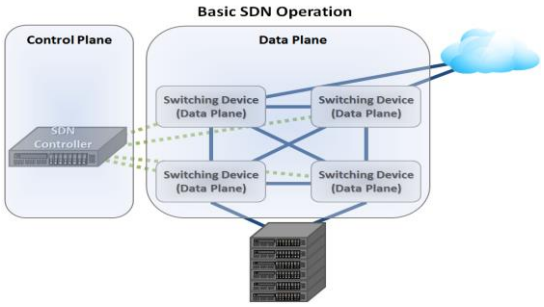
Custom Policy Generation and Firewall Policies in SDN		
Muhammad Shahriyar Sheikh	**Abu Bakar Sarwar**	**Hatim Ali Asghar**
K132015@nu.edu.pk	K132212@nu.edu.pk	K132260@nu.edu.pk

INTRODUCTION TO SDN:

Software defined network (SDN) is an approach to computer networking that allows network administration to manage network services through abstraction of higher level functionality. This is done by decoupling 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).



GETTING HANDS ON MININET:

1. $ sudo mn –h
   This command display a help message describing mininet startup options.

2. $ sudo mn
   This command runs the default toplogy.

3. minet > nodes
   This command displays all the node in the network.

4. mininet > net
   This command displays all the links in the network.

5. mininet > dump
   This command dumps complete information of all the nodes in the network

6. mininet> pingall
   This command simply passes pings to all the nodes in the network.

7. mininet > h1 ping h2
   You can also explicitly ping from host h1 to host h2 through this above command.

8. mininet > exit
   This command exits the CLI.

9. $ sudo mn –c
   This command try cleans up the cache and any residual state if mininet crashes for some reason.

10. mininet > xterm h1 h2
    The alternative - better for running interactive commands and watching debug output - is to spawn an xterm for one or more virtual hosts. In the Mininet console, run:

11. mininet> h1 ifconfig
    To run a single command on a node, prepend the command with the name of the node. For example, to check the IP address of a virtual host.

SAMPLE CODE FOR TOPOLOGY GENERATION:

This example shows how to create an empty Mininet Object (without a topology object) and add nodes to it manually. This is emptynet.py file which is located at Mininet>Examples. This help us set the base for topology generation. We further used this code make the custom topology python code.

```
from mininet.net import Mininet
from mininet.node import Controller
from mininet.cli import CLI
from mininet.log import setLogLevel, info

def emptyNet():

    "Create an empty network and add nodes to it."

    net = Mininet( controller=Controller )

    info( '*** Adding controller\n' )
    net.addController( 'c0' )

    info( '*** Adding hosts\n' )
    h1 = net.addHost( 'h1',ip='10.0.0.1' )
    h2 = net.addHost( 'h2', ip='10.0.0.2' )

    info( '*** Adding switch\n' )
    s3 = net.addSwitch( 's3' )

    info( '*** Creating links\n' )
    net.addLink( h1, s3 )
    net.addLink( h2, s3 )

    info( '*** Starting network\n')
    net.start()

    info( '*** Running CLI\n' )
    CLI( net )
```

```
    info( '*** Stopping network' )
    net.stop()

if __name__ == '__main__':
    setLogLevel( 'info' )
    emptyNet()
```

PSEUDOCODE:

```
Input number of hosts(n)

Create list H[size of n]

i=1

While(i<=n) {

  H[i]=net.addHost('H[i]',ip='10.0.0.%i')

  i++ }

Input number of switches(s)

Create list S[size of s]

i=1

while(i<s){

  S[i]=net.addSwitch('S[i]')

  i++ }

While(h!=0 and s!=0){

Print('Type 0 in host and switch to end')

Input h

Input s

Net.addLink(H[h], S[s])}
```

TRAFFIC GENERATION WITHOUT GUI:

We have made our code generic by embedding the above mentioned pseudo code. The main purpose was to permit the user so that he can add as many hosts, links and switches in the network and explicitly we allow the user to define the host names and their IP addresses. Moreover users have the control to link any of the hosts to any of the switches which he has defined in the network. In this way user can make his own network topology and can generate traffic among various flows.

TRAFFIC GENERATION WITH GUI:

In order to make our program attractive and user friendly we have design the GUI. Users now can easily design their own topology and control the flow of traffic with the help of the GUI.

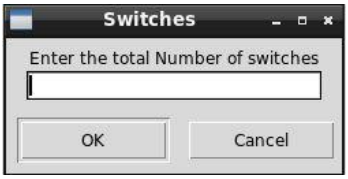
To start this program, type

~ sudo –E python 'filename'.py

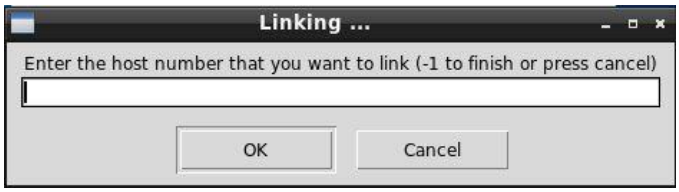
The program will start to run indicating the user's to first enter the number of hosts

After entering, it will ask all the ip's of the hosts as

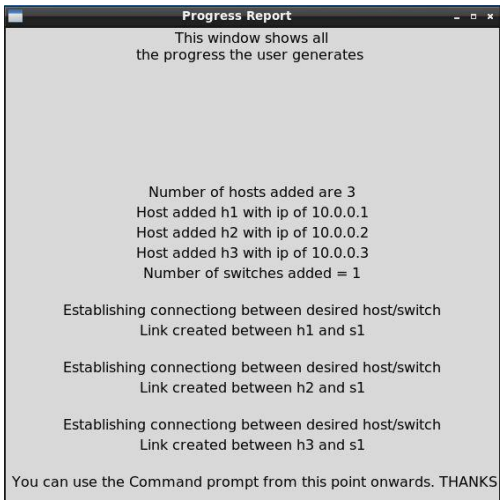
After entering ip's of all hosts, the user will be prompted with entering the number of switches



The user will then create the links



A progress window will appear at the start of the program, informing the user of his appropriate actions



After all the necessary requirements have been fulfilled, the command prompt can be used to test the custom topology that is generated by the program.

Code snippet of GUI traffic generation

```
#!/usr/bin/python
## Created by Muhammad Shahriyar Sheikh
##            Hatim Ali Asghar
##            Abu Bakar Sarwar



import Tkinter
from Tkinter import *
import tkMessageBox
import tkSimpleDialog

from mininet.net import Mininet
from mininet.node import Controller
from mininet.cli import CLI
from mininet.log import setLogLevel, info

root = Tk()



def add_new(textToDisplay):

    b = Label(root,text = textToDisplay, fg = "Black", font = "Fixedsys")
    b.pack()



def emptyNet():

    #root = Tk()


    #"Create an empty network and add nodes to it."

    ######################
    hostCount=0
    a=1 #Loop Variable to assign hosts of hostCount

    ipstring = ""
    i=1 #Loop Variable to assign ip to hosts

    switchCount=0
    b=1 #Loop Variable for allocating switchCount switches
    j=1 #Loop Variable for assigning switches of switchCount

    hostLink=0 #Number of host to be connected in link
    switchLink=0 #Number of switch to be connected in link
    hostCounter = 0 #For it to stop asking for input when the number of hosts are completed

    ######################

    add_new("This window shows all\nthe progress the user generates \n\n\n\n\n\n\n")
    #roo = Tk()
    #roo.withdraw()

    net = Mininet( controller=Controller )

    info( '*** Adding controller\n' )
    net.addController( 'c0' )


    #hostCount=input('Enter the total number of hosts ')

    hostCount = tkSimpleDialog.askinteger("Number of Hosts","Enter the number of hosts? ")
    add_new("Number of hosts added are %s"%hostCount)

    hosts=[Mininet(controller=Controller)] #List for hosts
    #Loop for allocation host count hosts to list

    hosts.append(Mininet(controller=Controller)) #To avoid h0


    while a<=hostCount:
    hosts.append(Mininet(controller=Controller))
    a=a+1
    #Loop to assign hosts of hostCount




    while i<=hostCount:
    k=str(i)
    #ipstring = raw_input("Enter an ip for host h"+k + " ")
    ipstring = tkSimpleDialog.askstring("IP for host h%s"%k,"Enter an ip for host h%s (Must be of xx.xx.xx.xx format) "%k)
    hosts[i]=net.addHost('h'+k,ip=ipstring)
    i=i+1
    info('Host added h'+k)
    info(' with ip of ' + ipstring)
    add_new("Host added h{0} with ip of {1}".format(k,ipstring))
    info('\n')
```

```
    #switchCount=input('Enter the total number of switches: ')
    switchCount = tkSimpleDialog.askinteger("Switches","Enter the total Number of switches")
    switches=[Mininet(controller=Controller)] #List for Switches
    add_new("Number of switches added = %s"%switchCount)


    switches.append(Mininet(controller=Controller)) #To avoid s0
    #Loop for allocating switchCount switches
    while b<=switchCount:
    switches.append(Mininet(controller=Controller))
    b=b+1



    #loop for assigning switches of switchCount
    while j<=switchCount:
    l=str(j)
    switches[j]=net.addSwitch( 's'+l)
    j=j+1
    info('Switch added s'+l)
    info('\n')



    #Loop for making links


    while hostCounter < hostCount or hostLink == -1:
        #hostLink=input("\nEnter host number that you want to link(-1 to finish or press cancel)\n")
    hostLink = tkSimpleDialog.askinteger("Linking ...","Enter the host number that you want to link (-1 to finish or press
cancel)")
    if hostLink ==-1:
       break
        #switchLink=input("Enter switch number that you want to link it with\n")
    switchLink = tkSimpleDialog.askinteger("Linking ...","Enter switch number that you want to link h{0} with ".format(hostLink))

    add_new("\nEstablishing connectiong between desired host/switch")
    #Establishing connectiong between desired host/switch
        net.addLink(hosts[hostLink],switches[switchLink])
    m=str(hostLink)
    n=str(switchLink)
    info( "Link created between h"+m )
        info( " and s"+n)
    add_new("Link created between h{0} and s{1}".format(m,n))

    hostCounter = hostCounter + 1

    add_new("\n You can use the Command prompt from this point onwards. THANKS\n")
    info( '\n*** Starting network\n')
    net.start()

    info( '*** Running CLI\n' )
    CLI( net )

    info( '*** Stopping network' )
    net.stop()
    exit()


if __name__ == '__main__':


    root.geometry('+900+50')
    root.minsize(420,600)
    root.wm_title("Progress Report")

    setLogLevel( 'info' )
    emptyNet()
    exit()
    #roo = mainloop()
    #roo.quit()
```

POLICY / FIREWALL IN SDN:

Policy is an overarching term that describes network constraints, configurations and settings. It includes the enforcement of access control, traditionally provided by Access Control Lists (ACLs).The major constraint that we have looked at is blocking the access between 2 hosts. For example H1 and H2 are two hosts, we have given the user the choice to block packet sending between them. We basically found a sample code for firewall that could block only h1 and h2. The code as following and the CSV file is also given below:

POLICIES WITH GUI

Once we had this and understood the meaning of policies, we decided to make the code more generic and give more authority to the user. So we converted the code into a more generic form. Now the user can choose which host he wants to block on his own choice. But that is not enough. We also managed to implement the code with a good looking graphical user interface for the user. The method for running is given below:

Step 1: After opening 'LX Terminal' type in command

~ cd /pox/pox/misc

Step 2: Whilst inside the /misc directory, check for the necessary .py files which are needed for applying policies to a controller.

Step 3: In the command prompt, type

~ sudo –E python askpolicies.py

To enter all the necessary policies the user wishes to implement on the controller, A Graphical User Interface will guide the user with the necessary steps

Step 4: After entering the polices for the controller, go back to the first /pox directory by typing TWICE

~ cd ..

Step 5: On the command prompt, type

~ python pox.py log.level --DEBUG forwarding.l2_learning misc.pox_firewall &

To start the controller with the necessary policies. The controller will start listening for incoming mininet topologies

Step 6: Create a custom topology by typing

~ sudo mn - -topo single,3 - -controller remote - -mac

On the command prompt, which will create a mininet topology

Code Snippet for **askpolicies.py**

```
##Created by Muhammad Shahriyar Sheikh '_'
##           Hatim Ali Asghar .-.
##           AbuBakar Sarwar ._.


import csv
import tkMessageBox
import tkSimpleDialog
from Tkinter import *




###################################
root = Tk()
root.withdraw()
noOfPolicies = 0
host1=0
host2=0
macstring = "00:00:00:00:00:0"

###################################


writer = csv.writer(open('firewall.csv','wb'))
writer.writerow(["id","mac_0","mac_1"])

noOfPolicies = tkSimpleDialog.askinteger("Number of Policies","Enter the number of policies you wish to apply on the controller ? ")


for i in range(0,noOfPolicies):
    hoststr = tkSimpleDialog.askstring("Traffic Generation Scheme","Type in x,y which means x will not be able to ping y (Like 1,2)")
    writer.writerow([str(i+1),macstring+hoststr[0:1],macstring+hoststr[-1:]])
```