ISSN XXXX XXXX © 2017 IJESC

Research Article Volume 7 Issue No.6

Early Detection of DDoS Attacks in a Multi-Controller Based SDN

Dr.T.Pandikumar[1], Frew Atkilt[2], Capt.Abdulkadir Hassen[3]
Associate Professor[1], M.Tech Student[2], Lecturer[3]
Department of Computer & IT
College of Engineering, Defence University, Debre Zeyit, Ethiopia

Abstract:
Networks used to depend on hardware devices which have the control and data plane in a single plane. This made networks susceptible to failure because if the hardware fails the network stops. This is the reason software defined network structure has come into existence. By separating the control plane from the data plane, software-defined networking offers several benefits for networking. But that didn't mean software-defined networking solves every problem in the network. Areas like networks' scalability, reliability and availability remain as the issues yet to be addressed. The main concept behind SDN is the separation of the network's control and forwarding planes with the control plane moved to the centralized controller, which provides an ideal platform for distributed detection and mitigation of DDoS attacks. This research work takes advantage of this special ability of SDN to propose a solution with an implementation running at the multi-controller to detect DDoS attack at the early stage. The method not only can detect the attacks but also identify the attacking paths and start a mitigation process to provide protection for the network devices the moment an attack is detected. The proposed method is based on the entropy variation of the destination host targeted with its IP address and can detect the attack within the first 250 packets of malicious traffic attacking a particular host in the SDN.

Keywords: DDoS attack, Entropy, Multi-controller, SDN

1. INTRODUCTION

Networking principles have remained mostly unchanged over the past decade. Networks are built using more or less sophisticated switches and routers. These devices are being developed by tens of vendors usually using proprietary operating system and interfaces. Building heterogeneous networks on devices from different vendor's means that organization has to employ a specialist on every router brand. Configuration of different systems also increases the probability of configuration mistakes. This issue coupled with incompatibility of different versions of systems from one vendor makes heterogeneous networks difficult or very expensive to manage. There is a need for a new technology to make networks more scalable, dynamic and to allow easier management of network devices from different vendors. These needs could be fulfilled by programmable networks, i.e., by Software Defined Networking (SDN) (Martin Vizv´ary, January 2015). SDN could replace traditional networking. It is based on the abstraction of a control and a data plane. The main idea is to produce less sophisticated data plane devices, e. g., switches, which only forward the traffic according to a set of rules defined by the software in the control plane. This should remove the differences in proprietary interfaces of devices and makes the network administration independent of data plane devices vendors. SDN also enables applications and network services to treat the network as one logical entity and grants unified access to all devices through the SDN control plane. This opens the upper layer of the network to software that can manage how traffic in the network is forwarded. The research in the field of SDN and general security in SDN is still in its early phase. The SDN will not erase the DDoS attacks from the Internet. Moreover, every new technology and level of abstraction opens new attack vectors. However, we believe that the attributes of SDN can help to detect and mitigate the attacks.

Our research will be dedicated to analysis of security challenges in SDN from the point of view of DDoS attacks and development of a new DDoS attacks mitigation technique. We believe that SDN gives us a new powerful tool against DDoS attacks. The higher flexibility and easier management of networks could be a powerful tool for detection and mitigation of DDoS attacks. However, one should be aware of upcoming security threats accompanied with the deployment of SDN. The research focused on the security in SDN is still in its early phase. This research focused on the security of the data plane, security of the control plane, security of the communication between these planes and on enhancing the network security using SDN, which is also our goal. The result of our research is going to be a novel method for mitigation of DDoS attacks using the benefits of Software Defined Networking, which enhance the network security. Combination of the existing detection methods and management of SDN forms a new way of DDoS mitigation in future networks.

1.2 Statement of the Problem

SDN is a new networking approach that is introduced with the goal to simplify the network management by separating the data and control planes. SDN has brought with itself programmability in the network control plane. The shift of the control logic from networking devices, such as switches and routers, in traditional networks to a centralized unit known as the controller permits the physical network hardware to be detached from the control Plane. This separation simplifies the design of new protocols and implementation of new network services such as access control, QOS, enforcement of new policies, bandwidth management, traffic engineering and etc. No longer does every small change need to come at the cost of reconfiguring all the network devices. The SDN networks and its controller can be seen as slice of the network. We are focusing on each of these slices to protect it against DDoS. If

the connection between the switches and the controller is lost, the network will lose its processing plane. That means packet processing is no longer done in the controller and by losing the controller, the SDN architecture is lost. The aim of this research is detecting a DDoS attack in its early stages. The term early depends on the network itself. Since the controller software can be run on a laptop or a powerful desktop, the term early would depend on the tolerance of the device and traffic properties. However, if the detection happens in the first few hundred packets, the mitigation is applied before the controller is completely swamped with the large number of malicious packets. To accomplish this goal, a fast and effective method is needed that works within the controller. At the same time, it must be lightweight to avoid excessive processing power usage, specially, at the peak of an attack.

1.3 Objective of the Study

1.3.1 General objective

The general objective of this research work is to Early Detect DDoS attacks by using multi-controller Software Defined Networks.

1.3.2 Specific objectives

- In order to achieve the aforementioned general objective the following specific objectives need to be achieved:
- Detection of attack in a multi-controller SDN structure.
- To analyze mitigation of the attack.
- To develop a prevention mechanism to avoid DDoS attack in its initial stage before harming our network.
- Implement the proposed mechanism using Mininet.

1.4 Scope of the Study

- Find a solution to detect DDoS in SDN before it overwhelms the controller.
- Proposed a lightweight and fast DDoS detection mechanism based on entropy, to protect the controller.
- To Show the effectiveness of the solution through extensive simulations.

1.5 Limitation of the project

The Proposed Policy will be implemented in simulator. The Attack is only concentrating on traffic attack sending a huge volume of TCP, UDP and ICMP packets to the target.

1.6 Significance of the study

In order for SDN to deliver on its full promise, it must be enabled by open networking standards that can be easily integrated with current infrastructures. Adopting an SDN methodology has a myriad of benefits including flexibility, scalability, redundancy, and performance. In a traditional network, there might be certain limited hardware and software pieces. When a network requires additional resources, there will be considerable cost in buying new hardware and licensing. With SDN, the network is abstracted onto software, leaving more choice and flexibility in purchasing hardware. In addition, a growing network can be more easily supported by SDN because a network administrator or engineer can simply add more virtual switches or routers rather than purchase costly equipment and licensing.

II. SDN AND OPEN FLOW ARCHITECTURE

2.1 Introduction

This chapter provides discussion on fundamental concepts of the study. This chapter is organized into four main sections. The main important points of the chapter consists an in depth study of the Software Defined Networks, SDN Controller, Open Flow Protocol and DDoS Attacks.

2.2 Software Defined Networks

Software-defined networking (SDN) is an approach to computer networking that allows network administrators to manage network services through abstraction of lower-level functionality. SDN is meant to address the fact that the static architecture of traditional networks doesn't support the dynamic, scalable computing and storage needs of more modern computing environments such as data centers. This is done by decoupling or disassociating the system that makes decisions about where traffic is sent (the control plane) from the underlying systems that forward traffic to the selected destination (the data plane).

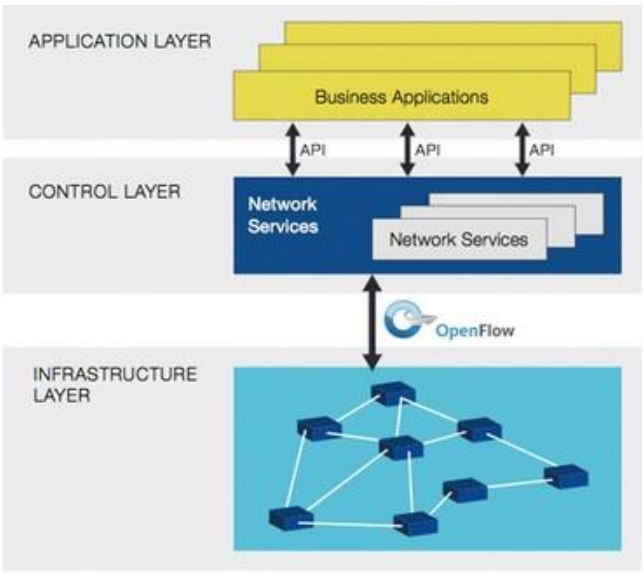

Figure.1. Software Defined Networking (SDN) Framew

Software-defined networking (SDN) is an architecture purporting to be dynamic, manageable, cost-effective, and adaptable, seeking to be suitable for the high-bandwidth, dynamic nature of today's applications. SDN architectures decouple network control and forwarding functions, enabling network control to become directly programmable and the underlying infrastructure to be abstracted from applications and network services (Open Networking foundation, 2017). Software-defined networking (SDN) has gained a lot of attention in recent years, because it addresses the lack of programmability in existing networking architectures and enables easier and faster network innovation. SDN clearly separates the data plane from the control plane and facilitates software implementations of complex networking applications on top. There is the hope for less specific and cheaper hardware that can be controlled by software applications through standardized interfaces. Additionally, there is the expectation for more flexibility by dynamically adding new features to the network in the form of networking applications. This concept is known from mobile phone operating systems, such as Apple's iOS and Google's Android, where "apps" can dynamically be added to the system (Wolfgang and Michael, 2014).

The SDN architecture is:

- *Directly programmable*: Network control is directly programmable because it is decoupled from forwarding functions.

- *Agile*: Abstracting control from forwarding lets administrators dynamically adjust network-wide traffic flow to meet changing needs.
- *Centrally managed*: Network intelligence is (logically) centralized in software-based SDN controllers that maintain a global view of the network, which appears to applications and policy engines as a single, logical switch.
- *Programmatically configured*: SDN lets network managers configure, manage, secure, and optimize network resources very quickly via dynamic, automated SDN programs, which they can write themselves because the programs do not depend on proprietary software.

2.2.1 SDN Architecture

SDN architecture contains six major components. First is the management plane, which is a set of network applications that manage the control logic of a software-defined network. Rather than using a command line interface, SDN-enabled networks use programmability to give flexibility and easiness to the task of implementing new applications and services, such as routing, load balancing, policy enforcement, or a custom application from a service provider. It also allows orchestration and automation of the network via existing APIs. Second is the control plane that is the most intelligent and important layer of an SDN architecture. It contains one or various controllers that forward the different types of rules and policies to the infrastructure layer through the southbound interface.

Third, the data plane, also known as the infrastructure layer, represents the forwarding devices on the network (routers, switches, load balancers, etc.). It uses the southbound APIs to interact with the control plane by receiving the forwarding rules and policies to apply them to the corresponding devices. Fourth, the northbound interfaces that permit communication between the control layer and the management layer are mainly a set of open source application programming interfaces (APIs). Fifth, the east-west interfaces, which are not yet standardized, allow communication between the multiple controllers.

They use a system of notification and messaging or a distributed routing protocol like BGP and OSPF. Sixth, the southbound interfaces allow interaction between the control plane and the data plane, which can be defined summarily as protocols that permit the controller to push policies to the forwarding plane. The OpenFlow protocol is the most widely accepted and implemented southbound API for SDN-enabled networks (Othmane, Mouad, and Redouane, 2016).

2.2.2 SDN Controllers

One of the core ideas of the SDN philosophy is the existence of a network operating system placed between the network infrastructure and the application layer. This network operating system is responsible for coordinating and managing the resources of the whole network and for revealing an abstract unified view of all components to the applications executed on top of it. This idea is analogous to the one followed in a typical computer system, where the operating system lies between the hardware and the user space and is responsible for managing the hardware resources and providing common services for user programs. Similarly, network administrators and developers are now presented with a homogeneous environment easier to program and configure much like a typical computer program developer would.

III. LITERATURE SURVEY

3.1 Introduction

DDoS Attack have recently recognized as one of the most threats to the SDN based networks. Many researches have been conducted to analyze and detect DDoS Attack and the results of these researches contributed to security enhancement and draw new idea on strengthening the detection of DDoS Attack in SDN based networks. In this chapter, previous research works which are directly or indirectly related to this study are reviewed. The area of focus and limitations of these works are also discussed.

3.2 Reviewed Literature

One of the works published on the subject matter of Detection of DDoS Attacks in SDN Controller is a paper titled as "Early Detection of DDoS Attacks in Software Defined Networks Controller" by Seyed Mohammad (2015). In his solution, randomness of the incoming packets is measured. A good measure of randomness is entropy. Entropy measures the probability of an event happening with respect to the total number of events. For instance, in a network of 64 hosts, all hosts should have a reasonably close probability of receiving new incoming packets. This will results in, reasonably, high entropy. New packet, in the sense that there is no flow for it in the switch table and it has to be sent to the controller to be validated for a new flow. If one or a number of hosts starts to receive excessive incoming packets, the randomness decreases and entropy drops. This research makes use of this property of entropy to detect an attack at its early stages. Based on the tests that are done in this research, we choose a threshold for entropy and lower values will be considered as attacks. Being programmable is one of the major advantages of SDN. Any time the network configuration changes, the threshold can be adjusted. And, it can be adjusted while the network is running live traffic so there is no restriction. Depending on the network, the entropy can be of the destination IP address, VLAN tag, destination port or any other applicable field. If it is lower than the set threshold, it will be considered an attack.

The need of Entropy

When packets arrive at the controller, the source address is always new. This is the reason they come to the controller. There has not been an instance of them in the table of the switch so they are passed on to the controller. For every new incoming connection, the controller will install a flow in the switch so that the rest of the incoming packets will be directed to the destination without further processing. The other known fact about the new packets coming to the controller is that the destination host is in the network of the controller. The network consists of the switches and hosts that are connected to it. Knowing the packet is new and the destination is in the network, the level of randomness can be quantified by calculating the entropy based on a window size. The window size is the number of incoming new packets that are used for calculating entropy. In this case, maximum entropy occurs when each packet is destined to exactly one host. Minimum entropy occurs when all the packets in a window are destined for a single host. The paper also discusses the possibility of losing the controller and identifies the need for a backup one. The paper proposes a second controller that runs in parallel to the current running controller. If the switches lose connection to the controller, they will look for the second controller,

which is added to the configuration of the switch. One of the mentioned scenarios is losing the controller in a DDoS attack. Figure 2 shows running two controllers in parallel. The first controller will continuously send status updates to the backup controller announcing that it is alive. If the first controller goes to an unknown state or becomes unreachable, the second controller will take control and starts running the network normally.

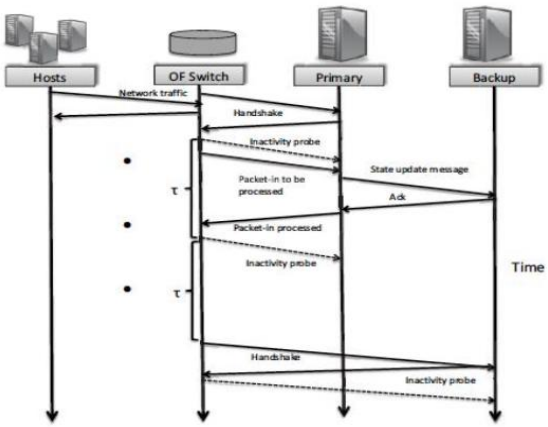

Figure .2. Two controllers for resiliency in Open flow

The Paper by Sandeep Singh, R.A. Khan and Alka Agrawal (2015) developed four steps strategy to avoid the attack from being happen. According to the first step by increasing the available queue size for incoming packets and simply direct them to wait until the queue becomes empty. This will give some relaxation to the traffic handling. After applying this step if still the traffic is continuously increasing and the buffering policy came in to failure mode then it shall go for the second step. Setting Timer/ Time stamping: when buffering does not work they use timer for a particular source of IP address. The IP of generated traffic can be easily identified and marked for time stamping. This time stamping policy is simply like debar a particular source of IP for some specified time limit. By setting a timer in the Visual Network Description (VND) scenario they can block a particular IP for some time interval. In this way other users can smoothly communicate with servers of SDN network and a specified IP will be put on hold for some time. After specified time limit has reached controller automatically allow all sources to transfer data. In this way if the problem resolved then we have nothing to worry if still a heavy traffic is coming it reconfirming about something is wrong there in network. In this situation it shall go for the third step of mechanism. Warning cum request packets: when none of the policy is working it replicates about the confirmation of attack. In third step it shall generate an Eco request packet to the specified source of IP to slow down the transfer rate of sending traffic. This warning message will be applied thrice for slowing down the rate of packet transfer. If at this stage the particular source of IP did not slow down the rate of sending traffic it is identified that a particular source of IP is an attacking point and some strict action must be taken against it. In the fourth step after applying the above policies to make sure about an attack is happening it simply trace back the source and block this IP by sending a command to the controller. After applying the mechanism developed for prevention of Infrastructure based DoS attack the link congestion is avoided at its early stage and servers are also secured by getting into deadlock. The Paper by Nayana Y, Mr.JustinGopinath and Girish.L (June 2015) tried to mitigate DDoS attack with SDN Controller. The SDN controller detects the DDoS attack by using threshold value and helps to remove DDoS attack in the network. In their project they are providing security challenges in DDoS attacks mitigation in SDN environment. The paper uses for mitigation the output of developed DDoS detection method. Selecting threshold value is necessary to help the DDoS detection to make a good decision in identifying the attacker at the fast attack especially. This value is helpful for differentiating normal activity and abnormal activity in network traffic. If we select inaccurate threshold value will cause an excessive false alarm especially if the value is too high or too low. Detecting the intrusion as quickly as possible is very important to provide the security. The paper sets threshold value, if the traffic i.e., number of packets crosses the threshold value the controller will take action and mitigate the attack immediately. Random policy used for balancing the load. Balances the traffic to backend servers depend on the source address and source port on every incoming packet.

IV. METHODOLOGY AND MATERIALS

In this chapter, we will describe how our work is organized. We will give a detailed description of the materials and methods used to get our results. This section gives you the experimental setup of our study and the tools that we used.

41 Methods

4.1.1 Entropy Variation of Destination IP address

Entropy is a measure of uncertainty or randomness associated with a random variable which in this case is the destination address. A higher randomness will result in higher entropy. The entropy value lies in the range of $[0, \log_2 m]$ where m is the number of destination IP addresses. The entropy value is at its minimum when all the traffic is heading to the same destination. On the other hand the entropy value is at its maximum when the traffic is equally distributed to all the possible destinations (Maryam kia, 2015). In the normal network state we expect that the traffic spreads out to many different hosts. During a DDoS attack the number of packets destined for a specific host or a small set of hosts rises suddenly and the entropy decreases. A decrease in the entropy is an alarm for the network to watch out for a possible attack.
It is vital in SDN networks to have a fast detection method and to detect the attacks at its early stages. SDN networks are more vulnerable against the DDoS attacks than the traditional networks. If the detection time takes too long the attacker could break the switches or the controller and so an early detection is extremely important. For an early detection the window should not be too large. On the other hand a small window will add to the computational overhead. As proposed by Maryam Kia in this thesis we will use the window size of fifty to balance the two concerns. A module is added to the pox controller for the entropy calculations. For every fifty packets that arrive in the controller the relative frequencies are calculated. The calculated entropy is compared against the threshold value. If the calculated entropy is less than the threshold for five consecutive entropy calculations an attack is suspected and further analysis will be performed to determine if the attack is real.

4.1.2 Attack Mitigation

If a switch is reported as being under attack the algorithm should try to mitigate the attack. A number of possible attack mitigation approaches include installing flows in the attack paths to drop packets until the attack is stopped or blocking the incoming ports where the attack traffic is arriving at.

Although all these methods will mitigate the attack and will buy time for the network operators to find the attack sources before the break down of the controller or switches the adoptions of these methods will also affect the legitimate traffic as much as the attack traffic and the network services will become unavailable or respond slowly to legitimate traffic. The controller is usually designed with high capacities and therefore it will not crash very easily. The switches on the other hand have limited resources and are not very robust against attacks. When an attack is underway the flow table on the switches will be filled with a large number of short flows that will eventually break the switch. In the proposed mitigation algorithm the flow idle timer will be changed from the default value to the mitigated value to prevent the breakdown of the switches. The mitigated value is smaller than the default value; consequently, the short malicious flows will time out quickly and are deleted from the switch flow tables. The legitimate traffic flows on the other hand are expected to have a longer connection with a larger number of packets. If the mitigated value is chosen correctly it will not affect the legitimate flow entries significantly but will clear out the malicious flows quickly.

4.2 Materials

4.2.1 Pox Controller

There are few famous controllers available. The one that will be used in this experiment is POX. Pox is widely used for experiments; it is fast, lightweight and designed as a platform so a custom controller can be built on top of it. It is an improved version of its predecessor NOX, and both are running on Python. POX works on Linux, Mac OS and windows, and it has topology discovery. For completeness, three other controllers should be mentioned. Floodlight is another widely used controller that is open-source and written in Java. One advantage of Floodlight is facilitating application interface to the controller so they can run alongside it. Beacon is another Java-based controller that is open-source and has high throughput and low latency. Open Daylight controller is the most recent addition to Openflow controllers. It meant to be a common platform for all SDN users.

Table .1. Comparison of different SDN Controllers

	POX	**RYU**	**Floodlight**	**Open Day Light**
Language	Python	Python	Java	Java
OpenFlow Support	v1.0	v1.0 ,1.2, 1.3	v1.0	v1.0
OpenSource	Yes	Yes	Yes	Yes
GUI	Yes	Yes	Yes	Yes
REST API	No	Yes	Web GUI	Yes
Platform Support	Linux Mac Windows	Linux	Linux	Linux Mac Windows

POX is an open source development platform for Python-based software-defined networking (SDN) control applications, such as OpenFlow SDN controllers. POX, which enables rapid development and prototyping, is becoming more commonly used than NOX, a sister project.

4.2.2 Pox Configuration

The way POX initiate components, is to use the component name as the argument. To use a pre-built L2 learning switch component, we can use 'forwarding.l2_learning' as the first argument.

A POX controller consists of three parts:

- Listener
- Control logic
- Messenger

4.2.3 Mininet

Mininet is the network emulator that will be used for this experiment. It is the standard network emulation tool that can be used for SDN. Mininet can prototype a network on a laptop or PC by using kernel namespace feature. Network namespace provides individual processes with their own network interfaces, ARP tables and routing tables. Mininet makes use of this feature of the kernel. It uses process-based virtualization to run switches and hosts on the kernel. Large networks with different topologies can be emulated and tested. Mininet is a network emulator which creates a network of virtual hosts, switches, controllers, and links. Mininet hosts run standard Linux network software, and its switches support OpenFlow for highly flexible custom routing and Software-Defined Networking.

4.2.4 Scapy

Packet generation is done by Scapy. It is a very powerful tool for packet generating, scanning, sniffing, attacking and packet forging. Scapy is used here to generate UDP packets and spoof the source IP address of the packets. The code for generating random source IP addresses and host IP addresses is in Python. The function "randrange" is used which is inheriting the function "random". This function produces a uniform random float in the range [0.0, 1.0). This number shows a long period of random number generation which will result in generating random numbers with uniform distribution. These numbers are joined together to form spoofed source IP addresses. Two other parameters that we set in Scapy are: type of packets and interval of packet generation. UDP packets are used for both attack and normal traffic. The interval was set to suit the test case. For instance, for an attack with 25% rate, normal traffic interval is 0.1 seconds and attack traffic is 0.025. This gave us windows with 25% of packets destined to one host.

V. SIMULATION RESULTS AND ANALYSIS

5.1 Simulation Scenarios

The simulation and testing of the proposed method for DDoS attack detection is explained through the following sections. The algorithm is implemented on the python based pox controller in the Mininet virtualized network environment. Scapy scripts are used to generate the legitimate and attack traffics on the network hosts during the simulation.

5.1.1 Network Setup

The simulation is done on a Toshiba laptop with a dual core processor, 1.7 GHz of power, 2GB of ram, and 10/100/100/1000Mbitps network interface. The operating system is Linux Ubuntu 14.04 and Mininet version 2.0.0 was

run native on Linux. Mininet 2.0.0 supports Open flow version 1.0. Using Mininet, a tree-type network of depth one with 3 controllers, 3 switches and 32 hosts was created. Figure 8 shows the network. OpenFlow is used for the network switches.

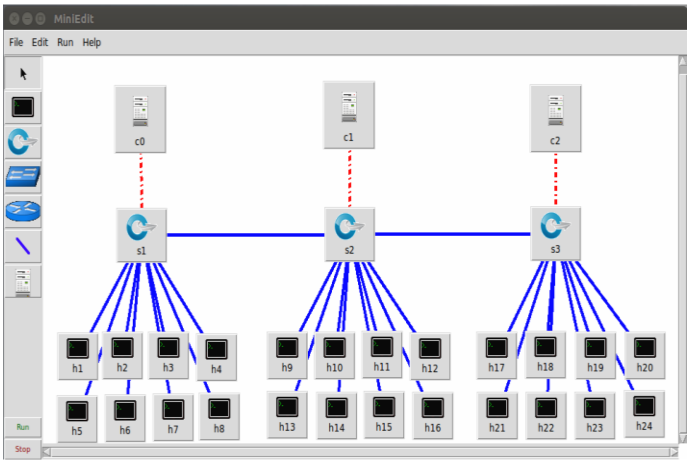

Figure.3. Experiment Network with 5 switches and 32 hosts

5.1.2 Choosing Threshold Value

We use the same threshold value that is suggested from the base paper. The detection mechanism in our solution dictates that if the entropy is lower than the threshold, and it persists for five windows in a row, an attack is in progress. The experiments cover an attack to one host and a subnet of four hosts. To compare different rates of incoming packets, we controlled the rate of normal and attack traffic to increase and decrease the intensity of DDoS on the controller. Equation 1 is used for showing the rate *R* of incoming attack packets to normal traffic attacks. Where *Pa* and *Pn* are the number of attack packets and normal traffic packets respectively.

$$R = \frac{P_i}{P_n} \text{ x } 100\% \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{ (1)}$$

Table 2 shows the threshold and compares it to normal traffic values. The threshold is set to 1.31. To get this value the following was done:

Table .2. Threshold Value Calculation

	Normal Traffic	25% Rate Attack
Mean	1.47	1.3
Standard Deviation	0.009	0.012
Confidence interval	± 0.0035	± 0.0047
Confidence interval Max	1.4735	1.3047
Confidence interval Min	1.4665	1.2953
Difference of Normal traffic min and Attack traffic Max.	0.1618	
Threshold	1.31	

When the network is running live, these values can be modified and this is one of the advantages of central control in SDN.

5.2 Results

The experiment covers four cases of normal and an attack traffic run. Normal traffic is run on all switches with randomly generated packets going to all hosts. Attack traffic is run from two hosts. Attacks were run manually (i.e. a script was run after one third of the length of our simulation). In Mininet, IP addresses for all hosts are assigned incrementally from 10.0.0.1 onward. For one host attack, we randomly chose a host in a switch to send attack packets to another host while all other hosts and switches are running normal traffic. Table 3 shows the attack traffic profile. All the traffic packets will be UDP, destination port is 80 and type of attack is DDoS. In Openflow, by default, only the packet header is sent to the controller so no payload was added to the generated packets.

Table .3. Attack Traffic Profile

Protocol	**Port**	**Payload**	**Types of Attack**
UDP	80	None	DDoS

DDoS attacks reach a much higher intensity. Attacks, often, they generate a traffic that is few times higher than the normal traffic. In order to show that the system functions as intended, we've considered different scenarios. The scenarios we've produced the results by simulating the system are as follows.

- Under normal traffic flow
- During an attack on a host
- As the entropy Changes between the Normal and Attack traffic
- After early detection and prevention work is done

5.2.1 Normal Traffic Packet Generation

In the normal traffic flow there is nothing the controller is expected to do except calculating the entropy value to determine whether the incoming packets are of attack types or normal. As it is depicted in the figure below the entropy value is stable and there is no sudden change that would make the controller to suspect the existence of an attack and therefore controller takes no action.

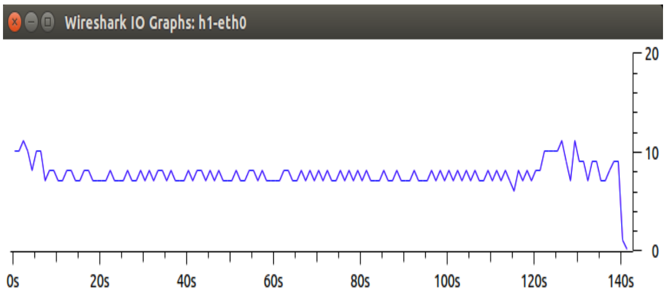

Figure.4. Normal Traffic Change that is generated from host 1

5.2.2 Attack on the host

In a condition where there is an attack detected in the network the entropy value does not stay as stable as normal environment. In fact, the entropy value goes down below the threshold value which in this case is equal to 1. In this scenario we have generated the attack traffic from two hosts making one of the other hosts a target. And as the figure 5 clearly shows, the sudden change in the packet flow occurs as expected. This implies the increase of the number packets

which in its turn results in the dropping down of the entropy value.

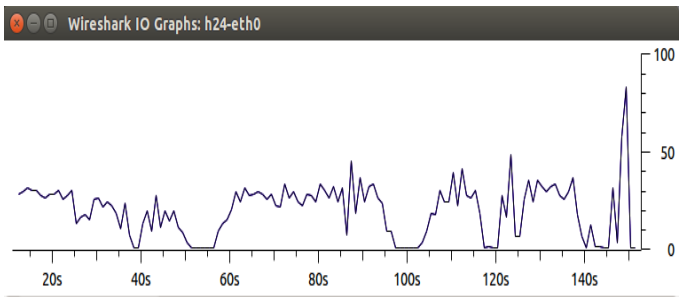

Figure .5. DDoS Attack results of Host 24

5.2.3 Entropy Change between the Normal and Attack traffic

Figure 6 shows the difference between the results of the normal traffic and the attack traffic entropy value changes. So, as you can see from the figure when the normal traffic is generated its entropy value will change and shows a reading of an increasing value from the given threshold. Whereas, in the case where the controller notices an attack traffic the entropy value show a decrease and drops down below the threshold value.

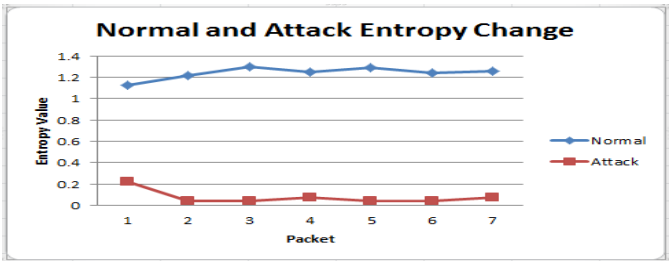

Figure.6. Entropy Change with Normal traffic and DDoS attack traffic

5.2.4 Detection and Prevention of DDoS Attack

The controller waits for a certain threshold value of packet size before it takes any action on the environment it suspected to have an attack. Therefore, the first 250 packets of attack traffic pass by until the detection is confirmed. Then it's after this confirmation that the controller takes the measure to prevent the attack from continuing to happen. Figure demonstrates the change in packet size as the attack happens and the point where the controller gets its defense on.

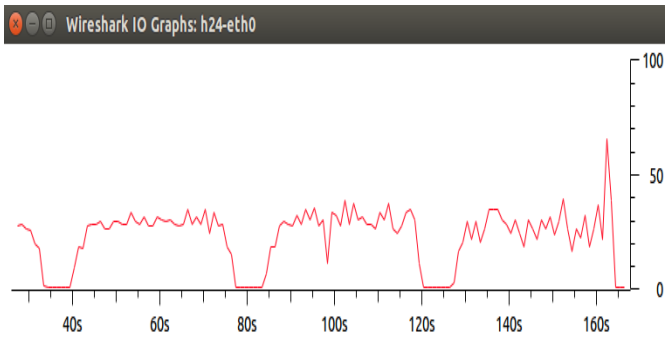

Figure.7. Detection and Prevention of DDoS Attack

VI. CONCLUSION AND FUTURE WORK

6.1 Conclusion

The academia and the industry in the networking field has come to the realization that the future of SDN relies on distributed architectures, because centralized architectures do not fulfill the needs of efficiency, scalability, and availability. In this paper, we've tried to provide a comprehensive solution of SDN multi-controller architectures by explaining their characteristics and presenting different scenarios of the implementation. In this thesis work, the effort to implement a multi-controller based SDN solution to detect a DDoS attack on its early stage is accomplished successfully. The environment is implemented using logically centralized pox controller. This brings many solutions to the shortcomings of the single controller based environment. The most important of the contributions made by this research are reduction of a single point failure, increased flexibility to enable the scalability of the network and the capability of backup functionality with redundancy. This research succeeded in detecting DDoS attack early in a multi-controller structure. The mitigation of the attack is analyzed and prevention mechanism is developed to avoid the DDoS attack in its initial stage before harming our network. The mechanism is implemented using Mininet network emulator. The Proposed method is based on the Entropy variation of destination IP address and can detect the attack within the first 250 packets of malicious traffic attacking a particular host in the SDN.

6.2 Future Work

Network researchers and designers will have to deal with many problems that distributed architectures face to enhance a multi-controller network, like developing an efficient communication process, creating an adequate network design, or integrating new applications into the northbound interface that support multiple controllers. As it is stated above the solution we've proposed used logically centralized controller. In the future, it would be better to implement logically distributed controllers in order to get the best out of multi-controller structure giving it the ability to have both the advantages of single and multiple controller based systems. One of the main features in a multi-controller structure is that the controllers have the responsibility of watching over the whole environment and this obviously creates an additional load that a controller in a single controller based environment would not. Therefore, it would enhance the performance of multi-controller based structures even more if a solution to find a way to balance the load between controllers could be achieved.

VIII. REFERENCES

[1]. C. Douligeris and A. Mitrokotsa. (2004) "DDOS attacks and defense mechanisms: classification and state-of-the-art," Computer Networks 44.

[2].Jiafu Wan, Di li & Athanasios Vasilakos. (January 2016). "Security in Software-Defined Networking threats and Countermeasures".

[3]. L. R. Prete, C. M. Schweitzer, A. A. Shinoda and R. L. Santos de Oliveira. (2014) "Simulation in an SDN network scenario using the POX Controller," *IEEE*.

[4]. Martin Vizv´ary. (January 2015). "Mitigation of DDoS Attacks in Software Defined Networks".

[5]. Maryam Kia. (2015). "Early Detection and Mitigation of DDoS Attacks in Software Defined Networks".

[6] .M. B. C. Dillon. (2014). "OpenFlow DDoS Mitigation," Amsterdam.

[7]. Nayana Y, Mr. Justin Gopinath and Girish. L. (June 2015). "DDoS Mitigation using Software Defined Network". International journal of Engineering trends and Technology.

[8]. Nhu-Ngoc Dao, Junho Park, Minho Park, and Sungrae Cho. (2013). “A Feasible Method to combat against DDoS Attack in SDN Network”.

[9]. Open Networking Foundation. (September 2012). "Openflow-spec-v1.3.0".

[10]. Open Networking Foundation. (April 2012). " Software-Defined Networking: The New Norm for Networks".

[11]. Open Networking foundation. (2017). “https:// www. opennetworking.org/sdn-resources/sdn-definition”. Retrieved on March 2017.

[12]. Othmane Blial, Mouad Ben Mamoud Ben Mamoun, and Redouane Benaini. (2016) “An Overview on SDN Architectures with Multiple Controllers,” Journal of Computer Networks and Communications.

[13]. Poojja & Manu Sood. (2015). “SDN and Mininet: Some Basic Concepts”.

[14]. Sandeep Singh, R.A. Khan & Alka Agrawal. (2015). “Prevention Mechanism for Infrastructure based Denial-of-Service attack over Software Defined Network” International Conference on Computing, Communication and Automation (ICCCA).

[15]. Seyed Mohammad Mousavi & Marc St-Hilaire. (2015). “Early Detection of DDoS Attacks against SDN Controllers” International Conference on Computing, Networking and Communications and Information Security Symposium.

[16]. Shishupal Kumar, Nidhi Lal & Vijay Kumar Chaurasiya. (February 2016). “Performance analysis of Software Defined Network with Multiple Controllers”.

[17]. Sukhveer Kaur, japinder Singh and Navtej Singh. (December 2015). “Network Programmability Using POX Controller”.

[18]. William Stallings. (March 2013). http://www. cisco.com/c/ en/us/about/press/internet-protocol-journal/back-issues/table-contents-59/161-sdn.html.

[19]. Wolfgang Braun and Michael Menth. (2014). “Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices”.

[20]. Xenon Foukas, Mahesha k. Marina & Kimon Kontovasilis. “Software Defined Networking concepts”.