Aum Amriteswaryai Namah

Name: J Viswaksena Roll.No: AM.EN.U4AIE21035

LAB Assignment 3

For each question submit the final screenshot.

Q1. In this exercise, you need to use miniedit to create a Network Topology.

Invoke miniedit using the miniedit.py in the examples folder under mininet installation directory.host

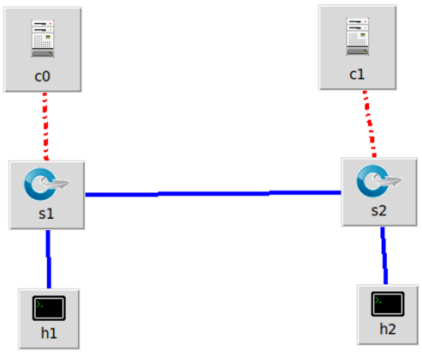
a) You add two controllers by selecting the control icon. Give the controllers port as 6653 and 6654.
b) Then add two OVS switches s1 and s2 by selecting appropriate icon.
c) Add two hosts h1 and h2. Then using Link connect h1 with s1 and h2 with s2.
d) Right click on Switches and hosts and give them unique IP addresses.
e) Connect a link between s1 and s2 and both of the switch should be connected to a controller.
f) After building the topology, you may select the following option "Export Level 2 script"
g) You may get more help from the following site.

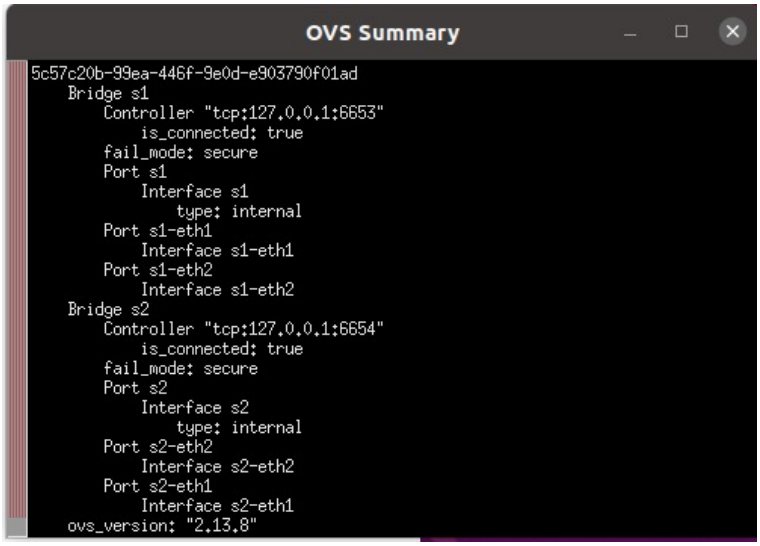
   https://www.brianlinkletter.com/2015/04/how-to-use-miniedit-mininets-graphical-user-interface/

h) The saved python script should be modified by adding/modifying the following lines for Ubuntu20.04 users:
   a. from mininet.node import Controller, OVSBridge
   b. net = Mininet(controller=Controller, switch=OVSBridge)

i) Run the pythin script to have a topology with two controllers.
j) h1 should be able to ping h2

$ sudo python2 miniedit.py






```
viswaksena@ubuntu:~$ sudo python2 miniedit.py
*** Adding controller
*** Add switches
*** Add hosts
*** Add links
*** Starting network
*** Configuring hosts
h1 h2
*** Starting controllers
*** Starting switches
*** Post configure switches and hosts
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
```

```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=12.8 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.453 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.093 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.094 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.095 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.043 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.085 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.142 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.044 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.093 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=0.091 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=0.344 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=0.103 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=0.173 ms
64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=0.098 ms
64 bytes from 10.0.0.2: icmp_seq=16 ttl=64 time=0.280 ms
64 bytes from 10.0.0.2: icmp_seq=17 ttl=64 time=0.360 ms
64 bytes from 10.0.0.2: icmp_seq=18 ttl=64 time=0.116 ms
64 bytes from 10.0.0.2: icmp_seq=19 ttl=64 time=0.137 ms
64 bytes from 10.0.0.2: icmp_seq=20 ttl=64 time=0.333 ms
64 bytes from 10.0.0.2: icmp_seq=21 ttl=64 time=0.118 ms
64 bytes from 10.0.0.2: icmp_seq=22 ttl=64 time=0.100 ms
^C
--- 10.0.0.2 ping statistics ---
22 packets transmitted, 22 received, 0% packet loss, time 21488ms
rtt min/avg/max/mdev = 0.043/0.737/12.833/2.641 ms
```

Q2. In the next experiment, we use two Routers to build a topology with three subnets

Python API given in http://mininet.org/api/annotated.html to create the topology.

a) You need to create two routers with two interfaces. Make sure they have IP address belonging to two different subnets:
    a. E.g. Router1 can have 192.68.1.1/24, 10.0.0.3/8, and Router2 can have 10.0.0.2/8, 172.16.0.1/16
    b. Router should use static IP routing (It is specified as an option in addNode)

b) Add one switch connected to each of the router interfaces
c) Add one host connected to each switch. Make sure that host belongs to the same Subnet as the router's interface.
d) Router1 eth2 and Router2 eth1 should be connected. Router1 eth1 should be connected to a host h1 and Router2 should be connected to Switch s2 and s3 which in turn should be connected to hosts h2 and h3.
e) H1 (h1) should have default gateway as Router1 and h2 and h3 should have default gateway as Router2
f) Use the skeleton given in myrouter2.py and add more statements required for the steps given above.
g) Start the network simulation.
h) H1, h2, h3 should be able to ping each other.

```
from mininet.net import Mininet
from mininet.node import Node
from mininet.link import Link
from mininet.cli import CLI
def create_topology():
    net = Mininet()
    # Adding routers
    router1 = net.addHost('router1', cls=Node, ip='192.68.1.1/24')
    router2 = net.addHost('router2', cls=Node, ip='172.16.0.1/16')
    # Adding switches
    switch1 = net.addSwitch('s1')
    switch2 = net.addSwitch('s2')
    switch3 = net.addSwitch('s3')
    # Adding hosts
    host1 = net.addHost('h1', ip='192.68.1.2/24', defaultRoute='via
192.68.1.1')
    host2 = net.addHost('h2', ip='10.0.0.3/8', defaultRoute='via 10.0.0.2')
    host3 = net.addHost('h3', ip='10.0.0.4/8', defaultRoute='via 10.0.0.2')
    # Creating links
    net.addLink(router1, switch1, intfName1='eth1')
    net.addLink(router1, switch2, intfName1='eth2')
    net.addLink(router2, switch2, intfName1='eth1')
     net.addLink(router2, switch3, intfName1='eth2')
    net.addLink(switch1, host1)
    net.addLink(switch2, host2)
    net.addLink(switch3, host3)
    # Starting the network
net.start()
    # Configuring static routes on routers
    router1.cmd('ip route add 10.0.0.0/8 via 10.0.0.2')
    router2.cmd('ip route add 192.68.1.0/24 via 10.0.0.3')
    # Enabling IP forwarding on routers
    router1.cmd('sysctl -w net.ipv4.ip_forward=1')
    router2.cmd('sysctl -w net.ipv4.ip_forward=1')
    # Setting up NAT on router2 (assuming it's the one connecting to the
Internet)
    router2.cmd('iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE')
    # Running the CLI
CLI(net)
    # Stopping the network
net.stop()

if __name__ == '__main__':
    create_topology()
```

```
viswaksena@ubuntu:~$ sudo mn --custom ~/mininet/custom/q2.py
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> h1 ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: h1-eth0@if48: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UP group default qlen 1000
    link/ether 92:f3:d8:14:e4:6d brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.0.1/8 brd 10.255.255.255 scope global h1-eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::90f3:d8ff:fe14:e46d/64 scope link
       valid_lft forever preferred_lft forever
mininet> h1 ping -c 3 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=16.6 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.974 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.229 ms

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2013ms
rtt min/avg/max/mdev = 0.229/5.939/16.616/7.555 ms
```

Q3. In this question, you will install Pox controller unless it's already installed on your system.

a) Install pox following the instructions in https://noxrepo.github.io/pox-doc/html/

b) Then run a L2 learning switch component by running the following command in the directory where pox.py is found. This component makes OpenFlow switches act as a type of L2 learning switch.

./pox.py samples.pretty_log forwarding.l2_learning

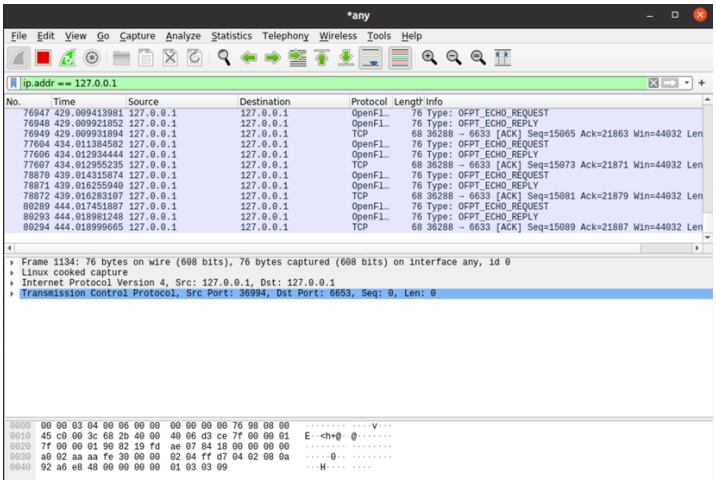
```
viswaksena@ubuntu:~/pox$ ./pox.py samples.pretty_log forwarding.l2_learning
POX 0.7.0 (gar) / Copyright 2011-2020 James McCauley, et al.
[version                ] Support for Python 3 is experimental.
[core                   ] POX 0.7.0 (gar) is up.
[openflow.of_01         ] [00-00-00-00-00-01 2] connected
```

c) In another window, run mininet with any sample topology and with the option remote controller as pox.

sudo mn –topo=single,4 --controller pox

```
viswaksena@ubuntu:~$ sudo mn --topo single,4 --controller remote,ip=127.0.0.1,port=663
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
```

d) Invoke Wireshark in another window or inside mininet and start capturing the packets.



e) Do a ping between two hosts. Once you find ping is working, Stop capturing the packets in Wireshark.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
```

f) Now locate the following Openflow messages

a. Packet in

```
62 16.218632485  fe80::200:ff:fe00:1  ff02::2  OpenFl…  156 Type: OFPT_PACKET_IN
63 16.222297713  fe80::200:ff:fe00:1  ff02::2  OpenFl…  162 Type: OFPT_PACKET_OUT
```

b. Echo,

```
51 5.006403905  127.0.0.1  127.0.0.1  OpenFl…  76 Type: OFPT_ECHO_REQUEST
52 5.008259373  127.0.0.1  127.0.0.1  OpenFl…  76 Type: OFPT_ECHO_REPLY
```

c. Hello and Barrier

```
1850 24.357698568  127.0.0.1       127.0.0.1       TCP       68 41748 → 6633 [ACK] Seq=1349 Ack=113 Win=4
1851 24.357724891  127.0.0.1       127.0.0.1       OpenFl…   76 Type: OFPT_BARRIER_REQUEST
1852 24.357730128  127.0.0.1       127.0.0.1       TCP       68 41748 → 6633 [ACK] Seq=1349 Ack=121 Win=4
1853 24.357823307  127.0.0.1       127.0.0.1       OpenFl…   76 Type: OFPT_BARRIER_REPLY
1855 24.398055300  127.0.0.1       127.0.0.1       TCP       68 6633 → 41748 [ACK] Seq=121 Ack=1357 Win=6
1862 24.506499393  ::              ff02::1:ff19:f098 OpenFl…  172 Type: OFPT_PACKET_IN
1760 23.848206680  127.0.0.1       127.0.0.1       OpenFl…   76 Type: OFPT_HELLO
1761 23.848217583  127.0.0.1       127.0.0.1       TCP       68 6633 → 41748 [ACK] Seq=1 Ack=9 Win=65536
1777 23.890620780  127.0.0.1       127.0.0.1       OpenFl…   76 Type: OFPT_HELLO
1778 23.890632120  127.0.0.1       127.0.0.1       TCP       68 41748 → 6633 [ACK] Seq=9 Ack=9 Win=44032
1779 23.891880745  127.0.0.1       127.0.0.1       OpenFl…   88 Type: OFPT_STATS_REQUEST
1780 23.891884103  127.0.0.1       127.0.0.1       TCP       68 41748 → 6633 [ACK] Seq=9 Ack=29 Win=4403
```

Using ryu controller:

```
viswaksena@ubuntu:~$ ryu-manager ryu.app.simple_switch
loading app ryu.app.simple_switch
loading app ryu.controller.ofp_handler
instantiating app ryu.app.simple_switch of SimpleSwitch
instantiating app ryu.controller.ofp_handler of OFPHandler
packet in 1 00:00:00:00:00:01 33:33:00:00:00:02 1
packet in 1 00:00:00:00:00:03 33:33:00:00:00:02 3
packet in 1 00:00:00:00:00:02 33:33:00:00:00:02 2
packet in 1 00:00:00:00:00:01 33:33:00:00:00:02 1
packet in 1 00:00:00:00:00:03 33:33:00:00:00:02 3
packet in 1 00:00:00:00:00:02 33:33:00:00:00:02 2
packet in 1 00:00:00:00:00:01 ff:ff:ff:ff:ff:ff 1
packet in 1 00:00:00:00:00:02 00:00:00:00:00:01 2
packet in 1 00:00:00:00:00:01 00:00:00:00:00:02 1
packet in 1 00:00:00:00:00:01 ff:ff:ff:ff:ff:ff 1
packet in 1 00:00:00:00:00:03 00:00:00:00:00:01 3
packet in 1 00:00:00:00:00:01 00:00:00:00:00:03 1
packet in 1 00:00:00:00:00:02 ff:ff:ff:ff:ff:ff 2
packet in 1 00:00:00:00:00:03 00:00:00:00:00:02 3
packet in 1 00:00:00:00:00:02 00:00:00:00:00:03 2
packet in 1 00:00:00:00:00:01 33:33:00:00:00:02 1
packet in 1 00:00:00:00:00:03 33:33:00:00:00:02 3
packet in 1 00:00:00:00:00:02 33:33:00:00:00:02 2
```

```
viswaksena@ubuntu:~$ sudo mn --topo single,3 --mac --controller remote --switch
ovsk
*** Creating network
*** Adding controller
Connecting to remote controller at 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> pinall
*** Unknown command: pinall
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
```