Mitigation of DDoS Attack in SDN Technology Using Load Balancing Policy

H. Alshrif*, B. Ghariba[2]
[1] hmalshrif@elmergib.edu.ly, [2] bmghariba@elmergib.edu.ly
[1] Department of Electrical and Computer engineering, College of Engineering, Elmergib University, Libya
[2] Department of Electrical and Computer engineering, College of Engineering, Elmergib University, Libya

1 ABSTRACT

Software Defined networking (SDN) is a new network technology that has been introduced to address the common issues in traditional networking. It provides central control by decoupling the control plane from network nodes and placing it in the controller. This allows the programmers to manage the entire network from one side. security issues, however, are challenging in this technology. There are many security issues in SDN networks, but Distributed Denial of Service (DDoS) is one of the biggest threats to SDN technology. This paper focuses on creating load balancing policy that can exploit the unutilized paths and use them for rerouting the packets in datacentres, we demonstrate how DDoS attack can exhaust the SDN network and provides the proper approach to deal with this attack in a short time. This method can be considered the first step to prevent the severe impact of attacks and alarm the system to block the attack ports.

Keywords: DDoS attack, SDN security, Load balancing.

2 Introduction

Software Defined Network (SDN) technology is a new networking architecture which is designed to allow programmers to manage the entire network from one side as well as to tackle some security issues that have increased in the last decade [1]. SDN was designed by decoupling the control plane from network devices such as routers and switches etc., and place it in a centralized controller; this controller is responsible for managing and configuring the whole network. Thus, applying new security policies and mechanisms would be much easier compared traditional networks. Therefore, designing new protocols and functions will be easy and powerful in order to enhance network security mechanism [2].

Security issues however, are considered an essential concern due to the increasing number of attacks which typically have harmful consequences, from stealing data to making the network collapse. These attacks include Denial of service (DoS), Distributed Denial of service (DDoS), Spoofing etc. they can cause tremendous

losses for companies and organizations. The security issues of the SDN model can be categorized depending on the SDN layer affected, i.e. Application layer, Application-Controller Interface, Control layer and Controller-Data Interface threats [3]. This paper will focus on the Control layer threats, and how they can be mitigated. This layer is the brain of the SDN architecture that manages the entire network, and if it fails, then the network will collapse. For this reason, intruders focus on attacks against the controller to cause severe damage such as system failure or crashes. The distributed denial of service attack is one of the biggest threats not only in traditional networks but also in SDN technology, and it mainly occurs at the network layer or application layer of the compromised system [4]. When the attack is launched to the network it causes bottleneck problems and overwhelms the CPU resources. Therefore, understanding the characteristics of the incoming data can be considered as a first step in detecting attacks. There is also another approach to deal with DDoS which is the load balancing policy. This policy can be implemented in the controller which is then responsible for managing the traffic within the network [5].

3 Related Works

The security threats of SDN are common to traditional networking, but in SDN network (centralized controller), the impact of attacks such as DDOS could be worse than that directed against a single router [6]. Attacking network devices, such as Routers and Switches, cannot be done in an SDN network as they only have a data plane, as the control plane is taken from these devices and placed in the SDN controller. However, SDN networks are not fully secured, there are some issues with detecting abnormal traffics and identifying whether the intrusion has a severe impact or not. Fast intrusion detection is also vital to react immediately, especially when DDOS takes place in the network. The early detection method based on Entropy (randomness of incoming packets) can classify the type of attacks using a fixed window widths which are 50 packets for DDOS attacks and 500 for slow DOS attacks [7]. Although the effectiveness of this method is exceptional, using a small window size can cause confusion to differentiate between normal and malicious traffic. There is also another approach to deal with DDOS which is the load balancing policy, this policy can be implemented in the controller which is responsible for managing the traffic within the network [8]. Basically, it reroutes the traffic and balances it between the links that can be used to get to the destination. Thus, during the attack, this policy will reduce the impact of exhausting the network with loads by balancing it between nodes and links; it can be considered a first step to deal with DDOS attacks.

4 Materials and Methods

4.1 SDN controller

OpenFlow controller is considered a strategic point or a brain in Software-defined network [9]. It simply manages the network and handles all communications between applications and network devices (such as Routers, Switches, etc.) to effectively modify the flow to meet changing needs by using a communication protocol called OpenFlow. There are many types of SDN controller such as Floodlight, Ryu, Nox, and Pox. Each one has its own advantages and programming language. Ryu controller has been used in this paper due to its advantages as it is an open source, supports OpenStack and is well-tested. It is also written in python and supports various protocols for managing network devices such as OpenFlow 1.0, 1.2, 1.3, 1.4, 1.5, and Nicira Extensions [10]. Ryu controller can be used remotely to manage the network topology in Mininet by using its IP as follows [10]:

```
$ sudo mn --controller=remote,ip=[controller IP],port=[controller listening  port]
```

4.2 System design and requirements

The SDN network of Open V Switches, hosts, links, and controllers has been created virtually by using a network emulator called Mininet. It supports research, prototyping, testing, and any task that could benefit from having a complete experimental network that can be deployed on real hardware for performance evaluation [11]. Mininet provides a very good alternative to physical SDN networks for development, testing, and evaluation. The network that has been created by this program simulates a datacenter called a fat tree network as is shown in figure 1; it consists of two types of switches, Core switches, and Edge switches. Edge switches are connected to the servers; three servers in each Edge switch, whereas Core switches are connected to all Edge switches to maintain availability. Two other servers were added by Mininet, the first one is a testing server which is used to test the behavior of the network as it is like an attempt to access the network by normal users, and ping packets will be used to simulate this matter, from which we will gather some information such as the response time between the testing server and the datacenter server, and the number of dropped packets. The other servers (Zombies) were added for launching controlled attacks (DDoS) using an IPERF (Internet Performance Working Group) tool which is a commonly used network testing tool that can create TCP and UDP packets and measure the throughput of a network between two nodes.It allows the user to set various parameters that can be used for testing a network, or alternatively for

optimizing or tuning a network [12].

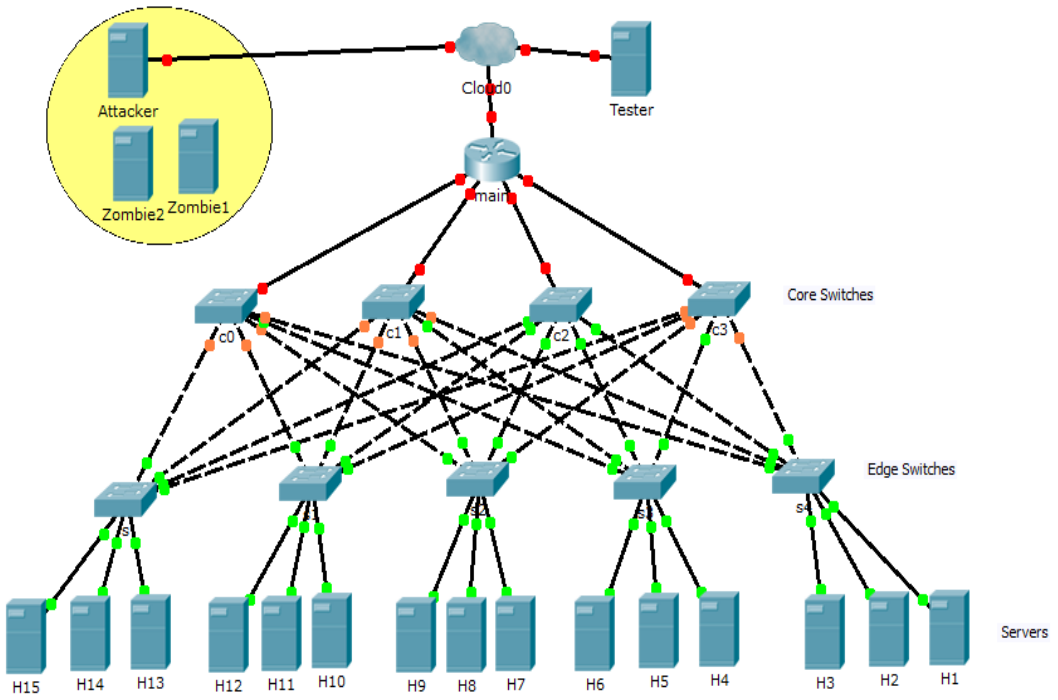

Figure 1. Fat tree topology of Datacenter with zombie's area

5 Theory and Calculation

5.1 Static policy

In the static policy, the SDN controller pushes the open flow table to the open V switches depending on the best route between each server. So the switches will assign each port a priority value and the MAC address of a server. Then the switches can identify where to forward the packets by using previous values as long as the network remains fixed (unchanged). However, if there are new devices connected to the network or the target of the incoming packet is not located in the routing table. Then the switch will send it to the controller which will update the routing table of the switch by sending back the proper route for that packet.

5.2 Load balancing policy

Load balancing policy distributes traffic intelligently between links in the network, it is adaptive, which continuously keeps making changes to the OpenFlow table of SDN switches. Using this policy in the controller allow us to find the best route at a specific time and send the update information to all

switches. Under high load flow, this policy will determine the unutilized switches and change the OpenFlow tables depending on that traffic. Figure 2 shows load balancing policy updating flow table diagram.

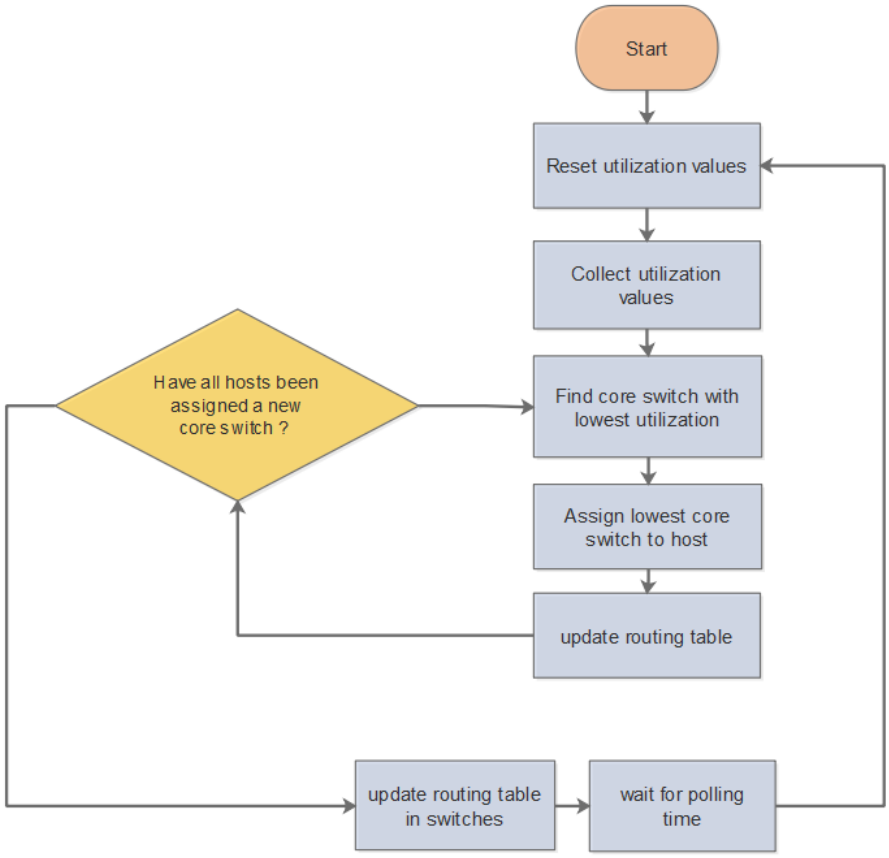

Figure 2 : Load balancing policy flow diagram

Evaluating these policies depend on two factors, time latency and throughput. These factors are considered the most crucial characteristics that measure the performance of a network. Latency is defined as the total time taken for a complete message to arrive at the destination. Networks with a longer delay have high latency, while those with fast response times have low latency. The second factor is throughput which is defined as the number of messages successfully transmitted per unit of time as in the equation below [13]:

$$Throughput = \frac{T}{D} \qquad (1)$$

Where T is the amount of the transferred data and D is the time duration (specific period of time).

6 Results and Discussion

The network topology that has been conducted in this paper is fat tree topology with one external server for testing and zombies for launching DDoS attacks. Table 1 shows experiments that has been conducted with low, medium and high load traffic flow against server 2 and 5.

Table 1: *Requirements of DDoS experiment*

Requirements	Experiment1	Experiment2
Datacenters links	Bandwidth = 5MB, queue_size = 1000, and delay = 1ms.	Bandwidth = 5MB, queue_size = 1000, and delay = 1ms.
Main switch links	Bandwidth = 10MB, queue_size = 1000, and delay = 3ms.	Bandwidth = 10MB, queue_size = 1000, and delay = 3ms.
Attack Loads	two levels of traffic flow are used, low (1MB), medium (2MB).	level of traffic flow are used, high load (10MB).
Victim	Server 2 and server 5 are the targets	Server 2 and server 5 are the targets

6.1 low and medium load evaluation

The graphs in Figure 3 below show how both policies, static and load balancing, manage the traffic flow of the network under low and medium loads. Experiments 1 and 2 show no noticeable change of average time response either in static or load balancing policies. The reason for that is that the network characteristics (especially Bandwidth) can accommodate the traffic flow without dropping any packets.

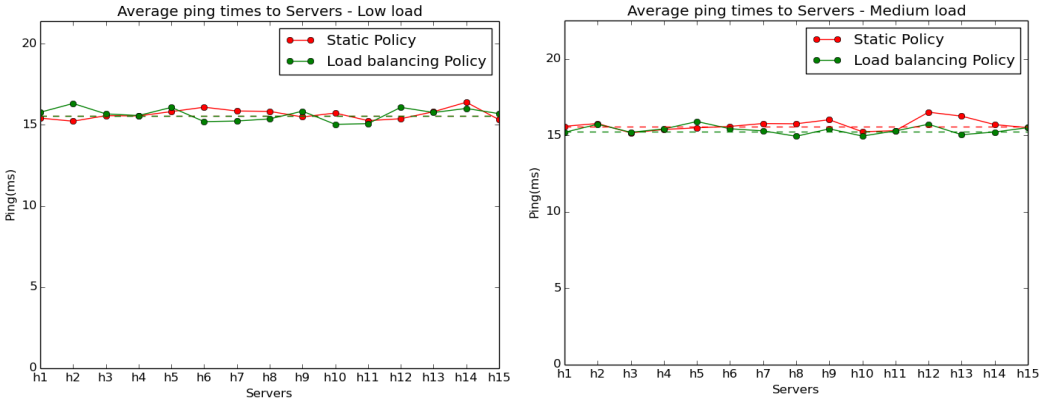

Figure 3: Average ping times to servers under low and medium loads

From the figure above, the ping response time is nearly the same in both policies, but in terms of power resources, load balancing policy consumes power and time to calculate the unutilized path continuously, unlike the static policy that uses a fixed routing table to forward packets to the destination.

6.2 high load evaluation

zombies servers have been used to attack two servers at the same time in order to exhaust the network or even to fail it. The targets were h2 and h5, and the load was different from the first experiment (high load 10MB), the dropped packets in static policy were extremely high and we can say that the network crashed under this policy, whereas the response time to servers under load balancing policy was incredibly short compared to the static policy.

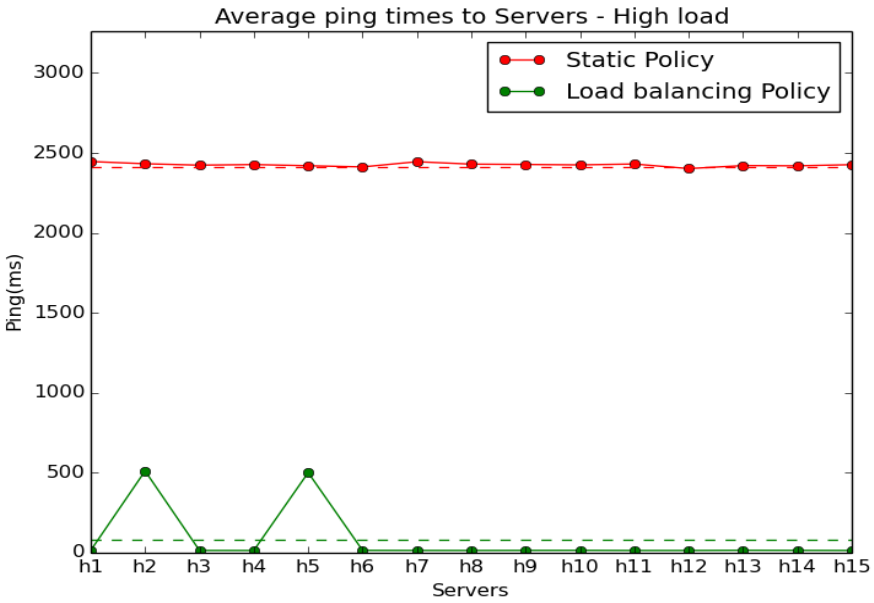

Figure 4 : Average ping times to servers

As is clear in figure 4, the average response times of servers under load balancing policy have an acceptable efficiency compared to static policy; moreover, the targets that have been attacked by zombies show an adequate improvement by utilizing the unusable paths, even the dropped packets in load balancing policy are almost nothing (only 11%) , while in static policy, around 70% of ping packets are dropped. The following Table 2 shows the comparison of both policies in terms of dropped packets, successful pings and ratio of dropped packets.

Table 2: *Ratio of dropped packets under high load attack*

Controller' s policy	**Dropped packets**	**Successful pings**	**Ratio of dropped packets**	**Servers that dropped pings**
Static	219	81	73%	All server, but server 2 and 5 are unreachable.
Load Balancing	33	267	11%	Server 2,5

The throughput in static policy is approximately 17.408 B/s whereas in load balancing policy is 143.104 B/s which indicates the noticeable adaptation of load balancing policy to cope with high traffic flow by utilizing all the network links to avoid network failure.

Generally, We can say that rerouting the network traffic between links provides an incredible improvement in network behaviour which indeed assists the network to cope with DDoS attack, and reduces the impact of this attack to give the network administrator enough time to block vulnerabilities in the SDN system.

7 Conclusions

SDN networks have been investigated in terms of security, and there was an adequate study of policy approaches that prevent or even mitigate DDoS attacks. From the results we found how load balancing policy mitigate the severe impact of DDoS attack by rerouting the traffic between links in the centres. However, static policy caused SDN architecture failure. Furthermore, load balancing policy in high loads attacks against many servers coped with the change and showed a high efficiency in preventing bottlenecks and maintaining availability. Alarming the system is recommended to inform network administrator which server was under attack.

References

[1] Kirkpatrik, K. , "Software-Defined Networking", *Communication of the ACM*, vol. 56, no. 9, pp. 16-19. 2013.

[2] Shin, Seungwon, Lei Xu, Sungmin Hong, and Guofei Gu. "Enhancing network security through software defined networking (SDN)." In *2016 25th international conference on computer communication and networks (ICCCN)*, pp. 1-9. IEEE, 2016.

[3] Scott-Hayward, Sandra, Sriram Natarajan, and Sakir Sezer. "A survey of security in software defined networks." *IEEE Communications Surveys & Tutorials* 18, no. 1 (2015): 623-654.

[4] Barki, Lohit, Amrit Shidling, Nisharani Meti, D. G. Narayan, and Mohammed Moin Mulla. "Detection of distributed denial of service attacks in software defined networks." In *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 2576-2581. IEEE, 2016.

[5] Belyaev, Mikhail, and Svetlana Gaivoronski. "Towards load balancing in SDN-networks during DDoS-attacks." In *2014 international science and technology conference (modern networking technologies)(MoNeTeC)*, pp. 1-6. IEEE, 2014.

[6] Hu, Zhiyuan, Mingwen Wang, Xueqiang Yan, Yueming Yin, and Zhigang Luo. "A comprehensive security architecture for SDN." In 2015 18th International Conference on Intelligence in Next Generation Networks, pp. 30-37. IEEE, 2015.

[7] Oshima, Shunsuke, Takuo Nakashima, and Toshinori Sueyoshi. "Early DoS/DDoS detection method using short-term statistics." In 2010 International Conference on Complex, Intelligent and Software Intensive Systems, pp. 168-173. IEEE, 2010.

[8] Belyaev, Mikhail, and Svetlana Gaivoronski. "Towards load balancing in SDN-networks during DDoS-attacks." In 2014 international science and technology conference (modern networking technologies)(MoNeTeC), pp. 1-6. IEEE, 2014.

[9] Sdxcentral (2017) 'What is an OpenFlow Controller?' [online] available from https://www.sdxcentral.com/sdn/definitions/sdn-controllers/openflow-controller/

[10] Ryu (2017) Building SDN agilely [online] available from < https://osrg.github.io/ryu/>

[11] Mininet (2017) Mininet: An Instant Virtual Network on your Laptop (or other PC) [online] available from < https://mininet.org/>

[12] iPerf (n.d.) iPerf - The ultimate speed test tool for TCP, UDP and SCTP [online] available from https://iperf.fr/>

[13] ETSI. 2012. "Throughput Measurement Guidelines." Etsi. European Telecommunications Standards Institute. Access online on 22 september 2022 at https://docslib.org/doc/9842978/throughput-measurement-guidelines