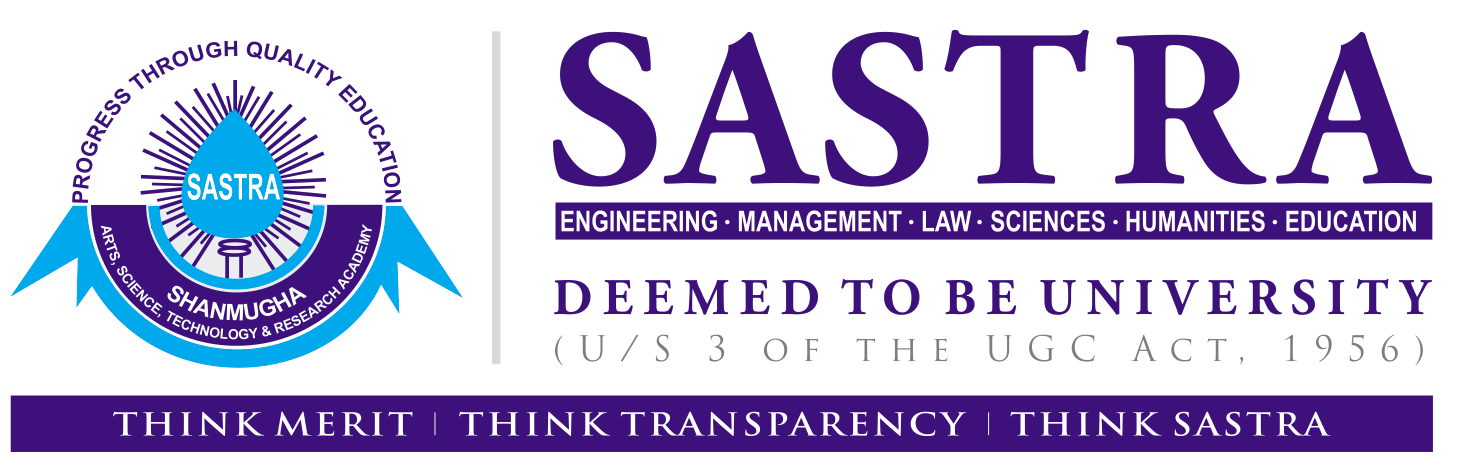
Project Report

Detection of DDoS­­­ Attack in SDN environment



122003096

Jaisaiarun P Srinivasan

3rd B.Tech Computer Science

Declaration:

This is to certify that the project entitled

**“Detection of DDoS­­­ Attack in SDN environment”**

is a bonafide record of the work carried out

by

JAISAIARUN P SRINIVASAN (122003096)

students of third year B. Tech., School of Computer Science Engineering of the SASTRA DEEMED TO BE UNIVERSITY, Thirumalaisamudiram, Thanjavur – 613 401 during the academic year 2020 – 2021.

*NAME OF THE INTERNAL GUIDE*:

Mrs. N. Hemavathi

SIGNATURE:

Mini-project-1 viva-voce held on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

EXAMINER-1 EXAMINER-11

Contents:

1. Abstract
2. Introduction
3. Existing System
4. Limitation
5. Proposed system
6. Implementation
7. Result and discussion
8. Conclusion and further enhancement
9. Reference

Abstract

Software defined networking (SDN) is a new emerging networking architecture, it fixes the needs of shortage of needs the traditional network does not support such as dynamic and scalable computing and storage needs for more computing environment. However, SDN also faces security problems, it is vulnerable to DDos attack. Distributed Denial of service (DDos) is a well know attack, but unlike in traditional networks, DDos not only damages the targeted server but also the SDN network by taking advantage of the devices limited buffer space and disturbing the resource allocation. Therefore it is very important have a rea-time detection system to identify the attack in the early stages.

Introduction

In this project we explore security measures to detect DDos attack on a SDN environment at real-time and prevent them before the system is damaged.

* *Software defined networking:*

Software-defined network (SDN) is an emerging architecture that decouples the network control plane and data plane, which simplifies network management and also makes the network programmable by making decisions about how to send traffics from the bottom component forwarding traffics. This feature of SDN is facilitating. SDN is an emerging architecture that is dynamic, manageable, cost-effective, and adaptable, makes it ideal for the high-bandwidth, dynamic nature of today’s applications. The centralized controller of SDN has the real-time feedback control capability, and open interfaces which offer modular plug-in features. The centralized controller provides an abstract network view, defining tasks by APIs and greater programmability of the network. It can integrate security devices within the network topology, which can lead to increase in accuracy, detecting security incidents and simplify management.

* *Distributed Denial-of-Service:*

Distributed Denial-of-Service (DDoS) attacks are widely used to run out of the target’s network bandwidth or process resources. DDoS attacks are not only effective in traditional networks but also active in SDN environments. And due to the mechanism of SDN environments, the switches need to hold all uninstructed packets in memory buffer before it gets respond from the controller, the size of memory buffer is very less which is exploited thus paving ay to the new type of DDoS attacks which easily flood this space and lead to packets drop.

* *SDN matching process:*

Packet matching process in SDN has limited storage spaces and process resources. These resources could be easily run out when DDoS attacks occur in SDN. Flow tables in a switch contains rules or flow entry describing how the switch should deal with the incoming traffic. In the OpenFlow switch specification it is stated that the pipeline processing is able to look up at different flow tables in a switch, we consider that there is one flow table in a switch for better understanding.

When processed by a flow table, the packet is matched against the flow entries of the flow table to select a flow entry. If a flow entry is found, the instruction set included in that flow entry is executed. If a packet does not match a flow entry in a flow table, this is a table miss. A table-miss flow entry in the flow table may specify how to process unmatched packets: Options include dropping them, passing them to another table or sending them to the controller over the control channel. The packets which are not dropped are stored in a temporary buffer in the switch waiting for instructions from the controller. When the buffer are is filled and it has no space for a new packet, existing packets are dropped by either the FIFO (first in first out) or the LIFO (last in first out) policy depending on the switch.

When the controller receives the messages it searches for a match in its flow tables, if a match is found the flow table is sent to all switches to install a new flow entry, if no match is found then the controller sends a message to the switches to find the target host, if there is no match the controller has to wait till the timeout.

* *The new type of DDos attack on SDN:*

The new type of DDoS attack is different from traditional DDoS attack, the destination of packet is randomly chosen. This attack is not aimed at one fixed target server, but the SDN network system. Therefore will be no server detecting the attack, therefore no server will alarm the attack, therefore harder to be detected and reported.

The buffer stores all the packets awaiting the controller’s instructions, until an instruction is received from the controller the packet is not dropped or the packet is dropped automatically after a time out. Only the first packet of a new flow is kept in the buffer, therefore if the flow has occurred before, its first packet is not buffered. With the new type of DDoS since the destination IP address is chosen at random a packet from each flow is buffered and wait for the controllers instructions.

The packets have different origin and destination, so a new table flow in needed for every packet .Since there are no pattern between the packets, it is very hard to classify table flows, which means it will cost a lot of flow table space. Furthermore, when each packet consumes a table flow, the table could be easily filled with attack flow.

*Impact of the attack*

1. *Impact on switches.*

Let’s assume that the DDos attack occurs throughout the full network and that the bots of the “botnet” controlled by the attacker are in connected to separate switch. When there is no matching flow table entry, all DDoS packets will be stored in the buffer area. The buffer area space is easily exhausted, and old or new packets must be dropped (depending on the policy) when a new uninstructed packet comes in. Also if the flow table is managed poorly by the controller, then each packet with different source and different destination address called OD (origin-destination) pair will need an individual entry in flow table, which exhausts the flow table space faster

1. *Impact on controller.*

During a DDos attack, huge number of uninstructed packets passing through different switches wait for the controller’s instruction, this leads to diminishing the controller’s process ability, which cause the latency of instruction and cause time out leading to packet loss, it may also lead to controller shutting down and thus making the network unable to work.

When traditional DDoS occurs, there is packet burst occurs in the network. However, for those packets there are two patterns that only contains in DDoS packets. One is there are lot of packets be sent from different sources to one destination, and the other is they start in short time. Unlike other occasion like a burst hot topic, the topic requires time for separation, therefore they usually have exponential growth rate. DDoS attack usually grows like a spark. Another different between those hot topics with DDoS is that one people usually view this topic once, but DDoS requires those bots continually access to destination to increase its effectiveness and strength. There will be two extra side effect on SDN environments.

Existing System

In traditional systems, a specific host is selected as a target in a DDoS attacks. The targeted host is bombarded with TCP packets, to flood the system bandwidth. Similarly in SDN the targeted hosts is flooded with TCP packets from other hosts in the network. Entropy is one detection mechanism used for detection traditional DDoS attack which is also used to SDN environment. A new type of attack DDoS that is exclusive to the SDN environments, where the network itself is targeted by exploiting the small buffer area of switches. In the new DDoS attack random source and destination address are used to generate a false flow entry which floods the buffer area. It also hijacks the controller’s computation by holding the controllers processing pipeline with timeout.

Limitations

A new mechanism for detecting DDos attack and the new type of DDoS attack must be identified, which must have faster detection time so that there are no permanent damage to the components in the SDN environments. One of the ways overcome the new DDoS attack in SDN environment is with higher specification device of switch and controller, that has larger space and faster process spend, allows SDN environment to handle more packets, but this can’t fix the problem. There are also other methods to amplify these side effect.

Proposed Scheme

In a SDN environment, the data collection, matrix calculations a d result comparison is done at the controller side .Considering the difference in bandwidth and computation capabilities of controller and switches, the controller has high chances of shutting down during a DDoS attack.

To handle the network traffic we partition the network into subnets with a switch on top to report to the controller. Therefore we use a tree topology to handle the traffic load to a controller. Therefore let consider a tree topology with depth 2 and number of nodes connected to a switch or fanout be 8, resulting in a network consisting of 64 hosts and 9 switches , so there are 64 origin node and 64 destination node and such that origin node and destination node are not same. For each OD-pair (O : D) we could represent that:

(O : D) ; O = 1, 2…64; D = 1, 2 … 64; such that O ≠ D

*Depth*: 2

*Fanout*: 8

*Number of hosts*: 64 (fanoutdepth)

*Number of switches*: 9 (fanout + 1)

We detect the DDoS attack by two methods:

***Sample Entropy:***

The most common, where statistics or traffic feature from the switches and calculate the randomness or entropy of the packets in the network. If the randomness of the packet is higher, the value of entropy is higher and vice versa. By setting a threshold value for entropy we can identify the attack.

Sample Entropy is a measure of calculate the degree of dispersal or concentration. Let’s assume that that the total amount of traffic be S, where N OD pairs exists, and ni stands for the traffic amount of OD-pair i. Therefore OD-pair i will occur ni times in this observation. So that

i

And sample entropy of this network is defined as following:

The range of Sample entropy *H(X)* ranges from *(0,)*. Where 0 indicates maximum concentration and indicates highest dispersion.

***Principal Component Analysis:***

A new method of detecting anomalies in traditional networks in principal component analysis (PCA). PCA transform data into new datasets called principal components. The principal components contain a property that points them in direction of maximum variation of energy left in the data.

In order to facilitate discussion in subsequent sections, we first introduce relevant notation. Let

* ***p*** denote the number of **OD flows in a network**.
* ***t*** denote the number of successive **time intervals of interest**
* where *t > p*
* ***X*** be a matrix of dimension ***t × p*** which denote **time series of all OD flows.**
* Each column ***Xi*** denote the time series of the **ith ­OD flow**
* Each row ***Xj***denote all the **OD flows at time j**.

OD flow consists of all traffic entering the network from a given point, and exiting the network through some other point. Each network ingress and egress point serves a distinct customer population1. Thus, each OD flow arises from the activity of a distinct user population. Calculating the principal components is equivalent to solving the symmetric eigenvalue problem for the matrix *XTX*.

The matrix *XTX* is a measure of the **covariance between flows**.

Each principal component ***vi*** is the ith **eigenvector** computed from the spectral decomposition of *XTX*:

*XTX vi = λivi* (i = 1,2… p )

Where ***λi*** is the **eigenvalue** corresponding to ***vi* eigenvector*.***

By convention, the eigenvectors have unit norm and the eigenvalues are arranged from large to small, so that *λ1 > λ2 \_>… >λp.*

Finding the first ***r*** non-negligible principal component, could approximate the original matrix. Detecting anomalies relies on the separation of *x* (Matrix X’s i-th row, a vector of all flows at i-th interval) into normal and anomalous components. And could separate x into:

x = x̂ + x̅

In which x̂ is modeled part and x̅ is the residual traffic part.

It can be shown that the eigenvector corresponding to the maximum energy of the residual by using the Rayleigh Quotient of *XTX*. We can write the k-th principal component vk as:

Thus, computing the set of all principal components, is equivalent to computing the eigenvectors of *XTX*. The principal component space can be used to examine the transformed data. The contribution of principal axis i as a function of time is given by Xvi, and can be normalized to unit length via dividing. Thus, we have each principal axis i,

*, (i=1,…,p )*

The ui are orthogonal by construction. The equation above shows that all the OD pairs, when weighed by vi, produce one dimension of the transformed data. ui captures the i-th strongest temporal trend common the all OD pairs, and the set of captures the time-varying trends common to the OD pairs, refer to them as the ***eigenflow*** of X. The set of principal components can be arranged in order as columns of a principal matrix V, which has size *p x p*.

Likewise, we can form the *t x p* matrix U in which column i is ui, that V, U and can be arranged to write each OD flow Xi as:

*, (i=1,…,p )*

The elements of are called the singular values, and

And therefore we have two methodologies to detect a DDoS attack.

Implementation

There are several SDN controllers such as OpenDaylight, Ryu, and POX. The SDN controller used in this project is POX, since it is written in python thus making it user friendly are understandable .The POX controller is connected with mininet emulator.

A custom topology is created using mininet which can generate normal traffic among nodes or implement an attack. And use POX controller to control the flow among nodes.traffic.py and attack.py are python scripts written to generate a normal traffic or launch an attack on a specific node respectively. Also python scripts are written in the POX controller to detect and block a DDoS attack using both entropy and PCA methodology.

*Prerequisites*

1. Install Python
2. Install mininet along with POX controller
   1. mininet installation : ​ <http://mininet.org/download/>
   2. pox controller : Clone the repository : ​ <http://github.com/noxrepo/pox>

*Python scripts used in mininet:*

1. traffic.py – to generate normal traffic
2. attack.py <target\_ip\_address> – to launch a DDoS attack on a target , that is by sending TCP packets to the target host
3. attackrand.py – to launch the new type of DDoS attack, TCP packets to with randomly generated destination address are sent to flood the controller.

*Python scripts used in POX:*

1. l3\_detectionEntropy.py – pox code to detect attack using Entropy
2. l3\_detectionPCA.py – pox code to detect attack using PCA
3. detectionUsingEntropy.py – code to calculate entropy
4. detectionUsingPCA.py – code to calculate PCA

The 4 scripts are written in the forwarding folder of pox. The forwarding function in pox is used to program each individual switches in the network to perform desirable tasks.

*forwarding.l3\_detectionEntropy*

This scripts calls the function is detectionUsingEntropy.py to calculate the entropy in run time. The arguments passed is destination IP address. The returned entropy value is checked with the threshold value to determine if the system in under DDoS attack. The entropy value is plotted against time duration. The threshold value is 1, which will signifying that the traffic is directed and therefore detect the DDoS attack.

*forwarding.l3\_detectionPCA*

This script calls the function in detectionUsingPCA.py to calculate the principal components, the distance of y from principal component and standard deviation value. The arguments passed is destination IP address. The returned values are plotted against time duration. If the distance of y from principal component is in the range of (-1,1), the system is under DDoS attack.

*Creating the test environment:*

* The POX controller must be run before the topology is created:
  + python ./pox.py forwarding.l3\_detectionEntropy

(or)

* python ./pox.py forwarding.l3\_ detectionPCA
* to create a network topology in mininet:
  + sudo mn --switch ovsk --topo tree,depth=2,fanout=8 --controller=remote,ip=127.0.0.1,port=6633

The topology created has 64 hosts and 9 switches, the port number of OpenFlow protocol is 6633

* Open xterm in mininet to launch traffic or attack

$ mininet > xterm h1

The following command opens the cmd window of h1

* run the following code in cmd of h1

$ mininet > python ./mininet/custom/traffic.py –s 2 –e 64

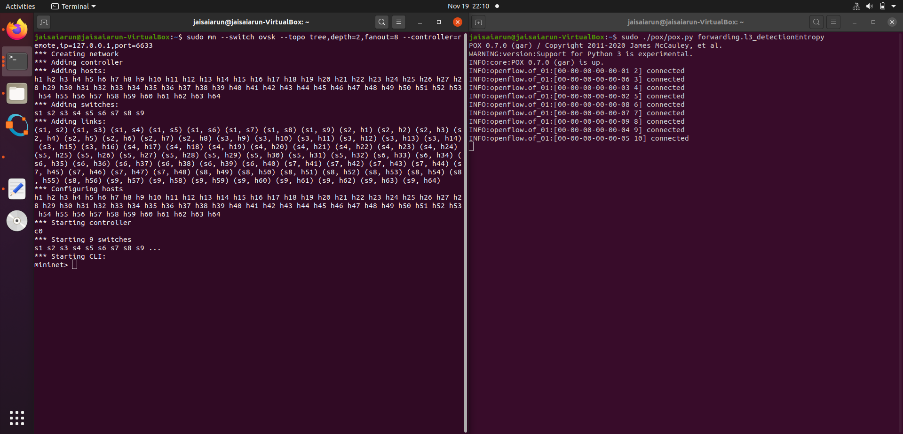
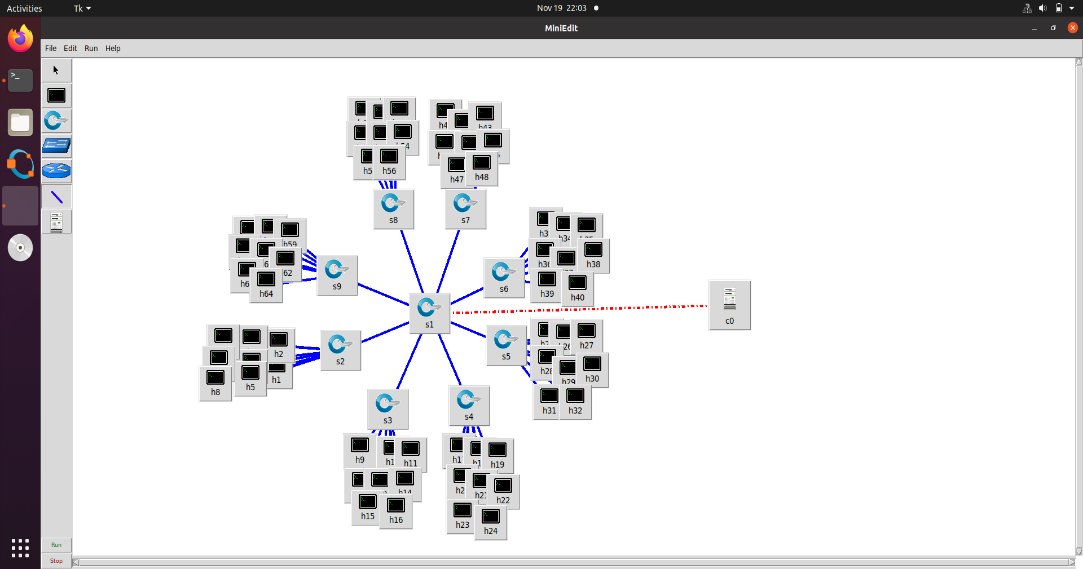
*To generate traffic to host whose ip address are 10.0.0.2 to 10.0.0.64*

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

*Launch DDoS attack on the host 10.0.0.64 ie. h64*

$ mininet > python ./mininet/custom/attackrand.py

*Launch the new type of DDoS attack*

**

*Establishing connection between POX controller in mininet*

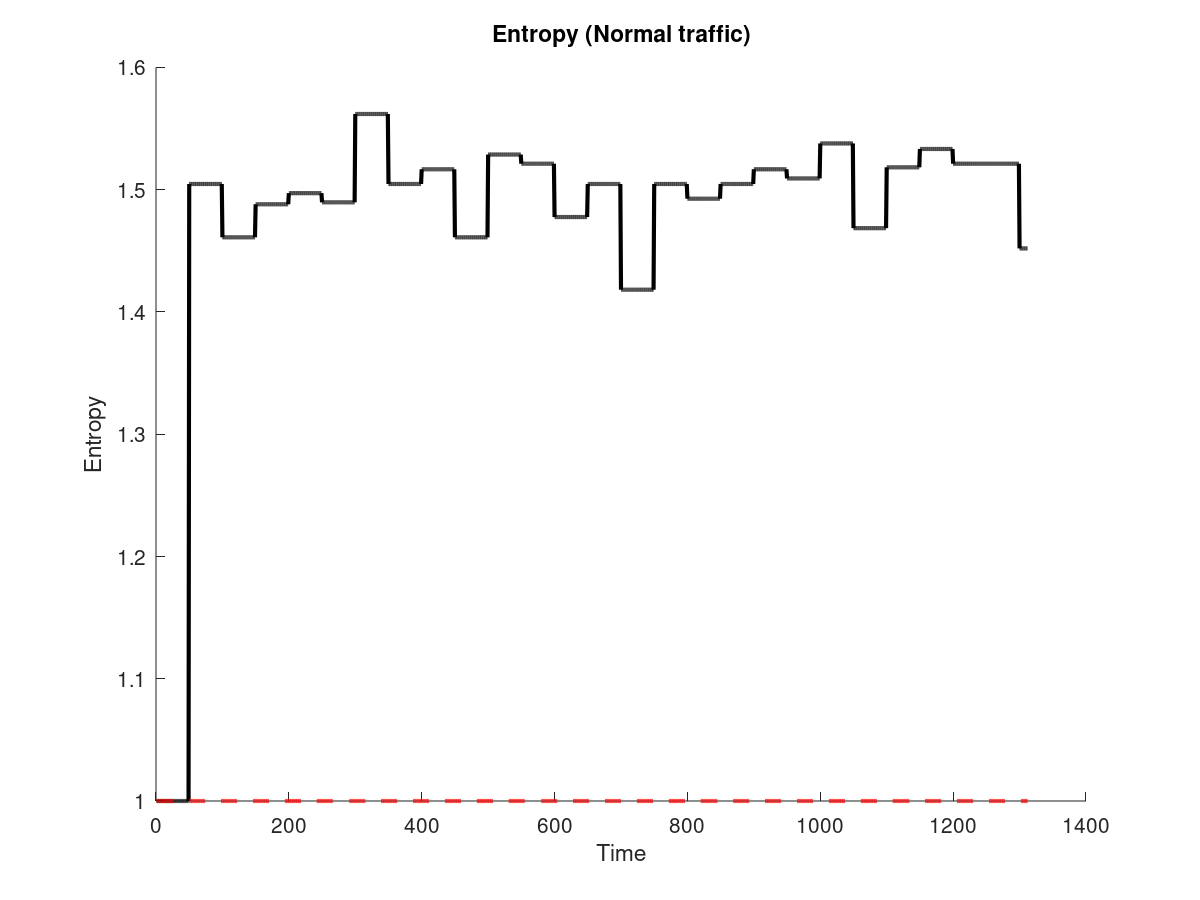
*Network diagram of topology*

Result and Discussion

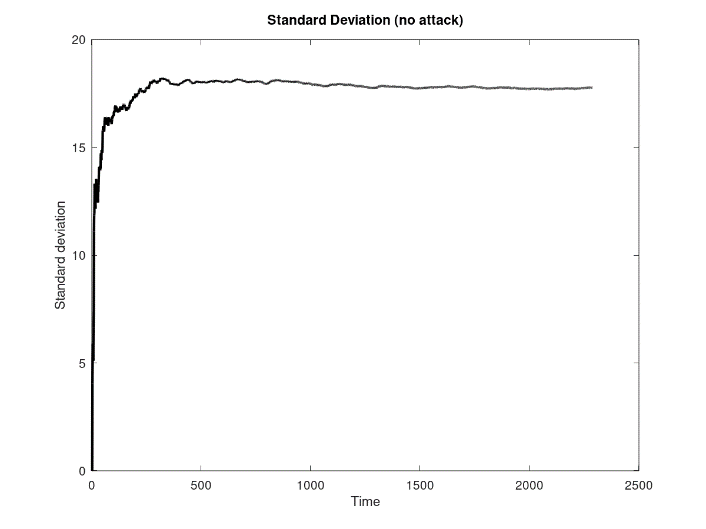
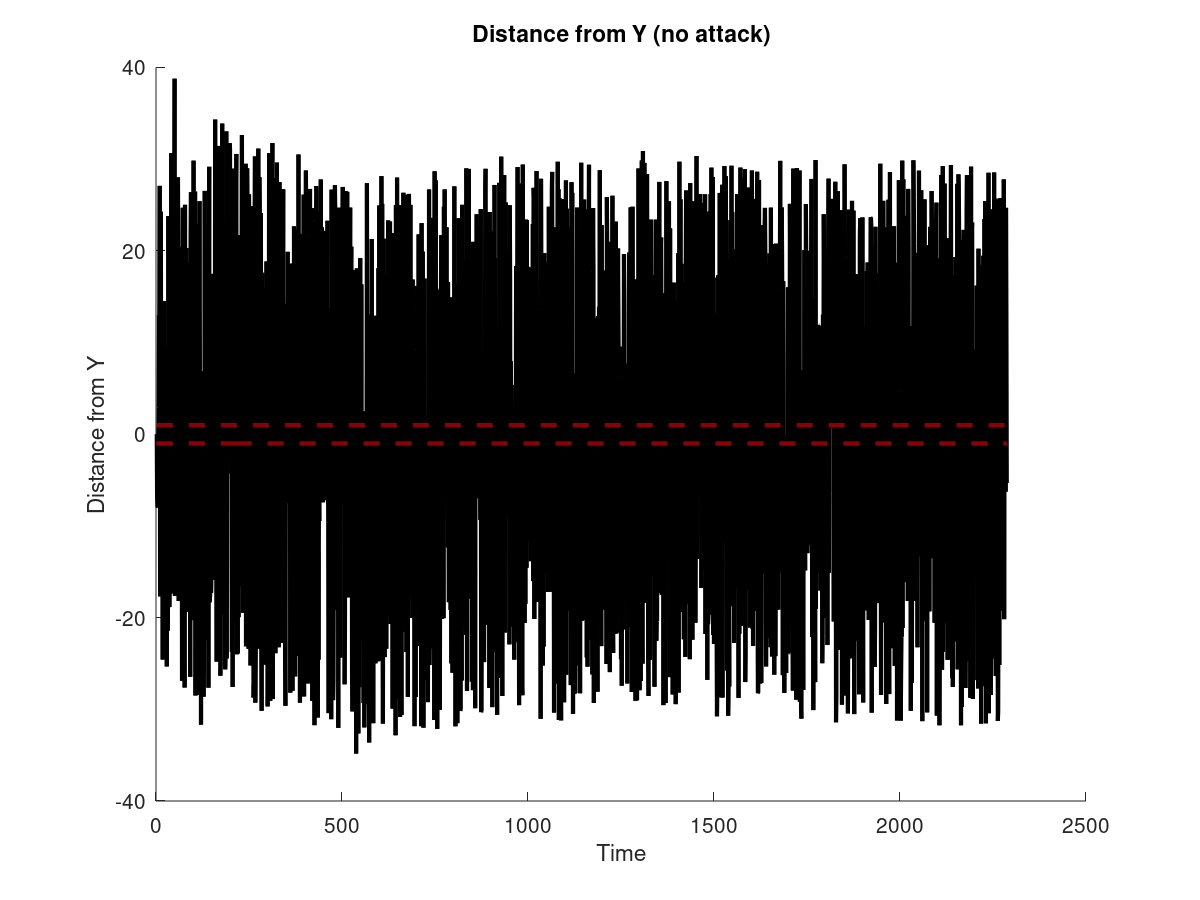
In this project, 3 cases are considered:

1. Normal traffic
2. Normal traffic with DDoS attack
3. Normal traffic with new type DDoS attack

*Case 1: Normal Traffic*

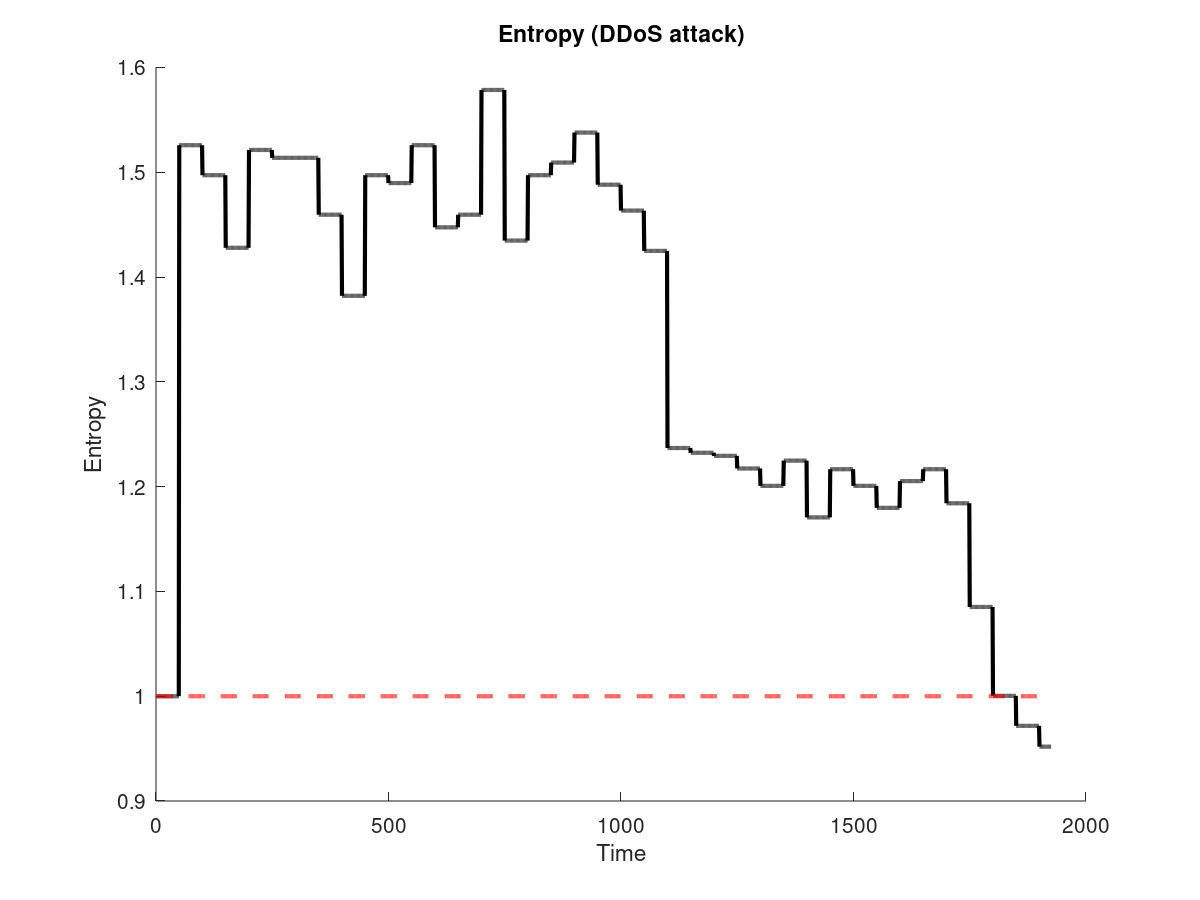
In mininet, traffic.py is run in 4 hosts which generate normal traffic my sending TCP packets to the 64 nodes in the network. Data is collected during this session and is compared with other 2 cases.

The entropy value is always higher than 1 which shows us that there is dispersion in traffic.

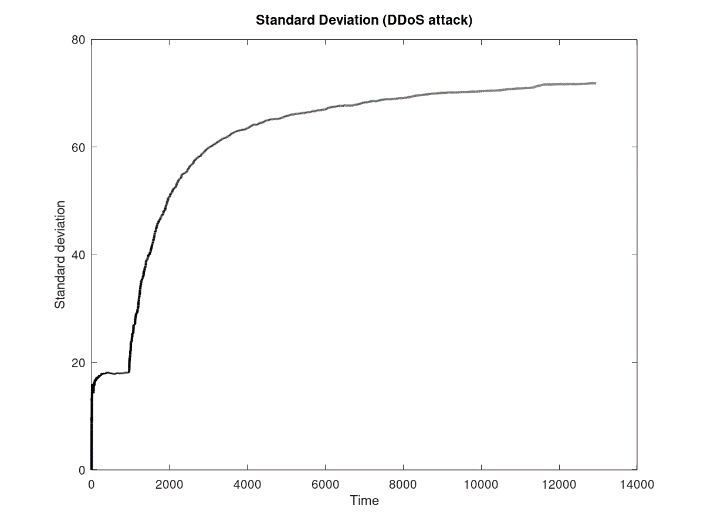
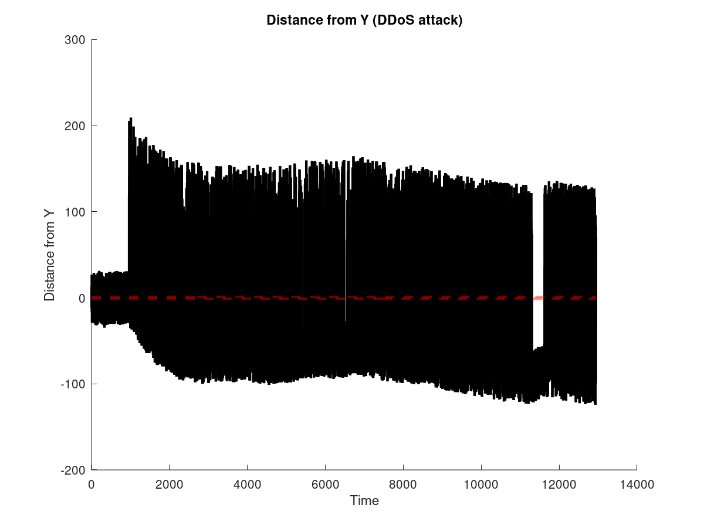


The above graphs are generated using PCA by plotting the distance of y from principal component and the standard deviation against time. The plots are used to compare with the other cases

*Case 2: Normal traffic with DDoS attack*

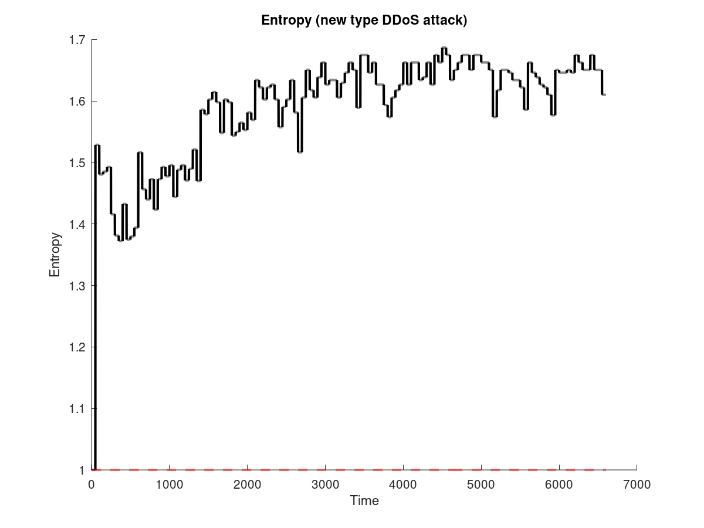
Similar to case 1, traffic.py is run in 4 hosts of the network and the values are recorded. Later the code attack.py with argument 10.0.0.64 is run to attack the host with IP 10.0.0.64.

From the plot we can identify that the entropy value decreases as time progresses, from which we say that randomness in the traffic decreased many packets are directed towards the host 10.0.0.64 .When the entropy value is below 1, we can confirm that system is under DDoS attack

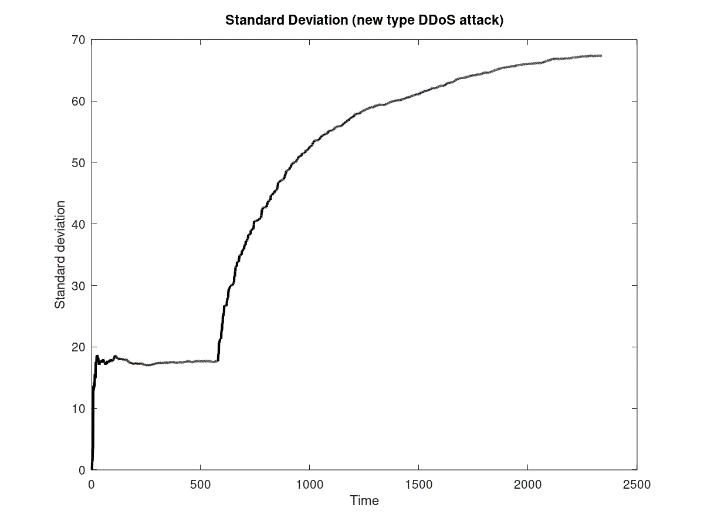
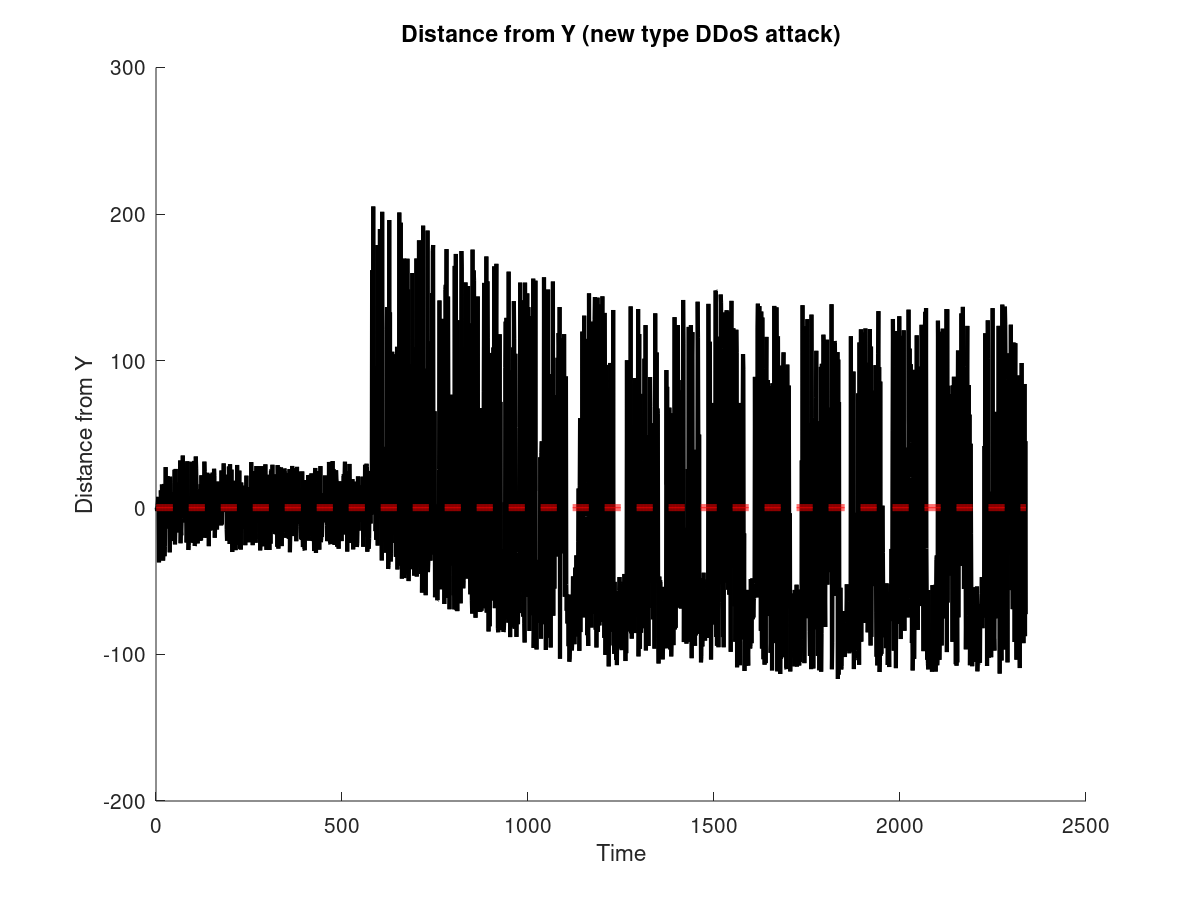


The above graphs generated using PCA, we can identify the distance of y from the principal component has high variations when the DDoS attack is initiated, also as time progresses the distance of y from the principal component converges in the range (-1,1) . The standard deviation of PCA also has a sudden increase when the DDoS attack is initiated. From which we can detect the system is under DDoS attack.

*Case 2: Normal traffic with new type DDoS attack*

Similar to case 1 and 2, traffic.py is run in 4 hosts of the network and the values are recorded. Later the code attackrand.py is run to generate packets with random destination.

From the graph we can infer, the entropy value keeps rising as time progresses. After the DDoS attack is initiated the entropy value does not move towards 1 therefore we cannot indicate the DDoS.

The obtained graphs are similar to that of case 2 with little variations. The distance of y from principal component converges at range (-1,1) after a long time duration. The values of the standard deviation is higher than the values obtained in case 2.

The results obtained from 3 cases show that entropy has faster detection time than PCA. While PCA detects the presence of the new type of DDoS attack sample entropy fails to do so. It is also found that in the new type of DDoS attack to the hijacking of the controller’s resources and completely occupying of buffer memory there are fluctuations in the entropy and distance of y from principal component in case 2.This fluctuations may also lead to the complete shutting of the SDN network.

Conclusion and further enhancement

In this project a new type of DDoS attack is introduced which unlike the traditional DDoS attack is focused on the entire SDN system. Early detection of the DDos attack during run time is important so that the network does not have permanent damage. We have considered two detection mechanisms sample entropy and principal component analysis. From the 3 cases tested the following facts are found, entropy has less detection time than PCA which is very important during real time attack. But it cannot identify the new type of DDoS attack since the randomness of the traffic does not decrease. While PCA needs more time for detection than sample entropy it can be used to detect the presence of the new type of DDoS attack in traffic. In an attackers target is the entire SDN system, PCA is the best option for detection and also since it also useful for detection of traditional DDoS attack, the best mechanism for detection of DDoS attack is Principal component analysis. By decreasing the detection time needed for PCA the value of the project increases. By identifying the bots in the ssystem we can send instruction from the controller to switches to drop the packets received from the interface.

Reference

1. “A Novel Distributed Denial-of-Service Attack Detection Scheme for Software Defined Networking Environments” *Di Wu, Jie Li, Sajal K. Dasy, Jinsong Wuz, Yusheng Jix, and Zhetao Li*
2. “DDoS Attack Detection Algorithm Based on IP Entropy Model ” *Wang xintong, Liu guqing, Yang jungang, Ran jinzhi*
3. “Structural Analysis of Network Traffic Flows” *Anukool Lakhina, Konstantina Papagiannaki, Mark Crovella, Christophe Diot, Eric D. Kolaczyk, and Nina Taft*
4. “[Analysis-of-DDoS-Attacks-in-SDN-Environments](https://github.com/aswanthpp/Analysis-of-DDoS-Attacks-in-SDN-Environments)” - *ASWANTH P P*
5. “Survey on SDN based network intrusion detection system using machine learning approaches” *Nasrin Sultana & Naveen Chilamkurti & Wei Peng & Rabei Alhadad*
6. “Openflow switch specification 1.3.1.” https://www.opennetworking.org/ images/stories/downloads/sdn-resources/onf-specifications/openflow/ openflow-spec-v1.3.1.pdf.
7. POX Manual Current documentation