information for DNS service, Abnormal info field is filtered by using the $Text\ Filter$ function and searching the field with the keyword dns to extract any records containing the word dns. Some of the records for the DNS's event information are listed in Table 5.13.

Table 5.13 List of Some Abnormal Information for DNS Service

Flow key information	Frames per second	Abnormal information
2019-06-13T01:58:04+0630 INFO: 000000000009,000000000002,10.0.0.9,10.0.0.2,53	4.761904762	dns_abnormal
2019-06-13T01:58:14+0630 INFO: 00000000000B,000000000002,10.0.0.11,10.0.0.2,53	6.285714286	dns_abnormal
2019-06-13T01:58:19+0630 INFO: 000000000012,000000000002,10.0.0.18,10.0.0.2,53	4.761904762	dns_abnormal
2019-06-13T01:58:20+0630 INFO: 000000000010,000000000002,10.0.0.16,10.0.0.2,53	4.761904762	dns_abnormal
2019-06-13T01:58:22+0630 INFO: 000000000010,000000000002,10.0.0.16,10.0.0.2,53	8.805756588	dns_abnormal

3) **DHCP Service:** For getting specific abnormal information for DHCP service, Abnormal info field is filtered by using the $Text\ Filter$ function and searching the field with the keyword $dhcp$ to extract any records containing the word $dhcp$. Some of the records are listed in Table 5.14.

Table 5.14 List of Some Abnormal Information for DHCP Service

Flow key information	Frames per second	Abnormal information
2019-06-13T01:58:04+0630 INFO: 00000000000E,000000000003,10.0.0.14,10.0.0.3,67	9.259591724	dhcp_abnormal
2019-06-13T01:58:04+0630 INFO: 00000000000E,000000000003,10.0.0.14,10.0.0.3,67	28.27168373	dhcp_abnormal
2019-06-13T01:58:04+0630 INFO: 000000000009,000000000003,10.0.0.9,10.0.0.3,67	4.761904762	dhcp_abnormal
2019-06-13T01:58:04+0630 INFO: 000000000009,000000000003,10.0.0.9,10.0.0.3,67	4.761904762	dhcp_abnormal
2019-06-13T01:58:04+0630 INFO: 000000000011,000000000003,10.0.0.17,10.0.0.3,67	16.60809384	dhcp_abnormal

4) **Web Service:** For getting specific abnormal information for Web service, Abnormal info field is filtered by using the $Text\ Filter$ function and searching the field with the keyword web to extract any records containing the word web. Some abnormal event information records for web service is as shown in Table 5.15.

Table 5.15 List of Some Abnormal Information for Web Service

Flow key information	Frames per second	Abnormal information
2019-06-13T01:58:04+0630 INFO: 000000000009,000000000006,10.0.0.9,10.0.0.6,80	4.761904762	web_abnormal
2019-06-13T01:58:08+0630 INFO: 00000000000F,000000000006,10.0.0.15,10.0.0.6,80	4.761904762	web_abnormal

5) **FTP Service:** For getting specific abnormal information for FTP service, Abnormal info field is filtered by using the $Text\ Filter$ function and searching the field with the keyword ftp to extract any records containing the word ftp. Table 5.16 listed some of the abnormal event information records for FTP service.

Table 5.16 List of Some Abnormal Information for FTP Service

Flow key information	Frames per second	Abnormal information
2019-06-13T01:58:03+0630 INFO: 000000000011,000000000005,10.0.0.17,10.0.0.5,20	4.761904762	ftp_abnormal

6) **Mail Service:** For getting specific abnormal information for Mail service, Abnormal info field is filtered by using the $Text\ Filter$ function and searching the field with the keyword $mail$ to extract any records containing the word $mail$. Some records are as shown in Table 5.17.

Table 5.17 List of Some Abnormal Information for Mail Service

Flow key information	**Frames per second**	**Abnormal information**
2019-06-13T01:58:09+0630 INFO: 00000000000C,000000000004,10.0.0.12,10.0.0.4,25	4.761904762	mail_abnormal

After grouping the specific abnormal information for each service, the baseline value of the respective service is identified as the maximum value of the service by using the Max function of the Microsoft Excel and rounding them to the nearest whole number. Then, the values are listed as shown in Table 5.18.

Table 5.18 List of Baseline for Each Service

Service	**Baseline**
NTP	64.60285 → **65**
DHCP	76.64288 → **77**
DNS	8.8075 → **9**
SMTP	4.761905 → **5**
FTP	4.761905 → **5**
WWW	4.761905 → **5**

In this scenario, the duration for testing and evaluation time is three minutes. During this time, all client hosts are accessing all available servers in the network concurrently and one attacker host launches the flooding attack for one minute as previously described in Figure 5.2. This scenario includes four steps:

Step 1: The NTP, DHCP, DNS, SMTP, FTP, and HTTP server were set up on the host h1, h2, h3, h4, h5, and h6, respectively by running $./ITGRecv$ command.

Step 2: After setting up the servers, all clients (from host h7 to h17) access all the servers concurrently by running the command $./ITGSend\ script\_file$ for three minutes (18000 seconds) as shown in Figure 5.9 with the different accessing time (i.e -t 18000). At the same time, the victim server was monitored by capturing all of its incoming and outgoing traffic with a packet capturing tool (i.e. $tcpdump$) such as: $timeout$ 180 $tcpdump - i\ any - w\ mata.pcap$.

Step 3: After one minute from the start of monitoring, attacker host h18 launches the flooding attack to a particular server for one minute by using $hping3$ tool such as:

a. For the FTP, SMTP, or WWW servers $timeout\ \ 60\ \ hping3\ \ -i\ \ u1\ \ -S\ \ Dst_{Port}\ \ Dst_{IP}$

b. For the NTP, or DNS servers $timeout\ \ 60\ \ hping3\ \ --flood\ \ -udp\ \ Dst_{Port}\ \ Dst_{IP}$

Step 4: After one minute attack, the number of flooding packets was checked that can be able to filter by this system. Since this flooding attack detection and mitigation system installs flow rule in the ingress switch s4 of the attacker hosts h18, the number of packet in the drop flow rule was checked at the switch s4 by using the command such as: $ovs-ofctl\ \ dump\ \ -flows\ \ s4$.

The value of each performance parameter for network security is defined according to the result of the scenario.

1) **True positive (TP):** The number of packets passing through the drop flow rule is defined as the value of TP.
2) **False Negative (FN):** The number of packets from the filtering traffic (i.e. from the attacker host h18 to victim server host) from the results of packet capturing tool (i.e. $mata.pcap$) is considered as the value of FN.
3) **True Negative (TN):** The number of packets getting from the subtraction of the number of packets of all capturing traffic from the captured result (i.e. $mata.pcap$) to the value of FN.
4) **False Positive (FP):** The value of FP is zero because this system is implemented with the avoidance of false alarms mechanism.

5.2.2.3 Experimental Results for Scenario 2

Depending on the methods used in the detection and mitigation process, the experimental results are described with two separate sections: detection results and mitigation results.

A. Detection Results

The detection results consist of five parts. The first part presents the dynamic threshold values adaptable with the incoming traffic produced by each algorithm. The second part shows the comparative results of various performance parameters (i.e. detection rate, false negative rate, and accuracy) over MATA and ATA algorithms to prove why the modified algorithm is chosen to use in detecting the flooding attack. The third parts shows the comparisons of incoming traffic and actual arrival traffic to the server for each service in order to demonstrate how this system can detect and mitigate the flooding attack effectively with the adaptive threshold. The evaluation results of the MATA with various rates of the attack are described in the fourth part of this section for indicating how the MATA can detect the various types of attacks. Moreover, the fifth part shows the evaluation results of the performance comparisons over the various attack time, and monitoring time for demonstrating the performance might be varied depending on the evaluating time taken for the network monitoring and attack launching.

i. Comparative Results of Adaptive Threshold over Incoming Traffic

Figure 5.11 and 5.12 show the DNS traffic's adaptive threshold dynamically produced by the sFlow analyzer based on the ATA and MATA, respectively. The main difference between the results of the two algorithms is that the false alarms can be seen at the initial state and normal state of the result produced by ATA while the MATA does not produce any of them. The reason is that the MATA reduces the occurrence of the false alarms significantly in the detection phase by using its modified technique. But, ATA raises some false alarms in the detection phase and avoids them in the mitigation phase. Each comparative result is divided into three states: initial state (before 10s), attack state (60s - 120s), and normal state ((11s - 50s) and (121s – 180s)).

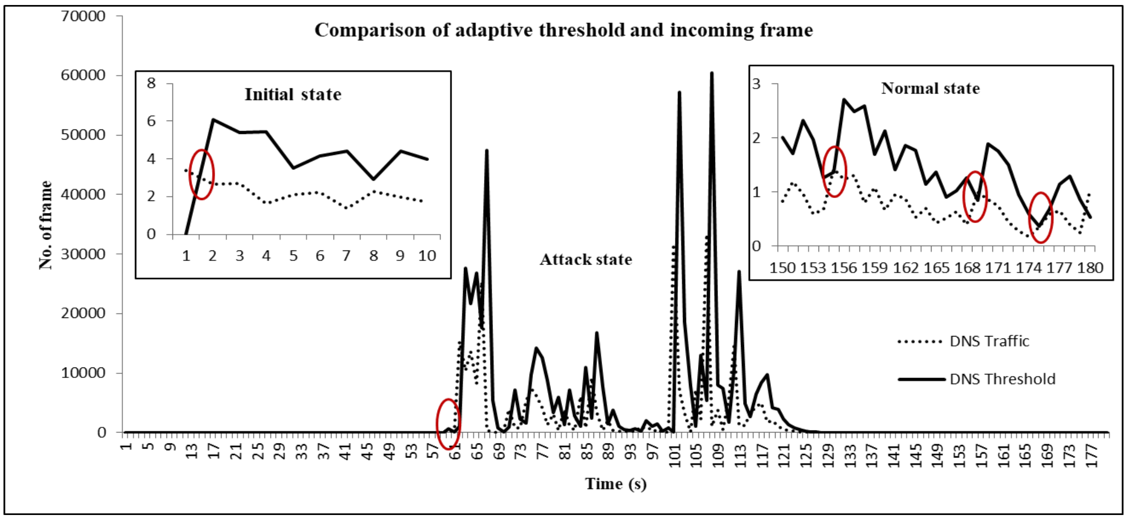


Figure 5.11 Adaptive Threshold Produced by ATA

1) **Initial state**:
   a. **ATA:** The threshold value is initialized as zero. Thus, the threshold is violated by the incoming traffic at the first comparison as shown in the initial state of Figure 5.11.
   b. **MATA:** The initial threshold value is defined as the value of the baseline. As a result, there are few false alarms at the initial state as shown in the initial state of Figure 5.12.

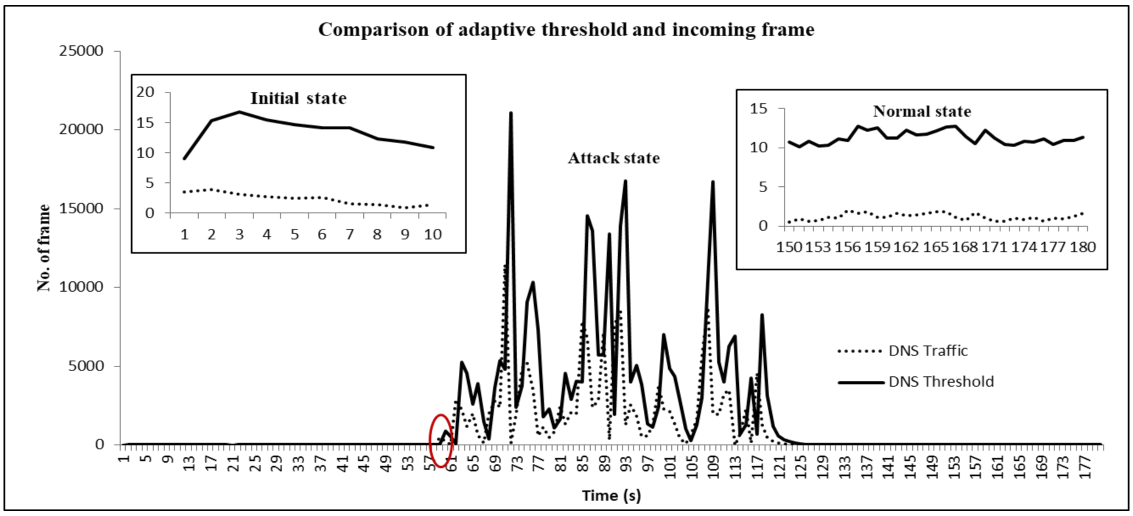


Figure 5.12 Adaptive Threshold Produced by MATA

2) **Attack state**: According to the result of the attack state of each figure, both algorithms could detect the flooding packets immediately when the attack is launching by the attacker. The delay time for the detection of both algorithms is at most one second.

3) **Normal state:**

a. **ATA:** The false alarms might be raised when the current rate of the incoming frames is slightly stronger than the previous rate as shown in the normal state of Figure 5.11.
b. **MATA:** There are few false alarms in both initial and normal states because the minimum threshold value itself is the same as the baseline traffic and then the thresholds are adaptable with the incoming frames.

ii. Comparative Results of Performance Parameters

In general, the performance parameters used in the evaluation of the network security include detection rate, false positive rate, false negative rate, and accuracy. Thus, this system also evaluates its performance by using general performance parameters. However, it does not describe the false positive rate because this system significantly reduces them in its implementation (i.e. the false positive rate is zero). The average percentage of each rate is obtained by calculating the average value of ten runs.

The comparison of detection rate produced by MATA and ATA over various types of services is described in Figure 5.13. The detection rate of MATA is slightly higher than the rate of ATA and the rate of the two algorithms is not extremely different because the original ATA is modified especially in reducing false negative rate for the avoidance of false positive rate.

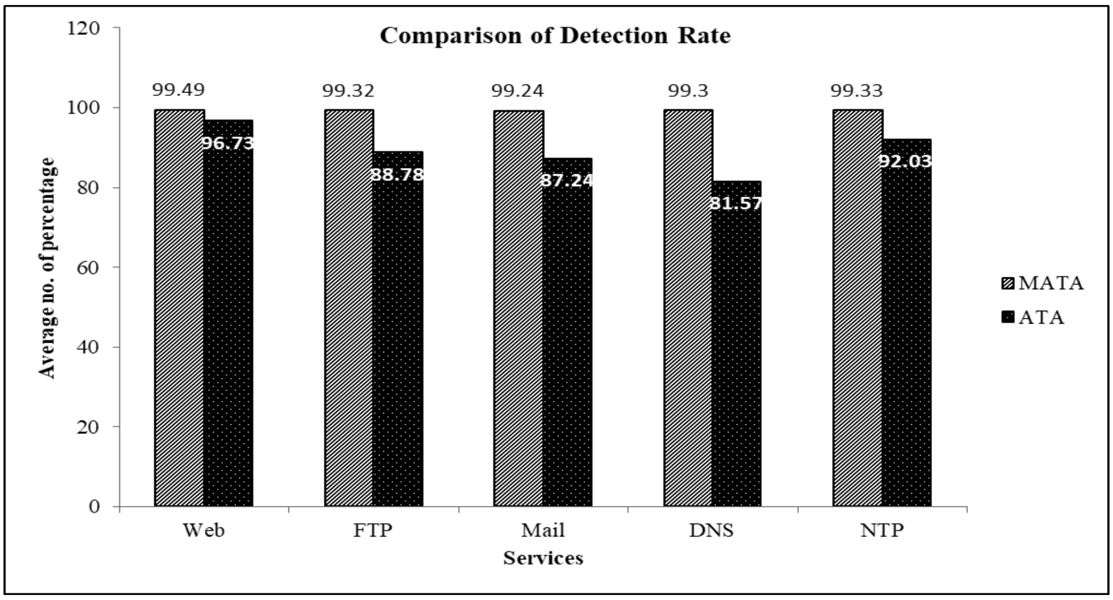


Figure 5.13 Comparisons of DR over Various Types of Services

The comparative results of the false negative rate for each service are shown in Figure 5.14. Each service has a different rate of incoming packets. The false negative rate of ATA is varied because this algorithm distinguishes the normal and malicious packets by analyzing the number of continuous incoming packets within a particular time. Thus, these rates are depending on the incoming rate of the packets of the service. MATA produces similar false negative rate for all services because it uses the same definition based on their baseline to differentiate the normal and malicious packets.

ATA produces a high value of FNR in DNS service because the incoming normal DNS packet rate is higher than the other services and the number of continuous incoming packets during the two seconds is about 6. Thus, ATA defines the incoming packet as the abnormal one when the number of the consecutive incoming packet within the two seconds is more than 6 packets as listed in Table 5.7. As a result, the number of the attack reaches the victim DNS server is high while the network is being in the attack. In contrast, the number of continuous incoming SYN packet within two seconds is 1 for the Web service. Since the normal incoming packet rate itself is very low, the attack can be detected early as soon as the number of the packet is greater than 1.

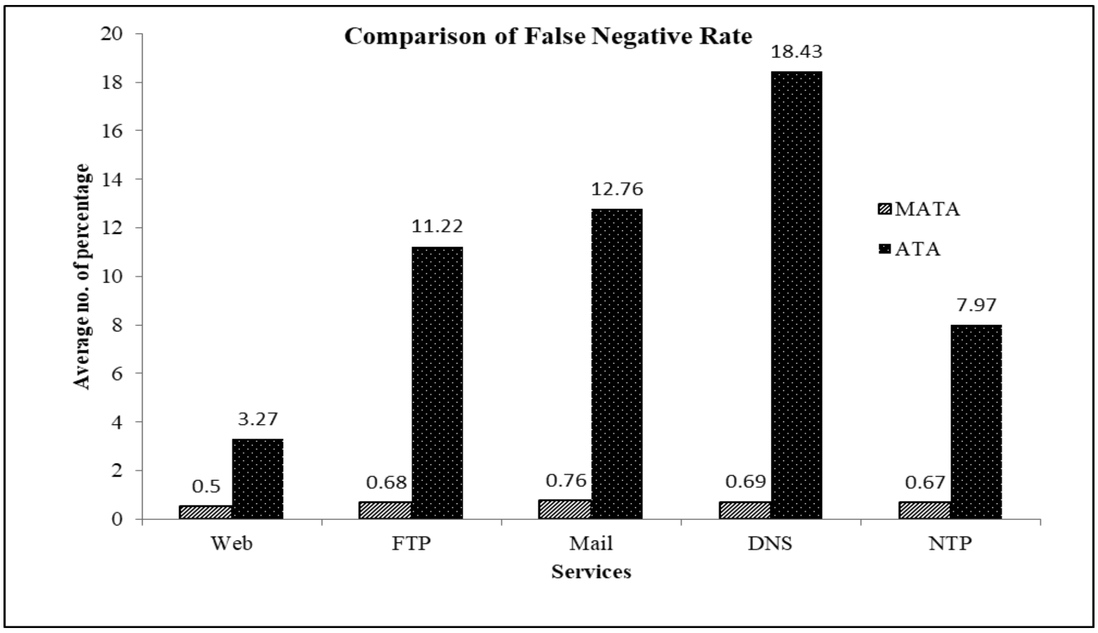


Figure 5.14 Comparisons of FNR over Various Types of Services

Similarly, ATA produces a different percentage of accuracy for each service because the calculation of accuracy is also depending on its false negative rate. It provides about 97.5% for the maximum percentage of Web service and around 83.2% for the minimum percentage of DNS service while MATA produces the accuracy above 99% for each service as shown in Figure 5.15.

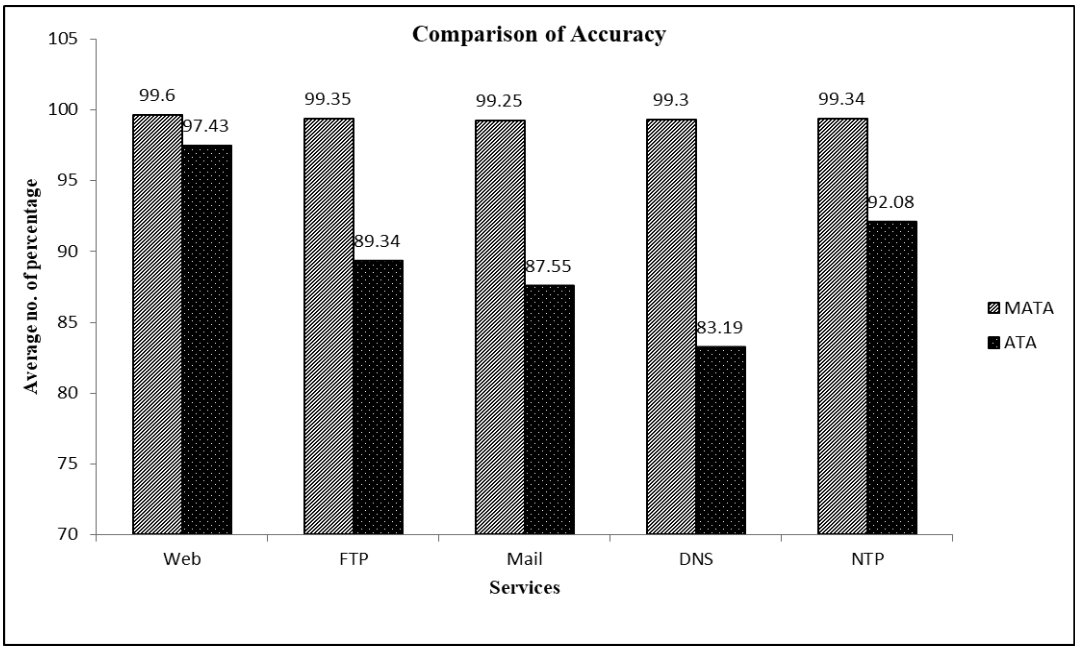


Figure 5.15 Comparisons of Accuracy over Various Types of Services

By reviewing the comparative average number of percentages for detection rate, false negative rate, and accuracy produced by MATA and ATA, MATA is an appropriate algorithm for the detection and mitigation of the flooding attacks because it provides a higher percentage of each performance parameter for all service than ATA.

iii. Comparative Results of Incoming Traffic in sFlow Analyser and Arrival Traffic in Each Server

The comparison of the incoming traffic and actual arrival traffic of each service is described in this section. The result of each figure has two parts: incoming traffic in sFlow analyzer and arrival traffic in each server. The sFlow analyzer collects the sample incoming frames from the switches and dynamically produces the adaptive threshold as shown in the upper part of each figure. In this part, all the flooding attack's frames can be seen obviously during the time of attacking (i.e. from the 60s to 120s). As soon as the flooding attack comes to the switch, the threshold is violated by the flooding frames and the analyzer raises the alarm.

As the $ddosmitigation$ application takes the event information from the analyzer in every second, it got the alarm within one second and then discards the flooding frames by installing the drop flow rule at the ingress switch. Thus, the flooding frames might reach the server about one second during the drop flow rule installation of the $ddosmitigation$ application as shown in the lower part of the figure at the time between the 60s and 61s. After the installation of the drop flow rule, the attack could not reach the server. As a result, the normal users can access the server without any interruption even the attacker is still sending the flooding attacks to the network. Thus, the arrival traffic of each service during and after the attack is remaining the same as the previous traffic before the attack as shown in the lower part of each figure. The amount of arrival traffic is different among the servers because the traffic is incoming according to the percentage defined in the experimental setup settings. Figure 5.16, 5.17, 5.18, 5.19, and 5.20 show the comparative traffic of Web, FTP, Mail, DNS and NTP, respectively.

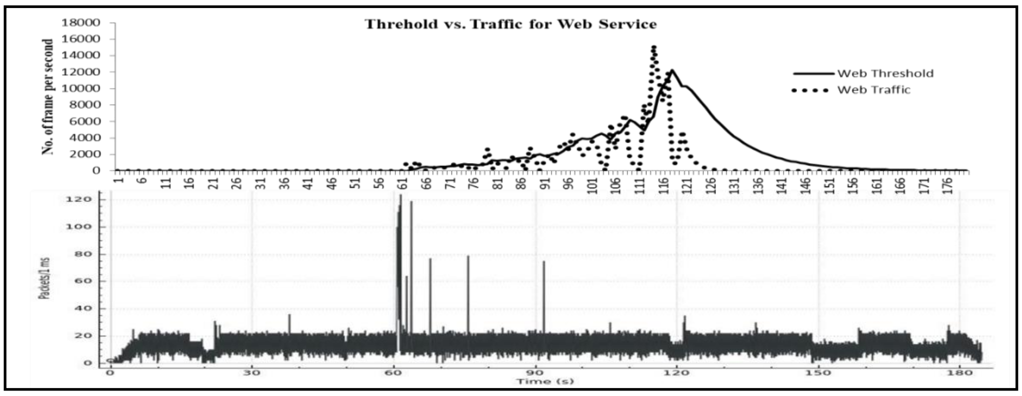


Figure 5.16 Web Traffic and its Adaptive Threshold

Since the amount of traffic generated for web service is assumed as seventy percent of all traffic, the web traffic in the lower part of Figure 5.16 is the thickest traffic among any other traffic as shown in Figures 5.17, 5.18, 5.19, 5.20.

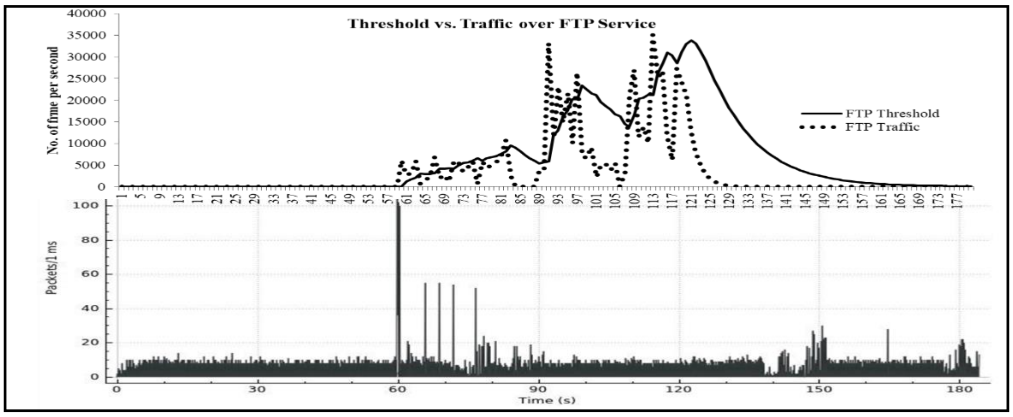


Figure 5.17 FTP Traffic and its Adaptive Threshold

As shown in the lower part of Figure 5.17, the FTP traffic is less thick than the web traffic. The reason is that the amount of traffic generation for FTP service is ten percent of all traffic in the network.

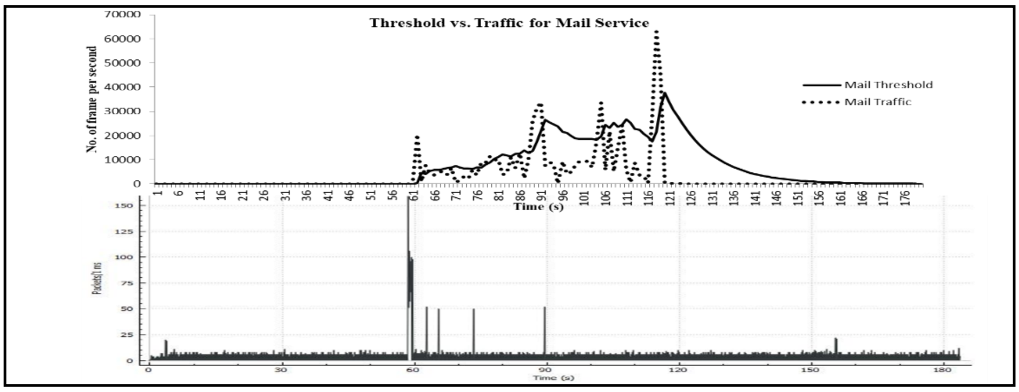


Figure 5.18 Mail Traffic and its Adaptive Threshold

According to the assumption of the traffic generation model, Mail traffic is generated only five percent of all traffic. Hence, the mail traffic as shown in the lower part of the Figure 5.18 is thinner than the other TCP traffic: Web and Mail traffic.

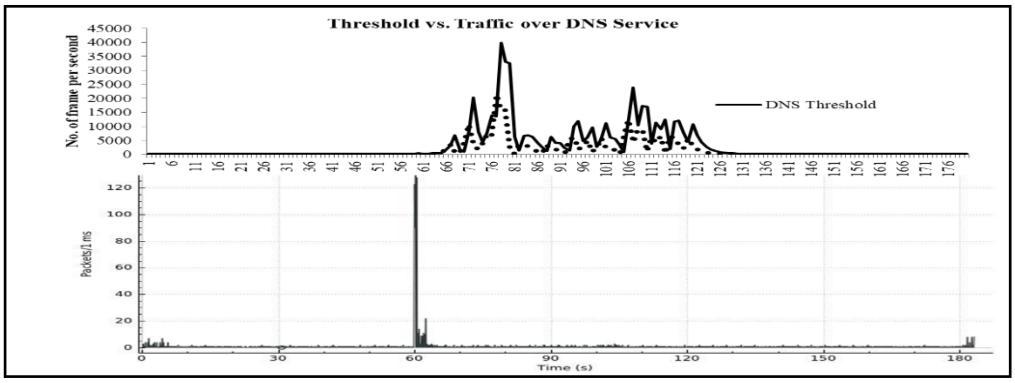


Figure 5.19 DNS Traffic and its Adaptive Threshold

The UDP traffic: DNS and NTP traffic are generated only five percent of all traffic similar to the Mail traffic. Although these two services are assumed the same amount of traffic generation with the Mail service, the traffic as shown in the lower part of the respective Figures 5.19 and 5.20 is thinner than the mail traffic shown in Figure 5.18. The reason is that the mail traffic contains the defined payload size while the two UDP traffic are generated without any payload size definition.

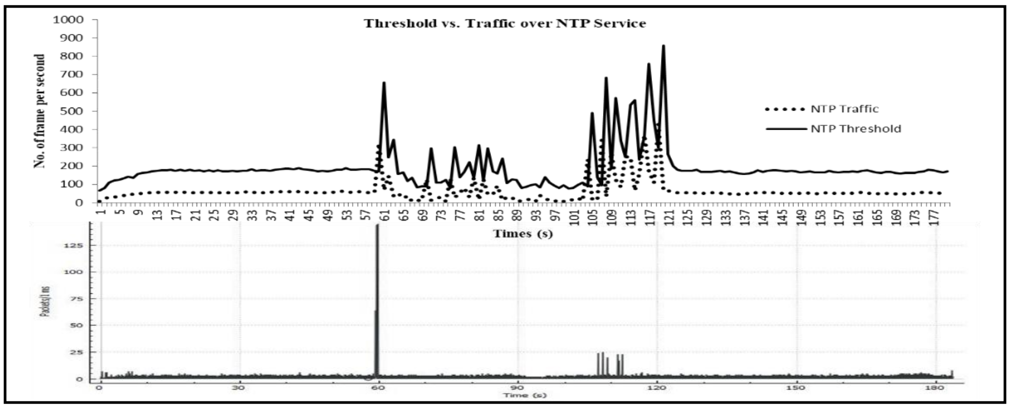


Figure 5.20 NTP Traffic and its Adaptive Threshold

iv. Comparisons of Performance over Various Attack Rates

Depending on the rate of the attack, the percentage of the detection rate and false negative rate are different. As this system is implemented in the flooding attack at the transport layer, the performance comparison of attack rate is described separately for TCP and UDP protocols. Web and DNS service is used to represent the TCP and UDP protocol, respectively.

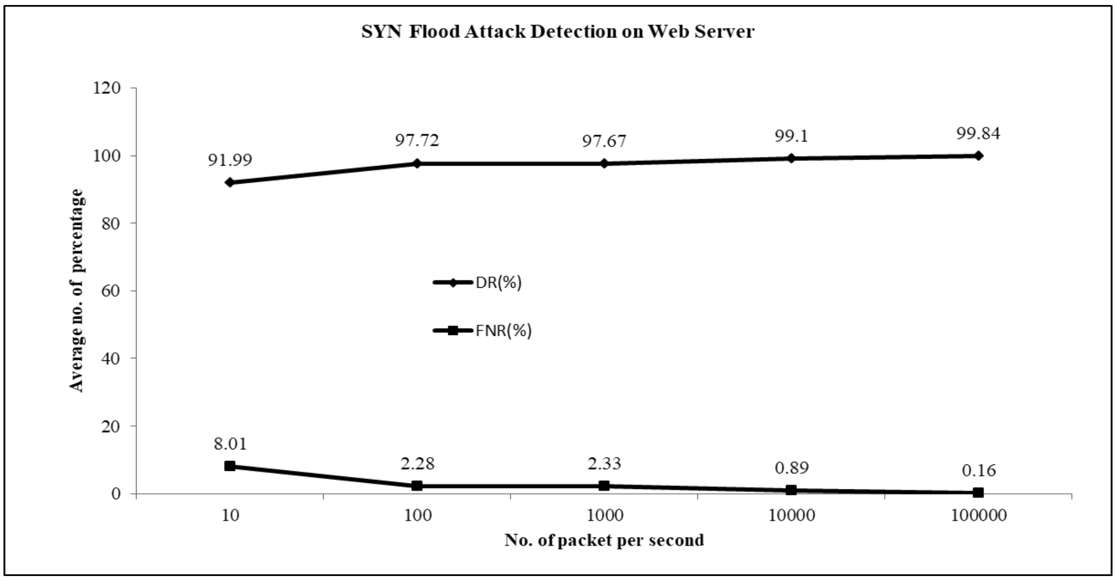


Figure 5.21 Comparisons of Various Attack Rates over Web Service

Five different rates of attack (i.e. 10 packets per second, 100 packets per second, 1000 packets per second, 10000 packets per second, and 10000 packets per second) are used to test the performance of the system. For testing each rate of attack, the $hping3$ command is used with u100000, u10000, u1000, u100, u10 and u1 to send the attack packet with 10 packets per second, 100 packets second, 1,000 packets per second, 10,000 packets per second, and 100,000 packets per second, respectively. These results are produced by averaging the results of ten runs.

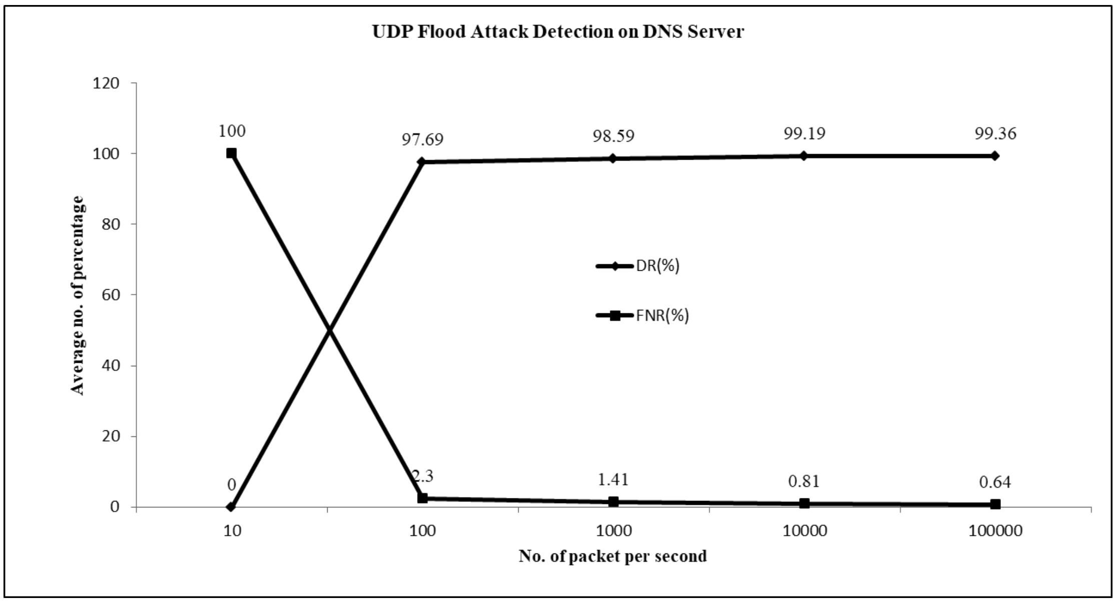


Figure 5.22 Comparisons of Various Attack Rates over DNS Service

According to the results of the detection rate and false negative rate as shown in Figure 5.21 and 5.22, the detection rate of the MATA algorithm can be determined as the higher the attack rate, the higher the detection rate. In contrast, the false negative rate can be defined as the higher the attack rate, the lower the false negative rate. Although the detection mechanism using this algorithm can detect all attack rates for Web traffic, it is not capable to detect the lowest rate of attack (i.e. 10 packets per second) for DNS traffic because the rate of the normal packet of the UDP protocol itself is high. But it can be starting to detect the attack with the rate of 100 packets per second.

v. Comparisons of Performance over Various Monitoring Time and Attack Time

The performance can be slightly varied depending on the evaluating time taken for the network monitoring and attack launching. The reason is that the formula used in performance evaluation is relying on the ratio of the percentage of attack and normal packet. The longer the attacking time, the more attack packets are increasing. Similarly, the longer the monitoring time, the more normal packets are observing. The experimental results in this book except this section are produced from the monitoring results taken in the duration of three minutes with one-minute attack.

For proving this system which can detect the attack effectively over the various flooding attack time and monitoring time, the various performance results are produced by evaluating the web service with the different amount of time taken for attacking and monitoring as shown in Figures 5.23 and 5.24. The results of Figure 5.23 are produced from the various attack time within a monitoring time. The monitoring time is seven minutes and the different attacking times are from one minute to six minutes. Although the detection rate is slightly increased and the false negative rate is gradually decreased as the attack time is increased, the detection rate is not lower than the 99% and the false negative rate is not higher than the 1%.

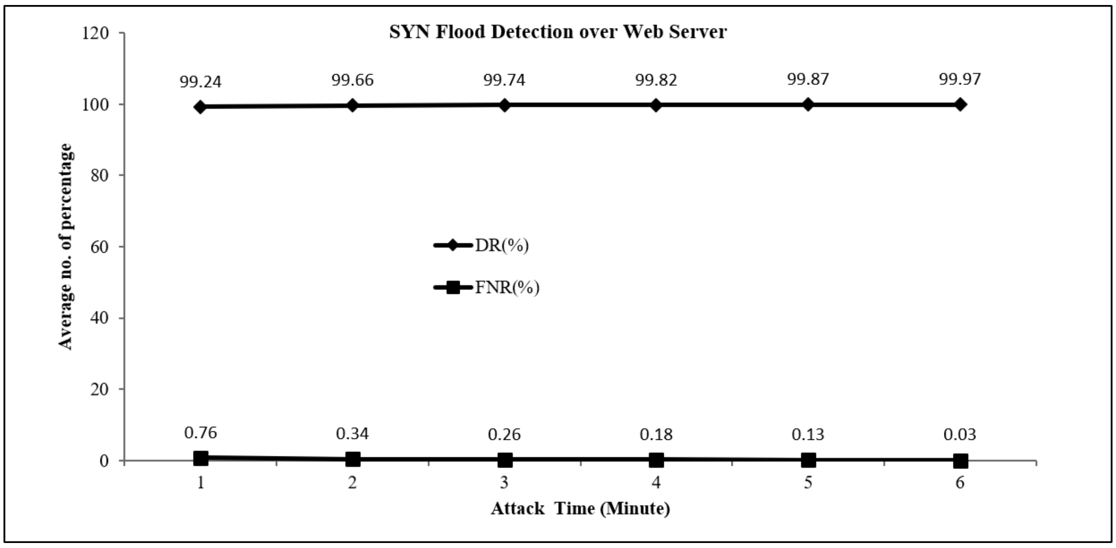


Figure 5.23 Comparisons of Various Attack Time over Web Service

Figure 5.24 shows the results produced from the various monitoring times with the same attack time. The monitoring times are from 3 minutes to 7 minutes. During each monitoring time, an attack is launched for one minute. As the monitoring time is more increased, the performance is weaker because the percentage of normal packets is increased more and more while the percentage of attack is being stable.

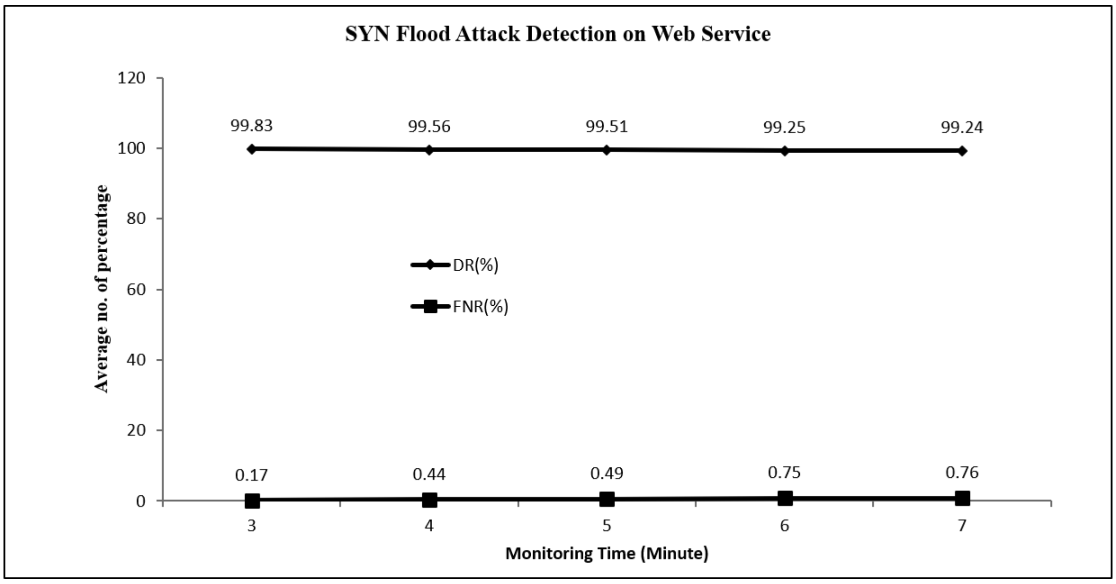


Figure 5.24 Comparisons of Various Monitoring Time over Web Service

B. Mitigation Results

The second section of the experimental results, the mitigation results, shows the filtering results using the *ddosmitigation* application and the results without using it. The comparisons of the percentage of filtering results produced by source-based and destination-based defense methods are also performed in this section to show why the source-based defense mechanism is used in this system. Moreover, the performance of the network during attack filtering is measured to prove that the source-based defense mechanism is more effective than the destination-based one.

i. Filtering Results

The $ddosmitigation$ application drops the abnormal traffic depending on the alert information obtained from the sFlow analyzer. The application takes the information from the analyzer every second by using REST API. Thus, the maximum time between the detection and mitigation is one second.

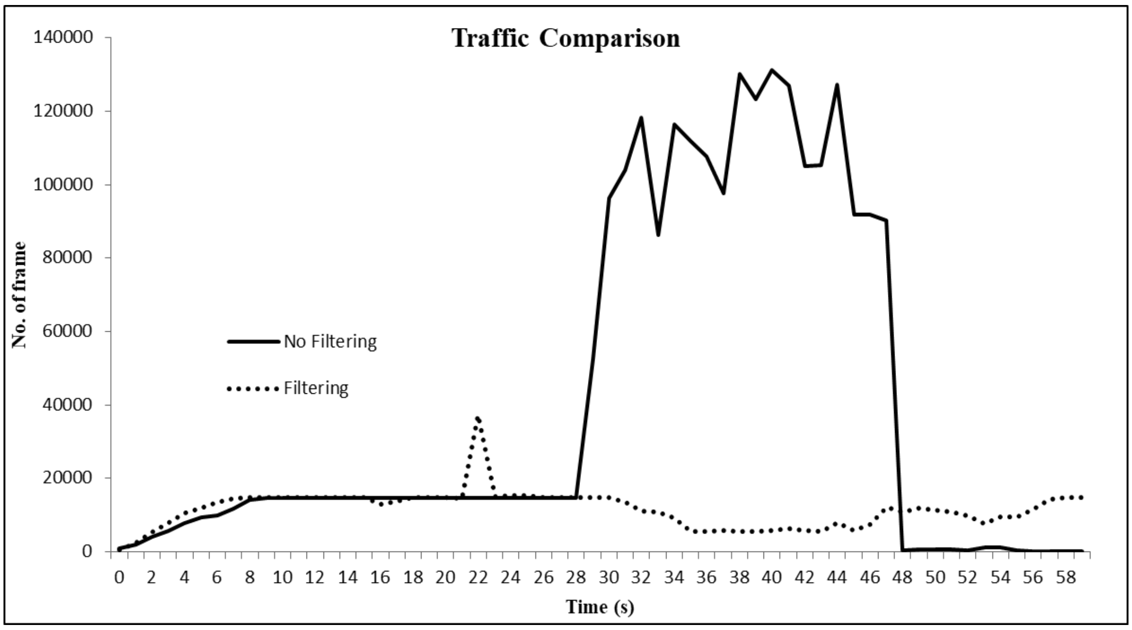


Figure 5.25 Comparison of Network Traffic with and without Filtering

By filtering the network with the application, the normal users can access a particular service without interrupting even though the server is in the attack. In contrast, without filtering the network with the $ddosmitigation$ application, the server can be down as soon as it is in the attack, and the service will be no longer available for normal users. The comparative result of filtering the network with and without $ddosmitigation$ application is as shown in Figure 5.25. These results are produced from the I/O graphs of the capturing results for one minute.

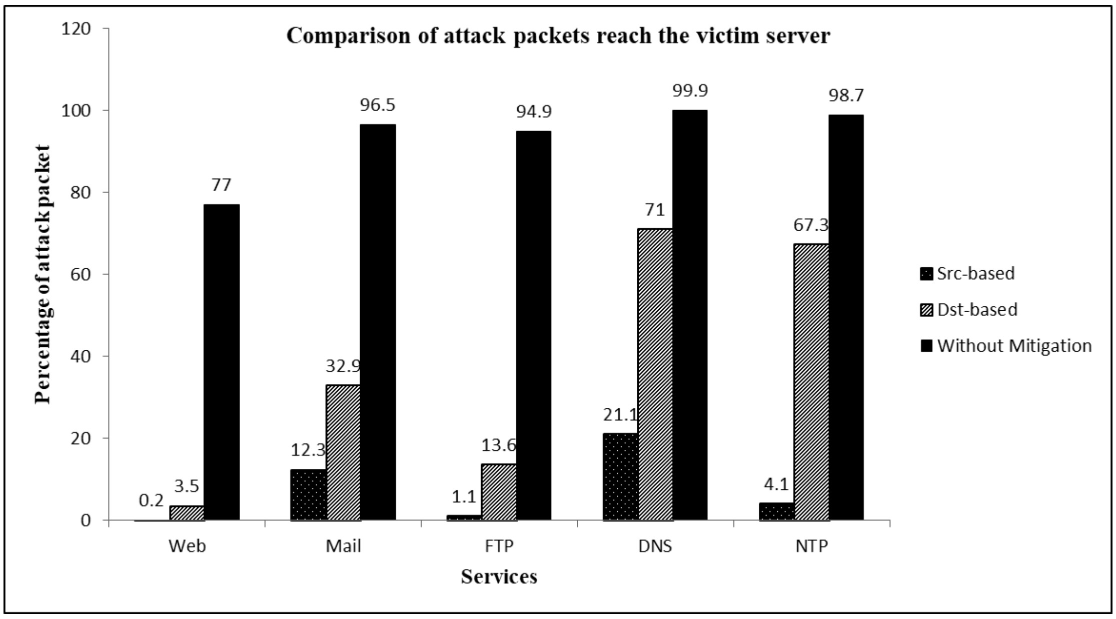


Figure 5.26 Comparison of Attack Packets reach the Victim Servers

Moreover, Figure 5.26 describes the percentage of the attack packet reach the victim server while the network is filtering with the $ddosmitigation$ application implemented with two different defense mechanisms (i.e. source-based defense mechanism and destination-based mechanism). To decide that how many percentages of the attack packet can be reduced by each mechanism, the figure also describes the percentage of attack packet reaches the victim server when the network is not filtering with the $ddosmitigation$ application.

Table 5.19 Reduction of Attack Packets Reach the Victim Servers

Mechanism	Web	Mail	FTP	DNS	NTP
Source-based defence	76.6	84.2	93.8	78.8	94.6
Destination-based defence	73.5	63.6	81.3	28.9	31.4

By reviewing the results of the percentage of the attack packet that reaches the victim server, the source-based defense mechanism could reduce the attack packet up to 94.6% while the destination-based mechanism only reduces them to 31.4% for NTP service as listed in Table 5.19. Thus, the source-based defense mechanism is more effective than a destination-based defense mechanism for the flooding attacks with the non-spoofing IP address.

ii. Network Performance

Figure 5.27 shows the comparative results of average network performance while the network is being in attack and filtering with the source-based and destination-based defense mechanism. The performance is measured by pinging with ten packets from client host h8 to h17. The average performance is also calculated by monitoring the ten times of average time to live (i.e ttl) from pinging and averaging the results.

Since the former mechanism is dropping the attack packets nearest to the source of the attack, the network will not be congested with those attack packets. Thus, the source-based defense mechanism maintains higher performance than the destination-based defense mechanism. As shown in Figure 5.27, the latency of the source-based defense mechanism for each service is about doubling the latency of the destination-based defense mechanism.

But, the source-based defense mechanism can only protect the direct attack because it must know the exact location of the attack so that it can install drop flow rule into the ingress switch of the attack host. If it does not know the location of the attack, the destination-based mechanism is preferred. Although this method can protect the attack with spoof IP addresses, the network might be congested because of attack traffic.

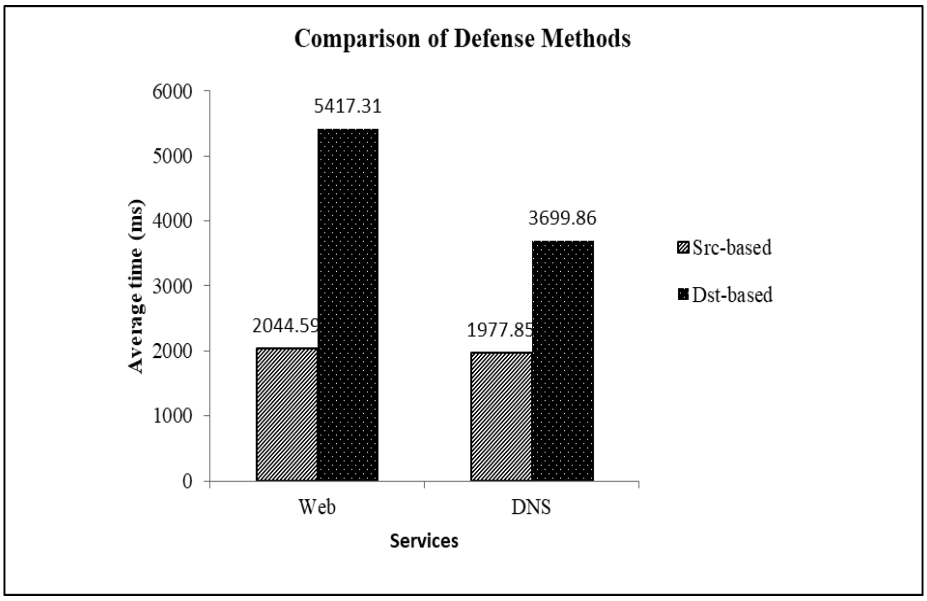


Figure 5.27 Comparative Results for Network Performance during the Network is being in Attack

5.3 Chapter Summary

This chapter described two types of scenarios for testing this system. The first type is a simple scenario with only one attack, SYN flooding attack and the network is solely generated the web traffic. The second scenario is to test the system as in the real network environment by generating all possible network traffic and launching the transport layer flooding attacks into the network. It also presented the various comparative results produced from the two scenarios.

CHAPTER 6

CONCLUSION AND FUTURE WORKS

This chapter describes the summary of the dissertation, advantages and limitations, and recommendations for the future works.

6.1 Summary of Dissertation

Flooding attacks might fail the whole SDN network or services running in the network during a short time. Thus, the effective and speedy mechanism for detecting and mitigating the flooding attack is necessary for the network. Various techniques are proposed for the detection and mitigation of the DDoS attack including flooding attacks. Statistical analysis techniques are commonly used to detect the flooding attack. Entropy and change point detection are belonging to statistical analysis techniques. Although entropy can measure the randomness of the network traffic within a given time, they could not differentiate the different distributions with equal uncertainty. Thus, the malicious traffic without randomness will not be detected. Change point detection techniques are effectively used in DDoS detection and mitigation. However, the function of the detection is implemented in the SDN controller itself. As a result, the controller might be overloaded with the detection operations. For avoiding the statistic overhead in the SDN controller, the whole detection operations can be delegated to the sFlow analyzer. The most commonly used dynamic threshold algorithms are the adaptive threshold algorithm (ATA) and the cumulative sum algorithm (CUSUM).

Both high-intensity and low-intensity attacks can be detected by using CUSUM. However, ATA can detect only high-intensity attack. In the CUSUM algorithm, the two static values for the Maximum threshold and Minimum Threshold must be defined. Moreover, the implementation of this algorithm cannot be accomplished in the sFlow analyzer because sFlow allows the comparison which is more than the maximum value and does not allow the comparison which is less than the minimum value. Thus, CUSUM is not adaptable with the default API of the sFlow. However, ATA can be appropriately implemented in sFlow. But it produces a high number of false alarms. The avoiding method of false alarms used in this algorithm might increase the false negative rate as a consequence. This method is defining the abnormal when the number of threshold violations consecutively occurs within a specific time. As a result, the tradeoff between false positive rate and detection rate depends on the predefined number of threshold violations and time for it. Since it does not discard the attack as soon as the threshold violation occurs, the false negative rate will be increased. For circumventing the consequence problem of increasing the false negative rate, the false alarms are significantly reduced by modifying the ATA with the consideration of the baseline of the network.

In order to get the baseline in the virtual network environment as in the real time network, $D-ITG$ tool is used in the traffic generation. Moreover, the experimental testbed is constructed with the various types of servers and clients like in the real world. Traffic is also generated by referencing the traffic models to get the real network traffic pattern. While all other clients are accessing the servers concurrently, the attacker launches the flooding attack on the victim server. During the experimentation, the network traffic is captured by using the traffic capturing tool $tcpdump$. After the experiment is finished, the performance parameters of network security (i.e. detection rate, false negative rate, and accuracy) are measured by calculating the percentage of false negative, true positive, and true negative values from the captured result. False positive rate is considered as zero because the ATA algorithm alrea6dy considered the avoidance method of false alarms.

All of the above described dissertation was presented in this book with six chapters. In chapter 1, the dissertation was starting introduced with the definition of the flooding attack, and the impact of the attack on the SDN network especially. This chapter also described the types of flooding attacks and the mechanisms for the detection and mitigation of these attacks. Moreover, this chapter described the motivation of the research by pointing out the incomplete effectiveness of legacy defense mechanisms with the list of recent DDoS attacks on various recognized organizations. It also introduced threshold is important in the statistical analysis technique and described the weakness of the calculation method for the dynamic threshold as the problem of this research. It also described the objectives which are related to overcoming the weak points of the original works. It sequentially presented the focus works along the way of the research. It then described the modification of the calculation method for producing the adaptive and dynamic threshold which supporting the effective detection and mitigation of the flooding attack as the main contribution of the research.

The literature reviews of the detection and mitigation of the flooding attack in SDN with the various types of mechanisms were described in chapter 2. Before describing these detection and mitigation mechanisms, this chapter described the limitation of traditional networking for defending of DDoS attack and the advantages of SDN over the limitations of existing DDoS defense mechanisms. Then, it presented separately the brief review of the previous works for each type of mechanism. It has two main sections for describing the mechanisms: detection, and mitigation. The detection mechanisms were furthered divided into five types: statistical analysis including entropy and change-point detection, machine learning, traffic pattern analysis, connection rate, and integration of traffic monitoring tool and OpenFlow. Likewise, each mitigation mechanism was separately described in each sub-section such as source-based, destination-based, and hybrid-based drop packet or block port, redirection, control bandwidth, and change network topology.

The background theory for the flooding attack detection and mitigation over the SDN network was detailed described in chapter 3. Thus, it included the description of the SDN architecture and its main components such as OpenFlow protocol, Open vSwitch, and ONOS controller. It also described three types of flow rule installation (i.e. reactive, proactive, hybrid), which are the main functions of the SDN controller. Since this dissertation used the sFlow analyzer for analyzing the incoming packets, this chapter presented the collection and manipulation of packet statistics with the sFlow-RT analyzer in SDN with the description of sampling rate and polling interval. Moreover, this chapter described the two algorithms (i.e., original algorithm, and modified algorithm) for the calculation of the dynamic threshold used in this dissertation. The performance evaluation methods of the network security which are used in this dissertation are also detailed described in this chapter with the confusion matrix and respective equations.

The overall system architecture for the flooding attack detection and mitigation in SDN was firstly described in chapter 4. As this system used the sFlow-RT analyzer in the attack detection phase and ONOS controller in the attack mitigation phase, each phase is detailed explained with the process flow. Moreover, as this dissertation used the dynamic threshold in traffic comparison of the sFlow analyzer, the step by step process for the implementation of the dynamic and adaptive threshold algorithm within the analyzer is then described as the form of procedure. Since this dissertation is mainly contributed to modifying the dynamic threshold algorithm, the implementation of both original and modified algorithms was included in each description of this chapter.

The experimental design and implementation of the flooding attack detection and mitigation in SDN were presented in chapter 5 with the various types of experimental results. For proving that the modified adaptive threshold algorithm is more effective than the original one, all experimental results are produced based on the comparison of the two algorithms. Firstly, it described the design for the testbed with the hardware and software requirements. Then, two types of scenarios were used to evaluate the results produced from the detection and mitigation of only the SYN flooding attack, and the various types of flooding attacks. For only SYN flooding attack detection and mitigation, the experimental results are described as the comparison of adaptive threshold and traffic, the filtering results, and the results from the performance parameters: FNR, DR, and ACC.

In the testing of the experiment with the various types of flooding attack, the false alarm avoidance methods for both algorithms are detailed described in this chapter with some testing data. Since this dissertation analyzed the traffic generated from the mininet emulator, this chapter also described the traffic generation model and tool for generating the traffic as in the real network traffic. The experimental results of the various types of flooding attack detection and mitigation are separately described with two main sections: detection and mitigation results. The detection results consist of five parts: comparative results of adaptive threshold over incoming traffic, comparative results of performance parameters, comparative results of incoming traffic in sFlow analyzer and arrival traffic in each server, comparisons of performance over various attack rates, and comparisons of performance over various monitoring time and attack time. The mitigation results also consist of two main parts: filtering results including the comparison of network traffic with and without filtering, comparison of attack packets reach the victim servers, and reduction of attack packets reach the victim servers, and the measurement of network performance during the network is being in attack.

6.2 Advantages and Limitations

By using the modified ATA, the average percentage of the false negative rate and the accuracy are around 0.7% and 99%, respectively. Although this method has a little overhead for finding the baseline of the network traffic, it considerably reduces the false alarms by producing the dynamic and adaptive threshold based on the baseline of the network traffic. Thus, the attack might be discarded immediately when the $ddosmitigation$ application received the abnormal event information. Moreover, since the application regularly takes the event information from the sFlow analyzer every second, the maximum delay time for detection and mitigation is one second while the detection time of previous works [13][52] was around four seconds. Therefore, it can be concluded that MATA is the effective and speedy algorithm for the various types of flooding attacks detection and mitigation.

The main limitation of this dissertation is that it can detect only the flooding attack with the non-spoofing IP address and non-distributed DoS attack because of the source-based defense mechanism. It can detect only the high-intensity attack because ATA solely checks the traffic rate is greater than the threshold value.

6.3 Recommendations for Future Work

Although this system is implemented to fulfill with its objectives, it still has some implementations as its extensions or future works.

The detection phase of the flooding attack will be needed to implement in SDN controller itself without using the assistance of the analyzer. Then, the comparison of the two methods: with, and without using the sFlow-RT analyzer can be made.

This system can only detect and mitigate the transport layer attack based on the non-spoofing address, and high-intensity attack. Thus, it is better to extend this system to be able to analyze the various types of attacks including application-layer flooding attacks, low-level attacks, and spoofing IP attacks.

Moreover, as this system is implemented based on the layer 2 forwarding network, it can be deployed within the same network. For applying it over the different networks, this system will be needed to implement the detection and mitigation of the flooding attack over layer 3 and above SDN network by using SDN-IP application in ONOS controller.

Since this system is evaluated by using the own generated network traffic, it is better to test and evaluate the implemented system with the dataset by replaying the traffic into the SDN network.

It is better to implement this system on the real SDN testbed than the emulated network topology. The final recommendation for the future work of this system is to implement the flooding attack detection and mitigation system as a network function for the intrusion detection system (IDS) running on the SDN network by using SDN/NFV architecture.

AUTHOR'S PUBLICATIONS

[p1] N. H. M. Oo and A. H. Maw, "Firewall Application for ONOS SDN Controller", in Proceedings of the 15th International Conference on Computer Applications 2017(ICCA 2017), Yangon, Myanmar, pp. 391-397, 2017 February.

[p2] N. H. M. Oo and A. H. Maw, "Stateful Firewall Application on Software-Defined Networking", in Proceedings of the 1st International Conference on Advanced Information Technology 2017 (ICAIT 2017), Yangon, Myanmar, pp. 39-45, 2017 November.

[p3] N. H. M. Oo and A. H. Maw, " State-aware Packet Forwarding on Software-Defined Networking", in Proceedings of the 16th International Conference on Computer Applications 2018(ICCA 2018), Yangon, Myanmar, pp. 368-373, 2018 February.

[p4] N. H. M. Oo and A. H. Maw, "SYN Flooding Attack Detection and Mitigation in SDN", in Proceedings of 2019 The 9th International Workshop on Computer Science and Engineering, WCSE_2019_SPRING, Yangon, Myanmar, pp. 126-131, 2019 February 27 - March 1, ISBN: 978-981-14-1455-8, doi:10.18178/wcse.2019.03.022.

[p5] N. H. M. Oo and A. H. Maw, "Effective Detection and Mitigation of SYN Flooding Attack in SDN", in Proceedings of the 19th International Symposium on Communications and Information Technologies (ISCIT 2019), Ho Chi Minh, Vietnam, pp. 300-305, 2019 November 21, IEEE.

[p6] N. H. M. Oo, A.C. Risdianto, L.T. Chaw, A. H. Maw, "Flooding Attack Detection and Mitigation in SDN with Modified Adaptive Threshold Algorithm", International Journal of Computer Network & Communication (IJCNC), ISSN: 0974 – 9322 [Online]; 0975 – 2293 [Print], 2020 January. (To be appeared).

BIBLIOGRAPHY

[1] A.A. Aizuddin, M. Atan, M. Norulazmi, M.M. Noor, S. Akimi, Z. Abidin, "DNS amplification attack detection and mitigation via sFlow with security-centric SDN", In Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication, pp. 3, 2017 January 5, ACM.

[2] A. Aleroud, I. Alsmadi, "Identifying DoS attacks on software-defined networks: a relation context approach", In NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium, pp. 853-857, 2016 April 25, IEEE.

[3] A. Arins, "Firewall as a service in SDN OpenFlow network. In 2015 IEEE 3rd Workshop on Advances in Information", Electronic and Electrical Engineering (AIEEE), pp. 1-5, 2015 November 13, IEEE.

[4] Avallone, S., Emma, D., Pescapé, A., & Ventre, G., "Performance evaluation of an open distributed platform for realistic traffic generation", Performance Evaluation, 60(1-4), pp. 359–392, 2005.

[5] N. Axel, B. Davis, "Software-Defined Networking", Cisco, 2013 August 30.

[6] N.Z. Bawany, J.A. Shamsi, K. Salah, "DDoS attack detection and mitigation using SDN: methods, practices, and solutions". Arabian Journal for Science and Engineering, 42(2), pp. 425-41, 2017 February 1.

[7] K. Bogineni. "Introducing ONOS—A SDN network operating system for service providers", White Paper, 2014.

[8] R. Braga, E. Mota, A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow", In LCN, 10, pp. 408-415, 2010 October 10.

[9] S. Brent, "OpenFlow: Proactive vs Reactive Flows", 2013 January 15.

[10] C. Buragohain, N. Medhi, "FlowTrApp: An SDN based architecture for DDoS attack detection and mitigation in data centers", In 2016 3rd International Conference on Signal Processing and Integrated Networks (SPIN), pp. 519-524, 2016 February 11, IEEE.

[11] T. Chin, X. Mountrouidou, X. Li, K. Xiong, "Selective packet inspection to detect DoS flooding using software-defined networking (SDN)". In 2015 IEEE 35th international conference on distributed computing systems workshops, pp. 95-99, 2015 June 29, IEEE.

[12] V. Christodoulou, Y. Bi, “A combination of CUSUM-EWMA for Anomaly Detection in time series data”, In 2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA), pp. 1-8, 2015 October 19, IEEE.

[13] M. Conti, A. Gangwal, M.S. Gaur, “A comprehensive and effective mechanism for DDoS detection in SDN”, In 2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pp. 1-8, 2017 October 9, IEEE.

[14] M. Conti, A. Gangwal, “Blocking intrusions at border using software defined-internet exchange point (sd-ixp)”. In 2017 IEEE Conference on Network Function Virtualization and Software-Defined Networks (NFV-SDN), pp. 1-6, 2017 November 6, IEEE.

[15] N.G. Dharma, M.F. Muthohar, J.A. Prayuda, K. Priagung, D. Choi, “Time-based DDoS detection and mitigation for SDN controller”, In 2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS), pp. 550-553, 2015 August 19, IEEE.

[16] C. Dillon, M. Berkelaar, “Openflow (d) dos mitigation”. Technical Report (February 2014). http://www. delaat. net/rp/2013-2014/p42/report. pdf, 2014.

[17] P. Dong, X. Du, H. Zhang, T Xu, “A detection method for a novel DDoS attack against SDN controllers by vast new low-traffic flows”, In 2016 IEEE International Conference on Communications (ICC), pp. 1-6, 2016 May 22, IEEE.

[18] S. Dotcenko, A. Vladyko, I. Letenko. “A fuzzy logic-based information security management for software-defined networks”, In16th International Conference on Advanced Communication Technology, pp. 167-171, 2014 February 16, IEEE.

[19] L. Dridi, M.F. Zhani, “SDN-guard: DoS attacks mitigation in SDN networks”, In 2016 5th IEEE International Conference on Cloud Networking (Cloudnet), pp. 212-217, 2016 October 3, IEEE.

[20] P.T. Duy, V.H. Pham, “A role-based statistical mechanism for DDoS attack detection in SDN”, In 2018 5th NAFOSTED Conference on Information and Computer Science (NICS), pp. 177-182, 2018 November 23, IEEE.

[21] A. Esage, “Poker Tournaments Cancelled after DDoS Attacks, Players want to be reimbursed”, 2018 May 4.

[22] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, V. Maglaris, “Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments”, Computer Networks, 62, pp. 122-36, 2014 April 7.

[23] T. Graf. “Underneath OpenStack Quantum: Software-Defined Networking with Open vSwitch”. Retrieved from. 2013.

[24] H. Guesmi, L.A. Saidane, “Using sdn approach to secure cloud servers against flooding based ddos attacks”, In 2017 25th International Conference on Systems Engineering (ICSEng), pp. 309-315, 2017 August 22, IEEE.

[25] S. Hilton, “Dyn Analysis Summary Of Friday October 21 Attack”, 2016 October 26.

[26] D. Hu, P. Hong, Y. Chen, “FADM: DDoS flooding attack detection and mitigation system in software-defined networking”, In GLOBECOM 2017-2017 IEEE Global Communications Conference, pp. 1-7, 2017 December 4, IEEE.

[27] L. Irwin, “Luxembourg government servers hit by DDoS attack”, 2017 March 8.

[28] R. Jin, B. Wang, “Malware detection for mobile devices using software-defined networking”, In 2013 Second GENI Research and Educational Experiment Workshop, pp. 81-88, 2013, IEEE.

[29] A. Kalliola, K. Lee, H. Lee, T. Aura, “Flooding DDoS mitigation and traffic management with software-defined networking”, In 2015 IEEE 4th International Conference on Cloud Networking (CloudNet), pp. 248-254, 2015 October 5, IEEE.

[30] R. Kandoi, M. Antikainen, “Denial-of-service attacks in OpenFlow SDN networks”, In 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 1322-1326, 2015 May 11, IEEE.

[31] P. Kumar, M. Tripathi, A. Nehra, M. Conti, C. Lal, “SAFETY: Early detection and mitigation of TCP SYN flood utilizing entropy in SDN”, IEEE Transactions on Network and Service Management, 15(4), pp. 1545-59, 2018 July 31.

[32] A. Lara, A. Kolasani, B. Ramamurthy , “Network innovation using openflow: A survey”. IEEE communications surveys & tutorials, 16(1), pp. 493-512, 2013.

[33] M. Latah, L. Toker, “A novel intelligent approach for detecting DoS flooding attacks in software-defined networks”, International Journal of Advances in Intelligent Informatics, 4(1), pp. 11-20, 2018 March 1.

[34] S. Lim, J. Ha, H. Kim, Y. Kim, S. Yang, “A SDN-oriented DDoS blocking scheme for botnet-based attacks”, In 2014 Sixth International Conference on Ubiquitous and Future Networks (ICUFN), pp. 63-68, 2014, IEEE.

[35] Y. Lu, M. Wang, “An easy defense mechanism against botnet-based DDoS flooding attack originated in SDN environment using sFlow”, In Proceedings of the 11th International Conference on Future Internet Technologies, pp. 14-20, 2016 June 15, ACM.

[36] J. zheng, Q. Li, G. Gu, J. Cao, D.K. Yau, J. Wu, “Realtime DDoS defense using COTS SDN switches via adaptive correlation analysis”. IEEE Transactions on Information Forensics and Security, 13(7), pp. 1838-1853, 2018 February 12.

[37] J. Mao, W. Deng, F. Shen, “DDoS Flooding Attack Detection Based on Joint-Entropy with Multiple Traffic Features”, In 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), pp. 237-243, 2018 August 1, IEEE.

[38] S.A. Mehdi, J. Khalid, S.A. Khayam, “Revisiting traffic anomaly detection using software-defined networking”, In International workshop on recent advances in intrusion detection, pp. 161-180, 2011 September 20, Springer, Berlin, Heidelberg.

[39] R. Mohammadi, R. Javidan, M. Conti, “Slicots: An sdn-based lightweight countermeasure for tcp syn flooding attacks”. IEEE Transactions on Network and Service Management, 14(2), pp. 487-97, 2017 May 8.

[40] L. Mutu, R. Saleh, A. Matrawy, “Improved SDN responsiveness to UDP flood attacks”, In 2015 IEEE Conference on Communications and Network Security (CNS), pp. 715-716, 2015 September 28, IEEE.

[41] T.M. Nam, P.H. Phong, T.D. Khoa, T.T. Huong, P.N. Nam, N.H. Thanh, L.X. Thang, P.A. Tuan, V.D. Loi, “Self-organizing map-based approaches in DDoS flooding detection using SDN”. In2018 International Conference on Information Networking (ICOIN), pp. 249-254, 2018 January 10, IEEE.

[42] L. H. Newman, “GitHub Survived the Biggest DDoS Attack Ever Recorded”, 2018 January 3.

[43] M. Özçelik, N. Chalabianloo, G. Gür, “Software-defined edge defense against IoT-based DDoS”, In 2017 IEEE International Conference on Computer and Information Technology (CIT), pp. 308-313, 2017 August 21, IEEE.

[44] Peter, “Network-wide visibility, Telemetry, analytics, and control with sFlow® standard”, 2009 May 15.

[45] Peter, “Sampling rates, Telemetry, analytics, and control with sFlow® standard”, 2009 June 26.

[46] P. Phaal and S. Panchen, “Packet Sampling Basics”, 2002.

[47] T.V. Phan, T. Van Toan, D. Van Tuyen, T.T. Huong, N.H. Thanh, “Openflowsia: An optimized protection scheme for software-defined networks from flooding attacks”, In 2016 IEEE Sixth International Conference on Communications and Electronics (ICCE), pp. 13-18, 2016 July 27, IEEE.

[48] R. Priyadarshini, R.K. Barik, “A deep learning based intelligent framework to mitigate DDoS attack in fog environment”, Journal of King Saud University-Computer and Information Sciences, 2019 April 24.

[49] A. Sangodoyin, B. Modu, I. Awan, J.P. Disso, “An approach to detecting distributed denial of service attacks in software-defined networks”, In 2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud), pp. 436-443, 2018 August 6, IEEE.

[50] M.J. Schwartz, “Bank of Spain Hit by DDoS Attack”, 2018 August 28.

[51] S. Shin, P. Porras, V.Yeneswaran, M. Fong, G. Gu, M. Tyson, “Fresco: Modular composable security services for software-defined networks”. In 20th Annual Network & Distributed System Security Symposium, 2013 February 26, Ndss.

[52] V.A. Siris, F. Papagalou, “Application of anomaly detection algorithms for detecting SYN flooding attacks”, IEEE Global Telecommunications Conference, 4, pp. 2050–2054, 2004.

[53] R.M. Thomas, D. James, “DDOS detection and denial using third party application in SDN”, In 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), pp. 3892-3897, 2017 August 1, IEEE.

[54] T.V. Tran, H. Ahn, “Challenges of and solution to the control load of stateful firewall in software-defined networks”, Computer Standards & Interfaces, 54, pp. 293-304.

[55] C.F. Tsai, Y.F. Hsu, C.Y. Lin, W.Y. Lin, “Intrusion detection by machine learning: A review”, expert systems with applications, 2009 December 1,36(10), pp. 11994-2000.

[56] T. Ubale, A.K. Jain, “SRL: An TCP SYN FLOOD DDoS Mitigation Approach in Software-Defined Networks”, In 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA), pp. 956-962, 2018 March 29, IEEE.

[57] V. Varadharajan, K. Karmakar, U. Tupakula, M. Hitchens, “A policy-based security architecture for software-defined networks”, IEEE Transactions on Information Forensics and Security, 14(4), pp. 897-912, 2018 August 31.

[58] A.N. Viet, L.P. Van, H.A. Minh, H.D. Xuan, N.P. Ngoc, T.N. Huu, “Mitigating HTTP GET flooding attacks in SDN using NetFPGA-based OpenFlow switch”, In 2017 14th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), pp. 660-663, 2017 June 27, IEEE.

[59] J. Wang, R. Wen, J. Li, F. Yan, B. Zhao, F. Yu, “Detecting and mitigating target link-flooding attacks using sdn”, IEEE Transactions on Dependable and Secure Computing, 2018 April 2.

[60] R. Wang, Z. Jia, L. Ju, “An entropy-based distributed DDoS detection mechanism in software-defined networking”, In 2015 IEEE Trustcom/BigDataSE/ISPA, 1, pp. 310-317, 2015 August 20, IEEE.

[61] Waqas, “Website security firm Sucuri hit by large scale volumetric DDoS attacks”, 2018 April 13.

[62] H.C. Wei, Y.H. Tung, C.M. Yu, "Counteracting UDP flooding attacks in SDN", In 2016 IEEE NetSoft Conference and Workshops (NetSoft), pp. 367-371, 2016 June 6, IEEE.

[63] T. Xing, D. Huang, L. Xu, C. Chung, P. Khatkar. "Snortflow: A openflow-based intrusion prevention system in cloud environment", In 2013 second GENI research and educational experiment workshop, pp. 89-92, 2013 March 20, IEEE.

[64] J. Zheng, Q. Li, G. Gu, J. Cao, D.K. Yau, J. Wu, "Realtime DDoS defense using COTS SDN switches via adaptive correlation analysis", IEEE Transactions on Information Forensics and Security, 13(7), pp. 1838-1853, 2018 February 12.

[65] "OpenFlow Switch Specification (Version1.3.0). Open Networking Foundation ONF), Technical Specification." https://www.opennetworking .org/images/stories/downloads/sdn-resources/onf-specifications/openflow/ openflow-spec-v1.3.0.pdf.

[66] "OpenFlow. Open Networking Foundation (ONF)", https://www.opennet-working.org/, accessed on 2014 November 13.

[67] Mininet [Online]. Available from: http://mininet.org.

[68] GNU Wget 1.18 Manual [online]. Available from: https://www.gnu.org/soft-ware/wget/manual/wget.html.

[69] Tcpdump [online]. Available from: https://www.tcpdump.org/manpages /tcp-dump.1.html.

[70] Hping3 Security Tool [online]. Available from: http://www.hping.org/hping3 .html.

[71] DITG Tool [Online]. Available from: http://www.grid.unina.it/software/ITG/.

[72] Traffic generation model, https://en.wikipedia.org/wiki/ Traffic _ generation _model.

[73] "SDN architecture", ONF TR-502, 2014 June 1.

[74] Polling-interval, TechLibrary, 2018 April 20.

[75] sFlow-RT, [Online]. Available from: https://www.inmon.com, 2014 May.

[76] ONOS [Online]. Available from: https://onosproject.org.

[77] ONF, "Software-defined networking: The new norm for networks," ONF White Paper, 2, pp. 2–6, 2012.

[78] Software-defined analytics, [Online]. Available from: https://blog.sflow.com/2013/05/software-defined-analytics.html, 2013 May.

[79] Open Networking Foundation, “Principles and practices for securing software-defined networks version 1.0”, 2015 January.

LIST OF ACRONYMS

ACL	Access Control List
ACC	Accuracy
ATA	Adaptive Threshold Algorithm
ACR	Ameriacs Cardroom
API	Application Programming Interface
Abf	Average of Bytes per flow
Adf	Average of Duration per flow
CAIDA	Center for Applied International Data Analysis (CAIDA)
CDP	Cisco Discovery Protocol
COTS	Commercial off-the-shelf
CI	Confidence Interval
CUSUM	Cumulative Sum
D-CPI	Data-Controller Plane Interface
DL	Deep Learning
DOS	Denial of Service
DR	Detection Rate
DDoS	Distributed Denial of Service
D-ITG	Distributed Internet Traffic Generator
DNS	Domain Name Service
DHCP	Dynamic Host Configuration Protocol
EWMA	Exponentially Weighted Moving Average
EWMA	Exponentially Weighted Moving Average
FAR	False Alarm Rate
FN	False Negative

FP	False Positive
FTP	File Transfer Protocol
GRE	Generic Routing Encapsulation
GDP	Growth of Different Ports
GSf	Growth of Single-flows
IA	Idle-timeout Adjustment
IDT	Inter Departure Time
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
LACP	Link Aggregation Control Protocol
LLDP	Link Layer Discovery Protocol
LFA	Link-flooding Attack
LISP	Locator/ID Separation Protocol
MATA	Modified Adaptive Threshold Algorithm
MTD	Moving Target Defense
NFV	Network Function Virtualization
NIC	Network Interface Card
NTP	Network Time Protocol
NNTP	New Network Transfer Protocol
NBI	North Bound Interface
ONF	Open Network Foundation
ONOS	Open Network Operating System
OSPF	Open Shortest Path First
OVS	Open vSwitch

OVSDB	Open vSwitch Database
PS	Packet Size
PPf	Percentage of Pair-flows
QC	Quality Control
QoS	Quality of Service
sFlow	Sample Flow
SOM	Self-Organizing Map
SPRT	Sequential Probability Ratio Test
SMTP	Simple Mail Transport Protocol
SNMP	Simple Network Management Protocol
SD-IXP	Software Defined-Internet Exchange Point
SDN	Software-Defined Networking
SBI	South Bound Interface
SPC	Statistical Process Control
STT	Stateless Transport Tunneling
SVM	Support Vector Machine
TCP	Transport Control Protocol
TLS	Transport Layer Support
TN	True Negative
TP	True Positive
UDP	Universal Datagram Protocol
VXLAN	Virtual Extensible Local Area Network
VLAN	Virtual Local Area Network
VM	Virtual Machine
WWW	World Wide Web

APPENDICES

APPENDIX A: FLOODING ATTACK DETECTION

The detection of flooding attack in this system is implemented by using the sFlow-RT analyzer. Thus, the three main functions including in the analyzer is detailed discussed in this section. All these functions are implemented in a script file (i.e. metric.js) by using the JavaScript API and this script file is under the directory of $sflow - rt/app/ddosmitigation/scripts/$.

The functions included in the script file are: flow definition, flow handling, and event handling.

1) **Flow Definition**

For getting the specific type of flow from each type of service, the set of flow keys and filtering keys are defined for each service.

a. Collection of the number of SYN frames per second from the Web flow with the flow name of "$tcpflow\_websyn$"

```
setFlow('tcpflow_websyn',
keys:'macsource,macdestination,
ipsource,ipdestination,
tcpdestinationport,',
value:'frames',
filter:'tcpdestinationport=80&tcpflags=000000010'});
```

b. Collection of the number of SYN frames per second from the Mail flow with the flow name of "$tcpflow\_mailsyn$"

```
setFlow('tcpflow_mailsyn',
keys:'macsource,macdestination,
ipsource,ipdestination,
tcpdestinationport,',
value:'frames',
filter:'tcpdestinationport=25&tcpflags=000000010'});
```

c. Collection of the number of SYN frames per second from the FTP flow with the flow name of "$tcpflow\_ftpsyn$"

```
setFlow('tcpflow_ftpsyn',
keys:'macsource,macdestination,
ipsource,ipdestination,
tcpdestinationport,',
value:'frames',
filter:'tcpdestinationport=20&tcpflags=000000010'});
```

d. Collection of the number of frames per second from the DNS flow with the flow name of "$udpflow\_dns$"

```
setFlow(‘udpflow_dns',
keys:'macsource,macdestination,
ipsource,ipdestination,
udpdestinationport,',
value:'frames',
filter:‘udpdestinationport=53’});
```

e. Collection of the number of frames per second from the NTP flow with the flow name of "$udpflow\_ntp$"

```
setFlow(‘udpflow_ntp',
keys:'macsource,macdestination,
ipsource,ipdestination,
udpdestinationport,',
value:'frames',
filter:‘udpdestinationport=123’});
```

f. Collection of the number of frames per second from the DHCP flow with the flow name of "$udpflow\_dhcp$"

```
setFlow(‘udpflow_dhcp',
keys:'macsource,macdestination,
ipsource,ipdestination,
udpdestinationport,',
value:'frames',
filter:‘udpdestinationport=67’});
```

2) **Flow Handling**

a. Initialization

The value of the baseline and threshold for each service is initialized according to the result produced from the observation of baseline in the Table 5.18 in section 5.2.2.2. Moreover, the number of previous frames is also initialized as zero.

i. Initial baseline value for each service

```
var baseline_mail = 5.0;
var baseline_web = 5.0;
var baseline_ftp = 5.0;
var baseline_dhcp = 77.0;
var baseline_dns = 9.0;
var baseline_ntp = 65.0;
```

ii. Initial threshold value for each service

```
var threshold_web = 5.0;
var threshold_ftp = 5.0;
var threshold_mail = 5.0;
var threshold_dhcp = 77.0;
var threshold_dns = 9.0;
var threshold_ntp = 65.0;
```

iii. Initial previous frame value for each service

```
var PFtminusone_web = 0.0;
var PFtminusone_ftp = 0.0;
var PFtminusone_mail = 0.0;
var PFtminusone_dns = 0.0;
 var PFtminusone_dhcp = 0.0;
 var PFtminusone_ntp = 0.0;
```

b. Alternate service handling

After defining the initial value of the baseline, threshold, previous traffic, and collecting the flow from the flow definition, the flow handling function for each service is implemented under the *setIntervalHandler* function.

Firstly, "*customSetTimeOut*" function is implemented for alternative handling all services within one second.

```
function customSetTimeOut (app, ms) {
  var now = (new Date()).getTime();
  var future = now + ms;
  var nownow = Date.now();
  while((Date.now() <= future)) {
      }
   return app();
}
customSetTimeOut(function(){
  ntp();
},0);
customSetTimeOut(function(){
  dhcp();
},0);
customSetTimeOut(function(){
 dns();
},0);
customSetTimeOut(function(){
 web();
},0);
```

```
customSetTimeOut(function(){
  ftp();
},0);
customSetTimeOut(function(){
  mail();
},0);
```

The invoked function of each service is implemented according to the algorithm 4.1 as described in section 4.1.1.2. It consists of threshold comparing, and new threshold calculation for the next second.

i. NTP service handling function

```
function ntp(){
  setThreshold(‘ntp_abnormal',
    {metric:‘udpflow_ntp', value:threshold_ntp,
     byFlow:true, timeout:1});
  udpCount_ntp = Count('udpflow_ntp');
  CF_ntp = CalFramePerSec(udpCount_ntp);
PFt_ntp = ewma(CF_ntp, PFtminusone_ntp,
                  alpha_critical);
  threshold_ntp_new = ((percentage + 1) * PFt_ntp) +
                                    baseline_ntp;
  PFtminusone_ntp = PFt_ntp;
  threshold_ntp = threshold_ntp_new;
  return;
}

```

ii. DHCP service handling function

```
function dhcp(){
  setThreshold(‘dhcp_abnormal',
    {metric:‘udpflow_dhcp', value:threshold_dhcp,
     byFlow:true, timeout:1});
  udpCount_dhcp = Count('udpflow_dhcp');
  CF_dhcp = CalFramePerSec(udpCount_dhcp);
  PFt_dhcp = ewma(CF_dhcp, PFtminusone_dhcp,
                  alpha_critical);

threshold_dhcp_new = ((percentage + 1) * PFt_dhcp)
                                  + baseline_dhcp;
  PFtminusone_dhcp = PFt_dhcp;
  threshold_dhcp = threshold_dhcp_new;
  return;
}
```

iii. DNS service handling function

```
function dns(){
  setThreshold(‘dns_abnormal',
    {metric:‘udpflow_dns', value:threshold_dns,
     byFlow:true, timeout:1});
  udpCount_dns = Count(‘udpflow_dns');
  CF_dns = CalFramePerSec(udpCount_dns);
  PFt_dns = ewma(CF_dns, PFtminusone_dns,
                  alpha_critical);
  threshold_dns_new = ((percentage + 1) * PFt_dns) +
                                  baseline_dns;
  PFtminusone_dns = PFt_dns;
  threshold_dns = threshold_dns_new;
  return;
}
```

iv. Web service handling function

```
function web(){
  setThreshold('web_abnormal',
 {metric:'tcpflow_websyn', value:threshold_web,
   byFlow:true, timeout:1});
  tcpCount_websyn = Count('tcpflow_websyn');
  CF_websyn = CalFramePerSec(tcpCount_websyn);
  PFt_web = ewma(CF_websyn, PFtminusone_web,
                  alpha_critical);
  threshold_web_new = ((percentage + 1) * PFt_web) +
                                  baseline_web;
  PFtminusone_web = PFt_web;
  threshold_web = threshold_web_new;
  return;
}
```

v. FTP service handling function

```
function ftp(){
  setThreshold('ftp_abnormal',
    {metric:'tcpflow_ftpsyn', value:threshold_ftp,
    byFlow:true, timeout:1});
  tcpCount_ftpsyn = Count('tcpflow_ftpsyn');
  CF_ftpsyn = CalFramePerSec(tcpCount_ftpsyn);
  PFt_ftp = ewma(CF_ftpsyn, PFtminusone_ftp,
                  alpha_critical);
  threshold_ftp_new = ((percentage + 1) * PFt_ftp) +
                                  baseline_ftp;
  PFtminusone_ftp = PFt_ftp;
  threshold_ftp = threshold_ftp_new;
  return;
}
```

vi. Mail service handling function

```
function mail(){
 setThreshold(‘mail_abnormal',
   {metric:'tcpflow_mailsyn', value:threshold_mail,
    byFlow:true, timeout:1});
 tcpCount_mailsyn = Count('tcpflow_mailsyn');
 CF_mailsyn = CalFramePerSec(tcpCount_mailsyn);
 PFt_mail = ewma(CF_mailsyn, PFtminusone_mail,
                    alpha_critical);
 threshold_mail_new = ((percentage + 1) * PFt_mail) +
                                   baseline_mail;
 PFtminusone_mail = PFt_mail;
 threshold_mail = threshold_mail_new;
 return;
}
```

3) **Event Handling**

According to the threshold violation result produced from the *setThreshold* function which compares the incoming traffic with the threshold, the *setEventHandler* function produces the respective abnormal event information for each type of service.

```
setEventHandler(function(evt) {
 if((evt.thresholdID == 'web_abnormal') ||
     (evt.thresholdID == 'ftp_abnormal') ||
     (evt.thresholdID == 'mail_abnormal') ||
     (evt.thresholdID == 'dns_abnormal') ||
     (evt.thresholdID == ‘dhcp_abnormal') ||
     (evt.thresholdID == 'ntp_abnormal'))
  logInfo(evt.flowKey+"  "+evt.value+"  "+
             evt.thresholdID);
 },['web_abnormal','ftp_abnormal','mail_abnormal',
 'dns_abnormal',‘dhcp_abnormal‘,'ntp_abnormal']);
```

APPENDIX B: FLOODING ATTACK MITIGATION

Flooding attack mitigation is implemented in the *ddosmitigation* application running in ONOS controller with the function of regular taking event information, tracing back the attack source and installation of drop flow rule.

1) **Getting event information from sFlow**

   The event information from the sFlow analyzer is periodically taken from the *ddosmitigation* in ONOS controller by using the URL: “*http*://*sFlowCollecterIP*: 8008/*events*/*json*”.

2) **Finding the source switch connected to the attacker host**

   According to the source-based defense mechanism, the source switch connected to the attacker host is needed to find for installing drop flow rule into it. Thus, the *ddosmitigation* application extracts the source MAC address of the attacker host from the event information and then finds the switch’s ID connected with it.

```
if (jsonText.contains("flowKey")) {
String key = "";
Double value = 0.0;
long timestamp=0;
long timestampNext=0, difftimestamp=0, initialdiffstable=0;
JSONArray arrayObj = new JSONArray(jsonText);
for(int i=0;i<arrayObj.length();i++)
{
JSONObject obj = arrayObj.getJSONObject(i);
timestamp=obj.getLong("timestamp");
key = obj.getString("flowKey");
String[] data = key.split(",");
Boolean match=false;
    if (match==false && timestamp>previousTimestamp) {
      String srcMac = data[0];
String delimiter = ":";
String Mac1 = srcMac.substring(0, 2);
String Mac2 = srcMac.substring(2, 4);
String Mac3 = srcMac.substring(4, 6);
String Mac4 = srcMac.substring(6, 8);
String Mac5 = srcMac.substring(8, 10);
String Mac6 = srcMac.substring(10, srcMac.length());
String completeStringMac =
    Mac1.concat(delimiter).concat(Mac2).concat(delimiter).concat(Mac3).
    concat(delimiter).concat(Mac4).concat(delimiter).concat(Mac5).concat
    (delimiter).concat(Mac6);
MacAddress srcMacAddress =
    MacAddress.valueOf(completeStringMac);
HostId srcHostId = HostId.hostId(srcMacAddress);
Host srcHost = hostService.getHost(srcHostId);
DeviceId sourceDeviceId = srcHost.location().deviceId();
```

3) **Checking the drop flow rule already installed for the current event**

In order to install drop flow rule effectively, the received events are checked whether it is new or not. If the new abnormal event information is received, the temporal drop flow rule will be installed to drop the abnormal frames from that flow. For the event that has been entered into this system and receive again after the expiration of its drop flow rule, the permanent drop flow rules are installed for this type of event.

```
IpAddress srcIpAddress = IpAddress.valueOf(data[2]);
IpAddress dstIpAddress = IpAddress.valueOf(data[3]);
SrcDstIpPair newIpPair = new SrcDstIpPair(srcIpAddress,
    dstIpAddress);
```

```
boolean found = false;
if (ipPairs.size() != 0) {
for (SrcDstIpPair st : ipPairs){
int p = ipPairs.indexOf(newIpPair);
if (st.equals(newIpPair)) {
if(p != -1){
found = true;
}
}
else {
found = false;
}
}
}
else {
log.info("Database is empty");
    }
    if(found == false) {
        ipPairs.addIfAbsent(newIpPair);
        installDropRuleT(sourceDeviceId,                srcIpAddress,
dstIpAddress);
         ruleInstallTime = timestamp;
         return;
      }
      else {
                timestampNext = obj.getLong("timestamp");
                difftimestamp = timestampNext - ruleInstallTime;
                if (initialdiffstable != difftimestamp) {
                    if (difftimestamp > 10000) {
                            boolean foundExist = false;
                            if (ipPairs.size() != 0) {
```

```
for (SrcDstIpPair st : ipPairExist){
                    int p = ipPairExist.indexOf(newIpPair);
                        if (st.equals(newIpPair)) {
                            if(p != -1){
                                foundExist = true;
                            }
                        }
                        else {
                            foundExist = false;
                        }
                    }
                }
                else {
                    //  log.info("Database is empty");
                }
    if(foundExist == false) {
        ipPairExist.addIfAbsent(newIpPair);
        installDropRuleP(sourceDeviceId,              srcIpAddress,
dstIpAddress);
         ruleInstallTime = timestamp;
         return;
      } else return;
        }
      initialdiffstable = difftimestamp;
        }else return;
          }
       } else
         {
log.info("It is normal flow");
             return;
           }
```

4) **Installation of flow rule with drop action**

Drop flow rule is installed by firstly define the source and destination IP for discarding the frames from the flow where is it incoming from and forwarding to. Then, the *DeviceID* is defined for the switch in which the drop flow rule to be installed and the *action* of the flow rule. Moreover, the flow rule can be installed permanently or temporarily. For temporal drop flow rule, the timeout value is needed to define when the installed drop flow rule is expired after any frame does not pass through the flow rule as shown in the function name of *installDropRuleT* with predefined timeout value for 60 seconds.

a. Temporal drop flow rule

```
private void installDropRuleT(DeviceId deviceId, IpAddress src, IpAddress dst) {
TrafficSelector.Builder selectorBuilder = DefaultTrafficSelector.builder();
IpPrefix srcIpPrefix = src.toIpPrefix();
IpPrefix dstIpPrefix = dst.toIpPrefix();
selectorBuilder.matchEthType(Ethernet.TYPE_IPV4)
.matchIPSrc(srcIpPrefix)
.matchIPDst(dstIpPrefix);
TrafficTreatment drop = DefaultTrafficTreatment.builder().drop().build();
flowObjectiveService.forward(deviceId, DefaultForwardingObjective.builder().fromApp(appId)
.withSelector(selectorBuilder.build())
.withTreatment(drop)
.withFlag(ForwardingObjective.Flag.VERSATILE)
.withPriority(40001)
.makeTemporary(60)
.add());
}
```

b. Permanent drop flow rule

```
private void installDropRuleP(DeviceId deviceId, IpAddress src,
IpAddress dst) {
TrafficSelector.Builder selectorBuilder =
DefaultTrafficSelector.builder();
IpPrefix srcIpPrefix = src.toIpPrefix();
IpPrefix dstIpPrefix = dst.toIpPrefix();
selectorBuilder.matchEthType(Ethernet.TYPE_IPV4)
.matchIPSrc(srcIpPrefix)
.matchIPDst(dstIpPrefix);
TrafficTreatment drop =
DefaultTrafficTreatment.builder().drop().build();
flowObjectiveService.forward(deviceId,
DefaultForwardingObjective.builder().fromApp(appId)
.withSelector(selectorBuilder.build())
.withTreatment(drop)
.withFlag(ForwardingObjective.Flag.VERSATILE)
.withPriority(40001)
.makePermanent()
.add());
}
```