# DDoS Attack Detection in SDN using Entropy

Gitanjali Chandwani Manocha, gc.manocha@thapar.edu

Ankit, aankit1_be18@thapar.edu

**ABSTRACT** By detaching the data plane from the control plane, the software-defined networking architecture framework simplifies the job of web administrators. This facilitates web setup by providing an interface for programming for application development related to administration, safety, and dependability, among other things. More access over the total network is provided by the centralized analytical controller, which also has the complete transparency in the network. The SDN and its benefits exposes the network to the threats and the outcomes of these security concerns are very dangerous as compared to the traditional networks, where the responsibility of protection from the attacks is provided by the network devices itself and they also limit the extent of the threat [7].

We are exploring various threats that can be attacked on the SDN at its different layers. We are also exploring different security techniques that are in use to mitigate such attacks [10]. To prevent DDoS Attacks, a possible solution can be using sample entropy method. A Denial-of-Service attack which utilizes multiple distributed attack sources is called Distributed Denial-of-Service or DDoS attack [11].

Every network in the system has an entropy. Fall in entropy is caused due to rise in randomness. To avoid the above-mentioned attacks, this work emphasizes the use of centralized SDN controls to detect attacks and introduces an effective, simple and quick fix that is also lightweight in number of utilized resources. Our project work also precisely demonstrates how such attacks can empty the SDN controller resources and along with that also proposes a solution for detecting DDoS attacks using the sample entropy differences of IP address of the destination. If the value drops below the threshold, the specific port in the switch is blocked and the port is brought down. The entropy approach can detect DDoS attacks within the first 500 packets of assault traffic.

## 1 INTRODUCTION

### 1.1 Project Overview

Why is SDN security important? We know SDN network has separated data plane and control plane unlike the conventional network architecture [6]. This network architecture can be implemented as cloud service. Cloud services are exploding and big organizations and enterprise network administrators are migrating to the SDN-based network implementations. These virtual technologies provide predictability, manageability and good quality of service. In parallel with added advantage, importance of security provision in this centralized managed network has become one of the main concerns. the single centralized virtual server running as controller, which basically install and manages the flows in the data plane network agents through overflow communication protocol which makes the controller a primary victim for the attacker.

### 1.2 Need Analysis

The use of OpenFlow for attacker, makes SDN controller the main victim because of the following reasons:

1) Implementation of security regulations in the OpenFlow communication is not yet defined properly and the developers of products are using their own methods [9].
2) The programmable aspect of SDN also puts it in a situation of greater risk due to a large number of dangerous attacks.
3) The interface of Software Defined Network can be a targeted of DDoS and other similar attacks.
4) Errors in SDN configuration may have more consequential outcomes than in conventional networks.
5) Establishment of trust is the most important task.

### 1.3 Problem Statement

There are a lot of security threats in the Software Defined Networks. We will be dealing with Distributed Denial-Of-Service in this project work as a result of these security concerns [10]. When a large number of packets are forwarded to a network device with an intent to either stop the service or decrease the performance then such attacks are termed as Distributed Denial-of-service attacks. In DDoS attacks, a large number of packets are sent to a host or a group of hosts in a network. If the source addresses of the incoming packets are spoofed, which they usually are, the switch will not find a match and has to forward the packet to the controller. The collection of legitimate and the DDoS spoofed packets can bind the resources of the controller into continuous processing that exhausts them. This will make the controller unreachable for the newly arrived legitimate packets and may bring the controller down causing the loss of the SDN architecture. Even if there is a backup controller, it has to face the same challenge.

This kind of attacks can be detected at an early stage by monitoring few hundreds of packets based on the entropy changes [11]. The early detection of DDOS attack prevents the controller going down. The term “early” is subjected to tolerance level and traffic being handled by the controller. If detection happens early say first few hundreds of packets, then, the impact of flooding of malicious packets can be controlled significantly. The early detection mechanism must be of light weight and should have a high response time. The high response time saves the controller in the period of attack to regain the control by terminating the DDOS attack.

### 1.4 Proposed Solution

Sample entropy, a general method for DDoS detection in SDN, is conducted by collecting the flow statistics or traffic features from the switches, and calculating the entropy measure randomness in the packets that are coming to a network [16]. The higher the randomness, the higher is the entropy and vice versa. By setting a threshold, if the entropy passes it or below it, depending on the scheme, an attack is detected.

An Openflow controller is connected to a network. We then observe the entropy of the traffic related to the controller under normal and attack conditions. We used the POX controller for this project that runs on Python [2]. The network emulator used is Mininet for creation of network topology [3]. Packet generation is done with the help of Scapy. Scapy is used for generation of UDP packets, sniffing, scanning, forging of packet and attacking. Scapy also spoof the source IP address of the packets [4].

### 1.5 Motivation to the Problem

The main goal 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 [12]. 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. For the purpose of this research, all packets will have spoofed IP addresses. This way, the switches do not have a match and all the packets are sent to the controller.

To accomplish this goal, a fast and effective method is needed that works within the controller. Collecting statistics is one of the functions of the controller [14]. In this study, this attribute is used for adding another set of statistics collections to the controller; destination IP addresses. In our 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.

### 1.6 Research Objective

This project work is carried out to find the weakest points in Software Defined Networks in DDoS attack scenario. This study of SDN led to SDN controller and when the Distributed Denial-of-Service attack occurs, we found the weakest part is the controller. Many different techniques in detection were studied so those can be used for the protection of the controller in the controller. Obstacles on the solution and the manner of its implementation is created by the SDN structure.

The barriers are:

1) Controller's finite number of resources.
2) The urgency of attack detection before a large number of corrupted packets reach to the controller and situation is out of control.

The main objectives are:

1) Analysing the behaviour of DDoS attacks on SDN environments.
2) Analysing the network by means of sample entropy.
3) When a DDoS threat happens, determine the weakest link in the SDN.
4) Implementing a mechanism for early detection of DDoS Attacks in SDN.

### 1.7 Prospective Outcomes

1) Understand the concept behind SDN, Entropy, and type of DDoS attacks in SDN environment.
2) Detect attacks on network hosts using entropy as a detection approach.
3) Implement an efficient algorithm to mitigate the attack using OpenFlow entries.

## 2 LITERATURE SURVEY

### 2.1 Literature Review

**Table 1.** This table shows information about different research papers I read, and all necessary information related to its publication date, name of a research paper, etc

| S. No. | Title (Publication,Year) | Authors | Objective of the Paper | Key Findings |
|---|---|---|---|---|
| **1.** | Computing in Communication Networks (Elsevier,2021) | Frank H.P. Fitzek, Fabrizio Granelli, Patrick Seeling | to provide knowledge about communication devices, virtualization and technology | Software-Defined Network, Network function virtualization, Mininet |
| **2.** | Detecting DDoS Attack on SDN Due to Vulnerabilities in OpenFlow (IEEE,2020) | Sarwan Ali, Maria Khalid Alvi, Safi Faizullah, Muhammad Asad Khan, Abdullah Alshanqiti, Imdadullah Khan | To asses security flaws in OpenFlow | Software-Defined Network, OpenFlow |
| **3.** | DDoS Attack Identification and Defense Using SDN Based on Machine Learning Method (IEEE,2019) | Lingfeng Yang, Hui Zhao | Machine Learning based approach for detecting DDoS attacks | Software-Defined Network, Machine Learning Techniques |
| **4.** | Mitigation of DDoS attack instigated by compromised switches on SDN controller by analyzing the flow rule request traffic (IJET,2018) | Sanjeetha R, Shikhar Srivastava, Rishab Pokharna, Syed Shafiq, Dr. Anita Kanavalli | To show DDoS attack using Flow-entry tables of switches and way to mitigate it. | Using Flow-entries as vulnerability for DDoS attack. |
| **5.** | A survey of distributed denial-of-service attack, prevention, and mitigation techniques | Tasnuva Mahjabin, Yang Xiao, Guang Sun, Wangdong Jiang | Glance at DDoS attack and methods for its prevention and mitigation. Also covers the possible limitation | Different types of attack, their motives, strategies and mechanisms. Different prevention |

| | | | | |
|---|---|---|---|---|
| | (International Journal of Distributed Sensor Networks,2017) | | and challenges of existing research. | and mitigation strategies. |
| **6.** | Early Detection of DDoS Attacks against SDN Controllers (IEEE,2015) | Seyed Mohammad Mousavi, Marc St-Hilaire | In an SDN controller, identify a DDoS attack in its early phases. To identify the threat, implement an efficient and lightweight method. | Software-Defined Network, DDoS attack, Entropy method to detect DDoS, Mininet(tool). |
| **7.** | An Entropy-Based Distributed DDoS Detection Mechanism in Software-Defined Networking (IEEE,2015) | Rui Wang, Zhiping Jia, Lei Ju | To design a flow statistics process in switch to identify DDoS attacks using entropy | Entropy to identify DDoS, Open vSwitch |
| **8.** | Combining OpenFlow and sFlow for an effective and scalable anomaly Detection and mitigation mechanism on SDN environments (Elsevier,2013) | K. Giotis, C. Argyropoulos, G. Androulidakis ⇑ , D. Kalogeras, V. Maglari | To present a combined mechanism for reduced data collection, entropy-based anomaly detection, and network-wide anomaly detection utilising OF. | Flow counters data collection for periodic analysis, sFlow, different parameters for entropy-based classification. |
| **9.** | A New Multi Classifier System using Entropy-based Features in DDoS Attack Detection (IEEE,2018) | Abigail Koay, Aaron Chen, Ian Welch, Winston K.G. Seah | To present a multi-classifier method for detecting DDoS based on a collection of several entropy-based features and a machine learning classifier to improve accuracy. | Entropy, multiple classifier system, machine learning to classify traffic. |
| **10.** | Early DoS/DDoS Detection Method using Short-term Statistics (IEEE,2010) | Shunsuke Oshima, Takuo Nakashima, Toshinori Sueyoshi | To assess the Denial-of-Service detection approach by short-term entropy. | Short-term Entropy method to detect DDoS |
| **11.** | An Advanced Entropy-Based DDoS Detection Scheme (IEEE,2010) | Jie Zhang, Zheng Qin, Lu Ou, Pei Jiang, JianRong Liu, Alex X. Liu | To establish the best threshold for reliably detecting DDOS attacks precisely using Entropy Method | Entropy to detect DDoS, Threshold calculation |
| **12.** | Lightweight DDoS flooding attack detection using NOX (IEEE,2010) | Rodrigo Braga, Edjard Mota, Alexandre Passito | Based on traffic flow properties, a lightweight approach for detecting DDoS attacks | NOX platform, traffic flow features, |

### 2.2 Research Gaps

I explored a variety of references to implement the detection of DDoS attacks via Sample Entropy. Previous research on rapid DDoS detection has shown some cure against DDoS threats on SDN, although there are gaps in this work. For example, a ML based approach is employed to study the network's behaviour and, determine whether or not an attack is underway on its basis [8]. This strategy is commonly employed in networks that are not SDN based. When utilised in SDN based networks, it takes the same technique as before, ignoring the influence of the Distributed Denial-of-Service on the controller. Before it can be utilised in the network, the solution must run concurrently with SDN and then even be trained for several sessions. Another difficulty is that SDN can regularly alter network at any time. This indicates that the solution based on Self-Organizing Maps must retrain in order to provide greater protection.

SNORT is a DDoS intrusion detection system in and of itself, and combining it with Software Defined Networking is, once again, employing a tool that is not based upon SDN for SDN [10]. This approach brings a stream processing component at the pinnacle of the original controller, which are renamed sub controllers.

To determine the shortest path, using Openflow, to Network Intrusion Detection System (NIDS) devices, the method necessitates installation of NIDS devices with network lines to monitor traffic for suspicious activities [9].

1) Controller in proposed architecture is vulnerable to DDoS attack.
2) An attacker can forge SRC_IP or DST_IP to generate the attack.
3) Generating huge no. of packet with forged IP address will flood the controller and lead to Denial-of-Service.
4) The proposed architecture has only detection module with no method to mitigate the attack.

### 2.3 Key Research Findings for Existing Literature

In none of above solutions the controller has been the centre of attention. As the controller drives the SDN and the operating system, it is the most crucial component in the architecture [16]. In our solution, we propose using the controller itself to detect any attack. The entropy method reduces the complexity by assigning the detection task to each controller. This work is mainly focuses on detecting DDoS threats that are endangering the controller which also protect the hosts.

## 3 FLOW CHART

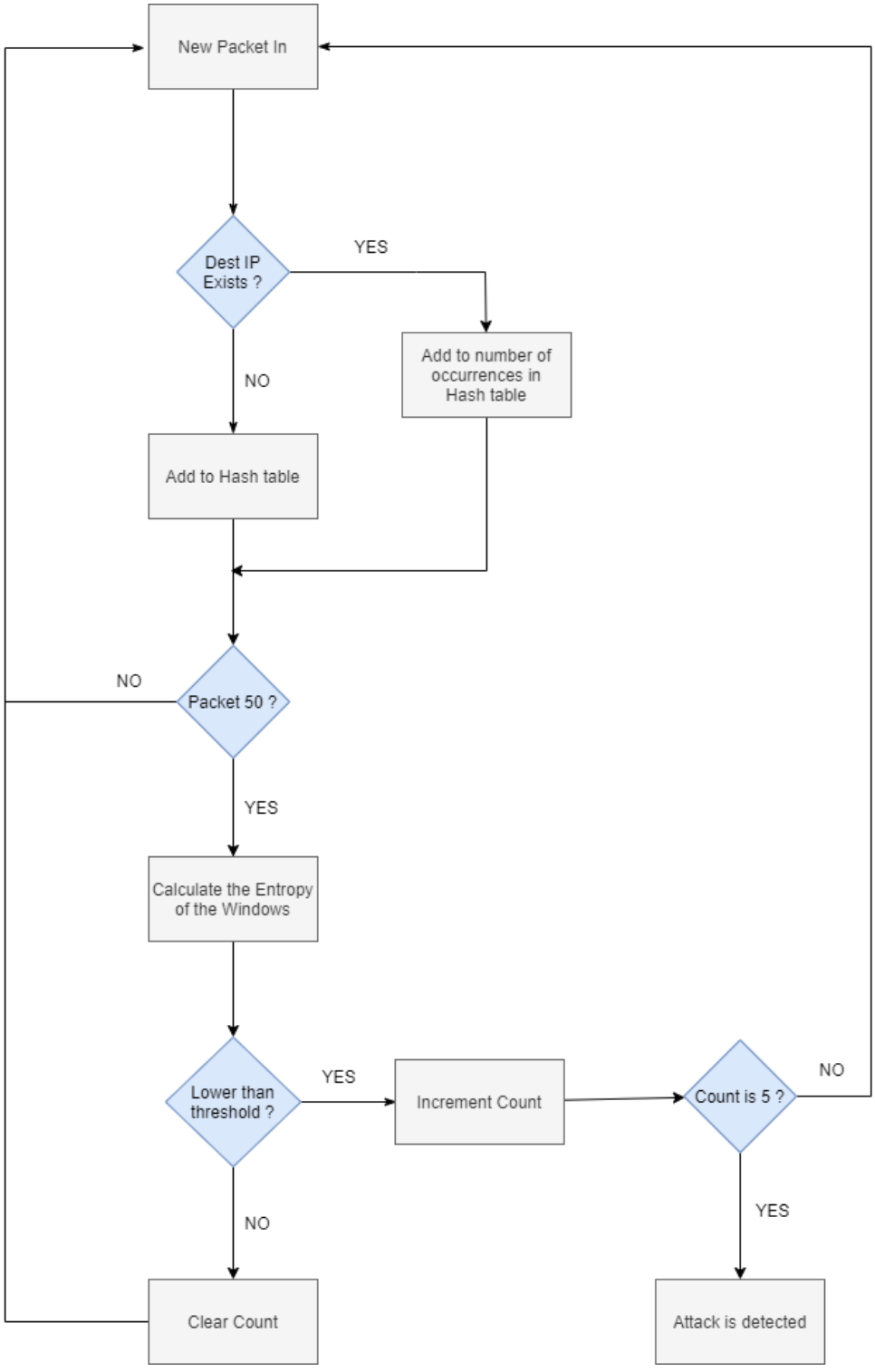


**Fig 1.** DDoS Attack Detection using Entropy

# 4 PROJECT DESIGN AND DESCRIPTION

## 4.1 Design Methodology

Module I: Creating Model

1) Implement SDN.
2) Creating a topology for simulation.
3) Creating scripts to generate traffic.

Module II: Simulation

1) Simulating the given topology with NORMAL traffic.
2) Simulating the given topology with ATTACK traffic.
3) Analyzing and comparing the recorded traffic.

Module III: Observation & Results

1) Observation of sample entropy
2) Detection of DDoS attacks using sample entropy.

## 4.2 System Architecture

1) SDN controller (virtual machine with configuration of 2.8GHz, 8GB memory, Fast Ethernet interface, and Ubuntu 20.04 as OS).
2) Topology (virtual machine with configuration of 2.20GHz, 4GB memory, and Fast Ethernet interface, and Ubuntu 20.04 as OS).
3) Mininet (to create topology of 10 devices, 3 switches and 1 controller).
4) Open Virtual Switch (OVS) (Mininet-compatible software switch that operates on both hardware and software)
5) POX Controller (networking software platform written in Python)
6) Python Scapy (to generate normal and attack traffic).

**POX Controller:** Selecting a controller is the first part of our experiment. There are many controllers available based upon different programming languages. POX and Ryu are both based upon Python. POX was employed in this experiment because it is commonly used for research, experimentation, and demonstrations. POX is designed as a platform for the development of bespoke controllers. It is quick, lightweight, and reasonably easy to comprehend compared to other controllers. POX's predecessor NOX runs on C++, but it based upon Python [17]. POX run under Linux, Mac OS and windows, but some features are not available on all platforms. Linux is the most feature rich platform.

**Mininet:** Mininet is the network emulator utilised in this experiment [3]. It can be used to develop, share, and experiment with Software Defined Networking. Rapid prototyping of Software-Defined Networks is possible using Mininet. Using Mininet, complex topologies may be evaluated without the need to link up an actual network. Mininet emulation topologies can be utilised in a genuine operational network. Mininet has an extensible Python API for network construction, exploration and experimentation. Mininet is distributed under a permissive BSD Open-Source licence and is constantly improved, actively developed and supported by a networking and SDN enthusiast community.

**Packet generation:** Scapy is used for packet generation [4]. It is a Python application that may be used to generate network packets, scan them, sniff them, attack them, deconstruct them, and forge them. Scapy can counterfeit or decode packets from a variety of protocols, transmit them over the network, collect them, and compare requests and responses. This functionality enables the development of tools that can explore, scan, or attack systems. Scapy produces UDP packets and spoofs the packets' originating IP address. Python is also utilised as a programming language in the POX controller. Python is used for writing the script for producing a random set of source and host IP addresses. The "randrange" function returns a randomly generated float inside the range [0.0, 1.0]. The float that was generated has a precision of 53 bits and a duration of 219937-1. Spoofed source IP addresses is formed by joining these numbers together.

### 4.3 Emulation

In SDN, Sample Entropy is a method for detecting DDoS attacks [11]. DDoS detection with entropy requires two components: a window size and a threshold. The size of the window is determined by either the time period or the number of packets. Within this timeframe, entropy is measured to assess the uncertainty in the incoming packets. A threshold is required to detect an assault [16]. An attack is detected if the computed entropy reaches or falls below a threshold, depending on the technique. The ability of entropy to assess unpredictability in a network is the primary rationale for using it. The more the entropy, the higher the randomness, and vice versa.

Let W be a data set with n entries and x be an event inside the set. The likelihood of x occurring in W is thus shown in equation 1. To compute the entropy, referred to as H in equation 2, we add the probabilities of all items in the set.

$$W = \{ x1, x2, x3, \ldots, xn \}$$

$$pi = xi / n \qquad \text{---(1)}$$

$$H = - pi \log pi \sum n\, i = 1 \qquad \text{---(2)}$$

If all elements have equal probabilities, the entropy will be at its maximum. If one element appears more frequently than the other, the sample entropy decreases. Window size is the size of W. Windows are formed by dividing continuous stream of incoming data into equal sets. In the window, every element and their occurrence are taken count of.

Each 50 Packet-In signals are parsed for the destination IP address in Algorithm, and thus the entropy of the list is calculated. After then, the computed entropy would be checked to a threshold. It is deemed an attack if the computed entropy is below the threshold and lasts for at least five successive entropy cycles. Detection inside Five entropy cycles is 250 packets in the intrusion, providing the network with an early warning of an assault. We tested with values ranging from one to 5 successive cycles, with five yielding the lowest false negative and positive rates for timely identification.

**Implementation:**

1) Install Python.
   $ sudo apt-get install python
2) Install Scapy.
   $ sudo apt-get install python-scapy
3) Install mininet along with pox controller:
   i. mininet installation: http://mininet.org/download/
   ii. pox controller Clone the repository: http://github.com/noxrepo/pox
4) Write the following files in *mininet/custom*
   i. traffic.py.
   ii. attack.py
5) Write the following files in *pox/pox/forwarding*
   i. detectionUsingEntropy.py
   ii. l3_detectionEntropy.py
6) Run the pox controller:
   $ cd pox
   $ python ./pox.py forwarding.l3_detectionEntropy.py
7) Create a mininet topology by entering the following command in another terminal:
   $ sudo mn --switch ovsk --topo tree,depth=2,fanout=3 --controller=remote,ip=127.0.0.1,port=6633
8) Open xterm for the following hosts:
   mininet>xterm h1 h2 h3 h9
9) To start traffic, run the following commands in h1's xterm window:
   $ cd ../mininet/custom
   $ python traffic.py –s 2 –e 65
10) The pox controller now generates a collection of entropy values. The typical traffic threshold entropy is the lowest value obtained.
11) To initiate the attack, repeat step (8) on h1 and simultaneously type the following instructions in the h2 and h3 xterm windows:

$ cd ../mininet/custom $ python attack.py 10.0.0.64

Take note of the entropy numbers inside the pox controller. The threat can be identified when these entropy values fall below the threshold entropy value for regular traffic.

### 4.4 Result Analysis

For analysis, a network of tree topology of 4 switches & 9 nodes directly connected to switch were set up using Mininet.

```
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(s1, s2) (s1, s3) (s1, s4) (s2, h1) (s2, h2) (s2, h3) (s3, h4) (s3, h5) (s3, h6) (s4, h7) (s4, h8) (s4, h9)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Starting controller
c0
*** Starting 4 switches
s1 s2 s3 s4 ...
*** Starting CLI:
```

**Fig 2.** Creation of custom topology in Mininet

The diagram above depicts the establishment of a tree topology with a controller, four switches, and nine nodes using mininet. Mininet configures the topology and runs its command line interface when the *sudo mn* command is issued.

```
mininet> nodes
available nodes are:
c0 h1 h2 h3 h4 h5 h6 h7 h8 h9 s1 s2 s3 s4
```

**Fig 3.** Available nodes in the topology

The *nodes* command is used to show the list of available nodes. This command returns the presence of a controller, nine hosts (host h1, host h2, …… host h9), and four switches (switch s1, switch s2, switch s3 and switch s4).

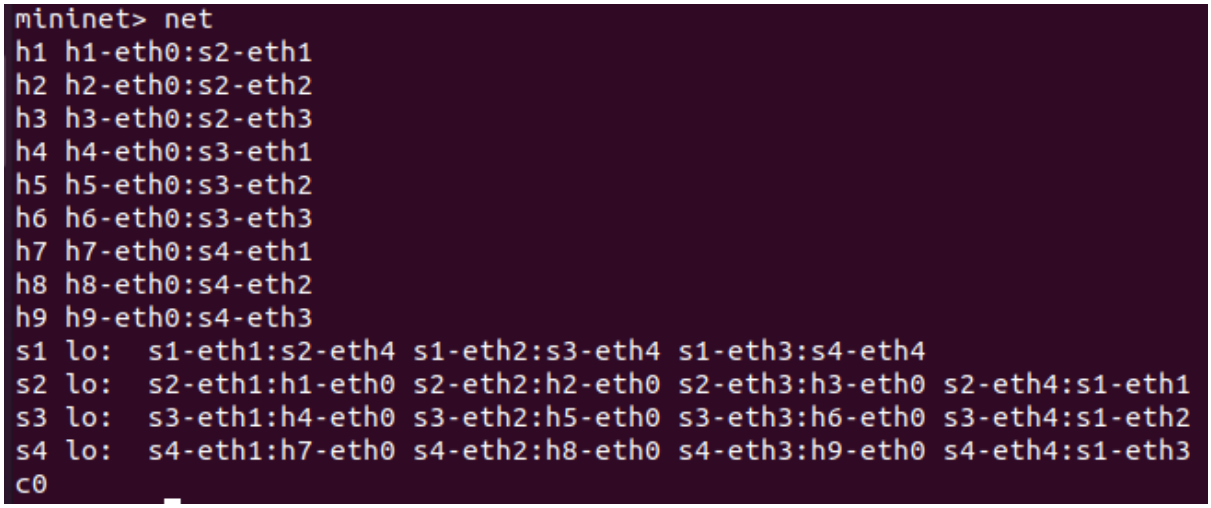


**Fig 4.** Links between different nodes in the topology

The result of *net* command demonstrates that hosts are linked to the switches on their network interfaces through their network interfaces. A loopback interface lo is available on switches. Controller c0 is the network's brain, and it possesses global knowledge of the network. A controller advises network switches on how to forward/drop packets.

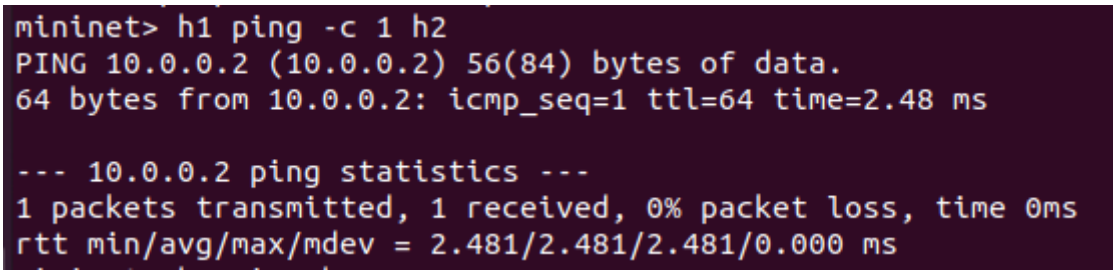


**Fig 5.** Connectivity verification by pinging from h1 to h2

The command *ping* can be used to test connection between them. The ping command communicates with a distant computer by delivering Internet Control Message Protocol (ICMP) Echo Request messages and waits for a response. The information supplied includes the number of responses returned and the time it takes for them to return.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9
h2 -> h1 h3 h4 h5 h6 h7 h8 h9
h3 -> h1 h2 h4 h5 h6 h7 h8 h9
h4 -> h1 h2 h3 h5 h6 h7 h8 h9
h5 -> h1 h2 h3 h4 h6 h7 h8 h9
h6 -> h1 h2 h3 h4 h5 h7 h8 h9
h7 -> h1 h2 h3 h4 h5 h6 h8 h9
h8 -> h1 h2 h3 h4 h5 h6 h7 h9
h9 -> h1 h2 h3 h4 h5 h6 h7 h8
*** Results: 0% dropped (72/72 received)
```

**Fig 6.** Connectivity verification among all hosts

The command *pingall* can be used to test connection between all the hosts in the network. Internet Control Message Protocol (ICMP) Echo Request messages are sent and waits for a response. The information supplied includes the number of responses sent and returned.

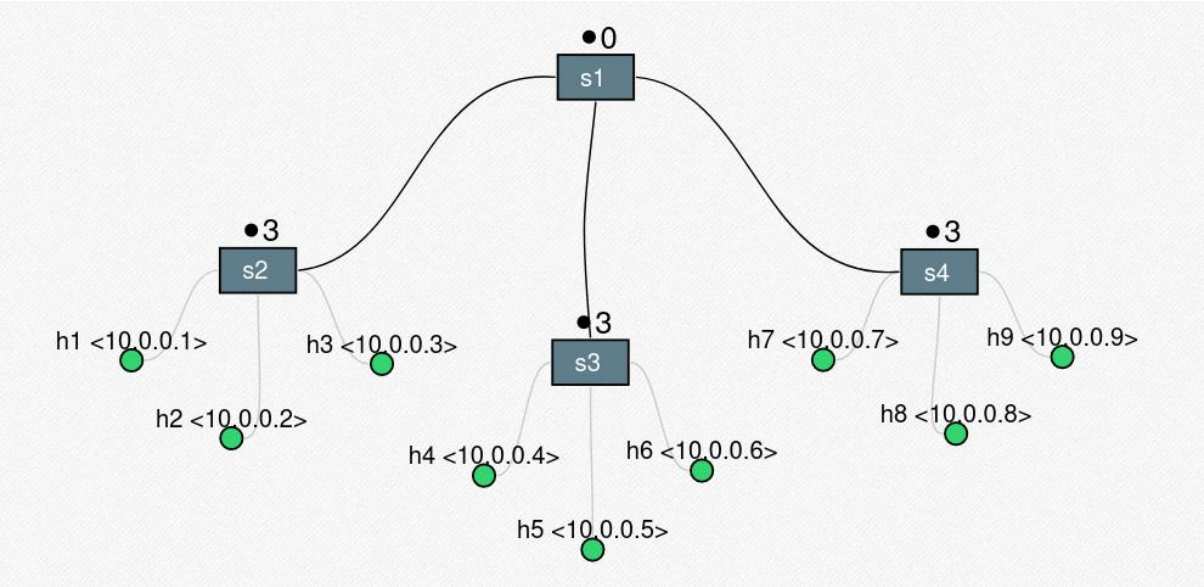


**Fig 7.** Topology

A network of tree topology of 4 switches & 9 nodes directly connected to switch were set up using Mininet.

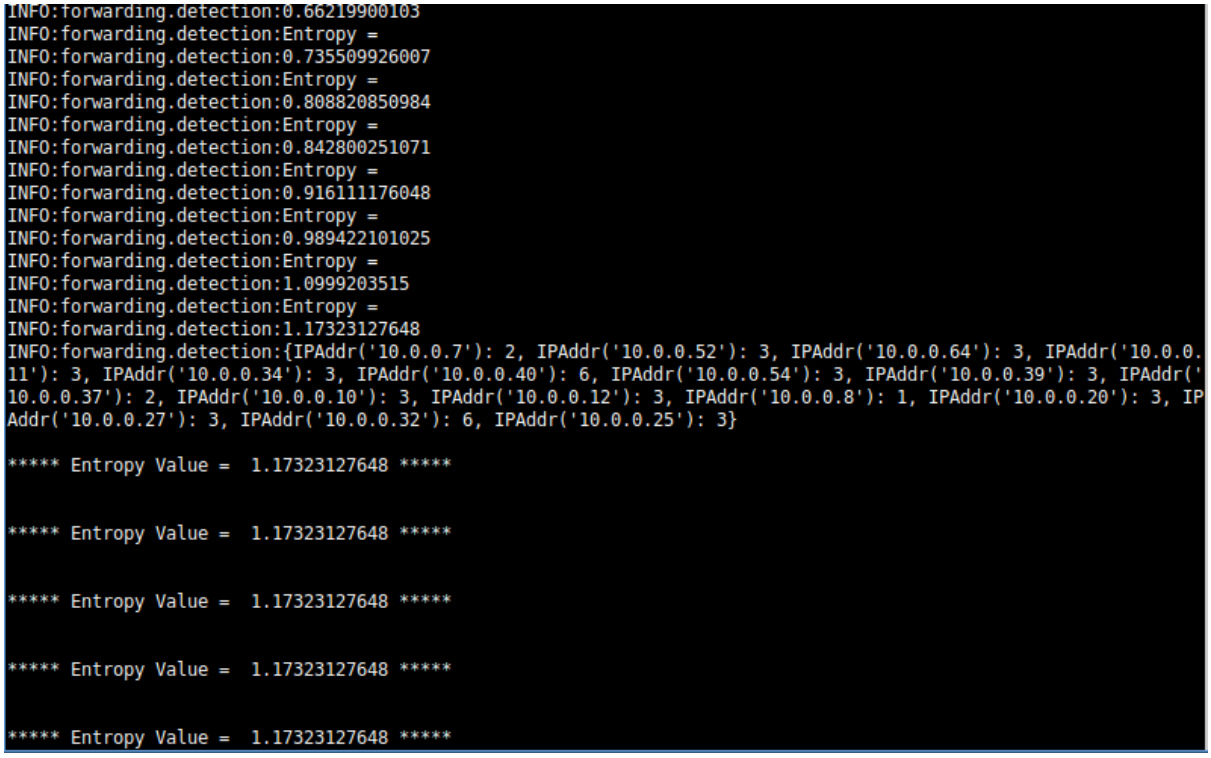


**Fig 8.** Entropy in the network before the attack

The pox controller now generates a collection of entropy values. The typical traffic threshold entropy is the lowest value obtained.

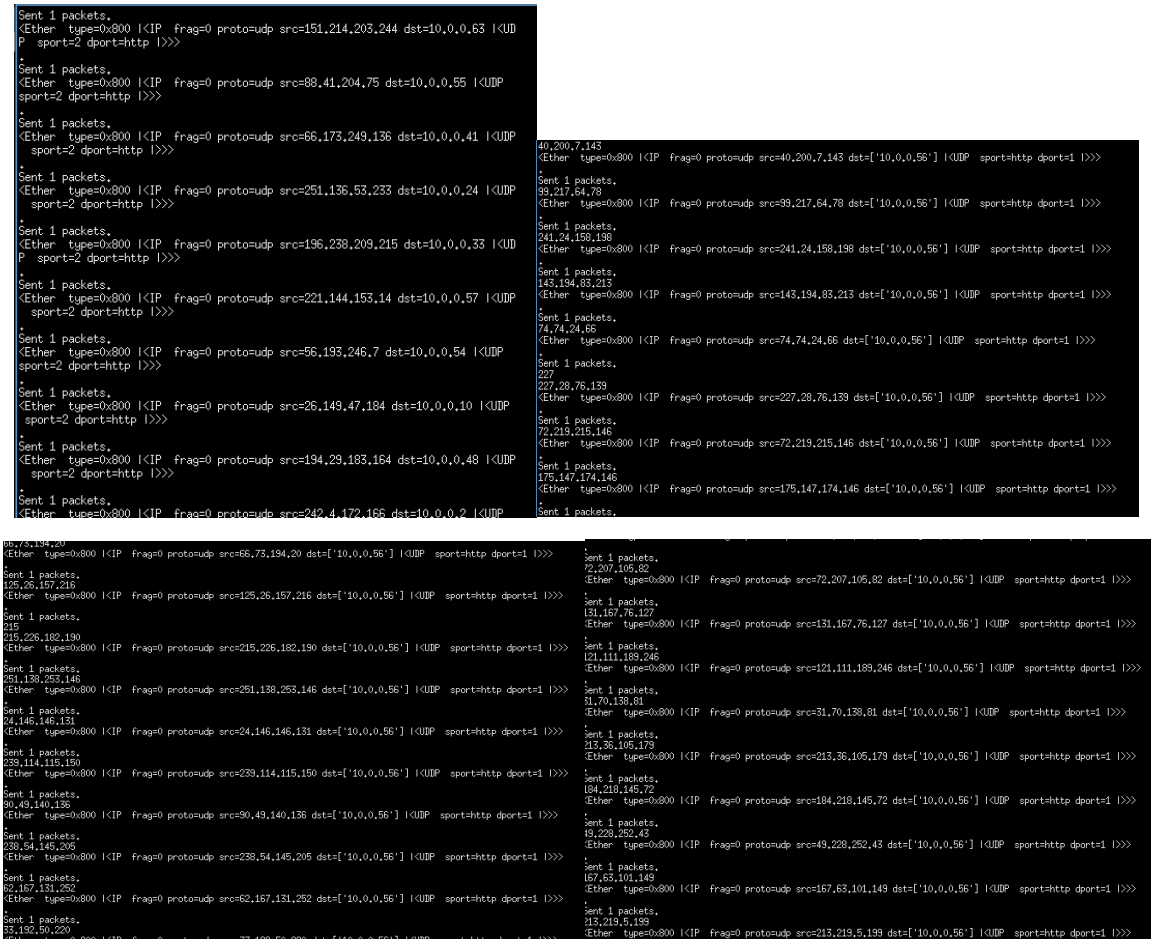


**Fig 9.** Launching traffic in the network

Traffic is launched by running the set of commands in h1's xterm windows. To initiate the attack, the above step on h1 and simultaneously typing the same instructions in the h2 and h3 xterm windows:

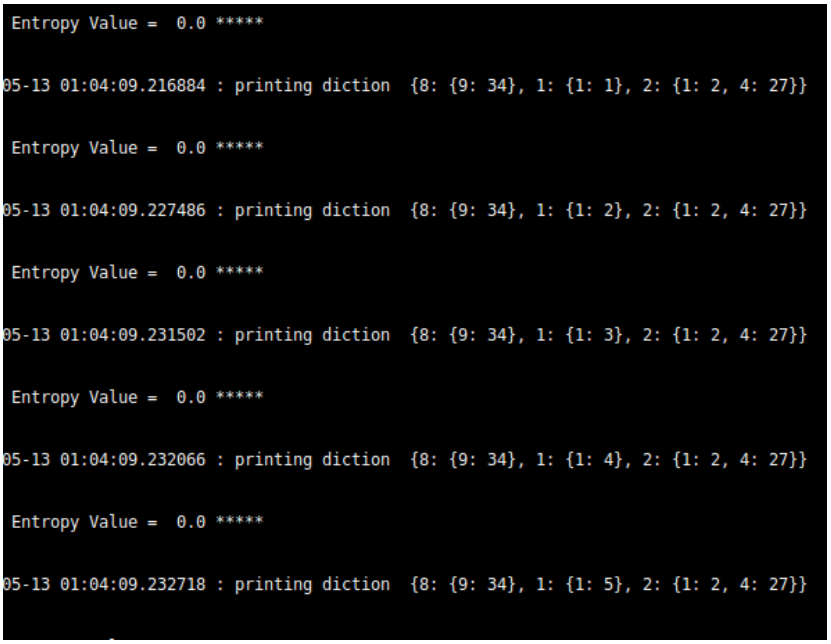


**Fig 10.** Entropy in the network after the attack

Observing the entropy values in the POX controller. The value decreases below the threshold value (which is equal to 0.5 here) for normal traffic which is the least value obtained of entropy in 1st iteration. Thus, we can detect the attack within the first 250 packets of malicious type of traffic attacking a host in the SDN network. After the hosts stop sending attack packets, the switches are started again by the POX controller after shutting them down during the attack.

# 5 CONCLUSION AND FUTURE SCOPE

## 5.1 Conclusion

Some of the key ideas of SDN are the introduction of dynamic programmability in forwarding devices through open southbound interfaces, the decoupling of the control and data plane, and the global view of the network by logical centralization of the ‘‘network brain’’. While data plane elements became dumb, highly efficient and programmable packet forwarding devices, the control plane elements, are now represented by a single entity, the controller. Protecting the controller by detecting DDoS attacks was the centre of this research. The challenge in detecting any threats to the controller is early detection.

We have implemented sample entropy method to detect DDoS attacks successfully. we quantified the early detection to the first 250 packets of traffic as minimum and 500 packets maximum. This solution is not only efficient in detection, it has minimal code addition to the controller program and does not increase CPU load in either normal or attack condition. There are different methods for detecting attacks and each method is used differently. In this project, we focused on a solution that works particularly well for SDN, based on its specifications, points of strength and limitations. We made use of the fact that SDN specification dictates the forwarding of new packets to the controller. We took into account the abilities of the controller and its broad view of the whole network and used that for adding entropy statistics collection. Finally, understanding the importance of keeping the controller connected to the network at all times, we came up with a solution to detect any threat at its very beginning.

### 5.2 Future Scope

1) The drawback of sample entropy method is when the complete network is being attacked by Distributed Denial-of-Service. So, our next task is to come up with a new mechanism other than sample entropy measuring to detect DDoS attack in SDN environments effectively.
2) For this project work, we worked on Detection of DDoS Attacks in one controller network, next we can work on multi-controller SDN structure to detect Distributed Denial-of-Service Attacks.
3) Having addressed the problem of detection of attack on the network, Mitigation of the Distributed Denial-of-Service attack in the SDN environment will be the next task in our project. It can be implemented with an efficient algorithm using OpenFlow entries.

## 6 REFERENCES


1. Openflow. [Online]. https://opennetworking.org/sdn-resources/customer-case-studies/openflow/
2. NOXREPO. [Online]. http://www.noxrepo.org/
3. Mininet. [Online]. http://mininet.org/
4. Scapy. [Online]. http://www.secdev.org/projects/scapy/
5. Python Standar Library. (2014, Jan) Python Douments. [Online]. https://docs.python.org/2/library/random.html
6. Frank H.P. Fitzek, Fabrizio Granelli, Patrick Seeling (2021). Computing in Communication Networks
7. Ali S., Khalid Alvi, Faizullah M., Asad Khan M., Alshanqiti A., Khan I. (2020). Detecting DDoS Attack on SDN Due to Vulnerabilities in OpenFlow (pp. 10-31) IEEE.
8. Yang L., Zhao H. (2019). DDoS Attack Identification and Defense Using SDN Based on Machine Learning Method (pp. 17-36) IEEE.
9. Raja, Sanjeetha & Srivastava, Shikhar & Pokharna, Rishab & Shafiq, Syed & Kanavalli, Dr. (2018). Mitigation of DDoS attack instigated by compromised switches on SDN controller by analyzing the flow rule request traffic. International Journal of Engineering & Technology. 7. 46. 10.14419/ijet.v7i2.6.10065.
10. Koay, A., Chen, A., Welch, I., & Seah, W. K. (2018, January). A new multi classifier system using entropy-based features in DDoS attack detection. In 2018 International Conference on Information Networking (ICOIN) (pp. 162-167). IEEE.
11. Mahjabin, T., Xiao, Y., Sun, G., & Jiang, W. (2017). A survey of distributed denial-of-service attack, prevention, and mitigation techniques. International Journal of Distributed Sensor Networks, 13(12), 1550147717741463.
12. Mousavi, S. M., & St-Hilaire, M. (2015, February). Early detection of DDoS attacks against SDN controllers. In 2015 International Conference on Computing, Networking and Communications (ICNC) (pp. 77-81). IEEE
13. Wang, R., Jia, Z., & Ju, L. (2015, August). An entropy-based distributed DDoS detection mechanism in software-defined networking. In 2015 IEEE Trustcom/BigDataSE/ISPA (Vol. 1, pp. 310-317). IEEE.
14. Giotis, K., Argyropoulos, C., Androulidakis, G., Kalogeras, D., & Maglaris, V. (2014). Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments. Computer Networks, 62, 122-136.
15. T. Nakashima, T. Sueyoshi S. Oshima (2010). Early DoS/DDoS Detection Method using Short-term Statistics, in International Conference on Complex, Intelligent and Software Intensive Systems, (pp. 168-173).
16. Z. Qin, L. Ou, J. Liu, A. X. Liu J. Zhang (2010). An Advanced Entropy-Based DDoS Detection Scheme, in International Conference on Information, Networking and Automation, (pp. 67-71).
17. E. Mota, A. Passito R. Braga (2010). Lightwight DDoS flooding attack detection using NOX/Openflow, in IEEE 35th conference on Local Computer Networks, (pp. 408-415).