



JIMMA UNIVERSITY
JIMMA INSTITUTE OF TECHNOLOGY
FACULTY OF COMPUTING

SDN based DDoS flooding attacks defense in cloud environment

By: Ashenafi Meshesha

**A THESIS SUBMITTED TO THE FACULTY OF COMPUTING
OF JIMMA UNIVERSITY IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF MASTER OF
SCIENCE IN COMPUTER NETWORKING**

Jimma Univeristy
Jimma, Ethiopia
June, 2018 G.C

JIMMA UNIVERSITY
JIMMA INSTITUTE OF TECHNOLOGY
FACULTY OF COMPUTING

SDN based DDoS flooding attacks defense in cloud environment

By: Ashenafi Meshesha

School of Computing Department of Computer Networking

Advisor: Dr. Towfik Jemal (Ph.D)

Approved by:

1. Dr. Towfik Jemal (Ph.D)

Advisor _____

Acknowledgements

I would like to take this opportunity to express my sincere gratitude to my supervisors, Dr. Towfik Jemal, for his extensive support and valuable comments in doing this thesis work. Also, I would like to sincerely thank my colleagues and friends during this work. I must acknowledge a number of peoples that I had make an email communication with them and they helped me a lot in regard to the research work. And last but not the least, I cannot forget my family's support during this entire phase – My parents, brothers and my wife. I know that I have given them enough confusion and disappointment throughout the journey, but I am glad that I have been able to catch up finally, to see them happy. I dedicate this thesis to my parents.

Thank you!!

Abstract

Cloud computing is one of the recent technology that provides different services from different platform for the users at any time, at anywhere using internet without any limitations. As cloud computing providing this service, the most serious challenge is that, a DDoS attack which interrupt an online service by generating a high volume of malicious traffic, which is called flooding-attack. Moreover, DDoS attack consumes all the available network resources thus rendering legitimate users unable to access the services. To tackle this problem different research works have been done and proposed to defeat this type of attack in traditional and SDN networks for the cloud computing systems.

In this thesis, we developed and investigated a pushback distributed defense mechanism or framework for private as well as public network domain DDoS attacks. The defense system has three major components: traffic monitoring with detection, attack identification and traffic control. The components are inter-dependent and working in hierarchical fashion. The traffic monitoring scheme monitors only high-rate outgoing flows at victim networks and identify the source of an attack in the network. Once the source of an attack is identified the traffic control daemon apply an ingress filtering to drops the packets belonging to these flows. Based on the rules implemented on the controller the rate limiting mechanism, limit the rate of an incoming traffic to the victim node and filter the traffic in its source network controller. For Distributed DDoS attack, the controller at the malicious source node network send a pushback request message to apply a rule to the victim node controller.

The proposed framework is evaluated with different performance metrics to analyze the detection of rate of an attack traffic, throughput, link bandwidth, attack and legitimate traffic drop rate, and system resource consumption during normal and attack state. The simulation model is designed and a

number of simulation experiments have been done on mininet virtual network setup. The results demonstrate that the scheme is capable of detecting flooding-based DDoS attacks, and the pushback defense framework can effectively mitigate attack traffic in order to sustain the quality of service for legitimate traffic.

Key words: Cloud computing, SDN, OpenFlow, DDoS attack, Pushback technique

Table of Contents

	Page
<u>Abstract</u>	<u>I</u>
<u>Table of Contents</u>	<u>III</u>
<u>List of Tables</u>	<u>VI</u>
<u>List of Figures</u>	<u>VII</u>
<u>List of Acronyms</u>	<u>VIII</u>
<u>Chapter 1.....</u>	<u>1</u>
<u>Introduction</u>	<u>1</u>
1.1 <u>Motivation</u>	<u>4</u>
1.2 <u>Statement of the problem</u>	<u>7</u>
1.3 <u>Objectives</u>	<u>8</u>
1.3.1 <u>Specific Objectives</u>	<u>9</u>
1.4 <u>Methodology</u>	<u>9</u>
1.5 <u>Scope</u>	<u>12</u>
1.6 <u>Organization of the Study</u>	<u>12</u>
<u>Chapter 2</u>	<u>14</u>
<u>Literature review</u>	<u>14</u>
2.1 <u>Cloud computing</u>	<u>14</u>
2.1.1 <u>Deployment models</u>	<u>16</u>
2.1.2 <u>Service models</u>	<u>17</u>
2.2 <u>DDoS attacks on Cloud environment</u>	<u>21</u>
2.2.1 <u>Infrastructure level attacks</u>	<u>24</u>
2.1.2 <u>Application level attacks</u>	<u>26</u>
2.3 <u>Impact of Dos/DDoS attack on cloud</u>	<u>26</u>
2.4 <u>DDoS Defense Scheme</u>	<u>29</u>

2.5	<u>DDoS defense deployment locations</u>	<u>32</u>
2.5.1	<u>Source based deployment</u>	<u>33</u>
2.5.2	<u>Access-point deployment</u>	<u>34</u>
2.5.3	<u>Intermediate-network deployment</u>	<u>34</u>
2.5.4	<u>Distributed defense deployment</u>	<u>35</u>
2.6	<u>Traditional and SDN Network</u>	<u>36</u>
2.6.1	<u>SDN</u>	<u>36</u>
2.6.2	<u>OpenFlow</u>	<u>39</u>
2.7	<u>OpenFlow Operation</u>	<u>41</u>
2.8	<u>SDN controllers</u>	<u>42</u>
2.9	<u>SDN for cloud computing</u>	<u>43</u>
Chapter 3		45
Related works		45
3.1	<u>Existing DDoS attack defense works</u>	<u>45</u>
Chapter 4		52
Proposed design		52
4.1	<u>Overview of Pushback</u>	<u>53</u>
4.2	<u>Cooperative defense components</u>	<u>58</u>
4.2.1	<u>Traffic Flow monitor</u>	<u>59</u>
4.2.2	<u>Attack identification</u>	<u>61</u>
4.2.3	<u>Attack mitigation</u>	<u>62</u>
4.3	<u>Work flow</u>	<u>64</u>
Chapter 5		63
Experimentation and Results		67
5.1	<u>Environmental Setup</u>	<u>67</u>
5.2	<u>Simulation topology</u>	<u>69</u>
5.3	<u>Mininet Simulation parameters</u>	<u>71</u>
5.4	<u>Traffic generation</u>	<u>72</u>

5.5	<u>Simulation Results</u>	<u>72</u>
5.5.1	<u>Throughput</u>	<u>73</u>
5.5.2	<u>Bandwidth measurement</u>	<u>76</u>
5.5.3	<u>Drop rate of an attack traffic</u>	<u>77</u>
5.5.4	<u>Drop rate of legitimate traffic</u>	<u>78</u>
5.5.5	<u>System resource consumption</u>	<u>79</u>
5.6	<u>Result discussion</u>	<u>80</u>
5.6.1	<u>Local defense</u>	<u>81</u>
5.6.2	<u>Cooperative defense</u>	<u>82</u>
Chapter 6	<u>.....</u>	<u>85</u>
Conclusion	<u>.....</u>	<u>85</u>
6.1	<u>Limitation</u>	<u>87</u>
References	<u>.....</u>	<u>88</u>
Declaration	<u>.....</u>	<u>91</u>

List of Tables

Table 4.1: Flow table entry	59
Table 5.1: Mininet Emulation parameters.....	72
Table 5.2: Drop rate of an attack traffic	78
Table 5.3: Drop rate of Legitimate traffic.....	79

List of Figures

Figure 1.1: DDoS attacks in cloud: Direct/indirect effects	6
Figure 2.1: Cloud computing architecture	15
Figure 2.2: Cloud computing deployment and service models	16
Figure 2.3: Virtualization	20
Figure 2.4: DDoS attack scenarios in cloud infrastructure	21
Figure 2.5: DDoS defense deployment location	32
Figure 2.6: Source-end based DDoS defense	33
Figure 2.7: Network-based DDoS defense	35
Figure 2.8: Monolithic and SDN paradigm.....	37
Figure 2.9: SDN architecture and it fundamental abstraction	38
Figure 2.10: OpenFlow protocol architecture	40
Figure 2.11: Flow table entry	40
Figure 2.12: Basic OpenFlow Operation	42
Figure 3.1: Detection Loop operation.....	48
Figure 4.1: DDoS attack in progress	54
Figure 4.2: DDoS attack Cooperative defense framework	56
Figure 4.3: Cooperative flow monitoring Algorithm	60
Figure 4.4: Cooperative DDoS attack defense Algorithm	63
Figure 4.5: Cooperative DDoS attack defense flow chart	65
Figure 5.1: OpenFlow Ryu controller	68
Figure 5.2: Simulation network setup	70
Figure 5.3: Mininet virtual network setup	71
Figure 5.4: Throughput of TCP and UDP without defense	74
Figure 5.5: Throughput of TCP and UDP with defense	76
Figure 5.6: Allocated Bandwidth during normal and attack periods....	76
Figure 5.7: System CPU consumption at different traffic rate.....	79
Figure 5.8: System CPU consumption with defense	80
Figure 5.9: Attack indentation at the local domain	81
Figure 5.10: DDoS defense cooperation between the controllers	83
Figure 5.11: DDoS defense cooperation between the controllers	84

List of Acronyms

Abbreviations	Description
SaaS	Software-as-a-Service
PaaS	Platform-as-a-Service
IaaS	Infrastructure-as-a-Service
SDN	Software defined network
DDoS	Distributed Denial of Service
UDP	User Data Gram Protocol
ICMP	Internet Control Message Protocol
DNS	Domain Name System
CPU	Central Processing Unit
DaaS	Desktop-as-a-Service
CaaS	Communications-as-a-Service
HaaS	Hardware-as-a-Service
DoS	Denial of Service
CIDoS	Cloud-internal denial of service
EDoS	Economic denial of service
ADos	Application denial of service
OVS	OpenFlow Virtual Switch
RPF	Router-based Packet Filtering
SAVE	Source Address Validity Enforcement
CBSPs	Cloud-based Security Providers
CPS	Cloud Service Providers
MTF	Multilevel Thrust Filtration

Chapter 1

Introduction

Today's Internet communication is changing. Cloud computing is an emerging new technology which provides a centralized pool of configurable computing resources and computing outsourcing mechanisms that makes available different computing services to different people [1]. With the use of cloud, many companies can scale up without having to invest large amount in new infrastructure, software license and building large data centers. Cloud computing technologies are saves costs of deployments, high availability of services, flexibility and easy scalability of nature when the services demand increases [1]. As a result of integration of many techniques such as grading, clustering, utilization computing and resource's sharing, cloud computing has been appeared as multi element's composition technology, it offers several computing services such as IaaS (infrastructure as service), PaaS (platform as service) and SaaS (software as service) based on pay as you use rule. But nevertheless, cloud computing end users participate in computing resources (co_ tenancy), and by which infrastructure computing can be shared by a number of users, some security challenges has been explained, reported and researched by the Internet community. One of the most serious security threats is bandwidth attack, which prevent other users from using cloud infrastructure services. This kind of attack can be done by a legitimate or illegitimate cloud computing users [2].

There are various network bandwidth attacks [3]. Distributed denial of service (DDoS) attacks and Internet worms are the two frequently occurred ones. The former attacks a specific IP address from many distributed sources at the same time to cause the communication congestion at the destination; and the later breaks out when a large number of Internet hosts are infected and then, scan vigorously for the vulnerable hosts to propagate.

Bandwidth attacks are typically distributed attacks. An attacker uses tools to gain root access to machines on the Internet. Once a machine is cracked, it is turned into a “zombie.” The attacker instructs the zombies to send bogus data to one particular destination. The resulting traffic can clog links, cause routers near the victim or the victim itself to fail under the load.

One major reason underlies the absence of a simple solution against bandwidth attacks: attackers can release high volumes of normal-looking packets on the Internet without being conspicuous or easily traceable. It is the mass of all packets together directed at one victim that poses a threat, rather than any characteristics of the individual packets. A dropping policy in routers based on per-packet characteristics will, therefore, not work. It is relatively easy, but rather useless, to detect a bandwidth attack in the vicinity of the victim: by measuring the traffic load on a link or in a router, the exceptionally high volume of packets can be detected. Unfortunately for the victim, determining that it is under attack will not make the packets go away. Harm has already been done the time the malicious packets reach (the vicinity of) the victim. A bandwidth attack should, therefore, be detected close to the attacker rather than close to the victim so that malicious packets can be stopped before they can cause any harm [4].

Traditional DDoS solutions are unable to respond in time or handle the immense bandwidth that these attacks impose on today’s networks. Both service providers and enterprises need a scalable solution that can detect and mitigate these behavioral security threats, immediately preventing them from spreading through the network and disrupting the services of customers and users [5].

SDN is a new paradigm in networking industry designed to solve the limitations of traditional networks flexibility, scalability and vendor locked features. Besides the fact that SDN has been proposed as a candidate of the next generation Internet architecture, companies like Google have already adopted SDN in their internal data centers [6]. With the decoupling of data and control plane and the

introduction of open communication interfaces between layers, SDN enables programmability over the entire network, promising rapid innovation in this area. If it is properly implemented SDN can actually be exploited to address the security challenges brought by cloud computing and the DDoS attack defense can be made more effective and efficient in the era of cloud computing and SDN [6].

In order to solve the security problems there are a number of DDoS defense mechanism are designed and proposed in traditional and SDN based networks for the cloud computing networks. By exploiting a number of features of OpenFlow protocol researchers designed a DDoS flooding attacks defense mechanisms at the number of locations: at Source node based, Network based, and Destination based networks. Each of the implementation of this defense approaches has its own advantage and disadvantage in protecting the network from different types of attacks.

In this thesis, we present a pushback scheme to detect possible attacks and defeat them at the source of an attack and at the destination, so that most of the attack packet are targeted to the victim node. This scheme makes the identification between good and bad packets easier, thus minimizing the amount of collateral damage. Moreover, it helps to reduce bandwidth usage by the attack traffic. Pushback monitors only high-rate outgoing flows at source networks and preferentially drops the packets belonging to these flows when it senses any existing Internet protocols. The approach uses a specific IP address or MAC address from many distributed sources at the same time to cause the communication congestion at the Victim. Based on this the detection, identification and mitigation of an attack can be done on a per-flow basis by matching either IP or MAC addresses.

Our work is an extension to the pervious works [3] and presents such collaborative mechanism to fight against DDoS attacks by making defense mechanism at the source node network and the victim node network domain

work together. Each of the controller in its domain collects local information which is extracted by the local defense module and makes filtering instructions in order to regulate the traffic.

1.1 Motivation

The growing of dependency of users accessing resources from different platforms at anytime and anywhere from where they are connected to internet is the newly observable trend in the interconnected community. In this regard the concepts of, cloud computing, big data and internet of things are recently evolved. Cloud computing is enables users and organizations to be access their files and resources easily through Internet. This situations enforced most of the organization to move their services and resources to the cloud computing system.

The basic objective for most of the companies working in this area is providing highly available service to the customer using the Internet is the recent need of the service providers and business organizations. Particularly in infrastructure based cloud services, companies like, Google are providing virtual infrastructure for the users in commercial base with distributed servers on the Internet. Cloud users from different sides of the world are with pay for a service system they use this network for their day to day business process. The resources and data's of the users should be protected, secure and always available in this system. And, users shouldn't be denied due to security breaches and attacks. But, this is common in cloud computing systems due to a number of reasons. One of this is that the feature of the technology by itself. Resource sharing with the users, direct access to Cloud infrastructures, etc.), are the main reason for the security problems in the cloud network. So, it need new and innovative solutions to protect both the users and the provider from security attacks, like DDoS.

As we described slightly the concepts in the cloud computing, the nature of the technology, and the architecture of the cloud computing network which is different from the traditional network. This nature makes the system and the services to be vulnerable to different types of attacks security threats. Due to

this, cloud computing have always been primary target for DDoS attacks.

In traditional network there are a number of mechanisms are designed to protect the network from different kind of DDoS attacks. And, this defense mechanisms are some are effective and some are ineffective in defending DDoS attacks. But, in cloud computing the DDoS attack defending mechanisms are a little different. Even though, some of the defense mechanism which are designed for traditional network are feasible and used in cloud network for DDoS attacks. In cloud computing system the way of attacker gain access to the network for the attack is very simple. Because, tenants are the owners of the Virtual machines they are using. So, this characteristics open the way for attackers simply to employ an attack from using a simple script to any of services and application which is connected to Internet.

As the number of cloud security alliance report reviewed by this [7], [8] work, the extent of the DDoS attack is still increasing and its impact also. In spite of deploying various security measures against cloud attacks, there are a number of issues that providers may face still now: for example, the tradeoff between public and private cloud, Insiders and outsiders threats, militancy, web-based access, and the simplicity of attacking methods. This reflects organization and cloud providers need to have a better security solution for their service availability. So, designing and proposing an effective DDoS attack detection and mitigation approach is an open research for the researchers.

For the distributed botnet attacks form multiple domain nodes to a single target node designing a single defense mechanism on the source node only can't be effective. Furthermore, defending attacking node at the victim also. A sophisticated attacker can easily evade detection by employing a large number of zombie machines around the world and make them send malicious traffic through different edge routers. As a result, if we only take a single router into accounts, the volume of malicious traffic accounts, the volume of malicious traffic might not be aggregated at a detectable level.

Designing an attack reaction and filtering malicious traffic at the source node, locally and globally with cooperation between the systems enable us to reduce the impact of an attack. The advantage of designing a distrusted defense system compared to source based, destination based defense system is more effective in defending an attacks in collaboration between the systems.

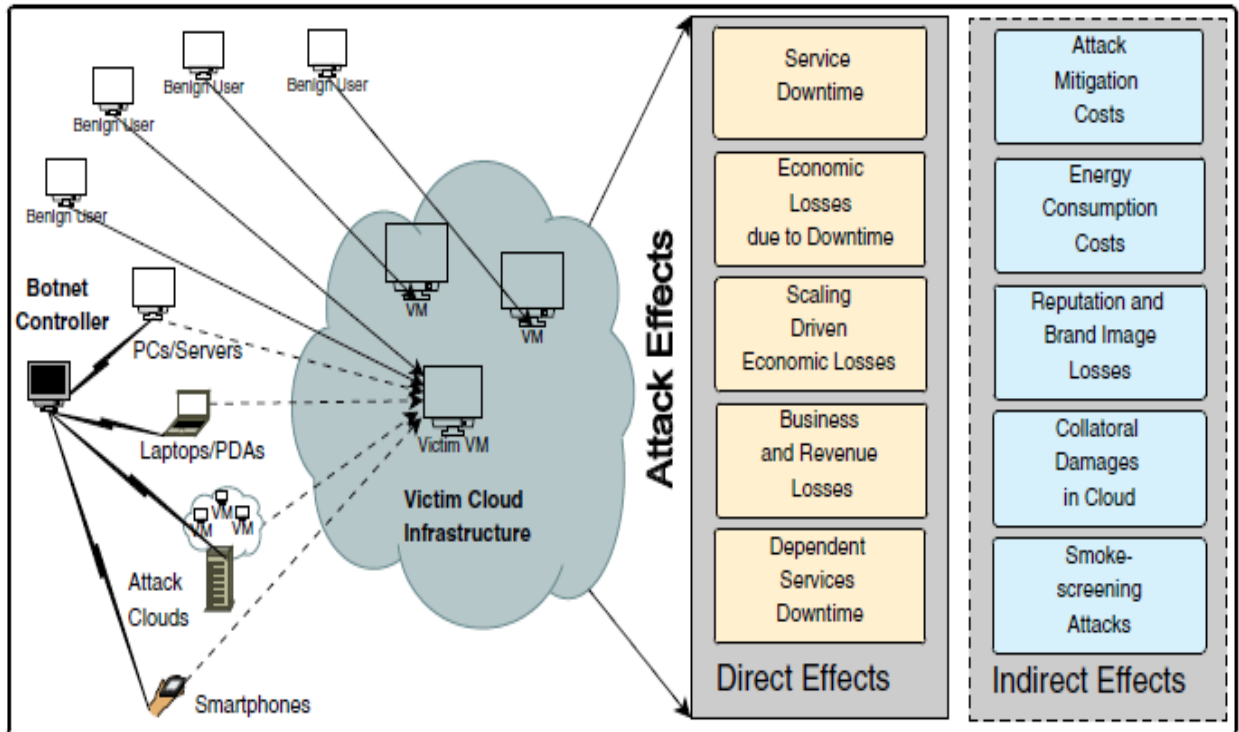


Figure 1.1: DDoS attack in cloud: direct and indirect effects [9]

In this regard, there are a number of works that has been done in traditional works in the detection of Botnet attacks at the Source node, destination (victim node), and also in a distributed manner. Even though, designing of this detection techniques was very complex and not reactive on the reaction and filtering of malicious traffic when the DDoS is noticed by the systems. Furthermore, some works has been designing the botnet attacks by employing an SDN methodology. Though, our work is designed with objective of reacting and filtering malicious traffic on locally by applying rate limiter at the source node and forward notification message to other controller in another domain.

In this thesis we design and investigate an SDN based pushback approach for DDoS attack detection and mitigation for cloud computing network. In this approach we monitor the rate of the traffic in the cloud network and identify an attacking node which generates malicious request to the target node from globally distributed private as well as public networks. Moreover, we investigate and analyze its feasibility in the detection and mitigation of DDoS flooding attack in the cloud system, scalability, flexibly, and dynamically.

1.2 Statement of the problem

Today, many of the enterprises are increasingly moving services and business applications from their internal data centers to external cloud service providers, whether by purchasing applications in the form of Software as a Service (SaaS) or by running those applications on cloud-based Infrastructure as a Service (IaaS) [10]. The basic needs for the enterprises moving their service to the cloud data center could be; to ensuring an availability and connectivity of the services for their mobile users they are accessing the cloud services using different networking devices or platforms, and good resource allocation for their cloud users or tenant's. As the nature of communication is changing in the cloud computing, security and vulnerability of different components of the systems is also a big problem for the cloud service providers and tenants. When physical resources are provided as logical re-sources via a virtualization layer, availability attacks on one virtual machine can affect other virtual machines which share the resources between them. Due to this feature, availability attacks on infrastructure are very intimidating on those virtual machines which share the physical resources.

A Distributed Denial of Service attack is one of the availability attacks. A DDoS attack generates a large amount of traffic to the specific system on the network. Therefore, it depletes the system resources, and end users do not receive reliable services. Botnet as the flood traffic comes from many machines, and is not a

single flow on the network. While it is possible to attack various types, it is difficult to detect this attack or to find its source.

As more applications, data, and websites move to the cloud systems, there is a need to design a flexible, scalable and easily manageable security mechanism for this environment. Traditional works have been tried to address DDoS botnet flooding attacks in the past, but DDoS attacks remain a major security problem and the detection and protection process is hard, especially when it comes to highly distributed implementations.

Therefore, a deployment of defense mechanism at different points in the network is an important consideration for creating an accurate filter to separate good traffic from attack traffic, and finding an efficient method to filter. Furthermore, cooperation between this defense mechanism in the detection process and reducing the impact of an attack on the system. Hence, providing a cooperative defense mechanism can be a significant improvement in this area. As a result of these problems, the following research questions have been identified:

- How to design a scalable, flexible cooperative DDoS attack defense mechanism to ensure service availability in cloud computing networks?
- What are the appropriate flow statistics features be used to detect DDoS attacks using a pushback algorithm?
- Does pushback is a feasible approach for detecting and mitigating DDoS flooding attack in the cloud computing environment when we employing SDN technology?

1.3 Objectives

The General objective of this thesis work is to develop and investigate a DDoS flooding attack defense framework and approach for the ever growing cloud computing services availability and security consolidation. By using the advantages and features of OpenFlow protocol, we design a system which enable

us to detect distributed attacks performed to the targeted node (victim node), and identity the attacking node in a flexible and scalable manner.

1.3.1 Specific Objectives

The specific objectives are enable us to achieve our general objective. So, in order to meet the above objective, the following specific objectives are defined:

- Reviewing and analyze various literatures that has been done as a defense mechanism for DoS and DDoS attacks in traditional and SDN based cloud networks and its features to designing an effective security mechanism
- Understand different types of services provided to cloud tenants by the cloud providers, and types of DDoS attacks that exist in cloud computing systems that impact the services reliability and availability with the methodologies in defending the attacks at different locations
- Design a cooperative DDoS flooding attack defense system for botnet attack to the target node by monitoring the rate of a traffic in the network and identify an attacking node in private as well as in public network domain. A proposed technique detects and identifies a DDoS attack at an early stage of the attack, and its mitigation process should drop most of the attack packets without sacrificing the QoS for legitimate traffic.
- Evaluate and measure the proposed technique with different performance measurement metrics in defending the DDoS attack when it is deployed in a distributed manner

1.4 Methodology

The research used different methodologies in order to accomplish the general and specific objectives of the study. This methodologies enable us to design a DDoS flooding attack defending technique for the cloud environment.

The first thing we conducted a comprehensive review of literatures to acquire a deeper understanding of the research area and its problem domains. Through

this literature we identify the importance of the previous works done in the area in DDoS flooding attack and the defending techniques used in the cloud environments.

Existing works done in defending DDoS attack whether in traditional network or SDN based network technologies assessed to identify and point direction in providing solution to identified problems on this network domain.

Secondly, we design and propose our defense mechanism which is relied on SDN based OpenFlow technology. The main reason of selecting SDN technology to design our solution is its flexibility, scalability, programmability and more of its features. In addition to this, SDN has an ability to globally view of the entire network which is separated or decoupled the data plane (the forwarding network infrastructure) and the control plane (decision maker in forwarding the incoming traffics). The separation of this functionalities and an interface to program the control plane, obliged us to design and implement our DDoS defense mechanism on the existing SDN framework. Our DDoS defense framework consist of three daemon in the detection, identification and mitigation of DDoS attacks.

In a DDoS attack scenario, the proposed distributed framework defends against attacks by coordinating between the local defenses systems at the source ends and the victim end. A native OpenFlow protocol approach is used to gathering, distributing and analyzing information from a distributed Ethernet network. The following concepts are the method of attack defending mechanism in the framework.

Detection of an attack: Each of the controller protects a server which provides services for the tenants on the cloud network by monitoring flows on all of switches in its domain for high bandwidth on the link. Moreover, the controllers monitoring the hosts to check if it is sending data to the victim and its data rate is above from the predefined threshold.

Source finding: Detect an attack and identify an attacker. To find source-end edge routers, traditional methods rely on the topological knowledge in each node and iterative communication among nodes. Finding a source an attacking node can be accomplished by the traceback component of the defense system at the victim node domain controller. We proposed a mechanism that identify filters that block attack traffic and allow legitimate traffic as close to the source node as possible, so that network resources are not wasted in propagating the attack.

Mitigation of an attack: The defense system at the victim network controller identifies the aggregate, it enables Pushback. It sends pushback message to its upstream adjacent OpenFlow switches asking them to rate limit the rate of its identified aggregates. When the pushback message arrives at the source OpenFlow switch, the controller identify which input links are the contributing links that are primarily responsible for the aggregate traffic. The OpenFlow switch at the source then probabilistically drop the flows coming from these contributing links until they do not receive more pushback messages from downstream routers.

The final step is an experiment. We performed several steps in our experiments for analyzing the incoming traffic and identifying an existence of an attack from the flow of the traffic and mitigate the rate of an attack on the cloud networks. For the experimenting we used a mininet virtual network traffic that is generated between the virtual machines. Ryu OpenFlow controller is selected for the design of our application codes in the framework. Using this tools, different DoS and DDoS attack scenarios are designed and performed. Then finally our approach is evaluated, with our evaluation metrics, like rate of attack traffic, throughput measurement, available bandwidth, time delay and system overload during normal and attack periods. The observation of the result from the experiment is collected and presented as graphs and tables. We also made a discussion on the performance of the overall technique proposed.

1.5 Scope

In this work, the DoS attack defense system and the security issue analysis of the cloud computing is on the network infrastructure only. The main reason for this is that, most of the DoS attacks are generated and arise from where the cloud network infrastructures. As most of DoS attack statics reports, a large number attack traffics are sent from malicious users or VM's in the data center to different cloud networks easily. Because, the intention of an attackers in this cloud model is that, denying services by flooding the traffic congesting the links, consuming the bandwidths and system resources. The aim of the proposed defense systems is also, to detect malicious traffic on the network and increase detection rate and examine cloud based infrastructure attacks. We monitor all the flow of information by using PACKET IN message from the connected hosts, extracting this flow information's, identifying the malicious from, the normal flow and also mitigate an attack when a malicious flow is detected.

Moreover, our DDoS defense mechanism is on botnet attacks generated with-in private as well as in public cloud computing environment. Furthermore, the feasibility of the proposed system is analyzed with simulated data traffic that is generated in the mininet simulation environment.

1.6 Organization of the study

On the rest of this thesis work, we discuss a part of our work. Chapter 2 starts with introducing what the cloud computing and its services and models. It also discusses about DoS attack and how DDoS attack impact the cloud systems particularly at the Infrastructure level. Furthermore, the defense schemes and deployment of this defense mechanism are looks like. We also reviewed, some background information on SDN, OpenFlow architecture and, its basic feature that helps us in defending a DDoS flooding attack on the SDN virtual environment. Chapter 3 discusses related works that have significant relation with this thesis. Even if there are a number of works done on this area, we selects the most related works to our thesis and presents them based on their attack

defense approach. Chapter 4 shows the designed and the proposed work on the existing SDN framework. The section also detailed the coordination of the different components in attack defending process. Here, a pushback DDoS attack detection system is selected and designed due to its effectiveness in detection and identification of attack on the system without any overhead on the controller for the distributed cloud systems. On chapter 5 the proposed system implemented and simulated using Software defined network emulation tools. The simulation is done on virtually created mininet network. The result of this experimental analysis is described and explained. Finally, we summarize the contributions made in the thesis, and conclude our work based on the results obtained from the thesis. Furthermore, new issues that have been surfacing while working on the thesis will be suggested as future work.

Chapter 2

Literature Review

In this section we discussed the state of arts in DDoS attack in cloud, the major types of attack on the cloud computing environment, and the open security challenges in designing the defense mechanisms. DDoS attack is one of the security challenges on the cloud computing environment that results in minimizing the performance of the cloud services or blocking its availability from legitimate users. The existing DDoS attack detection methodologies, and the placement of this DDoS detection mechanisms is reviewed. Where the defense mechanisms can be deployed and effective in defending an attack. Moreover, we discussed features of SDN that helps us to design our defense mechanism. The motivations of the attackers and the tools they use for attacking, are also addressed. In addition to the primary motives of the attacker, the attacking techniques as well as their targets and the main problems due to these attacks are reviewed.

2.1 Cloud Computing

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

This cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models. The essential characteristics of cloud computing's are [11]:

- *On-demand self-service*: computing resources can be acquired and used at any time without the need for human interaction with cloud service providers. Computing resources include processing power, storage, virtual machines etc.

- *Broad network access:* the previously mentioned resources can be accessed over a network using heterogeneous devices such as laptops or mobiles phones.

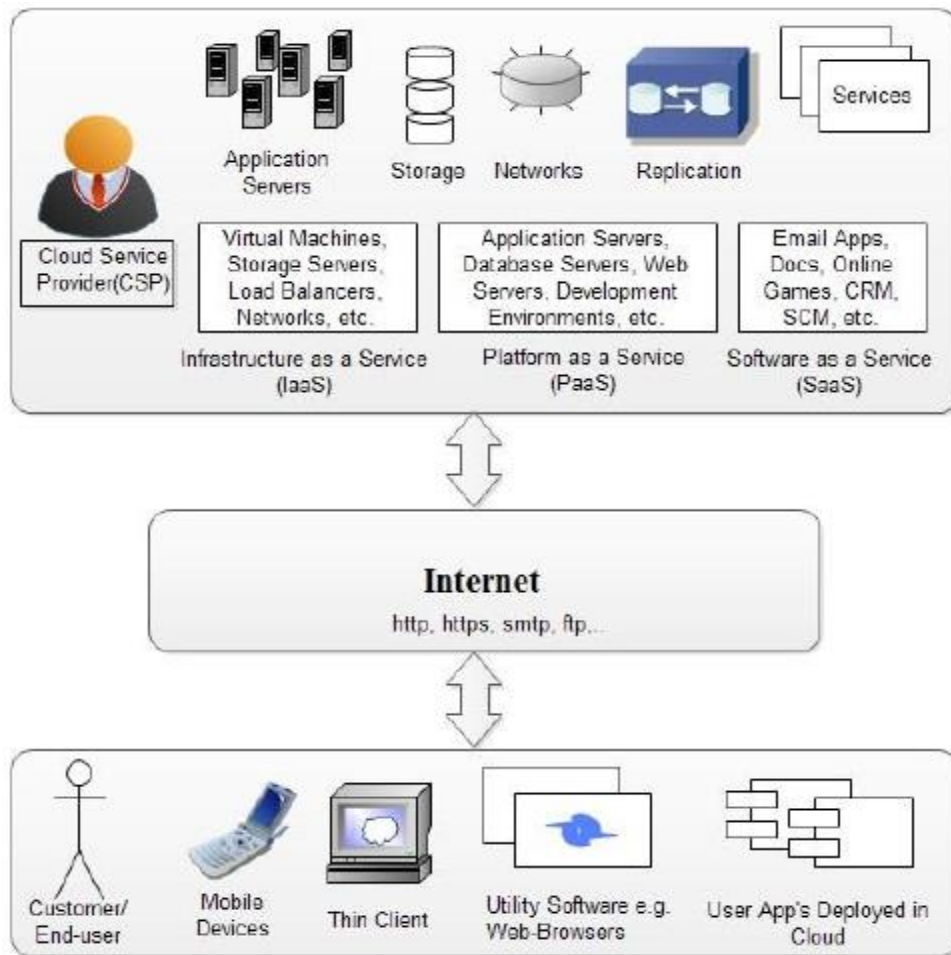


Figure 2.1: Cloud computing Architecture [12]

- *Resource pooling:* cloud service providers pool their resources that are then shared by multiple users. This is referred to as multi-tenancy where for example a physical server may host several virtual machines belonging to different users.
- *Rapid elasticity:* a user can quickly acquire more resources from the cloud by scaling out. They can scale back in by releasing those resources once they are no longer required.

- *Measured service:* resource usage is metered using appropriate metrics such as monitoring storage usage, CPU hours, bandwidth usage etc...

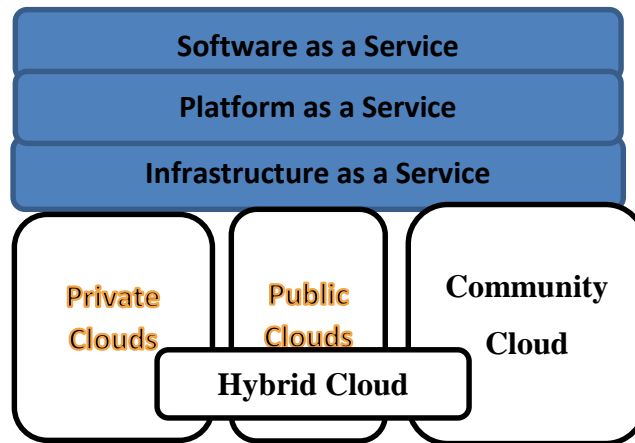


Figure 2.2: Cloud computing deployment and service models

2.1.1 Cloud deployment models

There are four types of cloud deployment models:

- *Private cloud:* Private cloud is a new term that some vendors have recently used to describe offerings that emulate cloud computing on private networks. It is set up within an organization's internal enterprise datacenter. In the private cloud, scalable resources and virtual applications provided by the cloud vendor are pooled together and available for cloud users to share and use. It differs from the public cloud in that all the cloud resources and applications are managed by the organization itself, similar to Intranet functionality. Utilization on the private cloud can be much more secure than that of the public cloud because of its specified internal exposure. Only the organization and designated stakeholders may have access to operate on a specific Private cloud [13].
- *Community cloud:* The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and

compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises.

- *Public cloud:* The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider.
- *Hybrid cloud:* The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds).

2.1.2 Service models

Clouds use architectural models in order to provide different services to the users. Service models are not tied to a specific deployment type, public, private, hybrid and community, rather each deployment type can use each service model. Just as with the different deployment methods the service models can have implications for a clouds security state, it is therefore important to have knowledge of these service models. The common service models are explained below.

A. Cloud Infrastructure-as-a-Service (IaaS)

The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which could include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, deployed applications, and possibly limited control of select networking components (e.g., host firewalls).

B. Cloud Platform-as-a-Service (PaaS)

This model provides a complete development environment to the customers, which includes all phases of SDLC with an appropriate support of APIs. PaaS facilitating a customer organization in developing software applications without investing huge on infrastructure which will be delivered to the users over Internet on-demand & rent (pay-as-you-use) basis. Web servers, application servers, development environment, runtime environment, etc. are the example components with respect to PaaS. In this model customers need not maintain underlying infrastructure including maintaining server machines, cooling, operating systems, storage, etc. Google AppEngine, force.com, Microsoft Windows Azure, RedHat, etc. are example of PaaS vendors [12].

C. Cloud Software-as-a-Service (SaaS)

Following this model, the cloud service provider makes both the software and the cloud infrastructure to run it available to the client, while it retains complete control over the underlying physical settings of the cloud (i.e., the operating system, network, storage, etc.) and the individual application capabilities. Thin client interfaces such as web browsers are often used to allow access to these applications [14].

D. Cloud Hardware-as-a-Service

Hardware as a Service, often abbreviated to 'HaaS'. It brought forth a significant improvement because it allows for easy access to physical hardware devices, distributed among several geographical locations. If the cloud consumers subscribe to this service, it will appear as if they are connected to the local machine. The HaaS cloud middleware will ensure transparency between data exchanges while the local system considers all connected hardware to be locally connected, even though this is not always the case.

E. Cloud Data-as-a-service

Data in various formats and from multiple sources could be accessed via services by users on the network. Users could, for example, manipulate the remote data just like operate on a local disk or access the data in a semantic way in the Internet. Amazon Simple Storage Service (S3) provides a simple Web services interface that can be used to store and retrieve, declared by Amazon, any amount of data, at any time, from anywhere on the Web. The DaaS could also be found at some popular IT services, e.g., Google Docs and Adobe Buzzword. Elastic Drive is a distributed remote storage application which allows users to mount a remote storage resource such as Amazon S3 as a local storage device [15].

Virtualization and multi tenancy are two of the core technologies that enables cloud computing to be used as we know it today. A traditional way of hosting applications and data storage involves running one operating system (OS) on one physical server. This traditional hosting method can also be used to create a functioning but inefficient cloud. This is achieved by linking multiple servers using a Virtual LAN (VLAN). This is secure but inefficient in the long term as a large part of the physical hardware available end up being unused.

Virtualization was created in order to solve this efficiency problem. By using a Virtual Machine Monitor (VMM) a single physical server can host multiple instances of an OS. This means that a single server can utilize the available hardware power in a more efficient manner. The figure below (Figure 2.3) is a basic illustration of a VMM running multiple instances of an OS using a virtualization layer. The virtualization layer is often known as hypervisor. There are two main ways of utilizing this hypervisor to run virtual machines (VM). These are known as full virtualization and paravirtualization. The difference between them lies in how much of the OS needs to be emulated. A VM deployed using full virtualization has to emulate the BIOS and drives of the OS, in addition to the other functions. A VM using paravirtualization runs a version of the OS

that has been modified to work without needing a BIOS or similar components [16].

There are also two major architectures used to deploy virtual machines, hosted architecture and hypervisor architecture. The difference here stems from the way the hypervisor is handled by the server. In a hosted architecture the hypervisor is a platform that the host OS runs as a normal application. The application is then charged with the upkeep of the virtual machines. On the other hand a hypervisor architecture skips the OS and is instead run directly on the hardware. Depending on which deployment method and architecture used different security aspects applied [17].



Figure 2.3: Virtualization of multiple operating system on one physical Machine

Multi tenancy is closely tied to virtualization. In short, multi tenancy allows several users to share computing resources with logical separation of the different users, a user in this case is a tenant of the system. In the context of cloud computing, each VM can be considered a tenant. However multi tenancy is not limited to multiple VMs running on the same hardware. Applications can also be utilised in a way that allows multiple tenants to use them, while at the same time separating the different users from each other [17] while virtualization and multi tenancy are core technologies needed for cloud computing to remain

efficient and viable they introduce new security risks. These are discussed in the next sections.

2.2. DDoS attack on the cloud network Infrastructure

Before dealing with possible detections and mitigations of attacks on Cloud Computing, the kinds of attacks and the types of attackers that are actually a threat to Cloud Computing shall be addressed. We shall first focus on the various forms an attack can take. There are multiple scenarios involved in the Cloud infrastructure itself and its environment. In a DDoS attack, some hosts (VM, PC or laptops), also called “bots” or “zombies”, can be controlled remotely. A collection of such bots controlled by a master entity (attacker) is known as a “botnet”. The typical attackers will be classified into three categories, according to their location, their motivation or their level of activity in the attack [16].

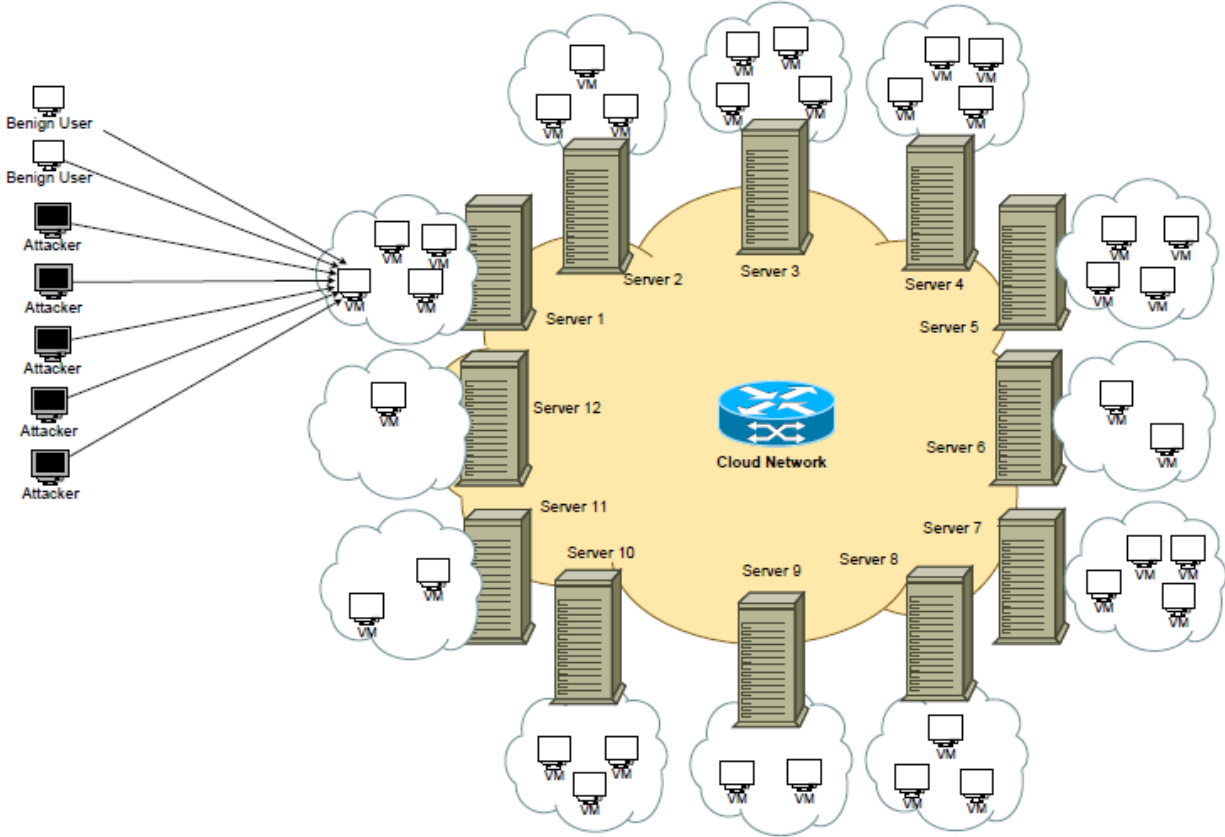


Figure 2.4 : DDoS Attack Scenario in Infrastructure Cloud [9]

Cloud computing infrastructures can be compromised in three ways: the attack can come from the outside and the target be inside (external to internal), it can even originate from within the system (internal to internal) and it can even occur from within to target the outside of the infrastructure.

1. External to internal. In such a case, the botnet used to perform the attack comes from outside the target system. The attack can target the internet gateway of the Cloud infrastructure, or the servers. If a particular client (in a VM) becomes the victim of an attack, it will also affect the other VMs present on the same physical server of the Cloud (performance interference between VMs).

2. Internal to external. In such a case, the attack begins by taking ownership of a VM running in the Cloud. This can be done with a Trojan horse. The choice of which customer's VM to infect is important because if this customer owns a large number of VMs, the Trojan horse can potentially spread over all those VMs, therefore forming a botnet. The great computing power and resource availability of the Cloud becomes a real threat for an external target.

3. Internal to internal. In the Cloud infrastructure, an internal botnet is formed and can attack another target inside the system (such as a VM or a group of VM). All Cloud infrastructures may break down under these kinds of attacks.

With the different kinds of attacks come different types of attackers. Indeed, each attack scenario corresponds to a particular attacker with a specific location and goals.

The scope of an attack may greatly vary, depending on who perpetrates the attack. System administrators take the appropriate actions: to exclude or to ensure a quick recovery and allow subsequent investigations. There are four categories of attackers that we will describe in the context of cloud computing.

1. Insider vs. Outsider. In such a case, the insider belongs to the network that is under attack: he is an authenticated user with privileged access to critical

data. Of course, the insider can do more harm than the outsider since the latter would be considered an intruder from the network perspective. Moreover, he would have fewer resources to begin an attack. In the case of Cloud Computing, an insider could be an employee of the Cloud infrastructure, or someone controlling one or several VMs inside the Cloud network, whereas an outsider would not be part of the network at all. For example, an insider attacker may be able to execute arbitrary commands on the behalf of a legitimate Cloud user, thus performing a DoS or DDoS on the user's services or to create a botnet for charging the Amazon Elastic Cloud Computing costs on the user's invoice.

2. Malicious or Rational. Malicious attackers have a general goal of harming the network or the network users (employees or customers of the network). Whatever the costs or the consequences, all means can be deployed to achieve his goal and such attackers are usually harder to stop or to track since no logic is involved. On the contrary, rational attackers can be more predictable in the way the attacks are led and which specific targets are reached. Consider the example of a DoS attack in Cloud Computing: a malicious attacker may want to destabilize an organization without any claim or consistent reasons to motivate his actions: he simply wants to be famous. However, a rational attacker could be a competitor desiring to create a commercial threat or an organization leading a DoS or DDoS against a company or a government for ideological reasons.

3. Active vs. Passive. Active attackers lead attacks by consciously or unconsciously sending packets or signals while passive attackers may simply eavesdrop. Victims may not even be aware that their machine is under the control of a master machine that forces it to contribute to the attack (a botnet is such an example). In DoS and DDoS attacks, this defines the difference between the zombies and the master entity (active attacker): both participate in the attack, but zombies are never aware that they are vehiculing an attack. In the context of Cloud Computing, an active attacker would have taken control of one or several VMs inside the Cloud network, for instance, and would send huge amounts of traffic or malformed packets to a specific host or subnet in the

network. Hence, a legitimate user such as a zombie whose VM was taken over by a master attacker, also performs the attack. A passive attacker consists on sniffing traffic to discover vulnerable links for future exploitations. In addition, passive attackers may launch eavesdropping attacks to capture the communication.

4. Local vs. Extended. The scope of the attacker depends on the number of machines he can control. More than just a number, it really is about how those machines are linked together and scattered across the network. An attacker controlling thousands of machines outside the cloud to perpetrate a DoS or DDoS would be considered an extended attacker. On the other hand, an attacker in the Cloud, with one or several entities, would be described as local.

2.2.1 Infrastructure level attacks

Network bandwidth, routing equipment and computing resources are considered infrastructure. In this attack, the intruder attempts to overwhelm the resource capacity of a private cloud's infrastructure by sending a large number of fake requests, which exploit the limitation of a specific application to cause performance degradation or ultimately crash remote servers. Some commonly used infrastructure level attacks are listed below.

a) Direct: A direct Denial-of-Service attack is characterized by an explicit attempt to prevent the legitimate use of a service. A Distributed Denial-of-Service attack deploys multiple attacking entities to attain this goal. A DDoS attack includes an overwhelming quantity of packets sent from multiple attack sites to a victim site. These packets arrive in such a high quantity that some key resource at the victim is quickly exhausted. The victim either crashes or spends so much time handling the attack traffic that it cannot attend to its real work.

b) Reflection/Indirect: It is a type DoS attack in which multiple compromised victim machines unwillingly participate in a DDoS attack. Flashes of requests to the victim host machines are redirected or reflected from the victim hosts to the target. Some reflection or indirect based attacks are mentioned below.

DNS (Domain Name Service) reflection or amplification attacks use botnets that send a large number of DNS queries to open DNS resolver using spoofed IP addresses of victims to produce an overwhelming amount of traffic with very little effort. Thus, such an attack can do a lot of damage as it is difficult to stop this type of attack at an early stage.

SSDP (Simple Service Discovery Protocol) reflection attacks are created using the Simple Object Access Protocol (SOAP) to deliver control messages to universal plug and play (UPnP) devices and to communicate information. These requests are created to elicit responses, which reflect and amplify a packet and redirect responses towards a target.

NTP (Network Time Protocol) reflection attacks are created by the attacker to send a crafted packet in which requests for a large amount of data are sent to the host. NTP is used to synchronize the time between client and server.

In an SNMP (Simple Network Management Protocol) reflection attack, the culprits send out a huge number of SNMP queries with forged IP addresses to numerous victim machines. SNMP is a network management protocol for configuring and collecting information from servers.

CHARGEN (Character Generator Protocol) is often misused when attackers use the testing features of the protocol to create malicious payloads and reflect them by spoofing the address of the source to direct them to the target. CHARGEN is a debugging and measurement tool and also a character generator service.

TCP SYN flood: Manipulating the 3-way handshake in a TCP connection, an attacker sends a lot of ordinary SYN segments to fill up resources causing a service to be denied for legitimate connections.

UDP flood: In this attack, massive amounts of UDP packets are sent to random ports on the victim side. Sometimes ports remain open without knowledge of administrators, causing the server to respond. A response to each UDP packet with an ICMP unreachable reply to the spoofed source IP address makes the situation worse by overwhelming the network environment of the victimized IP addresses.

ICMP flood: ICMP flood, occasionally referred to also as a Smurf attack or Pin flood, is a ping-based DoS attack that sends large numbers of ICMP packets to a server and attempts to crash the TCP/IP stack on the server and cause it to stop responding to incoming TCP/IP requests.

Crossfire Attack: A botnet can launch an attack with low intensity traffic flows that cross a targeted link at roughly the same time and flood it. For example, a botnet controller can compute a large set of IP addresses whose advertised routes cross the same link, and then direct its bots to send low-intensity traffic towards these addresses. This type of attack is called the Crossfire attack.

2.2.2 Application level attacks

Application layer DDoS attacks continue to grow in both complexity and prevalence.

Common application-layer DDoS attack types: When a heavy amount of legitimate application-layer requests or normal requests that consume large amounts of server resources or high workload requests across many TCP sessions are sent to the server, they can cause common application layer DDoS attacks.

HTTP flood attacks: Some application level DDoS attacks come in the form of HTTPGET floods. HTTP request attacks are those attacks where attackers send HTTP GETs and POSTs to Web servers in an attempt to flood them by consuming a large amount of resources. The HTTP POST method enables attackers to POST large amounts of data to the application layer at the victim side, and it happens to be the second most popular approach among the application layer attacks.

2.3 Probable Impact of DoS/DDoS on Cloud

As mentioned earlier, the cloud computing market continues to grow, and the cloud platform is becoming an attractive target for attackers to disrupt services and steal data, and to compromise resources to launch attacks. Miao et al. [12] present a large-scale characterization of inbound attacks towards the cloud and outbound attacks from the cloud using three months of NetFlow data in 2013

from a cloud provider. They investigate nine types (TCP SYN flood, UDP flood, ICMP flood, DNS reflection, Spam, Brute-force, SQL injection, Port scan, and Malicious Web activity (TDS)) of attacks ranging from network level attacks such as DDoS to application-level attacks such as SQL injection and spam. Cloud computing features a cost-efficient, “pay-as-you-go” business model. A cloud platform can dynamically clone virtual machines very quickly, e.g., by duplicating a gigabyte level server within one minute. Despite the promising business model and hype surrounding cloud computing, security is the major concern for a business that is moving its applications to clouds. When a DDoS attack is launched from a botnet with a lot of zombies, Web servers can be flooded with packets quickly, and memory can be exhausted quickly in an individual private cloud. So, we can say that the main competition between DDoS attacks and defenses is for resources. The increase of DDoS attacks in volume, frequency, and complexity, combined with the constant required alertness for mitigating Web application threats, has caused many Website owners to turn to Cloud-based Security Providers (CBSPs) to protect their infrastructure. In one recent analysis, DDoS attacks are considered one of the top nine threats to cloud based environments. This report concludes that cloud services are very tempting to DDoS attackers who now focus mainly on private data centers. It is safe to assume that, as more cloud services come into use, DDoS attacks on them will become more commonplace. Some key findings are provided by InfoWorld20, in 2013.

- 94 percent of data center managers reported some type of security attacks.
- 76 percent had to deal with distributed denial-of-service (DDoS) attacks on their customers.
- 43 percent had partial or total infrastructure outages due to DDoS attacks.
- 14 percent had to deal with attacks targeting a cloud service.

Unfortunately, the counterparts of clouds, e.g., client-server and peer-to-peer computing platforms, do not usually have sufficient resources to beat DDoS attacks. The public cloud infrastructure stands a better chance because a public cloud usually has a lot of resources that make it easy to handle a rapid increase

in service demands to counter the attack dynamically. It is almost impossible to shut down such clouds by attacking them. But, if an intense DDoS attack occurs on customers of an individual private cloud like a data center with limited resources, it cannot escape from the DDoS attack, and it becomes a battle of survival using all the resources there are to confront. The essential requirement to defeat a DDoS attack is to allocate sufficient resources to mitigate attacks no matter how efficient our detection and filtering algorithms are.

Cloud Service Providers (CPS) usually provide cloud customers two resource provisioning plans: short-term on-demand and long-term reservation. Giant cloud providers, like Amazon EC2 and GoGrid, provide both plans. If a customer chooses the first plan, it is charged based on resources used. This business model for resources is vulnerable to an Economic Denial of Sustainability (EDoS) attack. This kind of attack also disturbs the service of clouds that allocate resources based on spot instance. On the other hand, if a customer chooses the reservation plan, it makes a prior reservation for resources for the maximum usage for the business. In other words, the reserved resources for the application are limited from start. As a result, a threat of DDoS attack remains.

Some possible examples of DDoS attacks in cloud environments are Smurf attack, IP spoofing attack, Tear drop attack, SYN flood attack, ping of death attack, Buffer overflow attack, LAND attack, etc. From many news report we can state that large-scale IoT-enabled DDOS attacks will continue to dominate enterprise security. Darwish et al, discuss DDoS attacks as attacks that target the resources of these services, lowering their ability to provide optimum usage of the network infrastructure. Due to the nature of cloud computing, the methodologies for preventing or stopping DDoS attacks are quite different compared to those used in traditional networks, and new approaches published till now are usually adapted versions of older approaches. In the above mentioned papers, we can find descriptions about the effect of DDoS attacks on cloud resources and recommend practical defense mechanisms against different types of DDoS attacks in the cloud environment.

2.4 DDoS Defense Scheme

Defense mechanisms proposed in research literature against DoS (and DDoS) attacks can be divided into three main categories [17] :

1. Attack Prevention
2. Attack Detection
3. Attack Reaction

1) Attack Prevention: Attack Prevention aims to preempt attacks before they cause damage. This approach is effective against DoS attacks in which the source address of attack traffic is spoofed to hide the real source of the attack traffic and exploit protocol vulnerabilities. The main exponents of this approach is Ingress/Egress Filtering, Router-based Packet Filtering (RPF) and Source Address Validity Enforcement (SAVE).

Ingress filtering involves filtering the traffic coming into a local network, and egress filtering involves filtering the traffic leaving a local network. The purpose of ingress/egress filtering is to only allow traffic to enter or leave the network if its source addresses are within the expected ip address range in the network. Thus, as a result of deploying ingress/egress filtering, spoofed ip packets with source IP address not within the network are dropped, thereby mitigating the effect of DoS attacks.

RPF extends ingress filtering to the core of the Internet. It is based on the principle that for each link in the core of the Internet, there is only a limited set of source addresses from which traffic on the link could have originated. In the event that an unexpected source address appears in an ip packet on a link, we can infer that the source address has been spoofed, and hence filter the packet.

SAVE uses a protocol that can provide routers with information needed for source address validation. SAVE messages propagate valid source address information from source location to all destination, allowing each router along

the way to build an incoming table that associates each incoming interface of the router with a set of valid source address blocks.

The techniques described above involve changes in network infrastructure and protocols. Hence, unless policies or regulations are implemented for their enforcement, it is difficult to deploy such techniques.

2) Attack Detection: There are two main categories of DoS attack detection techniques - DoS-attack-specific detection and anomaly detection. DoS-attack-specific detection utilizes characteristics of DoS attack traffic. Since DoS traffic is generated by the attacker, it does not typically follow traffic control protocols. There is an imbalance in the traffic between source and victim since the victim is not able to handle all incoming packets. This is not the case for normal traffic. MULTOPS [4] is based on the assumption that packet rates between two hosts are proportional during normal operation.

It monitors traffic rates in up and down links to detect disproportional traffic between hosts in order identify DoS attacks. TOPS [2] uses a similar approach but is more memory efficient on account of use of hashing scheme with a small set of field length lookup tables. Other methods such as [14] define a statistical model of normal traffic and then identify traffic which does not match this model to be attack traffic.

Anomaly detection builds a model of normal traffic using training data. If the monitored traffic is statistically different from the model, then it can be inferred that a DoS attack is in action. The first real-time intrusion detection model was proposed in [8]. It detected attacks by monitoring a system's audit records for abnormal patterns of system usage.

The main drawback of using attack detection techniques mentioned above is that depend heavily upon a traffic model which may not be universally applicable. Building a traffic model and making online statistical comparison between normal and observed traffic is also time-consuming and costly.

3) Attack Reaction: The main aim of DoS attacks is to damage the target as much as possible. Attackers typically do not disguise the attack since the target will be aware of the attack damage eventually. The attack detection techniques mentioned attempt to detect an ongoing attack in the minimal possible time. In order to minimize the damage caused by DoS attacks, a reaction scheme must be employed after an attack has been detected. DoS attacks not only affect the end-host victim but also congest the intermediate links between the source and the victim. Attack reaction will be most effective if the attack traffic is filtered as close to the source as possible.

Attack reaction techniques can be classified into host-based reaction which takes place only at the end host and network-based reaction which takes place at intermediate routers (and optionally end-hosts). An example of network-based reaction is. It uses an online scheme in which intermediate routers learn a congestion signature based on the victim's IP address and the volume of traffic directed towards that IP address. Once a signature has been identified local congestion control is filter attack traffic. In addition, a Pushback mechanism is used to request upstream adjacent routers to rate-limit traffic matching a specified signature. [21], uses a Selective Pushback mechanism that sends pushback messages to the routers closest to the attack sources directly by analyzing the traffic distribution change of all upstream routers at the target.

The techniques mentioned above, though effective, require the co-operation of routers for their implementation. In many cases, a victim may not have access to such routers. In such a scenario, defense mechanism must be implemented on the end-host alone. While such techniques cannot completely stop an attack, they can mitigate the damage caused. An example of such a technique is SYN-cookies using which a host does not need to keep track of half-open connection states thereby mitigating the effects of a SYN-Flood.

Another technique is using system and network interface logs on the host to identify IP address from which malicious traffic is originating and then filtering

those using tools such as iptables/net filter. Malicious traffic can also be identified using a history maintained by the host.

2.5 DDoS Defense deployment location

For the detection and prevention of DDoS attack, four types of designs were proposed by the researches. Each of the designs have their own benefits and drawbacks in the detection and prevention of the different types of attacks on -

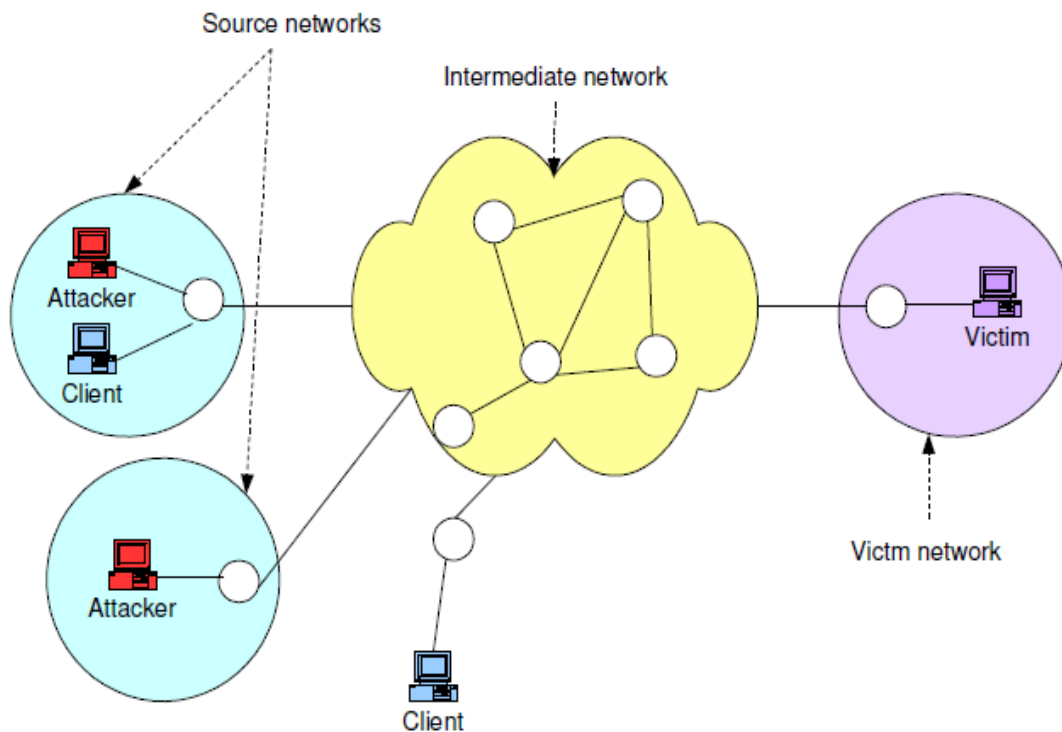


Figure 2.5: DDoS defense deployment locations [18]

the cloud system. Here below we explained the design of the defense mechanisms from the state of arts. From our analysis, our defense mechanism also uses one of the defense system design for our DDoS flooding attack detection in the cloud system. But, our analysis is limited and considers only the defense mechanisms to the cloud systems. In the cloud system mostly attacks are generated from the inside network or external networks. So, the locality of deployment, for DDoS defense the design of the defense system should have to consider this scenarios.

There are four defense systems are designed: source based, network based, destination and distributed based. When a DDoS attack is detected, and there is nothing that can be done expect manually fix the problem and disconnect the victim system from the network. DDoS attacks blocks a lot of resources such as CPU power, bandwidth, memory, processing time, etc., on the paths that lead to the targeted system. The main processing time, etc., on the paths that lead to the targeted system. The goal of any DDoS defense mechanism is based, and distributed form of attack defense mechanisms.

2.5.1 Source-end DDoS attack defense mechanism

A generic architecture of source-end preventive schemes is shown in Figure 2.6. This architecture is similar to the victim-end detection architecture. Here a throttling component is added to impose rate limit on outgoing connections. The mechanism. The observation engine compares both incoming and outgoing traffic statistics with some predefined normal profiles.

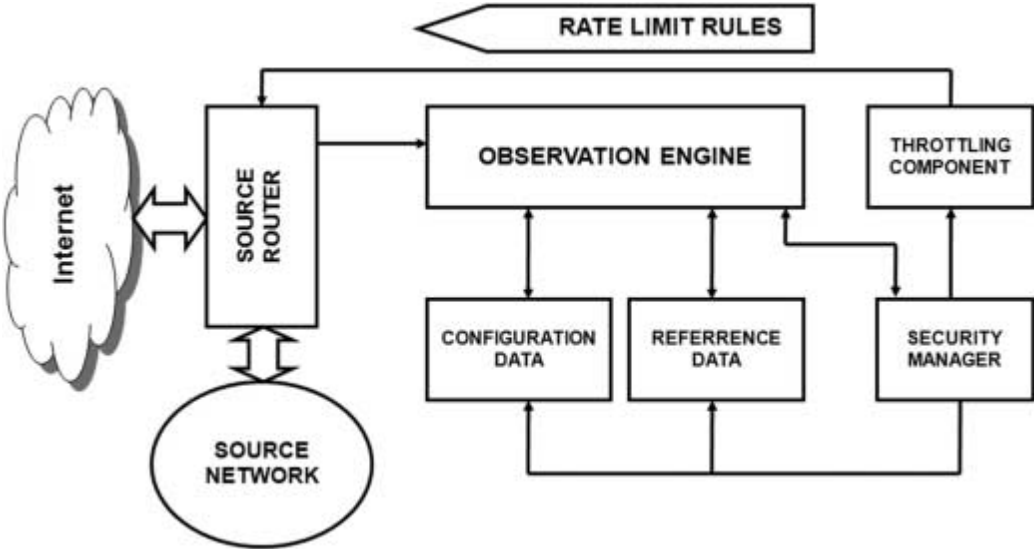


Figure 2.6: Generic architecture for source-end based DDoS defense [8]

Detecting and stopping a DDoS attack at the source is the best possible defense. It prevents the possibility of flooding not only on the victim side, but also in the whole intermediate network. The main difficulty with this approach is that,

detecting DDoS attacks at source end is not easy. This is because in these attacks, sources are widely distributed and a single source behaves almost similarly as in normal traffic. Another problem is the difficulty of deploying system at the source end [8].

2.5.2. Access point deployment

Access point deployment is usually deployed in the front-end, back-end or each virtual machines (VMs) in the cloud computing environment. The front-end is typically the administrative domain of the cloud service that serves as an interface between the cloud user and the various cloud components. In Eucalyptus, for example, this is referred to as the cloud controller, while in Xen, it is known as dom0. DDoS defenses deployed at the access point distinguish legitimate packets from malicious packets before granting access to the cloud computing resource and services. Key limitation of this deployment is that the access point is generally not the most suitable place for filtering or rate-limiting as bandwidth might be saturated. However, this approach is most commonly deployed due to the ease of deployment, and SBTA (Yang et al., 2012) is one popular example.

2.5.3 Intermediate-network defense mechanism

The intermediate network defense scheme balances the trade-offs between detection accuracy and attack bandwidth consumption, the main issues in source-end and victim-end detection approaches. Figure 2.7, shows a generic architecture of the intermediate network defense scheme, one that can be employed in any network router. Such a scheme is generally collaborative in nature and the routers share their observations with other routers. Like a source-end scheme, these schemes also impose rate limits on connections passing by the router after comparing with stored normal profiles.

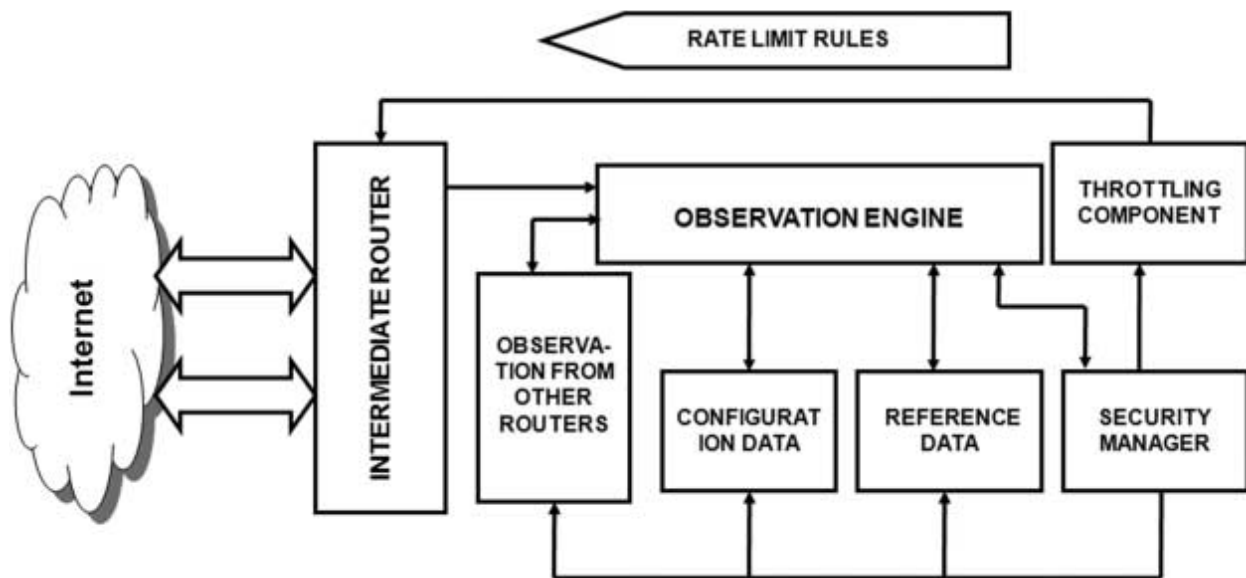


Figure 2.7: Generic architecture for intermediate network-based DDoS defense mechanism [8]

Detection and traceback of attack sources are easy in this approach due to collaborative operation. Routers can form an overlay mesh to share their observations. The main difficulty with this approach is deployability. To achieve full detection accuracy, all routers on the Internet will have to employ this detection scheme, because unavailability of this scheme in only a few routers may cause failure to the detection and traceback process. Obviously, full practical implementation of this scheme is extremely difficult by reconfiguring all the routers on the Internet [8]. Such a deployment can be effective but it is impractical in a cloud computing environment as the nodes are not controlled by the same provider and are in different administrative domains. This could, perhaps, work in private cloud deployment.

2.5.4. Distributed-end or Hybrid Defense architecture

Attack detection and mitigation at distributed ends can be the best strategy against DDoS attacks. The hybrid defense mechanisms are deployed at (or their components are distributed over) multiple locations such as source, Victim or intermediate networks and there is usually cooperation among the deployment

points. The core-end is best to rate-limit all kinds of traffic whereas the victim-end can accurately detect the attack traffic in a combination of legitimate and attack packets. Therefore, distribution of methods of detection and mitigation at different ends of the network can be more beneficial [19]. MTF (iyengar et.al, 2014) is an example of a distributed defense deployment.

2.6 Traditional and SDN Networks

Software defined networking is a promising technology which has an ability to view the entire network. The network infrastructure namely the data planes for forwarding traffics from cloud network administrative domain to the other remote domain. The controller on top of this network infrastructure domains manages the flow of traffic with the configure rules. Employing this methodology, helps us to design a simplified version of a DDoS attacks mitigation solution. The next section we will explain some of the concepts and features of this methodology.

2.6.1 SDN: A New Network Paradigm

SDN is a framework that separated the data plane and the control plane of the network switches and moves the control plane to a centralized application known as Network controller. The network controller maintains the entire network through a vendor-independent interface called as OpenFlow, which defined the low-level packet forwarding behaviors in the data plane.

The application layer will have a single view of the network through the control layer and the whole system looks like one logical switch. The control layer is where the controller abstracts the network infrastructure from the application layer. By using the control layer, any configurations and modifications can be done in real-time. In the infrastructure layer, there is no need for each device to learn different protocols and the only task left is forwarding.

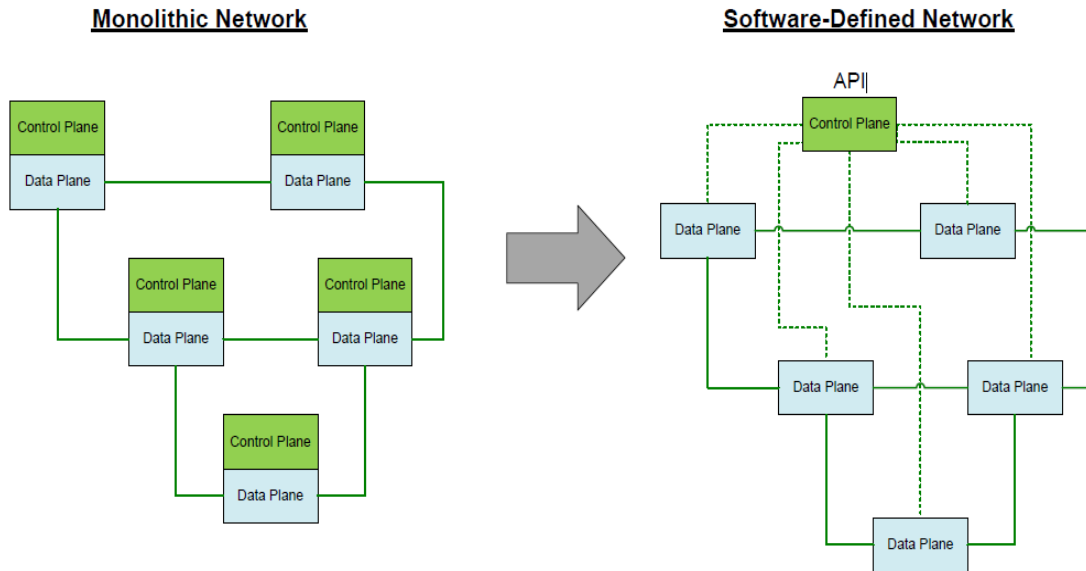


Figure 2.8: Monolithic vs Software-Defined Network Paradigms

In this new network architecture, control is decoupled from the network devices, and is directly programmable. The network devices become simple packet forwarding devices, which receive control instructions from a logically centralized entity known as the controller. By logically centralized we mean that control logic is to be designed and operated as if it was a centralized application, rather than a distributed state. However, the controller itself may be a distributed system, as is in fact the case with production SDNs, such as Google’s private Wide Area Network (WAN).

Current networks have no powerful control plane abstractions. SDN aims to solve this problem. The control plane is redefined as three abstractions: a forwarding abstraction, a state distribution abstraction and a global management abstraction. The forwarding abstraction allows a software controller to communicate directly with the data plane, using a common Application Programming Interface (API) to program the network hardware.

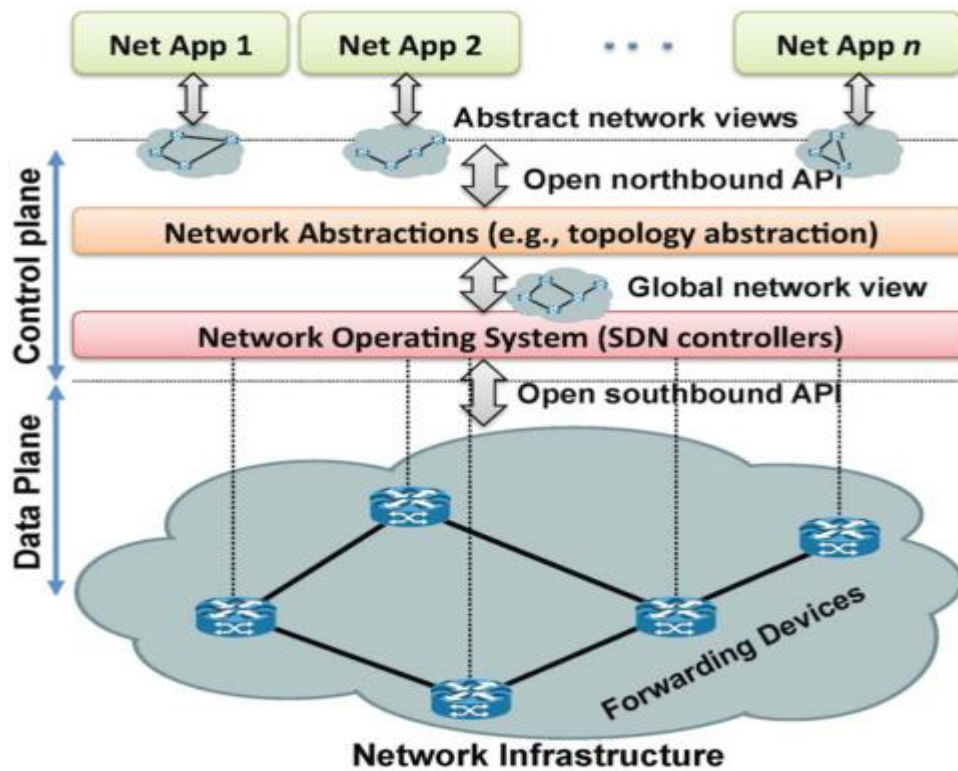


Figure 2.9 : SDN architecture and its fundamental abstractions [20]

In SDNs, the materialization of this abstraction is most commonly done using Open-Flow. The state distribution abstraction shields control programs from the vagaries of distributed state. Thus, management applications no longer have to worry about dissemination and collection of state. The logically centralized controller accomplishes the state distribution abstraction. With the global management abstraction the network has a logical appearance and can be managed as a single logical switch, rather than having to program each individual network device one at a time. The network becomes divided in three tiers, as seen in Figure 2.11. The switches —now “dumb” packet forwarding devices are located in the data plane tier; the controller and the network applications are in the control plane and application tiers, respectively.

2.6.2 OpenFlow

OpenFlow [21] is the most common forwarding abstraction in SDNs. It is the first standard communications interface defined for the exchanging of information between the controller and the packet forwarding devices. While it is not mandatory to use OpenFlow, it is nowadays the most common standard used for the communication between SDN controllers and packet forwarding devices.

OpenFlow started out as a way for researchers to run experimental protocols in networks used every day. As explained before, networks today are static. A lot of the algorithms that are used, as well as functions, are fixed in hardware, in the network device's chips. This results in a high barrier of entry for new ideas, due to the enormous installed base of equipment and protocols. Commercial solutions, meaning proprietary equipment, are closed and inflexible. Research solutions on the other hand, either have insufficient performance or are too expensive. OpenFlow, however, attempts to have switches support a broad range of applications, with high performance and low-cost implementations, all while being consistent with vendor's need for closed platforms.

OpenFlow operates on the switches' flow tables. While each vendor's flow table maybe different, OpenFlow exploits a common set of functions that run in many network devices.

The goal is to provide an open protocol to program the flow table indifferent network devices. This way, network traffic can be partitioned into production traffic and research traffic. Flows can be controlled, the paths that packets follow can be chosen, as well as the processing they receive. OpenFlow can be compared to the instruction setoff a Central Processor Unit (CPU), since it specifies basic primitives that can be used by external software (in SDN, the controller) to program the forwarding plane of the network devices.

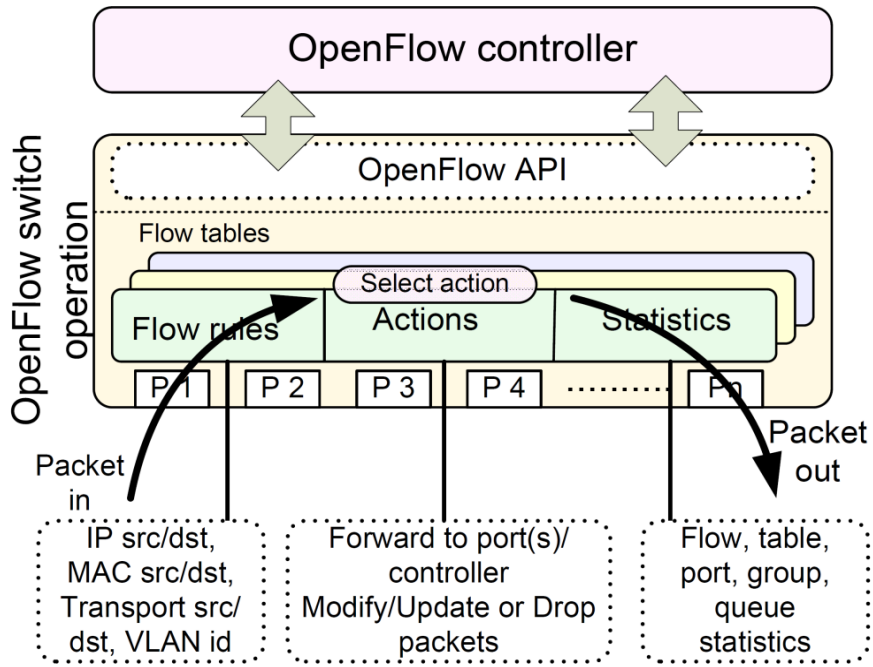


Figure 2.10: OpenFlow protocol architecture

We present the three building blocks of an OpenFlow switch in Figure 2.11: a flow table, with an action associated with each flow entry; a secure channel connecting the switch to a remote controller; and the OpenFlow Protocol, which provides an API for the controller to communicate with the switch. The flow table

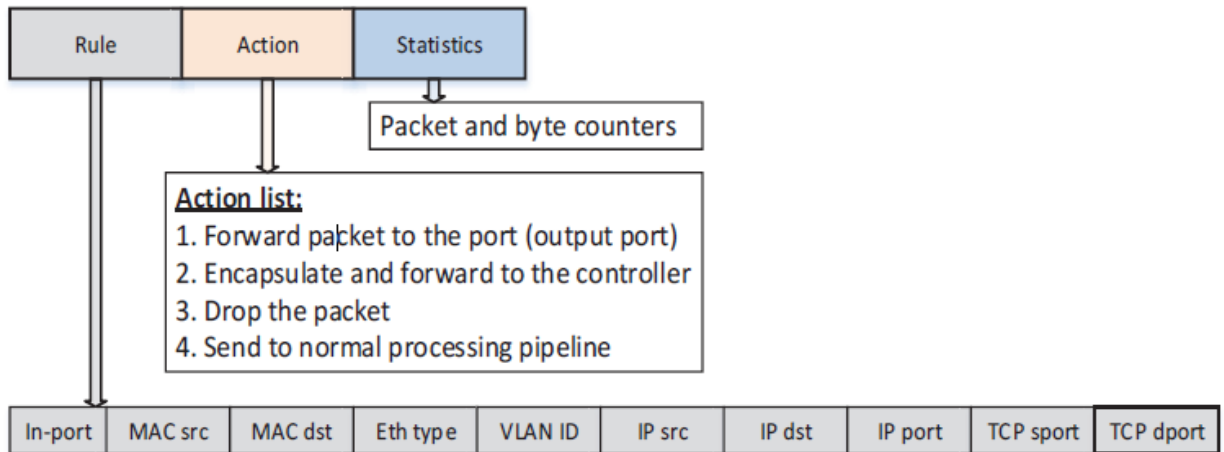


Figure 2.11: Flow table entries for matching fields [22], [23]

is populated with flow entries of the form h header; action i , as decided by the remote controller. Packet headers are compared with the header field of flow entries on the switch. If there is a match, the action associated with the matched entry is performed on the packet. The switch does not have to know what it means in term of distributed state, it only knows what it is supposed to do

2.7 OpenFlow Operation

Whenever any data packet from end host arrives at an OpenFlow-enabled switch, the switch will forward this packet to a control plane for verification. The function of switch is to encapsulate and forwards the first packet arrives from end host to an OpenFlow controller on secure link using OpenFlow Protocols (OFP). This in turn enables the controller to decide whether the flow should be added to flow table of switches or to discard. OpenFlow switch consists of flow table and secure channel to communicate with OpenFlow controller using OpenFlow Protocols (OFP). Each data flow through the network must first get permission from the OpenFlow controller in order to verify whether communication is permissible by network policies or not. If controller allows the flow than it will compute the route and inserts the flow entries in the flowtable of an OpenFlow switch. The flow table entries done by controller have three fields as shown in Figure 2.12. Once an entry is done by controller in a switch, all the succeeding packet arrives from hosts to a switch will match the entry and follow the same path dictated by a controller.

If entry not found in the flow table than either switch will discard the packet or it will send to the controller for further processing based on controller decision. A flow diagram of arrived data packet processing in OpenFlow-enabled switch is explained in [8].

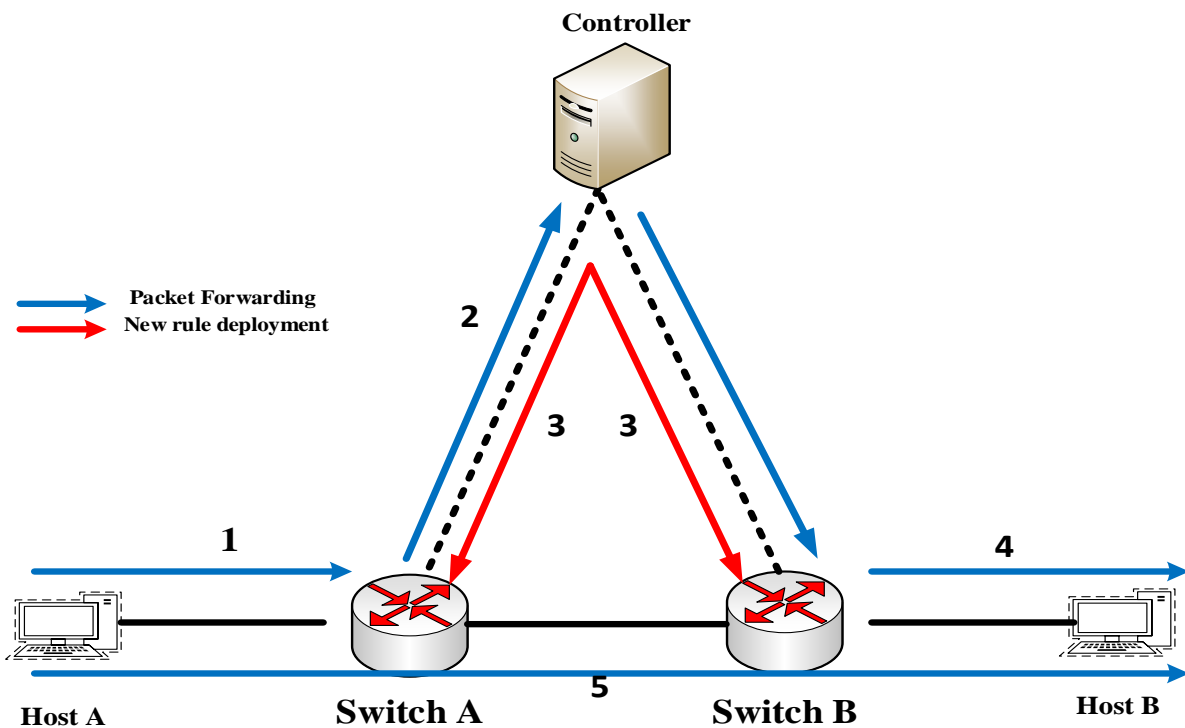


Figure 2.12: Basic OpenFlow Operations

2.8 SDN Controllers

Controllers offer a uniform and centralized programmatic interface to the entire network. Much like operating systems provide controlled access to high-level abstractions for the resources of a computer system, thus facilitating program development, the controller software is a “network operating system”, providing the ability to observe and control a network. The interface offered must be general enough to support a broad spectrum of network management applications.

The controller does not manage the network itself; applications implemented on top of it perform the actual management. The controllers form the control plane of the SDN network and the applications form the management plane. For example, the controller merely adds and removes flow-entries from the switches’ flow tables on behalf of the network management applications.

Some concerns about availability and scalability may arise when devising a network architecture based on a centralized controller. However, enough resilience can be achieved by applying standard replication techniques. In fact, the term logically centralized is an oversimplification. What is important to note is that the distribution model is our choice to make, and not the network's choice. Thus, the controller can be a distributed system built and configured based on the specific requirements of scalability, resiliency, availability, etc. All that is needed to maintain is a unified network view. As with any distributed system, the choice in consistency model offers a tradeoff between performance and overhead, which influences SDN scalability. Furthermore, the OpenFlow protocol allows for a switch to be controlled by more than one controller, for increased performance and resilience.

Controllers present programs with a centralized programming model, allowing applications to be written as if the entire network were present on a single machine. This is made possible by logically centralizing the network state. Controllers also allow programs to be written in terms of high-level abstractions, such as users and host names, instead of low-level configuration parameters, like IP and MAC addresses. Management rules can be enforced independent of the network topology; provided the controller maintains mappings between these abstractions and the low-level configurations.

2.9 SDN for cloud computing

SDN has many distinct features as aforementioned, and this distinct features offer many advantages to designing DDoS defending application on the controller [24].

Separation of the control plane from the data plane: SDN decouples the data plane from the control plane, and thus makes it possible to easily establish large scale attack and defense experiments. The high configurability of SDN offers clear separation among virtual networks, permitting experimentation in a real

environment. Progressive deployment of new ideas can be performed through a seamless transition from an experimental phase to an operational phase.

A centralized controller and view of the network: The controller has network-wide knowledge of the system and global views to build consistent security policies and to monitor or analyze traffic patterns for potential security threats. Centralized control of SDN makes it possible to dynamically quarantine compromised hosts and authenticate legitimate hosts based on the information obtained through requesting end hosts and remote authentication dial in user service (RADIUS) servers for users' authentication information and system scanning during registration.

Programmability of the network by external applications: The programmability of SDN supports a process of harvesting intelligence from existing intrusion detection systems and intrusion prevention systems. More intelligent algorithms can be flexibly used based on different DDoS attacks.

Software-based traffic analysis: Software-based traffic analysis greatly enables innovation, as it can be performed using all kinds of intelligent algorithms, databases, and any other software tools.

Dynamic updating of forwarding rules and flow abstraction: Dynamic updating of forwarding rules assists in the prompt response to DDoS attacks. Based on the traffic analysis, new or updated security policy can be propagated across the network in the form of flow rules to block the attack traffic without delay.

By Applying the above features of the OpenFlow protocol the proposed defense mechanism is deploy a flexible, scalable and very simple cost effective system as an application for the cloud computing environment.

Chapter 3

Related Works

This Chapter reviews some of the works that have been done on securing cloud infrastructure from possible attacks and assuring reliability of cloud services. Even though cloud computing is in its infant stage, some researches have been done in the field of security, particularly on DoS attacks in the cloud. Both DoS and DDoS attacks are serious threats to the Internet. Consequently, it is necessary to have a more intricate mechanism to determine the malicious traffic from the legal ones. In this regard, different detection and defense algorithms have been introduced in the literature for cloud computing and OpenFlow networks.

In this chapter, we will review some of the previous works on DDoS attacks in the cloud using SDN methodologies. We also, try to analyze mitigation as well as detection mechanisms of DDoS attacks in previous works.

3.1 Existing DDoS attack Detection and mitigation works

It is very interesting to know that implementing SDN architecture is proposed by, Seungwon shin [25] as a method for the intrusion detection in cloud environment. In the proposed scheme OpenFlow is integrated into the network structure to control the network flows and diverts the traffic through a path that it is inspected by the preinstalled security devices (e.g. network intrusion detection system (NIDS), firewall, etc.). Employing the SDN infrastructure will simplify the network operator's job in a huge cloud infrastructure. The changes in the flow directions and network policies can easily be performed by running simple scripts on the controller that will install new flow entries on the switches. The controller itself is not involved in the abnormal activity detection but it is responsible for calculating the best and shortest paths that will guide the traffic through the NIDS.

In a similar approach Snort, an Intrusion Detection Systems (IDS), is used to monitor network traffic and measures to identify mischievous activities in the network. Intrusion prevention System (IPS) is an IDS that has the power to spontaneously take action towards the suspect events upon attack detection. Tianyi Xing et al. [26] have implemented an IPS called snortflow by integrating Snort and OpenFlow modules. In this approach the cloud networking environment is dynamically reconfigured utilizing the power of OpenFlow switches in real time to dynamically detect and prevent the attacks.

Kreutz et al. [20] reveal the need of building protected and trustworthy SDNs in the design phase. Bringing replication, diversity and dynamic switch association to SDN control platform design are the main arguments described as mitigation methods for several threat vectors that enable the exploit of SDN vulnerabilities. In the proposed example by implementing a number of replicated controllers the backup controller will take over if one controller malfunctions. The controllers must be designed with interoperation capabilities. Meanwhile the switches must have the ability to dynamically associate to the controllers. To prevent simultaneous attack on all controllers, controllers' diversity must be considered to improve the robustness of the system. FRESCO [27] is an extension of this work that makes it easy to create and deploy SDN security services.

FRESCO is a framework proposed for easier design of secure SDN networks. FRESCO presents an OpenFlow security application development framework that assists in prototyping new compassable security services in OpenFlow networks. FRESCO offers a library of reusable security modules that can detect and mitigate different attacks. The scripting API offered by FRESCO enables the rapid design and development of these modular libraries. Essential security functions (e.g. firewalls, IDS, attack deflector, etc.) can be simulated by assigning values to the interfaces and connecting the necessary modules. The modules can produce flow rules used to enforce the security directives.

B. Wang et al. (2015) [6] proposes a DDoS attack mitigation architecture that integrates a highly programmable network monitoring to enable attack detection and an adjustable control structure to allow fast and specific attack reaction. To cope with the new architecture, paper proposes a graphic model based attack detection system that can deal with the dataset shift problem. The simulation results show that the architecture can effectively and efficiently address the security challenges brought by the new network prototype.

Chu, Yu Hunag et.al (2010) [28] proposes an OpenFlow DDoS Defender that monitors flows on an open flow switch. If the number of packets received in 5 seconds exceeds 3000 then the number of packets will be studied in per second duration. If the number packets per second exceed 800 for 5 continuous times then an attack is detected and the DDoS defender will start dropping the incoming packets until the flow entry times out.

S.A. Mehdi, (2011) [29] argue that network security tasks should be delegated to the home and office networks instead of ISPs. In the presented work security policy implementation is delegated to the downstream networks. Four prominent traffic anomaly detection algorithms, threshold random walk with credit based rate limiting, rate-limiting, maximum entropy detector and Network Traffic Anomaly Detector (NETAD) are implemented in NOX controller and it is observed that the anomaly detection can function well at line rates without any performance degradation in the home network traffic. It is suggested that this approach can monitor the network activities without the need of the excessive sampling.

K. Giotis et al. (2013) [30] proposed a combined mechanism comprised of data gathering with sampling, implemented with the use of the sFlow protocol and anomaly detection algorithm implemented by entropy-based algorithm. Their mechanism eliminates the flow statistics collection through forwarding tables' lookup and reduces the required communication between switches and OF controllers, thus easing the control plane overloading.

R. Braga (2010) [31] propose a DDoS detection method built into the NOX controller based on Self-Organizing Maps (SOM). SOM is an unsupervised artificial neural network trained with the features of the network flow that is periodically collected

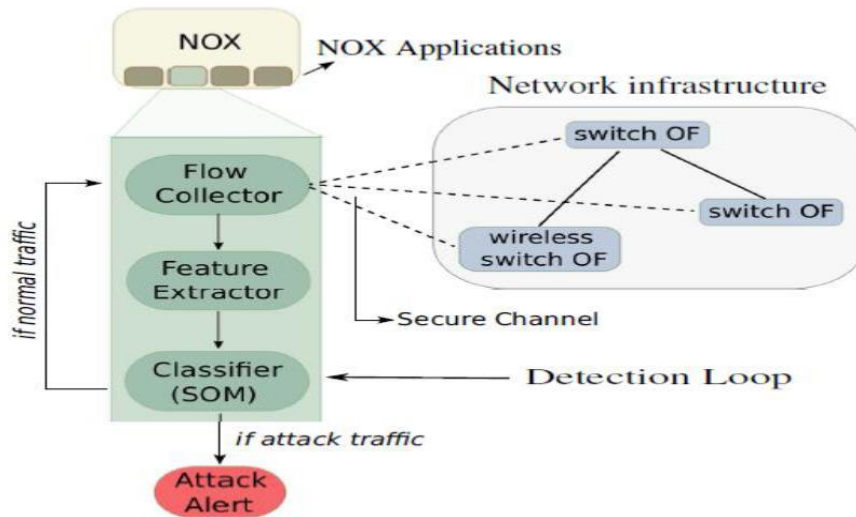


Figure 3.1: Detection Loop Operation [31]

from the switches. The traffic is classified as either normal or abnormal based on the SOM pattern. This detection method as shown in figure 3.1, runs in three modules running periodically within a loop in the NOX controller:

- The flow collector module queries the switches periodically for their flow tables.
- The feature extractor module extracts the main features that are studied for DDoS attack detection and gathers them in 6-tuples. The main elements that are calculated based on the collected features and will be studied in the next module for the traffic classification include average of packets per flow, average of bytes per flow, average of duration per flow, percentage of pair flows, growth of single-flows and growth of different ports.
- The classifier module must analyze and decide whether the given 6-tuple corresponds to a DDoS attack.

Querying the switches periodically especially in the large scale cloud architecture with large number of switches will put an extreme overhead on the system and will eventually affect the performance of the controller. Processing that high volume of flows in the flow tables is another issue that must also be well-thought-out.

Lee et al.(2013) [32] propose a collaborative defense model, called coDef, against large scale link-flooding attacks. CoDef consists of two complementary mechanisms: collaborative routing and collaborative rate control. They introduce a specialized server, called the route controller, into each participating AS, which has complete knowledge of the network topology by participating in the intra-domain routing protocol (i.e. IGP). The route controller is implemented in an SDN architecture. In collaborative routing, a congested router sends a congestion notification message to its route controller. Then, a route control message is exchanged between route controllers placed in individual ASs to instruct the source ASs to reroute their traffic, which relieves congestion at that router. The collaborative rate control mechanism helps to distinguish between bot-contaminated and uncontaminated ASs. In this case, a router that is subject to a flooding attack sends rate-control requests to all the ASs to establish the service priorities of their out-going flows (i.e. high-priority flows, low-priority flows, and flows to be filtered).

In Xuan, Bettati and Zhao (2001) a collaborative DDoS defense system is proposed in which routers act as gateways, detecting DDoS attacks locally and identifying and dropping packets from misbehaving flows. Gateways are installed and communicate only within the source and the victim domains, thus providing cooperative defence of a limited scope.

Sumanth M. Sathyanarayana (2011), demonstrates that SDN can be leveraged to mitigate DDoS attacks efficiently using the pushback technique with their Frenetic mechanism. Two daemons are implemented in the Open-Flow controller: a pushback daemon and a rate-limiter daemon. The use of the

OpenFlow controller prevents pushback messages from being sent by the routers, and all communications are handled by the central controller.

Another Collaborative DDoS defending system is pushback. Pushback has two components, Pushback [33] and Aggregate-Based Congestion Control (ACC). Pushback is the routers in the system assume that the congestion of local packet queue is the sign of DDoS attack and take action to rate limit the identified aggregates which are responsible for queue congestion according to local policy. If the congested router cannot control the aggregate itself, it issues a rate limit request to its immediate upstream neighbors who carry the aggregates traffic to apply rate limiting to specified excessive flows. These requests will be propagated upstream as far as the identified aggregates have been effectively controlled. This approach requests all the routers on the path of aggregate traffic be augmented with the pushback capability.

Local ACC is triggered when the traffic drop rate at the queue exceeds from the predefined value of 10%. ACC attempts to identify the aggregate traffic responsible for this series congestion. it then limits the rate of the traffic probabilistically dropping from this aggregate until the drop rate reduces below 10%. The amount of packet dropping depends on how much traffic exceeded the target bandwidth. ACC drops the excess traffic from its identification aggregates until the drop rate at the output queue is below the predefined level. ACC is not able to distinguish between the legitimate and attack traffic within an aggregate. it drops legitimate traffic that belongs to the aggregate as well.

Our work is related to this approach. The pushback messages are the (defense) rules that would be added to the switches by the controller. The "Preferential Dropping phase of the Pushback" mentioned by [33]. (Ioannidis, 2001) is implemented in this work, in which anomalous nodes are removed first from the flow table of the switch next to the victim, and then successively on all the switches, one after the other, from the victim node to the malicious node. On the other hand, the approach presented here does all the calculations at the

controller and makes the decisions quickly. The mitigation strategy presented in this thesis takes advantage of the fact that the SDN controller has information on the complete network and is able to gather statistics of all the ports in it; with all this information, a computation can be done using the complete knowledge of the network and a quick decision can be made so that the attack that is being performed is stopped.

Chapter 4

Proposed Design

As we discussed in the literature reviewed, DDoS attack is the most serious attack in the cloud environment which consumes the resources of the service and denying the legitimate user not able to access the service. Implementing DDoS attacks on the cloud systems and denying services is easy when it compared to the other networks. Most existing congestion control mechanisms, are designed to detect and drop packets at or near the destination network where the packets have already traversed the network and consumed considerable bandwidth. The aggregate traffic at the destination node may consists of too much flows. It is more difficult to distinguish between legitimate and attacking flow there. Therefore, some forms of congestion control mechanism that can identify and handle DDoS attack at their source and collaboration between the different defense systems for the distributed cloud network systems is required. In this regard we design a simple, scalable congestion control mechanism which can detect an attack at the source node and a report an existence of an attack on the system and performing the attack mitigation process cooperatively. Our defense solution is based on Software defined networking technology.

SDN is a network architecture that decouples the control plane and the data plane of network switches and moves the control plane to a centralized application called network controller. The network controller is in charge of the entire network through a vendor-independent interface such as OpenFlow, which defines the low-level packet forwarding behaviors in the data plane. Using this methodology researchers can program the network from a higher level without concerning the lower level detail of packet processing and forwarding in physical devices. As a result, rich functionality in traffic management, routing, firewall configuration, load balancing etc., which may pertain to specific flows they control, may be easily developed.

Based on this, multiple features and functionalities, of SDN we try to design an efficient and scalable mechanism for performing anomaly detection and mitigation in SDN architectures. In order to detect DDoS flooding attacks, the designed framework requires collaboration between client and server side cloud computing network architecture. For this reason, the proposed architectural components should have to suit like the real cloud computing environment. And, each of the component on the framework should have to coordinate in the traffic flow collection, analysis and in the prevention process.

In this chapter, we present our pushback approach which detects and mitigates DDoS attacks at the source node and sends a pushback messages to remote node, which the controller in our case, in different network domain. In the process of attack detection and mitigation, pushback has two procedures. The first one is identifying the flows with high sending rate and secondly, controlling those flow's sending rate in the case of DDoS attacks. So, the goal of our work is to detect and drop most of the malicious packets at the source close to the attacking sources instead of at the victim network, and sending a pushback message to the controller for an attack that is generated from the remote network domain. This ensures that the victim host is not seriously congested at the time of an attack and allows a minimal level of collateral damage to legitimate traffic. Normal client requests are therefore able to reach their destination server even though it is under attack.

In section 4.1 we described an overview of the proposed pushback scheme. Section 4.2 also present the proposed components for DDoS attack that is targeted to infrastructure of the cloud domains. In this section each of our modules are described and explained in detail, and finally we shows the flow of information and the procedure how the technique is working.

4.1 Pushback Overview

Pushback is a network based solution to prevent DDoS attacks. It contains a local aggregates-based congestion control (ACC) mechanism for detecting and

controlling an aggregate a single router, and cooperative pushback mechanism in which a router can ask adjacent routers to control an aggregate upstream. Let us consider figure 4.1 to illustrate the operation of the pushback technique under a DDoS attack.

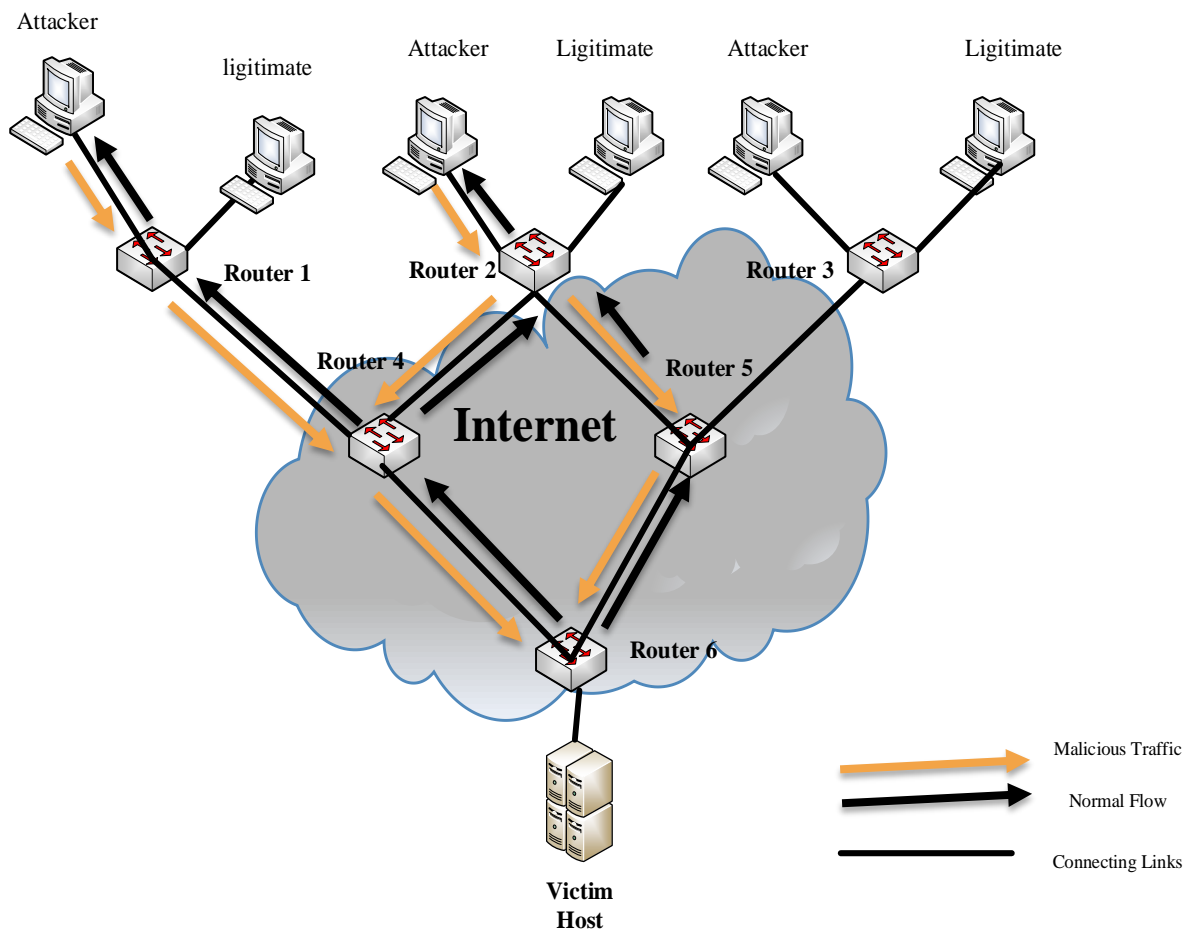


Figure 4.1: A DDoS attack in progress [33]

The victim host is the victim of an ongoing DDoS attack. The thick lines shows, the links for attack traffic flow. In contrast, the thin lines mean the links are in normal status. Especially, the last link between router R6 and the victim V is the bottleneck link which is congested by attack traffic. In this situation, the local aggregate congestion control (ACC) at router R6 detects incoming aggregated traffic. R6 therefore immediately starts to drop packets belonging to the

aggregated traffic. Because there are more than one aggregated traffic flows from different links, the pushback technique punishes them equally. Moreover, router R6 will attempt to cooperate with its upstream routers (R4 and R5) by sending pushback messages to them if the ongoing congestion is still severe. In fact, the operation can be recursive. This means that router R4 or R5 will send pushback messages to their upstream routers. The recursive operation will not end until congestion of the whole network is relieved.

The design decision in pushback is to separate the rate-limiting and packet dropping functionality. When congestion of links are detected, the router checks for anomalous aggregates. If anomaly is detected based on congestion signature (a congestion signature is something like a victim node's IP address or MAC address), then rate-limiting is done on such aggregates. The packets from those aggregates which are not rate limited are sent to the output queue of the router as usual. The packets which are dropped both by the rate limiter and the output queue are sent to the pushback module for an analysis on the attack estimated to be happening. The pushback daemon checks the dropped packets and update the congestion signature and adjust the rate limit based on how much congestion is still detected in the links. For e.g. even after setting a rate limit, if congestion is detected in the links, then either the rate limit has to be increased or rate limits have to be applied on the aggregates which might be anomalous. The router then send pushback messages to the upstream router in order to let them know that due to a particular signature, congestion is happening at the links and hence they in turn have to adapt accordingly by limiting their own rates and dropping certain packets.

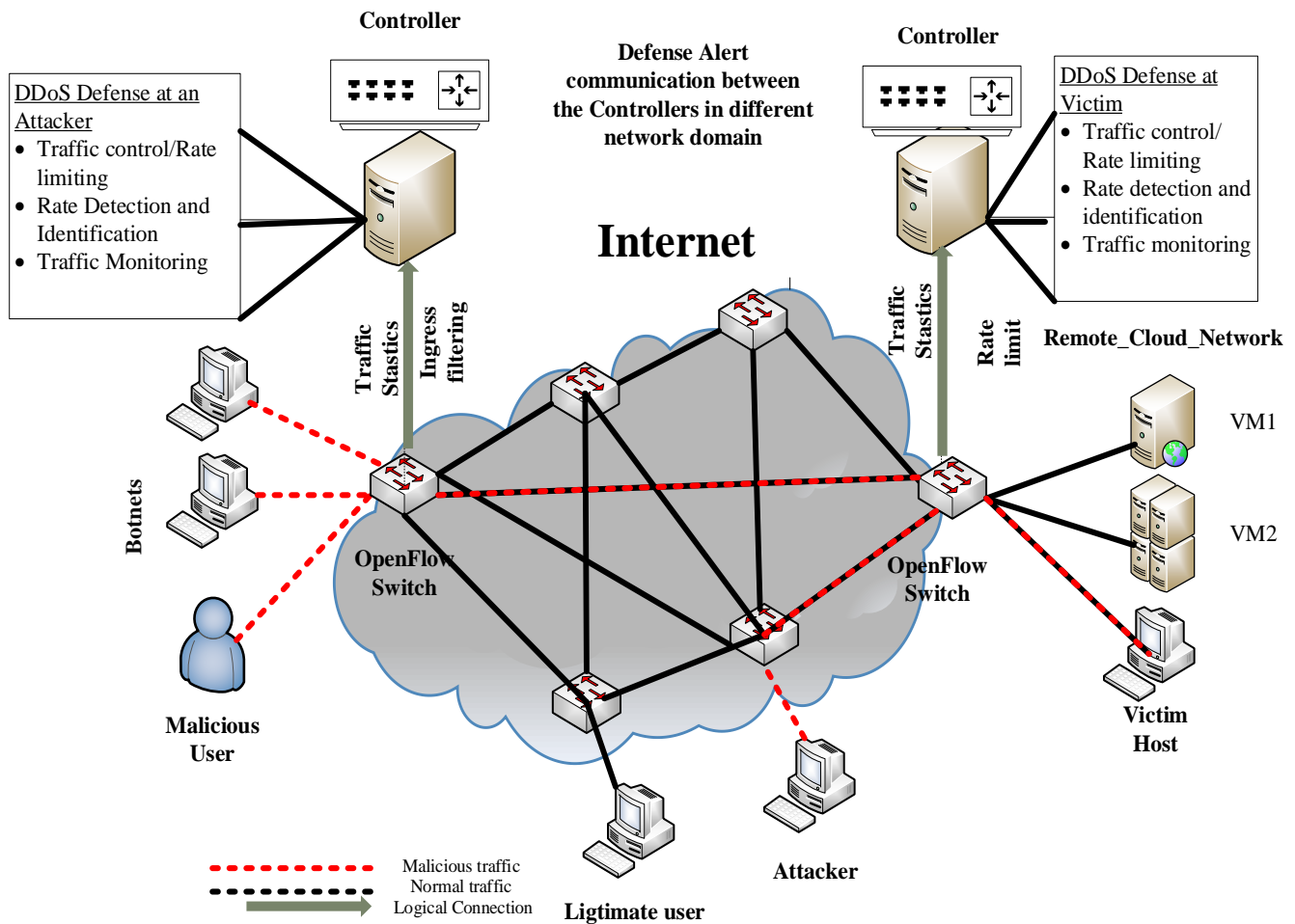


Figure 4.2: DDoS flooding attack and Collaborative defense mechanism

In SDN, controllers can be implemented as centrally to manage the entire network and in distributed manner. In this architecture distributed controller are implanted on the OpenFlow switches. The Ryu controller is running on top of the OpenFlow switches as the OpenFlow controller to function as both the pushback daemon and the rate limiter. As a traditional network pushback technique pushback messages are not sent by the router as the switches do not communicate and all the communication is done by the controller in case of OpenFlow networks. Thus, the pushback messages in this case are the rules

that would be installed by the controller on the switches, i.e. the communication between the controller and the switches.

On this implementation, the controller will program the switches to behave as learning switches, this means that for each MAC address of the hosts, the switches will store the port on which they have to forward the packet to reach their destination. If a packet arrives and the switch does not have the MAC address stored, it will flood all the ports and store the port that replies to the packet.

This control application will also have a graph that is constructed by obtaining the network information. This graph is a representation of the network and gathers information of nodes and links on it. Of the nodes, we know their type (either switch or host) and their identifier on the network. Each node has a list of links that contains information of the devices connected to it, the ports each link is using and statistics of the data that changes according to the network behavior; an example of this are the bytes that it has received or transmitted and the latest bytes per second measurement.

But here, our defense mechanism on Figure 4.2, uses different flow of traffic information of its attack detection, mitigation when the communication is takes place between a legitimate or attack node and victim node of the cloud network. In order to determine an existence of DDoS attack in the network, a simple application will periodically query the statistic of the network nodes traffic sending rate to the specific node is analyzed from this flow.

The statistics of interest to our implementation is the bandwidth that each host receives, with this information we can calculate the rate at which the host is inserting packets to the network, measured by Megabytes per second. Each of the host's traffic sending rate is pre-defined. For this implementation, the threshold that separates normal traffic from unusual traffic is set to a static value, equal for all the hosts in the network. Any node is allowed to receive the normal amount of data. When a node starts receiving larger data than from the

predefined threshold it is assumed that there has been a congestion which might be due to a DDoS attack. Once congestion seen in the network, the controller identifying the switch and the victim node as well as an attacking node. Then after the application on the controller install a rule on the switches so that they drop the packets coming from that port or a port that a malicious node is connected to.

By doing so, it is able to defend against a bandwidth attack before it can congest the network. For the performance of the above stated functionality, we deigned a flow monitoring and detection, traceback or an identification of an attack and mitigation module on the Ryu controller as a simple application. The property of this defending mechanism is performed locally and globally with the coordination between the distributed controllers in the private and public network domains.

4.2 Cooperative defense components

This section presents our scheme to identify attack nodes, and blocking the DDoS attacks at the source of their OpenFlow switches to minimize the impact both to the victim and to the cloud network during the attack time. The proposed DDoS mitigation mechanism consists of the three components. The first one is flow monitoring component. This component monitoring the rate of an incoming traffic and measure the flow thorough the connected ports. The measurement and the decision of an attack existence in the network is based on the calculated value of the bandwidth. Then the module checks the calculated byte transferred is above the predefined threshold it identify the situation to the next traceback or identification component. The module identify the victim node and an attacking node which carry the attack traffic, and activate packet filtering at selected points.

4.2.1. Traffic (flow) monitoring.

No matter what kind of anomaly detection mechanism, the flow collection is an important part to get the statistic data. To get this traffic flow information, from active OpenFlow switches we employed a native OpenFlow messages. More specifically, the “OFPT_STATS_REQUEST” and “OFPT_STATS_REPLY” OpenFlow messages have been implemented. The controller application periodically requests from all the switches to report statistics about the packet matches and the accesses to their flow tables occurred during the specified time window. Flow refers to a set of sequential packets which have the same properties traveling through the same network during a period of time[34]. Table 4.1: shows the flow entry of OpenFlow switch records the information of a flow naturally [30].

Table 4.1 An Example of flow table entries

LAYER 1	LAYER 2					LAYER 3				LAYER 4	
INPORT	ETHER			VLAN		IP				PORT	
	src	dst	type	id	PCP	src	dst	Proto	TOS	src	dst
X	A	D	G	*	*	B	E	H	*	C	F
Y	D	A	G	*	*	E	B	H	*	F	C

When the switch, namely the *OFA_switch* module, receives this message parses the flow table and collects all the current flow entries and their counters. All the information is included in the *OFPT_STATS_REPLY* OpenFlow message, which is sent back to the controller. When the controller receives replies from all the switches presented in the network, update the flow table when a forwarding lookup process matches a specific flow entry of the flow table.

Here we modified the functionality of an OpenFlow controller role for our requirement. Our flow monitoring component function as or performing: monitoring the flow, detect an attack, and identify an attacker. Once traffic received by the controller from the OpenFlow switches, the controller starts the monitoring process of incoming traffic. Here the controller keeps measuring each

incoming flows' byte counts for each FLOW and incoming flow PORT to calculate deltas for bandwidth usage calculation. Therefore, a record of the flow table's state for the previous time windows should be kept and compared to the current

Pseudo code 1: Cooperative Flow Monitor

Step 1: initialize the local threshold parameters, the collaborative detection threshold and the interval ΔT ;

Step 2: for each new incoming flow at any router or switch

Step 3: check the Source IP/MAC of the new flow

Step 4: Install OpenFlow rules for forwarding the new IP/MAC in switch flow

Step 5: continue the monitoring to get the statistics of the flow

Step 6: Check the received byte for this flow and calculate bandwidth

Step 7: Save the bandwidth calculated for this flow as f_1 , f_2 , f_3 ... f_n

Step 8: If the calculated bandwidth for the flow is above from the pre-defined threshold

Step 9: Call the identification module to identify the victim whether it is from the local or remote domain.

Step 10: Calculate no sustained attack counts

Step 11: If rate for all flows on the links is below safe level, increase the sustained no attack count for this link

Step 12: Call the mitigation function

Step 13: Go to step 2

Figure 4.3 Cooperative DDoS flow monitoring algorithm

number of bytes for each flow entry, in order to find the corresponding number of bytes for the current time window. The controller maintains different stats for

each high rate flow and updates them according to the average bandwidth at the victim. The rate of a flow of bytes is computed with a moving time window of 2 seconds, which means we are only concerned with the average arrival rate in the last 2 seconds. When the controller detects those flows whose arrival rate exceeds a threshold, it likely DDoS attack, tries to find out the OpenFlow switch where a high rate of traffic is generated from.

In the normal state, hosts in the network send the traffic with normal rate. And, the recipients are process and response the request normally, but in the case of an attack the rate of the incoming traffic to the receiving host is maximum. The recipient cannot process the request and denying the legitimate users request. The monitoring of the rate of the flow and calculate its average in order to determine the normal and attack flow. Once the attacking and victim node is identified, a new iteration of the method update is executed and the controller pass this information to the attack mitigation component for its measure. Figure 4.3 show, the proposed cooperative flow monitoring algorithm.

4.2.2 Attack Identification

Once an attack has been detected, the next step is identification of an attacking node and applying the mitigation technique to provide a service for those of the legitimate clients. Finding a potential group of attackers at the source network domain controller which has forwarded that group of attack traffic. For an identification of an attacking node we used some of the characteristics of attack traffics, such as source IP address or MAC address of a host to detect and identify attack traffic. The process of identifying a malicious host is based on the report of the flow monitoring algorithm. The monitoring algorithm check which host is sending malicious traffic that is above the pre-defined threshold. The malicious host identification process use, some characteristics of the flow for its attack identification. Which is the MAC address of the host. Once the MAC address of the tracing back the source of the router where the attacking node is connected to it. The control know all of the switches connected in the network through its

logical connection mechanism, and it sends flow modification message to the switch. The switch applies about the flow information that is decided by the controller. To forward a traffic or drop a traffic through some of its network interfaces. Identification and response is performed in both of the network domains. At the victim side network and at the source of the malicious node network. This helps us to identify attacks quickly and stop them as close to the source as possible. SDN is providing a better alternative to identify and block attacking traffic and a victim node on at the source. Because, the OpenFlow controller has an ability to globally view the entire network.

4.2.3 Attack mitigation on combination of filtering and rate-limiting

Once an attacking node and a victim host is identified, the next step is to rate-limit the traffic with the identified attack signature. As soon as the attack has been identified, the mitigation module is called. The purpose of the module is to rate-limit the traffic with the identified attack signature and to protection in the network and especially to the victim by dropping all the packets coming from the adversary. The module is responsible to prepare the match fields of the new flow entry and emit the “*Mitigation*” signal to the main controller application to notify about the new mitigation rule which will be installed.

In the scenario of a DDoS attack, the purpose of a rate limit is not only to lower the aggregated traffic under the bottleneck link's bandwidth, but also to decrease the percentage that attack traffic represents of the whole of aggregated traffic. To control attack traffic, we set up the rate limit on the controller which are close to the attackers OpenFlow switch interface. For rate-limiting we used the Token Bucket Filtering (TBF) queuing discipline, as a very light-weight method of allowing packets to pass only up to a specified maximum rate. To limit the rate of incoming traffic, we used a TBF rate-limiting mechanism. It triggered to rate-limit not only the total traffic that arrives at the victim, but also the traffic associated with each priority band separately.

Limiting the rate of the traffic incoming to the victim node is doesn't stop the problem in the network. But it limit the attack rate in the network. This need

Pseudo code 2: Mitigation of DDoS attacks

Step 1: *If an attacking node in the local domain*

Step 2: *Identify attackers for the victims and apply an ingress policy to each attacker*

Step 3: *If an attacking node is in different network domain*

Step 4: *Send current rate limit information to remote controller*

Step 5: *If there are no victims, for a sustained duration, try remove ingress policies*

Step 6: *Identify the set of victims attacked by hosts located in the other domain and directly apply policies to the attackers in the local domain*

Step 7: *Increase the count for confidence in a suspected attack by the identified attacker set if applicable*

Step 8: *If we have exceeded the confidence count for the local attacker set, apply ingress policies to all attackers or to a given attacker's switch/port*

Step 9: *If the confidence count for an attack exceeds the provisioned limit check and if the bandwidth consumption on one of the rate-limited links dropped below a "safe" level and remove ingress policy*

Step 10: *If rate for all flows on the links for this port have been below a safe level for the last couple of statistic readings, remove the ingress policy*

Step 11: *Stop forwarding the attack traffic to the victim network*

Figure 4.4 Cooperative DDoS attack defense algorithm

another better solution compared to this. Filtering of the traffic is impact or punish the attacking as well as the legitimate user traffics. Rather than pushing

and dropping all the traffic it is better to limiting the attack traffic. Here we applied rate limiting the impact. From then on, the protection mechanism is set into motion along the informed path. After detecting that the traffic is stable enough at the victim end, the last step of the recovery phase will remove rate limit at all source-end OpenFlow switches. This lets OpenFlow switches serve legitimate traffic fully. The pseudo code of the algorithm is show in Figure 4.4

4.3 Work Flow

On the process of the DDoS attack detection, the flow of information from source to destination is analyzed using the components on the design. Each of the components working in hierarchical manner. One's input is important to the other components for generating a result. The inputs from the flow collector and attack flow identification can be used by the DDos mitigation system. Once flow is crated to start communication between the server and the client, the flow is send to the switch using the OpenFlow message. The flow monitoring algorithm starts monitoring the flow of information between the receiver and the sending node per-defined with the predefined time duration. The statistics information that it monitored from all of the switches that are logically connected to the controller, enable us to me to determine if there is any DDoS happening in the network. The monitored traffic is calculated every two minute intervals. For our implementation purpose we have made use of a threshold amount of data = 5Mb, which every node can handle. When a destination node starts receiving of size of traffic that is greater than this predefined threshold amount, it is assumed that a DDoS be happening or it might be due to a temporary spike in the traffic. Once, it is detected that the threshold value of the data limit is exceeded on a particular node, we list out all the source nodes which are sending data to sending node as the first anomalous node. The controller starts the tracing procedure for this malicious node and switch where the host connected port. The controller have an ability to globally view all the switches connected to it and all the hosts' interfaces. Using this behavior the controller can simply identify the attacking node in all of the network domains. Then finally the controller installs flow

modification message to the switch to stop or rate limit the rate of an incoming flow to the victim node. The rate is limited for the host which is sending to much traffic than from the pre-defined rate by the controller.

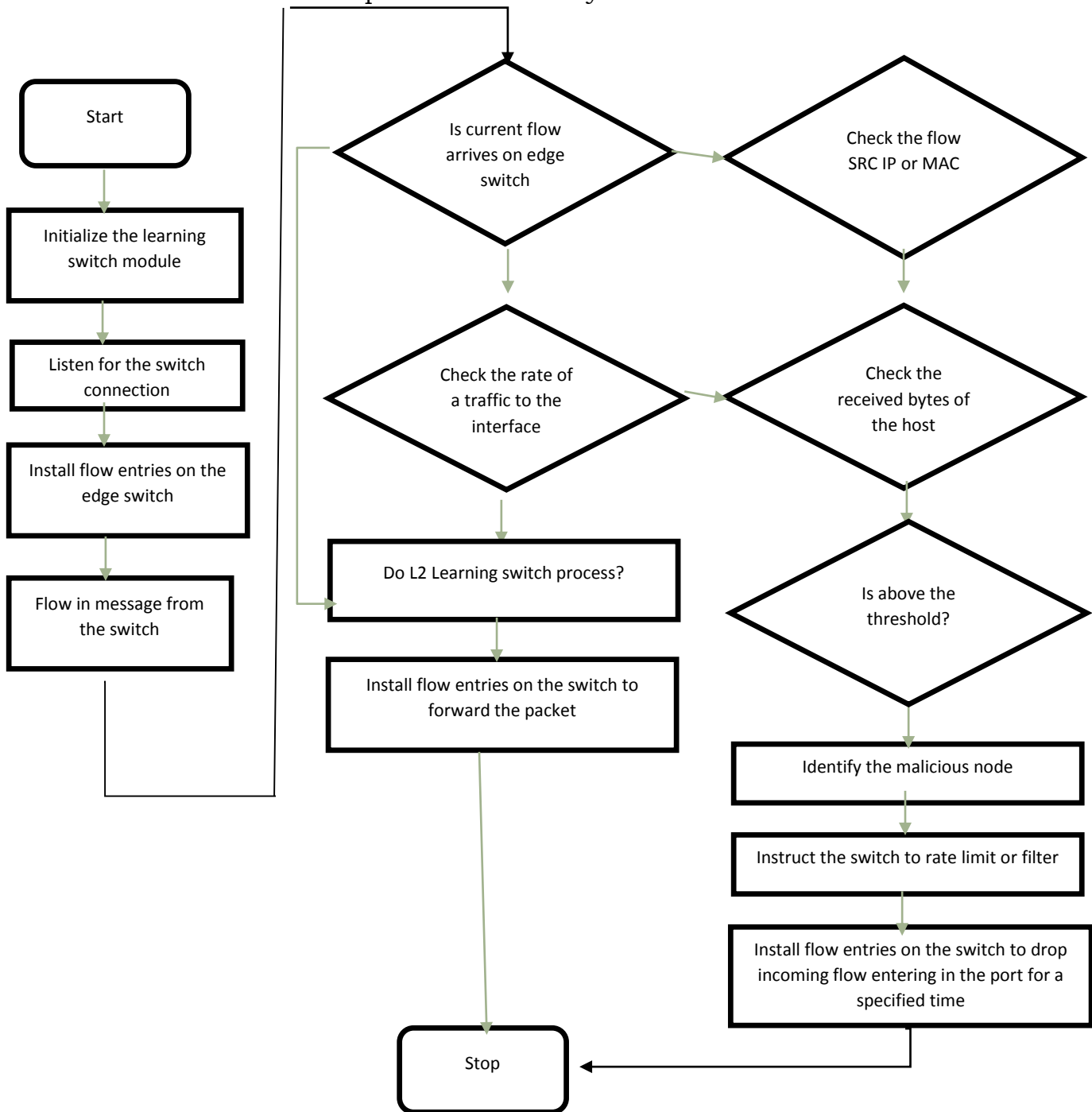


Figure 4.5: Cooperative DDoS attack defense Flow Chart

Once the rate of the traffic is identified by the controller in the local or remote network domain, the controller sends a pushback message for an attack traffic that is generated from a malicious node where it is connected. Once the pushback message received by the controller for remote network request, the controller sends a pushback reply message to rate limit the traffic to the victim node OpenFlow switch port interface. Figure 4.5 show the flow of how the controllers are cooperating in the process DDoS attack defending.

Chapter 5

Experimentation and Results

In this section, we evaluate the pushback approach as an attack detection, identification and mitigation mechanism. We mainly focus on the performance of our evaluation metrics with low to high traffic volumes. Moreover, we investigate the benefits of exploiting pushback in order to identify and mitigate any detected malicious traffic, using the capabilities of the OF Controller (Ryu).

Ryu is a modular event-based OF controller, which is exploited as a high-level programmatic interface upon network events. Through the API of the Ryu Controller we implemented all three components as a single Ryu application, responsible for flow monitoring, periodically rate of flow calculations for an existence of attack on the system and attack mitigation.

To accomplish the above tasks, we implemented the following Ryu Controller: for decision making, Mininet for the network simulation, OVS (OpenFlow switch) for the connection of the data plane and the control plane. Section 5.1 describes the simulation environment. Moreover, tools that are used in the experiment, and their usage. Section 5.2 also describes the simulation topology. The last section describes, the different scenarios with their respective simulation experiment and evaluation result.

5.1 Environmental Setup

Experiments are conducted on Lenovo Laptop-E51-80 computer which runs Ubuntu14.04 platform with the Long term support. The laptop has 8GB RAM and Intel® Core™ i7-6500U CPU@2.50GHz 2.50GHz processor. For the purpose our experiment we installed Ryu controller software on our Ubuntu 16.04 LTS laptop computer. Ryu is a component-based software defined networking framework. Ryu provides software components with well-defined API that makes

it easy for developers to create new network management and control applications. Ryu supports various protocols for managing network devices, such as OpenFlow, Netconf, OF-config, etc. Regarding OpenFlow, Ryu supports fully 1.0, 1.2, 1.3, 1.4 and Nicira Extensions.

Moreover, Ryu is written fully in Python, and therefore it is easy to develop for. It comes with various applications already written to deploy SDN network, like Spanning Tree manager, basic switch application for creating forwarding rules, firewall and routing application. The Ryu architecture is shown in Figure 5.1. It provides a rich set of APIs allowing developers to create specific application for managing different aspects of a network according to the operator requirement. It can support legacy networks along with SDN implementations, and recently a huge collection of documentation and references has been added that can be used to create specific application.

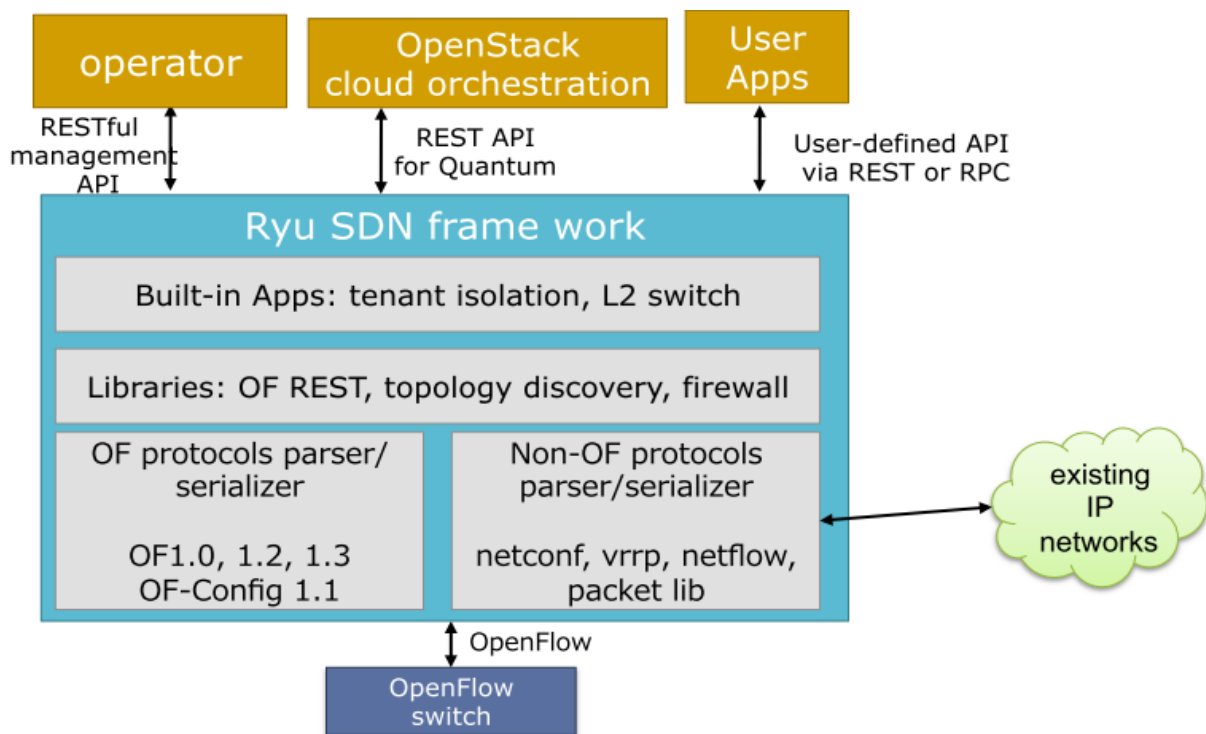


Figure 5.1: OpenFlow Ryu controller

Furthermore, we installed mininet for our network simulation. Mininet is a tool to simulate the Software Defined Networks that allows quick prototyping of a large virtual infrastructure network with the use of only one computer. It enables us to create virtual prototypes of scalable networks based on software such as OpenFlow, using primitive Virtualization Operating System. With these primitives, it allows us to create, interact and customize prototypes for Software Defined Networks in a very quick way.

In order to create a network, Mininet emulates links, hosts, switches, and controllers. Mininet uses the lightweight virtualization mechanisms built into the Linux OS, processes running in network namespaces and virtual Ethernet pairs, and creates a virtual network by placing host processes in network namespaces. For network connection of host processes, Mininet connects them with virtual Ethernet (veth) pairs.

After all system setup we started the experimentation. For the experiment we created a hierarchical network topology which is shown on Figure 5.2. The mininet python code is also modified for our simulation. With a simple mininet scrip on can build a large network consists of different network infrastructure

5.2 Simulation topology

For our experiment, we use six switches. From this six switches, three switches are connected to each other to form a hierarchal network which consists of an edge switch and an access switch that it directly connected to the end-devices. Each of this networks is managed by their own controller to act as a private network. Moreover, the controllers on OpenFlow switches are securely or logical connected to the OpenFlow switches in the domain. Figure 5.2: shows, the Network topology used in the experiment. The role of this devices whether they are in the private or in public networks is the same as one of the switch at the top of a topology used as an upper layer network device, function as routing traffic for a single network domain to the other network domains. Furthermore, which is connected to the same devices. Two of this act as an access switch and

they connected to four virtual machines that are members of this network are become the agents for DDoS attack.

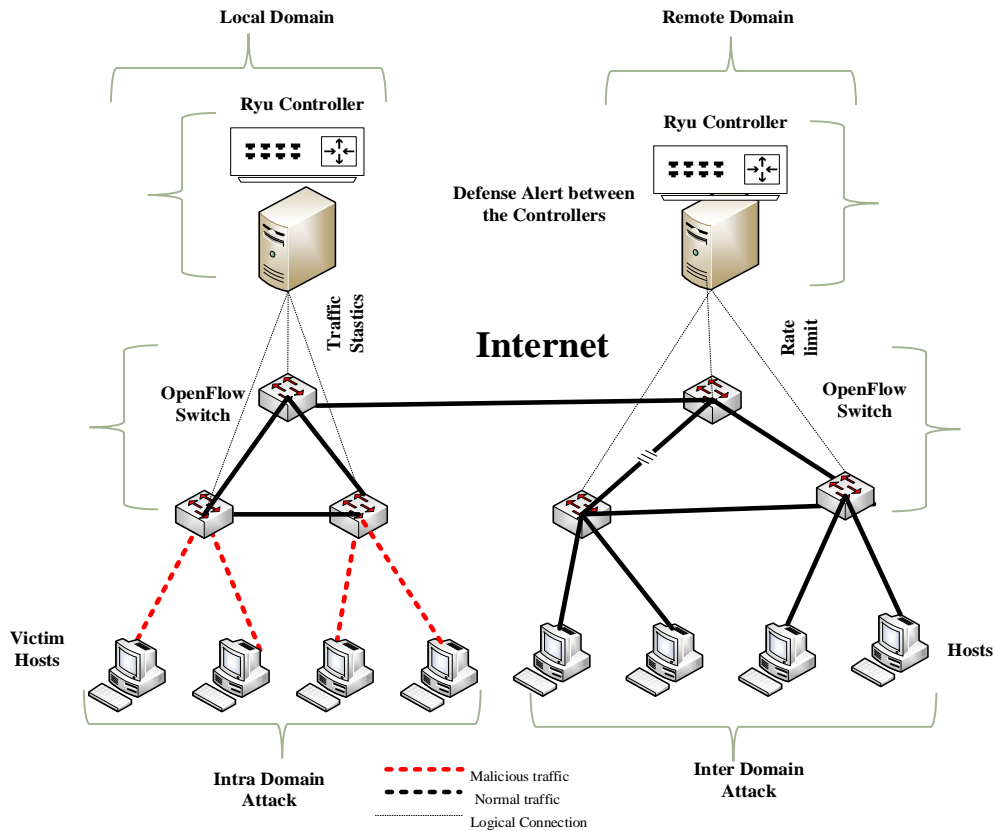


Figure 5.2 Simulation Network setup

One of the agent host then is used to generate malicious packets to attack flow table resource on each switch and finally, attacks flow table in the controller. The final goal is to flood the victim node, rendering them unable to provide normal service to legitimate users. In time of an attack, normal clients may also send legitimate request packets to the victim node. However, they may not pass through and suffer drops due to the series congestion in the victim network.

On the controller two applications are running on top of Ryu framework, one for configuring the switches to allow connectivity between themselves and host machines and another application developed by us get flow information, analyze and finally mitigate for an attacks in the system all as a single application.

5.3 Mininet Simulation parameter

To simulate our proposed system we designed two network domains on the mininet tool, each of which consists a legitimate machine and an attacking machine. As figure 5.3 depicts, there are 6 levels of OpenFlow switches. The links between the two OpenFlow switches S1 and S4 is configured with 10Mbps. The links between the 2 levels of OpenFlow switches is also, 5 Mbps bandwidth for each of the network domains. In addition to this the each of the virtual machines are connected with 2 Mbps of bandwidth with the OpenFlow switches interfaces. The victim (AAh1) is connected with the OpenFlow switch by a 2 Mbps bottleneck link. Rate limit decision is made by a centralized control point, which is a controller deployed on the network domains. In order to adapt to the dynamic changes, rate limit decision is refreshed every 2 seconds.

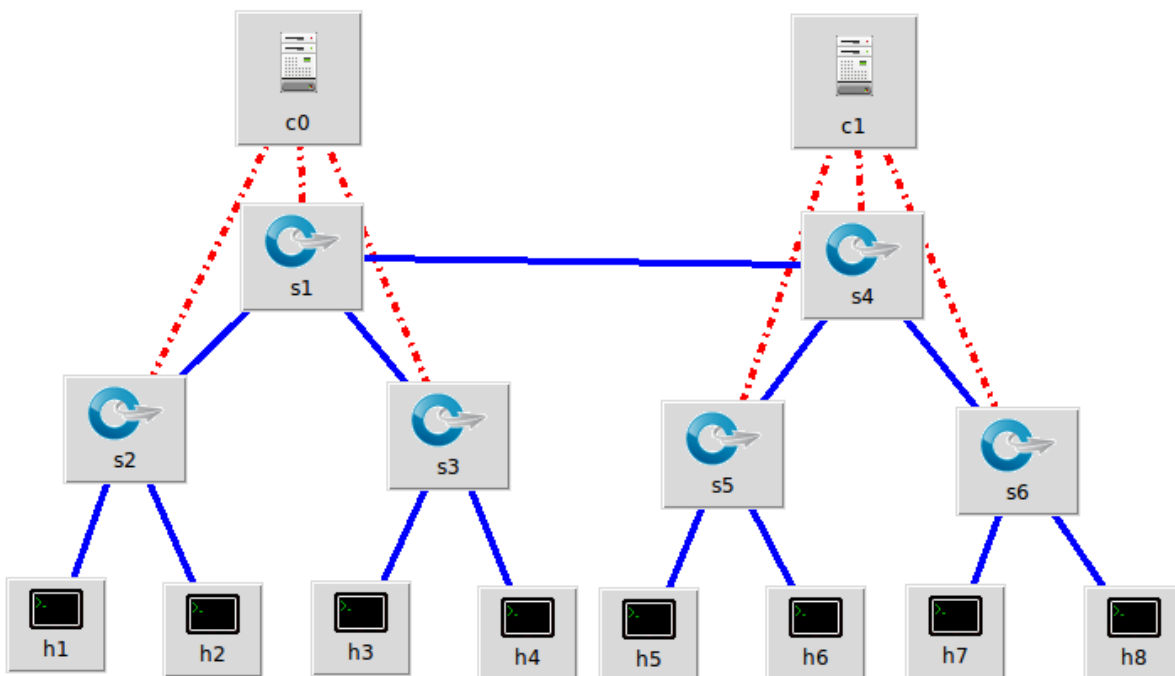


Figure 5.3: Mininet Virtual Network setup

Table 5.1, lists parameters of mininet configuration for the 3 groups of experiments. An attack threshold is set for the victim host when it receives more than 1Mbps of traffic in its compunction. Moreover, we set 4Mbps traffic that a

host should have to process an incoming traffic rate through its interfaces. Some group includes two experiments, in which different attack traffic patterns are used.

Table 5.1: Mininet Emulation parameters

Parameters	Values
Attack threshold bandwidth on host	1Mbps
Attack threshold bandwidth on port	4Mbps
Attack stopping threshold after applying an ingress policy	5 se
Decision time (sustained count)	5 sec
Traffic Monitoring interval	2 sec

All users including attackers and legitimate users begin to send packets at an independent random time between 0 and 4 seconds.

5.4 Traffic generation

To analyze our proposed system in the detection and mitigation process we used different data traffics. To simulate this traffic either it is normal or attack traffics we used Iperf. Iperf is a simple but powerful tool originally developed by NLANR/DAST for measuring performance and troubleshooting networks. It can be used to measure both TCP and UDP bandwidth performance by allowing the tuning of various parameters and UDP characteristics. Iperf uses the client-server architecture which needs to be at the opposite ends so that client can generate traffic and server can receive it. Once the generation is completed, both client and server report the performance results based on the protocol used.

5.5 Simulation and Result

Here we describe our experiments and analyze the results. Each of the experiment is simulated based on the performance evaluation metrics such as, Throughput, link bandwidth, and communication and computational overhead. All of the experiment is to evaluate the result of an attack and the defending

mechanism at the normal environment and in attack duration. On the experiment we measured and evaluated the rate of traffic flowing between nodes to identify an existence of an attack. Moreover, we monitored the bandwidth of the link between the legitimate client and the virtual application server.

5.5.1 Throughput measurement

We experimented the throughput between the client the server at normal traffic using TCP. Throughput is the total number of bits transmitted within a given amount of time. The servers are running with different port numbers. We monitored the communication traffic for every two second in the server side. Iperf tool is used for the measurement of bandwidth of the link in the performance analysis process. In order to analyze the throughput result we running Iperf command on each of the virtual machine.

Here, we have created host h1 as a server node and hosts h5 as a client nodes. We started Iperf measurement on both of the client node and server node in parallel. A TCP and UDP throughput performance analysis between server and client is shown in figure 5.4. The first experiment or the TCP protocol experiment is for the analysis of throughput during normal traffic flow between the client and the server in different network domain.

In our experiment a server starts listening on a default port 5001 having default TCP window size of 85.3 KB. From the experiment in Figure 5.4, we can see that the Y axis of the figure, represent the throughput measurement, and the X axis represents the time during that the experiment was done. The red line on the figure also represents the throughput of aggregate TCP traffic rate arriving at the bottleneck link of the traffic during normal traffic periods, and the pencil color represents the UDP traffic throughputs monitored within 2 seconds interval. The h5 client sends a 40.7 GB of data is to be transferred to its communication server which is h1 within 20 s of time interval. We also set a monitoring time interval for the throughput. Thus, the available throughput from the result is 15.2 Gbps. A throughput can be increase by increasing TCP window size.

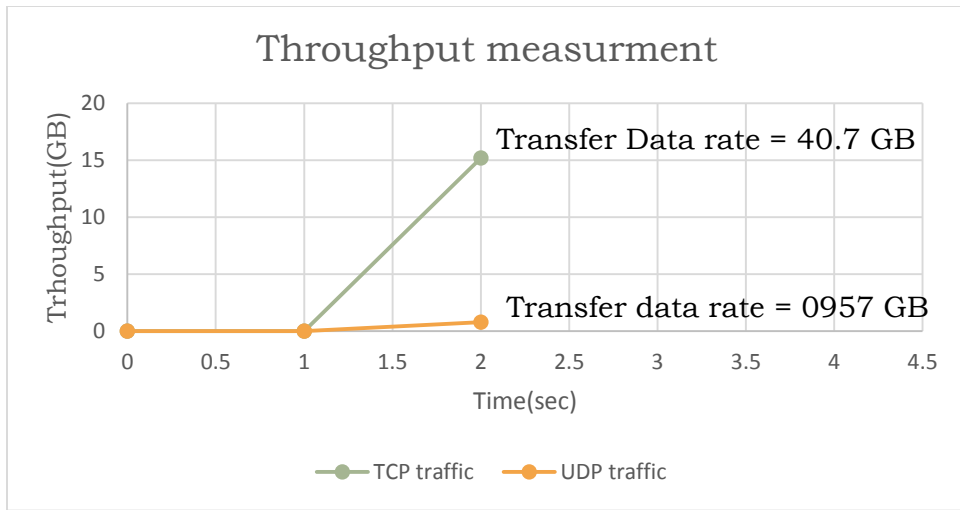


Figure 5.4: Throughput of TCP and UDP without defense

For measuring the throughput of UDP, we have done the same procedure like in TCP. The only difference in the experiment is setting some parameter changes. Which is the bandwidth of the link. At the client side we specified the bandwidth to use, 10 Gbps for this case, because iperf only uses 1Mbps of default bandwidth for UDP connection.

Unlike TCP, in UDP throughput get affected by different network parameters. Since, UDP is unreliable protocol, the factors affecting throughput are out of order delivery of packets, network jitter, and packet loss during transmission, etc. Moreover in UDP protocol, there is no sync between server and client nodes, hence, 936 MB of data noted at server site and 957 MB of data noted at client site is transferred in 20 s of time interval, which gives throughput of 787 and 803 Mbps at server site and client site respectively.

In the second of our experiment, we also measured the throughput of TCP and UDP traffic during normal traffic flow and in DDoS attack with the implementation of our rate limiting mechanism. Figure 5.5 shows as, the ability

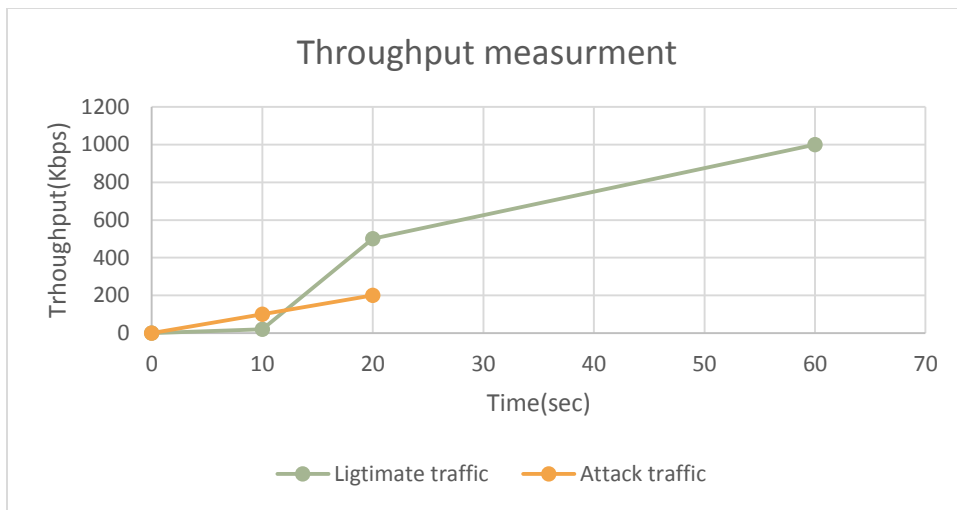


Figure 5.5: Throughput of TCP and UDP with defense

of our rate limiting technique in detecting the DDoS attacks. The throughput of TCP and UDP traffic without a defense mechanism is very high while the throughput of legitimate traffic is very low during an attack periods. The pushback technique shows good performance in protecting legitimate traffic in the figure 5.5. However, it seems to make no effort to control attack traffic once the legitimate traffic has been served properly. The reason is that the pushback technique lacks the ability to distinguish between attack and legitimate traffic when a diffused DDoS attack happens. Therefore, the pushback technique attempts to forward as much traffic as possible in order to lower the collateral damage for legitimate traffic.

Our DDoS defense framework, succeeds in lowering attack traffic while maintaining QoS for legitimate traffic. More aggressive attack traffic will be dropped more often because source-end edge OpenFlow switches have a lower rate limit value. However, we found a minor disadvantage in our rate limit mechanism during the process of traffic recovery. The phenomenon happens because the rate limit mechanism tries to remove the rate limit while an attack is still going. Moreover, our defense mechanism shows the cooperation between source node controller and the victim node controller in different network domains in the process of recovering.

In the experiments, our DDoS defense framework achieves even better performance and successfully maintains QoS of legitimate traffic by using the rate limit mechanism, while throttling back the amount of attack traffic.

5.5.2 Bandwidth measurement

In communication networks, bandwidth denotes the amount of data a link can carry. This is a metric which directly reflects the allocation of bandwidth in the normal and attack traffic is transmitted between the sending and the receiving host in the network. In the measurement of an allocation of bandwidth of the link, the fraction of attack traffic bandwidth is the value of attack traffic bandwidth over sum of attack and legitimate traffic bandwidth.

To analyze the link bandwidth allocation we started an IP flooding or attack traffic from the malicious node h5 to the server h1. Also, we a set the transmission duration for 15 seconds. The IP flooding attack traffic is monitored from the local virtual machine within 2 seconds monitoring time.

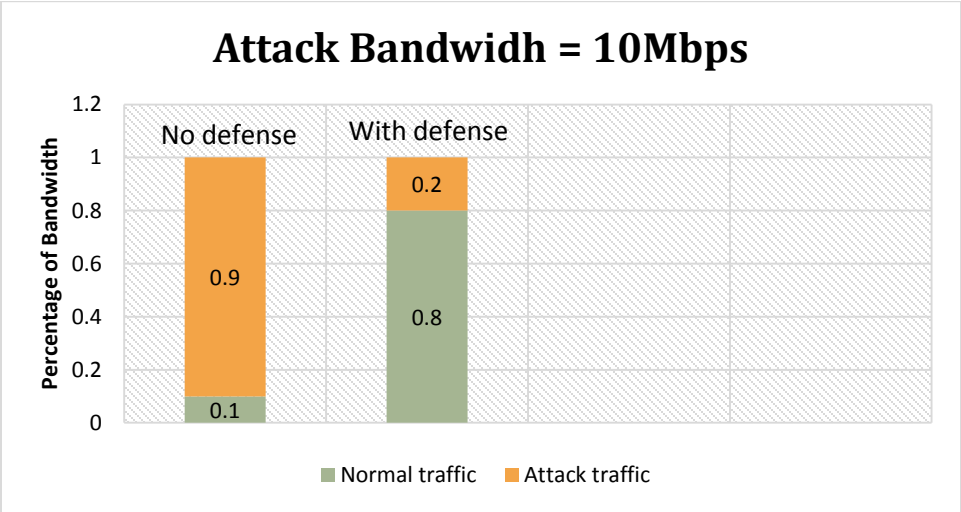


Figure 5.6: Allocated Bandwidth during normal and attack periods

Figure 5.6 illustrates the allocated bandwidth rate for the link under normal and attack periods. Moreover, the rate of a bandwidth without and with defense mechanism is used in the experiment.

In the experiment 10Mbps and 4Mbps of an attack IP traffic and normal traffic is sent to the victim host respectively. From the figure we see that an allocated bandwidth of the link without the implementation of any defense mechanism in the system is about 1 Mbps. This is calculated from an attack traffic is over the sum of an attack and normal traffic sent to the victim host. In the figure the yellow color shows the normal traffic and the green color is also showing the rate of an attack traffic. For an analysis an allocated bandwidth with implementation of our defense mechanism we generated the same rate of a normal and an attack traffic to the same host. The result shows as, 1Mbps of bandwidth is allocated for the link during this experiment.

According to the rule of a rate limiting implemented on the controller lower the amount of attack traffic reaches the victim node in the network. This is because fully deployed rate-limiters police attack traffic at different aggregation points on the traffic tree and thus manage to control it well. Moreover, this result also shows that the attack is detected and is stopped before reaching the victim node.

5.5.3 Drop rate of an attack traffic

The drop rate of attack traffic has been used for the purpose of evaluating the effectiveness of a DDoS defense system before. However, most researchers think it fails to capture whether legitimate service continues during the attack. For example, even if all attack traffic can be dropped by the edge router at the victim end, legitimate traffic may not be delivered properly simply because the edge router has no resources left to serve it. In fact, the edge router might be completely busy just in dropping attack traffic during the attack. In our experim-

Table 5.2: Drop rate of an attack traffic

Experiments	Drop rate
No defense	0.254
With Pushback	0.12

ents, we use the drop rate as a metric of the packet level to evaluate the distributed DDoS defense system. If we can demonstrate that the framework can effectively drop attack traffic at the source ends, it indirectly demonstrates that the framework can sustain QoS for legitimate traffic at the victim end. After we compare results in the two experiments in Table 5.2 we find the network without DDoS defense, attack traffic is only dropped in the edge router at the victim end due to a lack of resources. This shows that the normal congestion control mechanism cannot protect QoS for legitimate traffic from going down. Pushback drops attack traffic at upstream routers.

The reason is its lack of ability to differentiate between legitimate and attack traffic in a DDoS attack, and the consequent decision to throttle attack traffic just enough to maintain QoS for legitimate traffic.

5.5.4 Drop Rate of Legitimate Traffic

The drop rate of legitimate traffic is a direct metric which reflects the collateral damage for legitimate traffic. A good DDoS defense system always attempts to reduce collateral damage while lowering attack traffic as much as possible. In Table 5.3, we present the drop rates of legitimate traffic in the two experiments. The network without DDoS defense drops a large number of legitimate packets

Table 5.3: Drop rate of legitimate traffic

Experiments	Drop rate
No defense	0.56
With Pushback	0.29

due to congestion at the bottleneck link. Pushback DDoS defense framework show a much better performance to sustain QoS for legitimate traffic, meaning that collateral damage is very low.

5.5.5 Communicating and computational overhead

In this section, we discuss the communication and computation overhead of network resources in our scheme. According to our algorithm, when communication starts between the requesting node and victim node the controller starts monitoring of the flow of information and to calculate its rates. Monitoring the rates of the flow of information is a continuous process for the controller. Therefore, there is a communication overhead for the network during

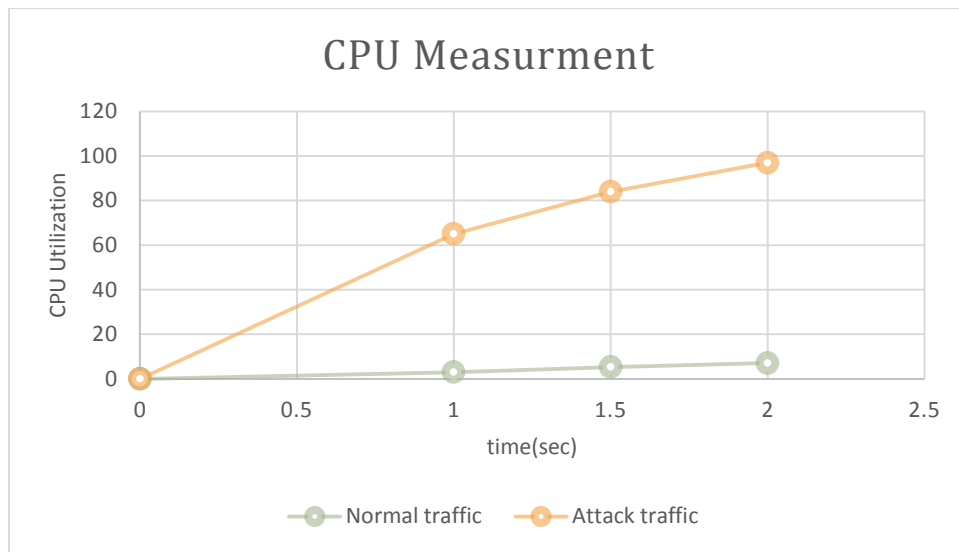


Figure 5.7: System CPU consumption at different traffic rate

the attack. Even though, there is a trade of between DDoS attack detection with the flow collecting and an effective attack detection and mitigation process. The communication between the controller and the OpenFlow switches is done by OpenFlow messages. So, the number of messages communication is not depends on the size of the topology.

When a switch receives the packet from any source for the first time it will install the flow entry in the flow table. When a switch learns all the nodes from the network, the complete flow establishment has taken place. The ping command in Mininet is used to check the connectivity of the entire network. It sends the message from every host to all other hosts. The CPU usage is measured for this

initial flow establishment. Figure 5.7 shows that the CPU usage is during an attack time rather than at normal state. When attacks occur, the CPU utilization

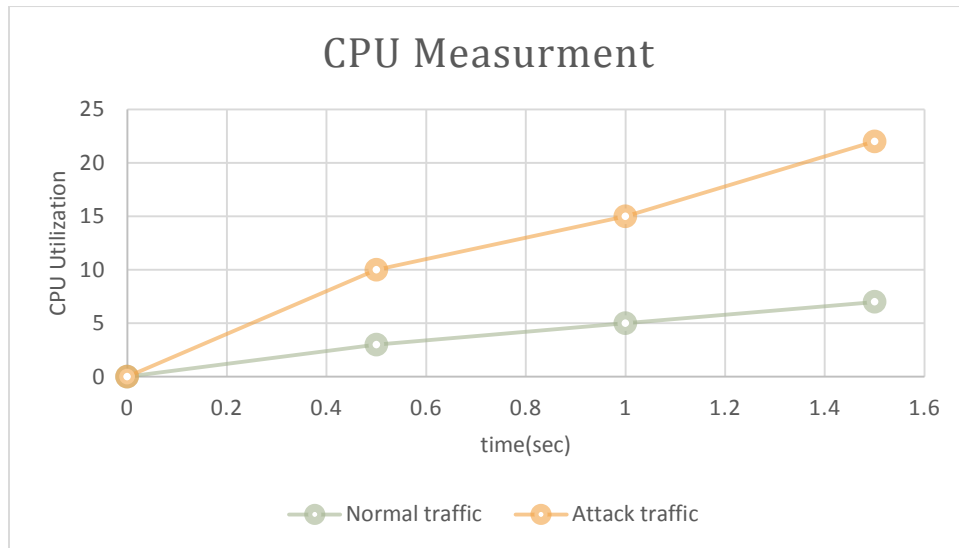


Figure 5.8: System CPU consumption at different traffic rate with defense

quickly reaches a peak (around 97%) in less than 2s. The possible reason is that the host is receiving too much IP flood traffic from a malicious node within a specific time period. When the Ping command is not fired there is a normal flow of traffic and the CPU usage result shows the normal process.

After we applying our rate traffic rate limiting mechanism the CPU consumption during normal and an attack duration is minimize. Figure 5.8, show the percentage of the CPU usage labeled with Red and in pencil color. The CPU usage during an attack period is limited which is compared to the previous result. The main reason is that the our proposed DDoS attack monitoring system sends flow modification message to the controller to limit the rate of an incoming traffic through the interfaces of this malicious node is connected.

5.6 Result Discussion

Form our experimental result we shown that our proposed system cant detected, identified and finally mitigated the attack in the cloud system in the distributed

manner. Each of the controller in SDN framework can identify an attacker in different network domains. The detection and identification of attack and an attacking node is done at the source of the node and replying a pushback response for the controller in other domain. The following discussion explains the overall result of the attack detection, identification and the mitigation of the proposed framework.

5.6.1 Local attack detection

For identify an attack controller monitoring the incoming traffic from the hosts to the OpenFlow switch periodically A protects hosts AAh1 and AAh2 by monitoring flows on all 3 switches in its domain for high bandwidth (5 Mbps) Identify an attacker check each host to check if it is sending data to the victim and its data rate is above a threshold (1 Mbps).

```

In Port 2 Eth Dst 0a:0a:00:00:00:01 Out Port 1 Bitrate 8054.032000
In Port 2 Eth Dst 0a:0b:00:00:00:01 Out Port 3 Bitrate 0.000000
In Port 2 Eth Dst 0a:0b:00:00:00:02 Out Port 3 Bitrate 0.000000
In Port 2 Eth Dst 0a:0b:0a:00:00:01 Out Port 3 Bitrate 0.000000
In Port 2 Eth Dst 0a:0b:0a:00:00:02 Out Port 3 Bitrate 0.000000
In Port 2 Eth Dst 0a:0b:0b:00:00:01 Out Port 3 Bitrate 0.000000
In Port 2 Eth Dst 0a:0b:0b:00:00:02 Out Port 3 Bitrate 0.000000
In Port 3 Eth Dst 0a:0a:00:00:00:01 Out Port 1 Bitrate 0.000000
In Port 3 Eth Dst 0a:0a:00:00:00:02 Out Port 2 Bitrate 0.000000
Identified victim: MAC 0a:0a:00:00:00:01 Host AAh1 Switch s11 Port 1
Attackers for vicim set(['AAh2']): AAh1
Identified victim: MAC 0a:0a:00:00:00:02 Host AAh2 Switch s11 Port 2
Attackers for vicim set(['AAh1']): AAh2
-----
----- Flow stats for switch s1 -----
In Port 1 Eth Dst 0a:0b:00:00:00:01 Out Port 2 Bitrate 0.000000
In Port 1 Eth Dst 0a:0b:00:00:00:02 Out Port 2 Bitrate 0.000000
In Port 1 Eth Dst 0a:0b:0a:00:00:01 Out Port 3 Bitrate 0.000000
In Port 1 Eth Dst 0a:0b:0a:00:00:02 Out Port 3 Bitrate 0.000000
In Port 1 Eth Dst 0a:0b:0b:00:00:01 Out Port 3 Bitrate 0.000000
In Port 1 Eth Dst 0a:0b:0b:00:00:02 Out Port 3 Bitrate 0.000000
In Port 2 Eth Dst 0a:0a:00:00:00:01 Out Port 1 Bitrate 0.000000
In Port 2 Eth Dst 0a:0a:00:00:00:02 Out Port 1 Bitrate 0.000000
In Port 2 Eth Dst 0a:0b:0a:00:00:01 Out Port 3 Bitrate 0.000000
In Port 2 Eth Dst 0a:0b:0a:00:00:02 Out Port 3 Bitrate 0.000000
In Port 2 Eth Dst 0a:0b:0b:00:00:01 Out Port 3 Bitrate 0.000000
In Port 2 Eth Dst 0a:0b:0b:00:00:02 Out Port 3 Bitrate 0.000000
In Port 3 Eth Dst 0a:0a:00:00:00:01 Out Port 1 Bitrate 0.000000
In Port 3 Eth Dst 0a:0a:00:00:00:02 Out Port 1 Bitrate 0.000000
In Port 3 Eth Dst 0a:0b:00:00:00:01 Out Port 2 Bitrate 0.000000
In Port 3 Eth Dst 0a:0b:00:00:00:02 Out Port 2 Bitrate 0.000000

```

Figure 5.9: Attack indentation at the local domain

An attack traffic generated from the local network identified by the host's MAC

address and its associated port the attack traffic initiated from. Identify the attacker and an interface which is connected to it.

Ryu controller and Mininet are launched in two different terminals as we can see on the figure 5.9. Ryu on the left, show how the controller measures the bit rate through the OpenFlow switch interfaces. Moreover, it identifies the victim host in the topology using its MAC address. On the right side of the figure node AAh2 which is a host from the public network domain generating an attack flow to the victim host IP address. An IP address of the victim host is 10.1.1.1 and, which is a cloud services exists in the private network domain. So, here the controller is easily monitor the rate of an incoming flow from all of OpenFlow switches connected to the distributed controllers. After the flow monitoring, the controller calculates the Bitrate. The Bitrate is calculated within 2 seconds interval. After the 2 second interval of flow monitoring, the controller identify the attacking host and the interface where which is connected to it. Once the attacking host is identified the controller install flow rule to drop or rate limit the rate of an incoming traffic from its local domain. This provides a space for those of legitimate clients to access the service.

5.6.2 Cooperative Defense

Implementing OpenFlow controller in the internet can be in centralized and distributed manner. Once it implemented in the distributed manner the controller oversees the entire network where they are connected to these network flows between the network devices. Controllers are sharing flow information for their analysis. For each of the commination traffic controllers are make a decision based on the flow. Based on this, the figure 5.10 shows, how the controllers in participating in the process of attack analysis and reaction for the Inter domain attack between the different cloud networks and send a push back request to the other controller for attack traffic control.

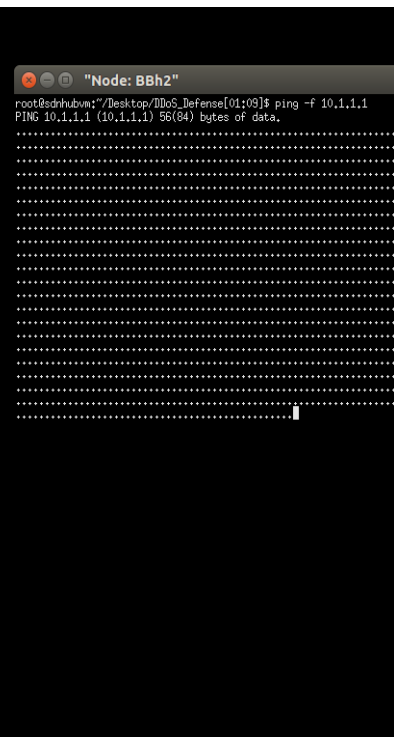
On the right side of the screen shoot, host BBh2 (a host from the public network domain) generating a flooding attack to the remote server in the private network

domain. The controller in its domain send a pushback request message to the upstream OpenFlow switches. After the pushback message received by the controller the controller responding to the pushback request to apply filtering of a traffic to an interface where the flooding traffic initiated from. After some time, the connected host is stopped the sending of malicious traffic to the victim host, and then the traffic filtering rule process is removed from the controller.

```

In Port      2 Eth Dst 0a:0b:0b:00:00:01 Out Port      1 Bitrate 0.000000
In Port      3 Eth Dst 0a:0b:0b:00:00:01 Out Port      1 Bitrate 0.000000
In Port      3 Eth Dst 0a:0b:0b:00:00:02 Out Port      2 Bitrate 6623.792000
-----
Received pushback message for victim: 0a:0a:00:00:00:01
-----
Flow stats for switch s2 -----
In Port      1 Eth Dst 0a:0a:00:00:00:01 Out Port      3 Bitrate 0.000000
In Port      1 Eth Dst 0a:0a:00:00:00:02 Out Port      3 Bitrate 0.000000
In Port      1 Eth Dst 0a:0b:00:00:00:01 Out Port      3 Bitrate 0.000000
In Port      1 Eth Dst 0a:0b:00:00:00:02 Out Port      3 Bitrate 0.000000
In Port      1 Eth Dst 0a:0b:0b:00:00:01 Out Port      2 Bitrate 0.000000
In Port      1 Eth Dst 0a:0b:0b:00:00:02 Out Port      2 Bitrate 0.000000
In Port      2 Eth Dst 0a:0a:00:00:00:01 Out Port      3 Bitrate 8492.680000
In Port      2 Eth Dst 0a:0a:00:00:00:02 Out Port      3 Bitrate 0.000000
In Port      2 Eth Dst 0a:0b:00:00:00:01 Out Port      3 Bitrate 0.000000
In Port      2 Eth Dst 0a:0b:00:00:00:02 Out Port      3 Bitrate 0.000000
164684bc-0d36-415e-958b-3c20f5d451f2
Out Port      3 Bitrate 0.000000
In Port      2 Eth Dst 0a:0b:0a:00:00:01 Out Port      1 Bitrate 0.000000
In Port      2 Eth Dst 0a:0b:0a:00:00:02 Out Port      1 Bitrate 0.000000
In Port      3 Eth Dst 0a:0b:0a:00:00:01 Out Port      1 Bitrate 0.000000
In Port      3 Eth Dst 0a:0b:0a:00:00:02 Out Port      1 Bitrate 0.000000
In Port      3 Eth Dst 0a:0b:0b:00:00:01 Out Port      2 Bitrate 0.000000
In Port      3 Eth Dst 0a:0b:0b:00:00:02 Out Port      2 Bitrate 8493.072000
Responding to pushback request, applying egress on s22
-----
Flow stats for switch s21 -----
In Port      1 Eth Dst 0a:0a:00:00:00:01 Out Port      3 Bitrate 0.000000
In Port      1 Eth Dst 0a:0a:00:00:00:02 Out Port      3 Bitrate 0.000000
In Port      1 Eth Dst 0a:0b:00:00:00:01 Out Port      3 Bitrate 0.000000
In Port      1 Eth Dst 0a:0b:00:00:00:02 Out Port      3 Bitrate 0.000000
In Port      1 Eth Dst 0a:0b:0a:00:00:02 Out Port      2 Bitrate 0.000000

```



The image shows a terminal window with two panes. The left pane displays network flow statistics for switches s2 and s21, including ingress and egress ports, destination MAC addresses, and bitrates. It also shows a received pushback message and the application of egress rules on switch s22. The right pane shows a terminal window titled "Node: BBh2" with a root user prompt and a ping command: 'ping -f 10.1.1.1'. The output of the ping command shows 'PING 10.1.1.1 (10.1.1.1) 56(84) bytes of data.' followed by a series of dots representing the ping results.

Figure 5.10: DDoS attack defense cooperation between the controllers

Moreover, in real scenario, with centralized architecture as in OpenFlow, the central network controller can co-ordinate with its switches and take the right decisions in a minimal time provided correct DDoS detection happens, by employing strategies like blocking the malicious node at the ingress router itself than going for a Pushback way of handling. But, our approach of Pushback takes care that even if the detection went wrong (false positive or due to spike in the network) and was not actually a DDoS attack, then during the phase of the pushback, the negation rule on the wrongly assumed malicious node can be removed and simultaneously ensure that the entire network is safe.

```

In Port 2 Eth Dst 0a:0b:0b:00:00:01 Out Port 3 Bitrate 0.000000
In Port 2 Eth Dst 0a:0b:0b:00:00:02 Out Port 3 Bitrate 0.000000
In Port 3 Eth Dst 0a:0b:0a:00:00:01 Out Port 1 Bitrate 0.000000
In Port 3 Eth Dst 0a:0b:0a:00:00:02 Out Port 2 Bitrate 0.000000
----- Flow stats for switch s22 -----
In Port 1 Eth Dst 0a:0a:00:00:00:01 Out Port 3 Bitrate 0.000000
In Port 1 Eth Dst 0a:0a:00:00:00:02 Out Port 3 Bitrate 0.000000
In Port 1 Eth Dst 0a:0b:00:00:00:01 Out Port 3 Bitrate 0.000000
In Port 1 Eth Dst 0a:0b:00:00:00:02 Out Port 3 Bitrate 0.000000
In Port 1 Eth Dst 0a:0b:0a:00:00:01 Out Port 3 Bitrate 0.000000
In Port 1 Eth Dst 0a:0b:0a:00:00:02 Out Port 3 Bitrate 0.000000
In Port 1 Eth Dst 0a:0b:0b:00:00:02 Out Port 2 Bitrate 0.000000
In Port 2 Eth Dst 0a:0a:00:00:00:01 Out Port 3 Bitrate 0.000000
In Port 2 Eth Dst 0a:0a:00:00:00:02 Out Port 3 Bitrate 0.000000
In Port 2 Eth Dst 0a:0b:00:00:00:01 Out Port 3 Bitrate 0.000000
In Port 2 Eth Dst 0a:0b:00:00:00:02 Out Port 3 Bitrate 0.000000
In Port 2 Eth Dst 0a:0b:0a:00:00:01 Out Port 3 Bitrate 0.000000
In Port 2 Eth Dst 0a:0b:0a:00:00:02 Out Port 3 Bitrate 0.000000
In Port 2 Eth Dst 0a:0b:0b:00:00:01 Out Port 1 Bitrate 0.000000
In Port 3 Eth Dst 0a:0b:0b:00:00:01 Out Port 1 Bitrate 0.000000
In Port 3 Eth Dst 0a:0b:0b:00:00:02 Out Port 2 Bitrate 0.000000
Removing ingress filters on BBh2, on switch s22 at port 2
----- Flow stats for switch s2 -----
In Port 1 Eth Dst 0a:0a:00:00:00:01 Out Port 3 Bitrate 0.000000
In Port 1 Eth Dst 0a:0a:00:00:00:02 Out Port 3 Bitrate 0.000000
In Port 1 Eth Dst 0a:0b:00:00:00:01 Out Port 3 Bitrate 0.000000
In Port 1 Eth Dst 0a:0b:00:00:00:02 Out Port 3 Bitrate 0.000000
In Port 1 Eth Dst 0a:0b:0b:00:00:01 Out Port 2 Bitrate 0.000000
In Port 1 Eth Dst 0a:0b:0b:00:00:02 Out Port 2 Bitrate 0.000000
In Port 2 Eth Dst 0a:0a:00:00:00:01 Out Port 3 Bitrate 0.000000

```

Figure 5.11: DDoS attack defense cooperation between the controllers

DDoS defense in real system have to cope with complicated scenarios like deception by the malicious nodes thought spoofing of their IP address, but still our implementation shows that defense can be erected quickly through software defined networking.

Chapter 6

Conclusion

The currently vulnerable internetworking cloud system and the growing number of Internet crimes inspires much research to increase Internet security and protect it from intrusions. Distributed denial of service is a major threat that cannot be addressed through isolated actions of sparsely deployed defense nodes. Instead, various defense systems must organize into a framework and inter-operate, exchanging information and service, and acting together, against the threat.

The purpose of this research work is to design and evaluate a SDN-based cooperative pushback defense framework for flooding attack in cloud computing environment. In our approach when a communication begins, each of the controller starts monitoring the rate of incoming flow periodically. The monitored traffic is above from the pre-defined threshold, the controller automatically identify an attacking node in the network. Then the mitigation of an attacking process is continues. Each of the components in the framework works hierarchical fashion in the process of defense. We simulated and investigated our DDoS cooperative defense mechanism with different tools. Mininet, Ryu and Iperf are tools used for virtual topology creation, application program developing and performance analysis respectively. Using this tools we conducted different experimental scenarios. Each of our experiment is evaluated with our performance metrics, throughput, link bandwidth measurement, time delay and communication and computational resources. We analyzed this results of each experiment on each approaches and draw a comparison about the result obtained during normal and attack periods. Finally, in the discussion section we clearly explained the how our cooperative DDoS defense mechanism is performing and handling for attacks in the private or local domain as well as attack in public domains.

In the final of our work, we able to contribute some solution to the cloud tenants, private cloud administration bodies about a simple defense mechanism for flooding attack that exists in the cloud network domain. To describe some our contribution:

- With the complete mechanism, we protect SDN-based cloud networks against both DoS and DDoS attacks. We identified the attackers or detecting the attackers and stopping this attack before causing large problem in the network. Traffic of trusted users is still handled normally throughout the process.
- We also designed a flexible, and scalable detection and mitigation of different kinds of attacks on the cloud computing. In addition to this we done an experiment and analysis how the proposed system detect attacks accurately and easily through extensive testing scenarios.
- The implementation of the proposed method brings no modifications on current routing software. On the other hand, our proposed method can work independently as an additional module on OpenFlow switches for monitoring and recording flow information, and communicating with its upstream and downstream distributed OpenFlow controller when the pushback procedure is carried out.
- In order to evaluate the efficiency of our detection system we used different important metrics for traffic and attack analysis. From the experiment performance evaluation we investigated pushback can be a defense mechanism for distributed attacks on the cloud environment by deploying distributed DDoS attack defending controllers at the local and remote network domain.

Even though a number of DDoS attack defense mechanism are proposed by many research works, the attacks and its impact in the Internet and cloud computing is exponential growing and the attacking techniques are changing from day to days. So that, the result of this research provides its own contribution to the ongoing researches in this domain area.

6.1 Limitation

There are a few aspects that need future discussion and some implementation and operational issues need to be addressed before pushback cloud be more robust in combating against DDoS attacks and deployed in the Internet. The main limitation of this filtering scheme is that if the attack does not use spoofed addresses then this filtering scheme will fail. Botnets with huge armies of thousands of zombies do not even care to spoof the source address; in such cases the strategy adopted by the Ingress/Egress filtering will fail.

Secondly, traffic monitoring is generates overhead when the controller requests trigger reports (to verify the data traffic behavior of a specific incoming flow) and periodic reports (to calculate the minimum number of packets per successful connection—parameter) from the switches. Hence, we must to balance between report period and overhead to get the optimal benefit.

References

- [1] P. Ankita and F. Khatiwala, "Survey on DDoS Attack Detection and Prevention in Cloud," vol. 3, no. 2, pp. 43–47, 2015.
- [2] R. Udendhran, "New Framework to Detect and Prevent Denial of Service Attack in Cloud Computing Environment," vol. 12, pp. 87–91, 2014.
- [3] N. Wu and J. Zhang, "Workshop on Information Assurance Investigation of Pushback Based Detection and Prevention of Network Bandwidth Attacks," pp. 416–423, 2004.
- [4] T. M. Gil and M. Poletto, "MULTOPS : a data-structure for bandwidth attack detection."
- [5] D. Attacks, A. Rising, and S. D. Mitigation, "Real-Time SDN and NFV Analytics for DDoS Mitigation."
- [6] B. Wang, Y. Zheng, W. Lou, and Y. T. Hou, "DDoS attack protection in the era of cloud computing and Software-Defined Networking," *Comput. NETWORKS*, vol. 81, pp. 308–319, 2015.
- [7] M. Aamir *et al.*, "Software-Defined Networking : A Comprehensive Survey," *IEEE Commun. Mag.*, vol. 3, no. 1, pp. 1–31, 2013.
- [8] M. H. Bhuyan, H. J. Kashyap, and D. K. Bhattacharyya, "Detecting Distributed Denial of Service Attacks : Methods , Tools and Future Directions."
- [9] G. Somani, M. S. Gaur, D. Sanghi, M. Conti, and R. Buyya, "1 DDoS Attacks in Cloud Computing: Issues, Taxonomy, and Future Directions," vol. 1, no. 1, 2017.
- [10] M. C. Fight and A. T. Attacks, "InfoSec Reading Room."
- [11] I. Sriram and A. Khajeh-hosseini, "Research Agenda in Cloud Technologies," 2008.
- [12] M. K. Srinivasan, K. Sarukesi, P. Rodrigues, M. S. Manoj, and P. Revathy,

- “State-of-the-art cloud computing security taxonomies - A classification of security challenges in the present cloud computing environment,” *Proc. Int. Conf. Adv. Comput. Commun. Informatics - ICACCI '12*, p. 470, 2012.
- [13] S. O. Kuyoro, “Cloud Computing Security Issues and Challenges,” no. 3, pp. 247–255, 2011.
- [14] E. Aguiar, Y. Zhang, and M. Blanton, “An Overview of Issues and Recent Developments in Cloud Computing and Storage Security,” pp. 1–31.
- [15] A. Shawish and M. Salama, “Cloud Computing : Paradigms and Technologies,” pp. 39–68.
- [16] A. Bonguet and M. Bellaiche, “A Survey of Denial-of-Service and Distributed Denial of Service Attacks and Defenses in Cloud Computing,” 2017.
- [17] B. B. Gupta, S. Member, R. C. Joshi, and M. Misra, “Distributed Denial of Service Prevention Techniques,” vol. 2, no. 2, pp. 268–276, 2010.
- [18] “D-WARD : Source-End Defense Against Distributed Denial-of-Service Attacks,” 2003.
- [19] K. M. Prasad, a R. M. Reddy, and K. V. Rao, “DoS and DDoS Attacks: Defense, Detection and TracebackMechanisms -A Survey,” vol. 14, no. 7, 2014.
- [20] D. Kreutz *et al.*, “Software-Defined Networking : A Comprehensive Survey,” pp. 1–61.
- [21] B. N. Astuto, M. Mendon, X. N. Nguyen, K. Obraczka, B. N. Astuto, and M. Mendon, “A Survey of Software-Defined Networking : Past , Present , and Future of Programmable Networks To cite this version :,” 2014.
- [22] N. Yu *et al.*, “Software-Defined Networking : A Comprehensive Survey,” *IEEE Commun. Mag.*, vol. 3, no. 1, pp. 1–31, 2013.
- [23] B. Heller, “OpenFlow Switch Specification,” *Current*, vol. 0, pp. 1–36, 2009.
- [24] Q. Yan, F. R. Yu, Q. Gong, and J. Li, “Software-Defined Networking (SDN

-) and Distributed Denial of Service (DDoS) Attacks in Cloud Computing Environments : A Survey , Some Research Issues , and Challenges,” vol. 18, no. 1, pp. 602–622, 2016.
- [25] S. Shin and G. Gu, “CloudWatcher: Network Security Monitoring Using OpenFlow in Dynamic Cloud Networks.”
- [26] T. Xing, D. Huang, L. Xu, C. Chung, and P. Khatkar, “SnortFlow : A OpenFlow-based Intrusion Prevention System in Cloud Environment,” 2013.
- [27] S. Shin *et al.*, “FRESCO : Modular Composable Security Services for Software-Defined Networks,” vol. 2, no. February, 2013.
- [28] “A Novel Design for Future On-Demand Service and Security,” pp. 385–388, 2010.
- [29] S. A. Mehdi, J. Khalid, and S. A. Khayam, “Revisiting Traffic Anomaly Detection Using Software Defined Networking *,” pp. 161–180, 2011.
- [30] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, “Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments,” *Comput. NETWORKS*, 2013.
- [31] R. Braga, E. Mota, and A. Passito, “Lightweight DDoS Flooding Attack Detection Using NOX / OpenFlow,” pp. 408–415, 2010.
- [32] S. B. Lee, M. S. Kang, and V. D. Gligor, “CoDef: Collaborative Defense Against Large-Scale Link-Flooding Attacks,” pp. 417–427.
- [33] J. Ioannidis and S. M. Bellovin, “No Title,” pp. 1–12.
- [34] R. Wang, Z. Jia, and L. Ju, “An Entropy-Based Distributed DDoS Detection Mechanism in Software-Defined Networking,” 2015.

Declaration

The work contained in this thesis has not been previously submitted to meet requirements for an award at this or any higher education institution. To the best of my knowledge and beliefs, the thesis contains no material previously published or written by another person except where due reference is made.

Declared by: Ashenafi Meshesha

Signature

Date

Confirmed by advisor: Dr. Towfik Jemal (PhD.)

Signature

Date