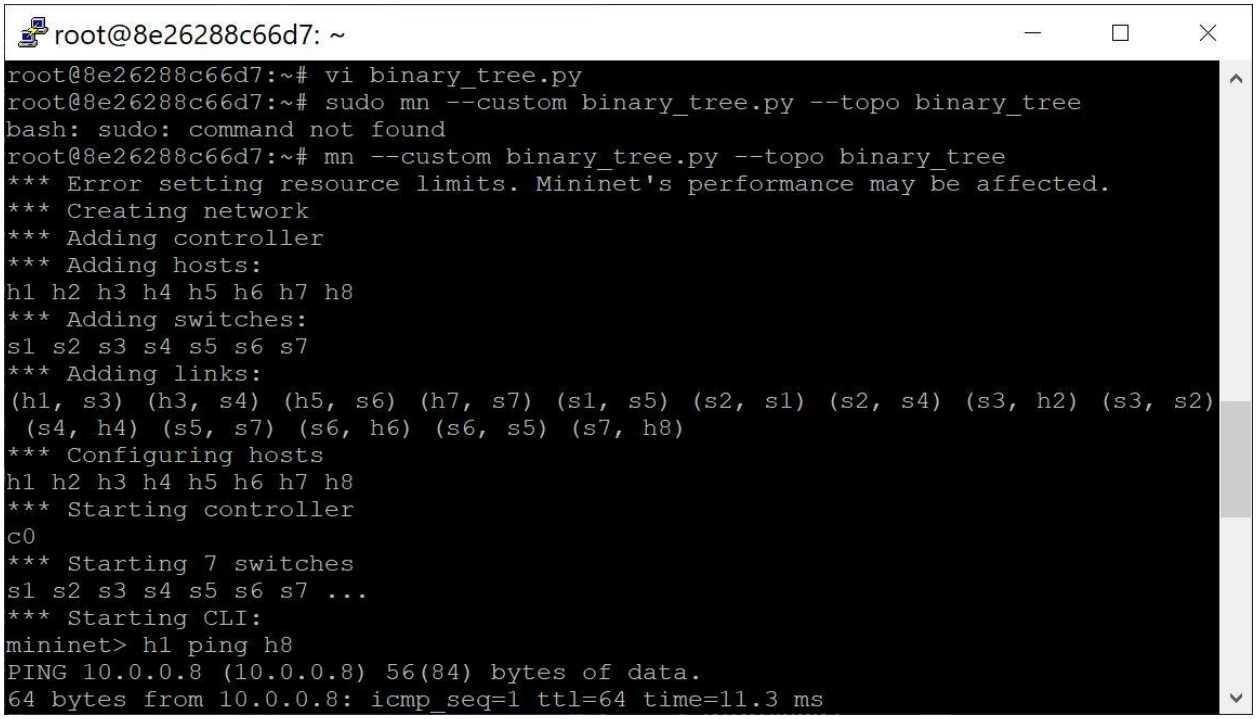
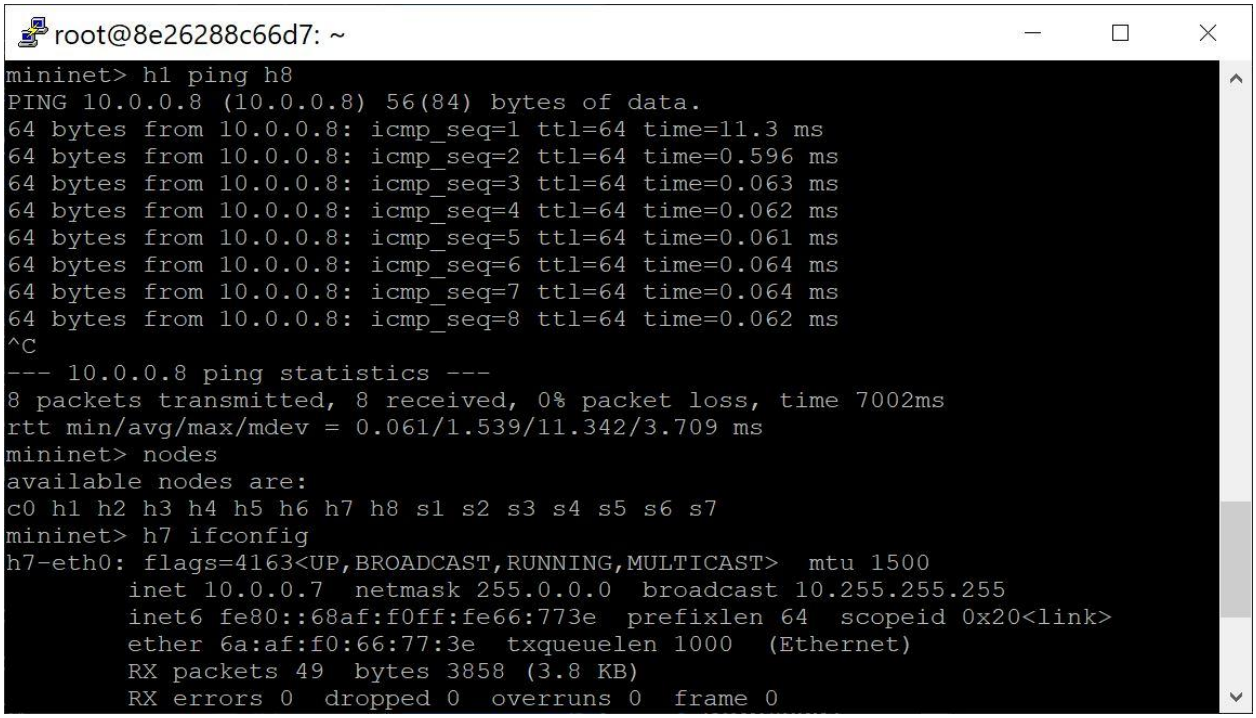
Praveen Vandeyar
COEN 241

Hw 3

sudo mn --custom binary_tree.py --topo binary_tree

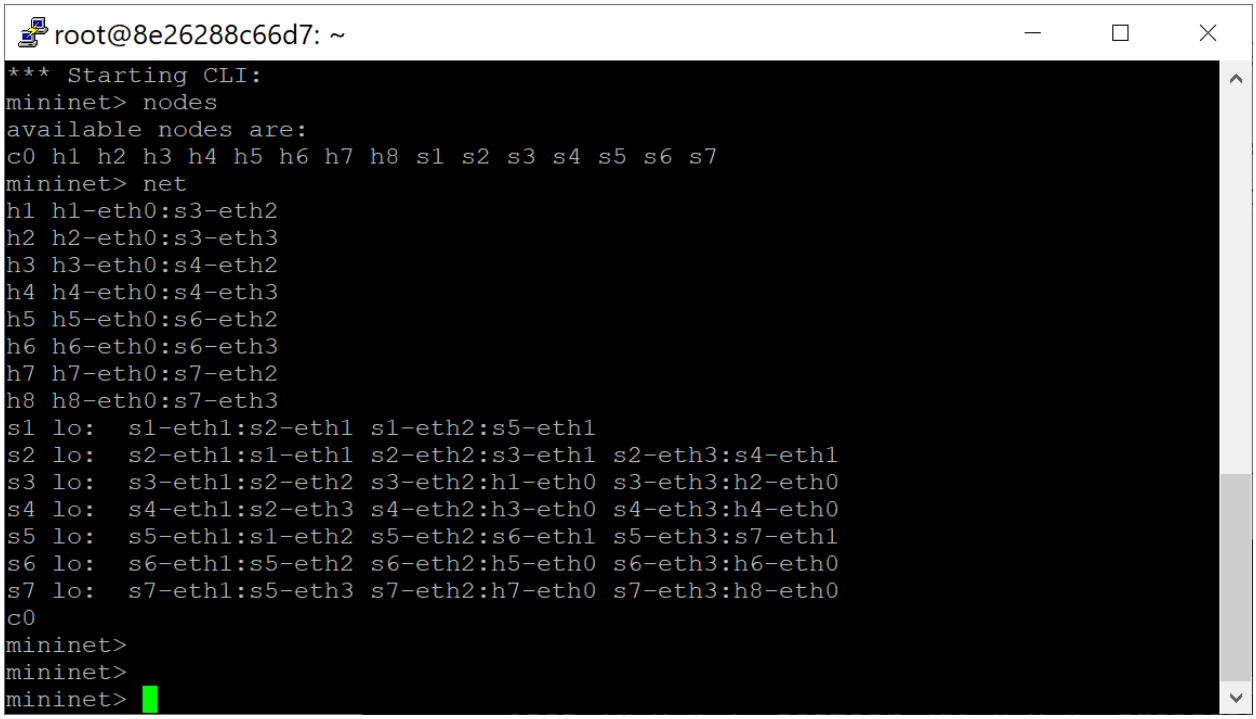


mininet> h1 ping h8

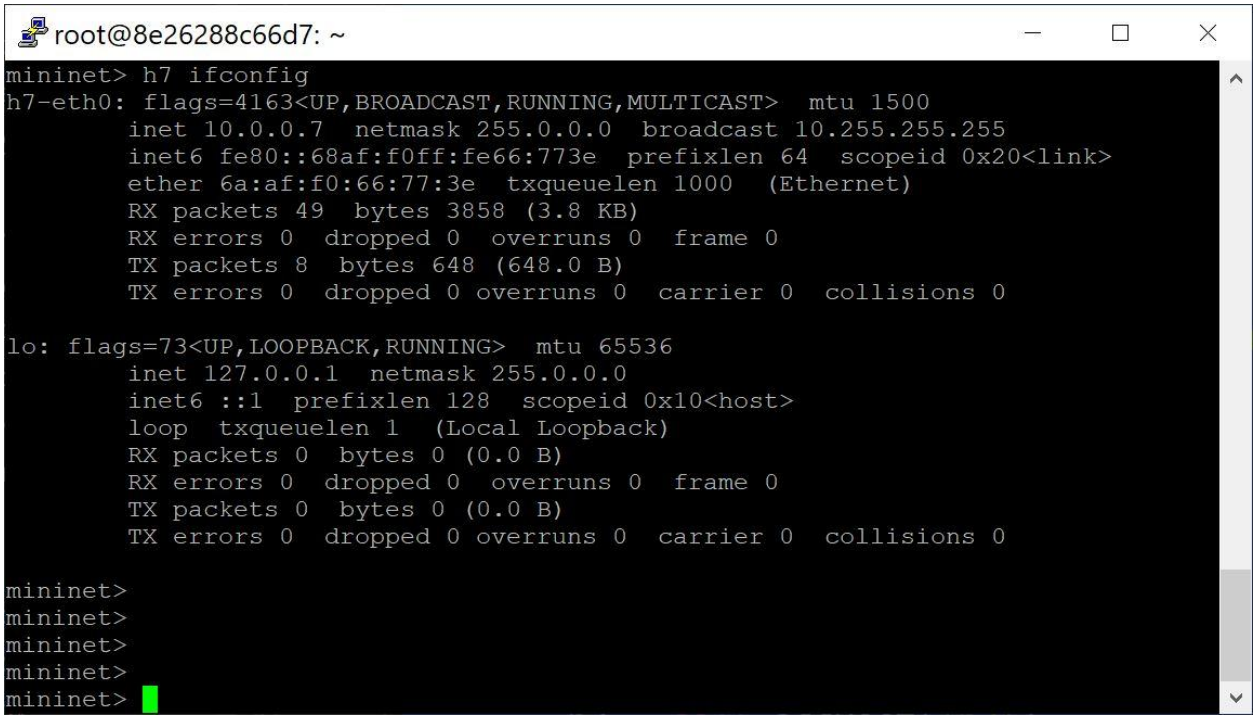

root@8e26288c66d7: ~
mininet> h1 ping h8
PING 10.0.0.8 (10.0.0.8) 56(84) bytes of data.
64 bytes from 10.0.0.8: icmp_seq=1 ttl=64 time=11.3 ms
64 bytes from 10.0.0.8: icmp_seq=2 ttl=64 time=0.596 ms
64 bytes from 10.0.0.8: icmp_seq=3 ttl=64 time=0.063 ms
64 bytes from 10.0.0.8: icmp_seq=4 ttl=64 time=0.062 ms
64 bytes from 10.0.0.8: icmp_seq=5 ttl=64 time=0.061 ms
64 bytes from 10.0.0.8: icmp_seq=6 ttl=64 time=0.064 ms
64 bytes from 10.0.0.8: icmp_seq=7 ttl=64 time=0.064 ms
64 bytes from 10.0.0.8: icmp_seq=8 ttl=64 time=0.062 ms
^C
--- 10.0.0.8 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7002ms
rtt min/avg/max/mdev = 0.061/1.539/11.342/3.709 ms
mininet> nodes
available nodes are:
c0 h1 h2 h3 h4 h5 h6 h7 h8 s1 s2 s3 s4 s5 s6 s7
mininet> h7 ifconfig
h7-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 10.0.0.7 netmask 255.0.0.0 broadcast 10.255.255.255
inet6 fe80::68af:f0ff:fe66:773e prefixlen 64 scopeid 0x20<link>
ether 6a:af:f0:66:77:3e txqueuelen 1000 (Ethernet)
RX packets 49 bytes 3858 (3.8 KB)
RX errors 0 dropped 0 overruns 0 frame 0

Task 1

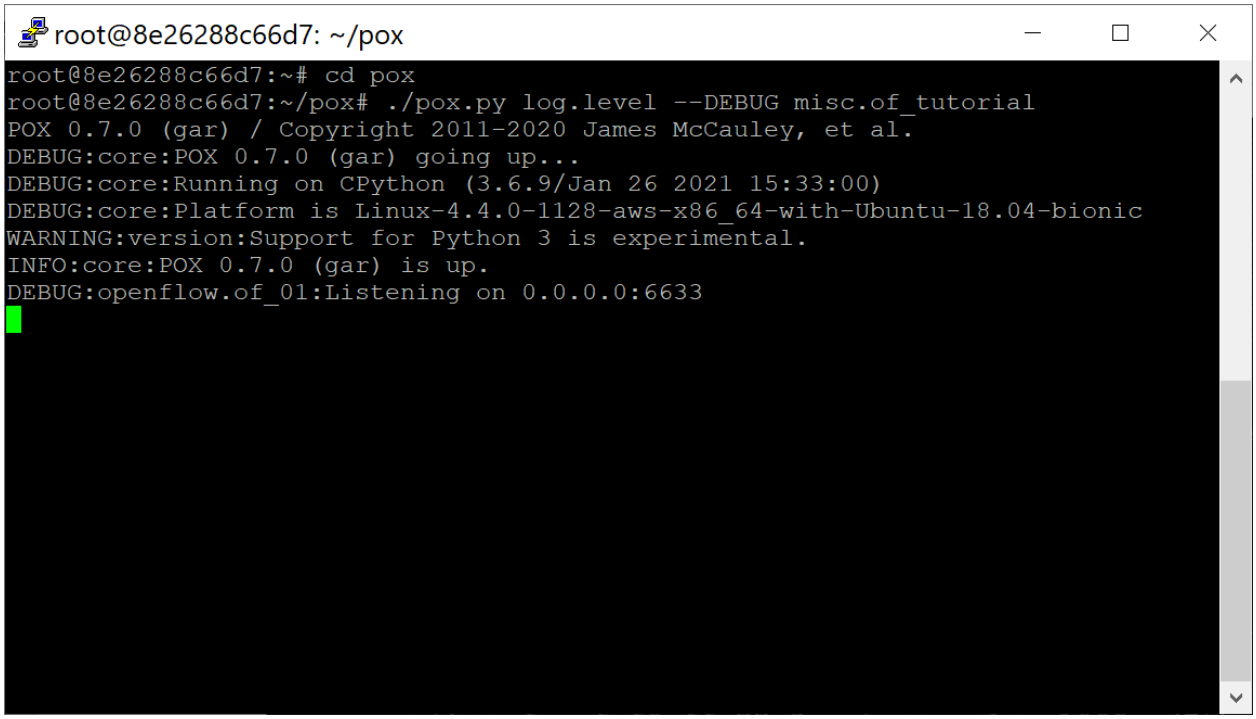
1.


root@8e26288c66d7: ~
*** Starting CLI:
mininet> nodes
available nodes are:
c0 h1 h2 h3 h4 h5 h6 h7 h8 s1 s2 s3 s4 s5 s6 s7
mininet> net
h1 h1-eth0:s3-eth2
h2 h2-eth0:s3-eth3
h3 h3-eth0:s4-eth2
h4 h4-eth0:s4-eth3
h5 h5-eth0:s6-eth2
h6 h6-eth0:s6-eth3
h7 h7-eth0:s7-eth2
h8 h8-eth0:s7-eth3
s1 lo: s1-eth1:s2-eth1 s1-eth2:s5-eth1
s2 lo: s2-eth1:s1-eth1 s2-eth2:s3-eth1 s2-eth3:s4-eth1
s3 lo: s3-eth1:s2-eth2 s3-eth2:h1-eth0 s3-eth3:h2-eth0
s4 lo: s4-eth1:s2-eth3 s4-eth2:h3-eth0 s4-eth3:h4-eth0
s5 lo: s5-eth1:s1-eth2 s5-eth2:s6-eth1 s5-eth3:s7-eth1
s6 lo: s6-eth1:s5-eth2 s6-eth2:h5-eth0 s6-eth3:h6-eth0
s7 lo: s7-eth1:s5-eth3 s7-eth2:h7-eth0 s7-eth3:h8-eth0
c0
mininet>
mininet>
mininet>

2.

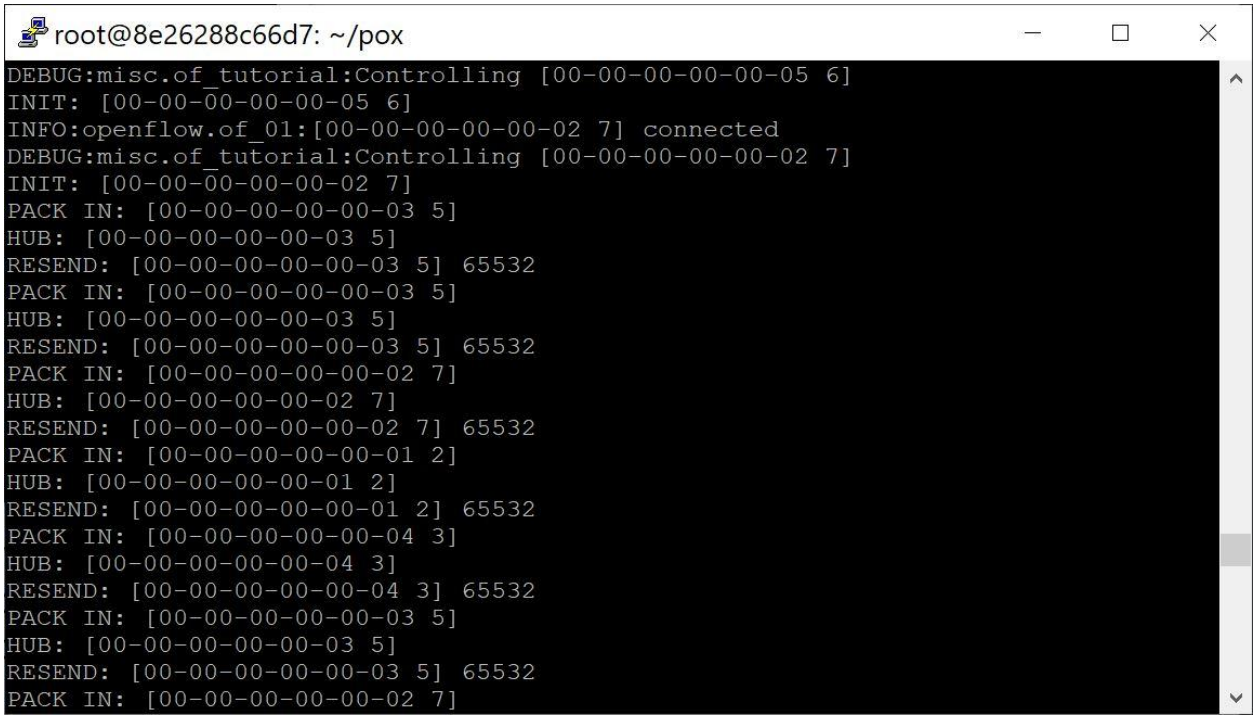

root@8e26288c66d7: ~
mininet> h7 ifconfig
h7-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 10.0.0.7 netmask 255.0.0.0 broadcast 10.255.255.255
inet6 fe80::68af:f0ff:fe66:773e prefixlen 64 scopeid 0x20<link>
ether 6a:af:f0:66:77:3e txqueuelen 1000 (Ethernet)
RX packets 49 bytes 3858 (3.8 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 8 bytes 648 (648.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1 (Local Loopback)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
mininet>
mininet>
mininet>
mininet>
mininet>

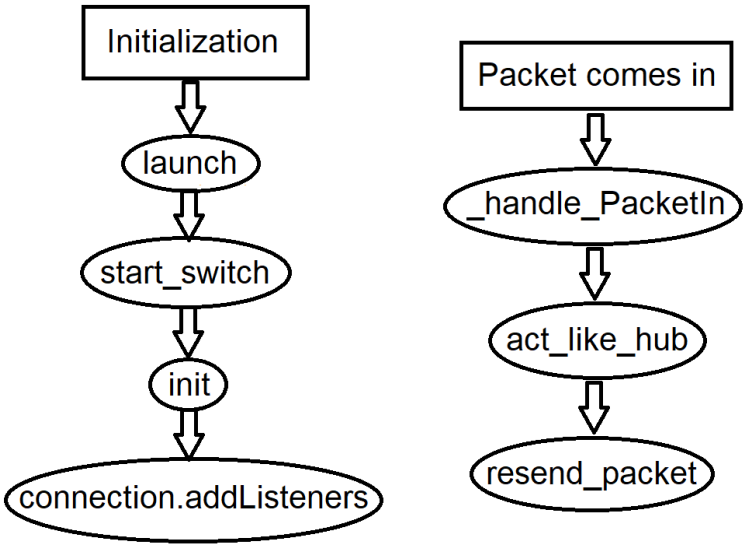
./pox.py log.level --DEBUG misc.of_tutorial



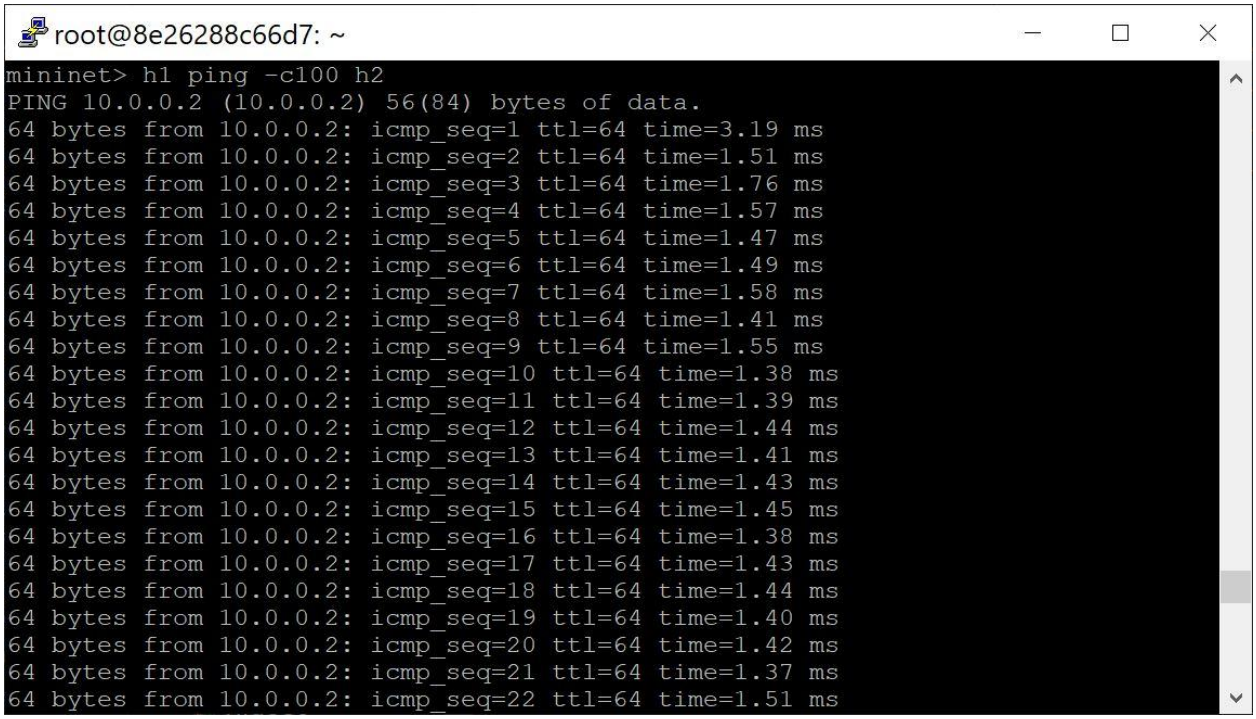
Task 2

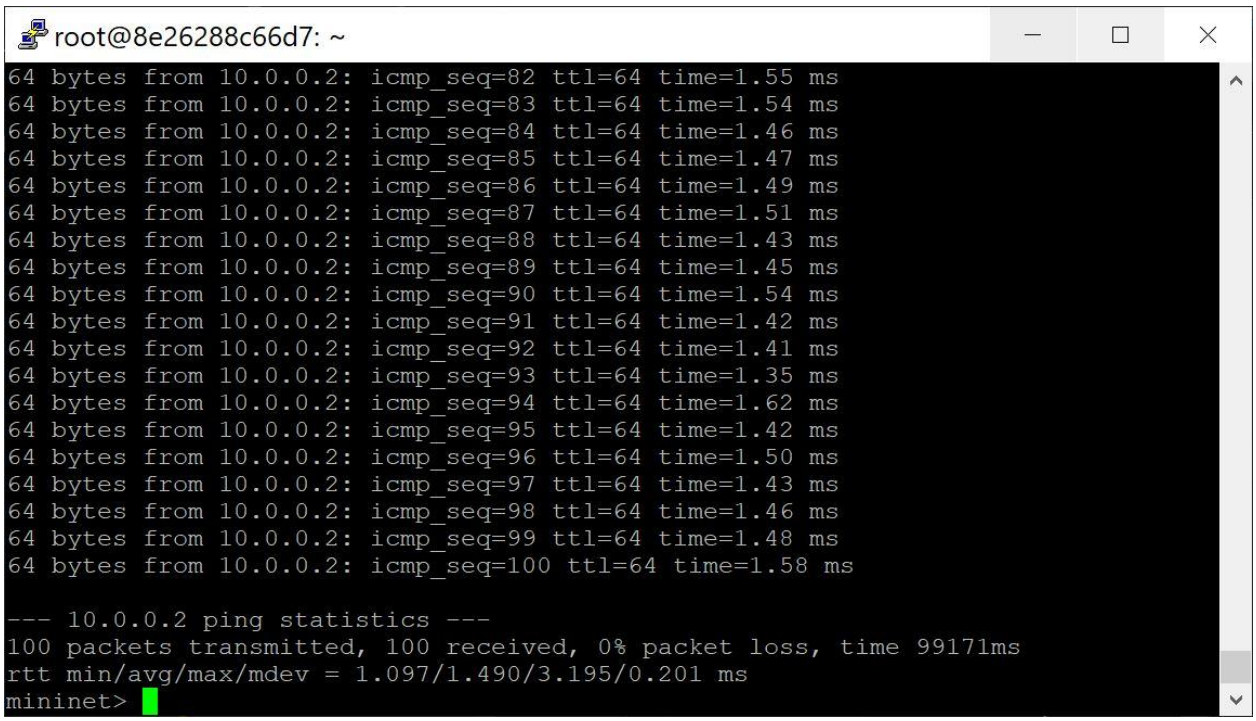
1.

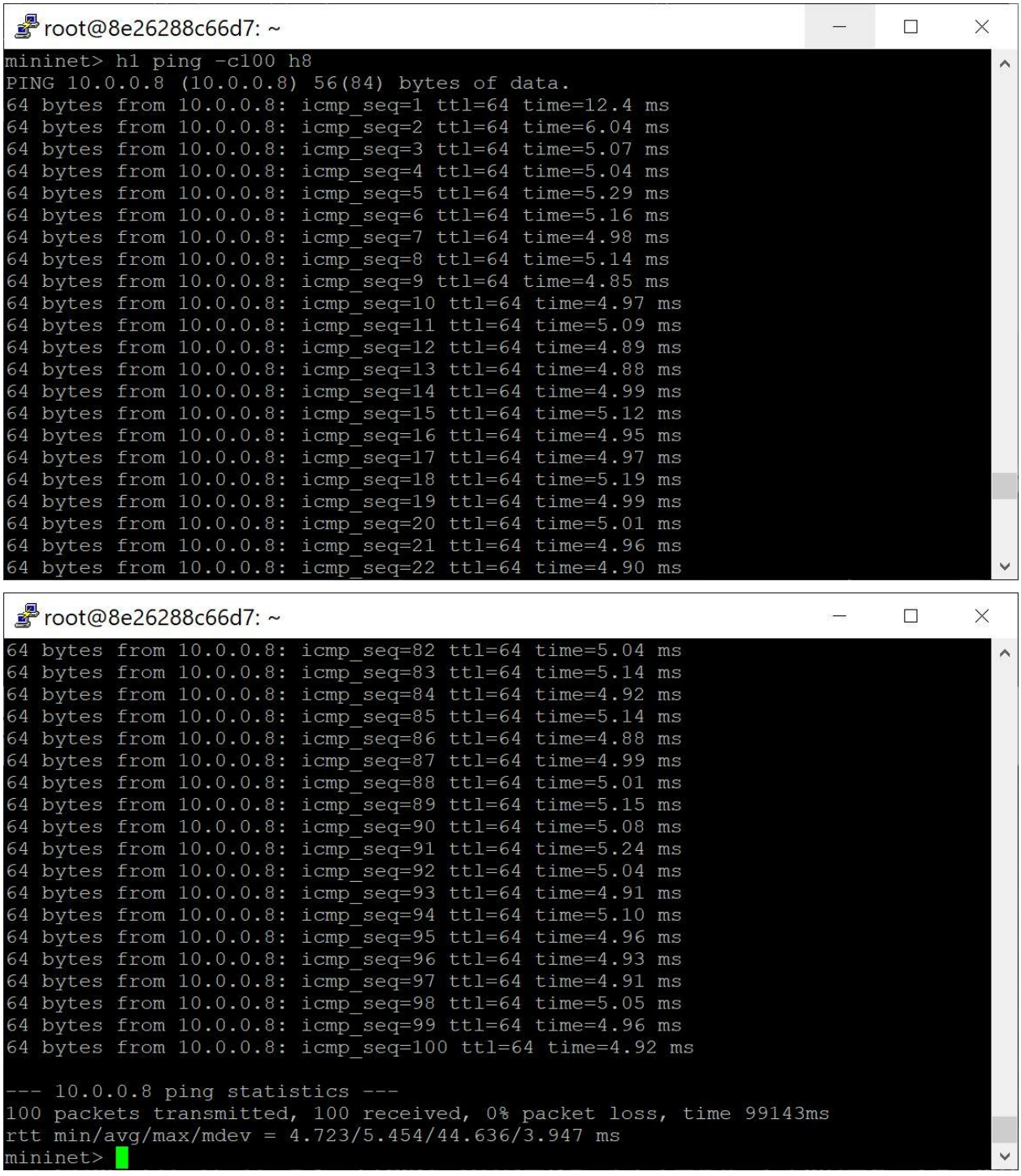

root@8e26288c66d7: ~/pox
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-05 6]
INIT: [00-00-00-00-00-05 6]
INFO:openflow.of_01:[00-00-00-00-00-02 7] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-02 7]
INIT: [00-00-00-00-00-02 7]
PACK IN: [00-00-00-00-00-03 5]
HUB: [00-00-00-00-00-03 5]
RESEND: [00-00-00-00-00-03 5] 65532
PACK IN: [00-00-00-00-00-03 5]
HUB: [00-00-00-00-00-03 5]
RESEND: [00-00-00-00-00-03 5] 65532
PACK IN: [00-00-00-00-00-02 7]
HUB: [00-00-00-00-00-02 7]
RESEND: [00-00-00-00-00-02 7] 65532
PACK IN: [00-00-00-00-00-01 2]
HUB: [00-00-00-00-00-01 2]
RESEND: [00-00-00-00-00-01 2] 65532
PACK IN: [00-00-00-00-00-04 3]
HUB: [00-00-00-00-00-04 3]
RESEND: [00-00-00-00-00-04 3] 65532
PACK IN: [00-00-00-00-00-03 5]
HUB: [00-00-00-00-00-03 5]
RESEND: [00-00-00-00-00-03 5] 65532
PACK IN: [00-00-00-00-00-02 7]


Initialization
launch
start_switch
init
connection.addListeners
Packet comes in
_handle_PacketIn
act_like_hub
resend_packet

2.


root@8e26288c66d7: ~
mininet> h1 ping -c100 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=3.19 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=1.51 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=1.76 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=1.57 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=1.47 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=1.49 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=1.58 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=1.41 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=1.55 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=1.38 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=1.39 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=1.44 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=1.41 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=1.43 ms
64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=1.45 ms
64 bytes from 10.0.0.2: icmp_seq=16 ttl=64 time=1.38 ms
64 bytes from 10.0.0.2: icmp_seq=17 ttl=64 time=1.43 ms
64 bytes from 10.0.0.2: icmp_seq=18 ttl=64 time=1.44 ms
64 bytes from 10.0.0.2: icmp_seq=19 ttl=64 time=1.40 ms
64 bytes from 10.0.0.2: icmp_seq=20 ttl=64 time=1.42 ms
64 bytes from 10.0.0.2: icmp_seq=21 ttl=64 time=1.37 ms
64 bytes from 10.0.0.2: icmp_seq=22 ttl=64 time=1.51 ms


root@8e26288c66d7: ~
64 bytes from 10.0.0.2: icmp_seq=82 ttl=64 time=1.55 ms
64 bytes from 10.0.0.2: icmp_seq=83 ttl=64 time=1.54 ms
64 bytes from 10.0.0.2: icmp_seq=84 ttl=64 time=1.46 ms
64 bytes from 10.0.0.2: icmp_seq=85 ttl=64 time=1.47 ms
64 bytes from 10.0.0.2: icmp_seq=86 ttl=64 time=1.49 ms
64 bytes from 10.0.0.2: icmp_seq=87 ttl=64 time=1.51 ms
64 bytes from 10.0.0.2: icmp_seq=88 ttl=64 time=1.43 ms
64 bytes from 10.0.0.2: icmp_seq=89 ttl=64 time=1.45 ms
64 bytes from 10.0.0.2: icmp_seq=90 ttl=64 time=1.54 ms
64 bytes from 10.0.0.2: icmp_seq=91 ttl=64 time=1.42 ms
64 bytes from 10.0.0.2: icmp_seq=92 ttl=64 time=1.41 ms
64 bytes from 10.0.0.2: icmp_seq=93 ttl=64 time=1.35 ms
64 bytes from 10.0.0.2: icmp_seq=94 ttl=64 time=1.62 ms
64 bytes from 10.0.0.2: icmp_seq=95 ttl=64 time=1.42 ms
64 bytes from 10.0.0.2: icmp_seq=96 ttl=64 time=1.50 ms
64 bytes from 10.0.0.2: icmp_seq=97 ttl=64 time=1.43 ms
64 bytes from 10.0.0.2: icmp_seq=98 ttl=64 time=1.46 ms
64 bytes from 10.0.0.2: icmp_seq=99 ttl=64 time=1.48 ms
64 bytes from 10.0.0.2: icmp_seq=100 ttl=64 time=1.58 ms
--- 10.0.0.2 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99171ms
rtt min/avg/max/mdev = 1.097/1.490/3.195/0.201 ms
mininet>

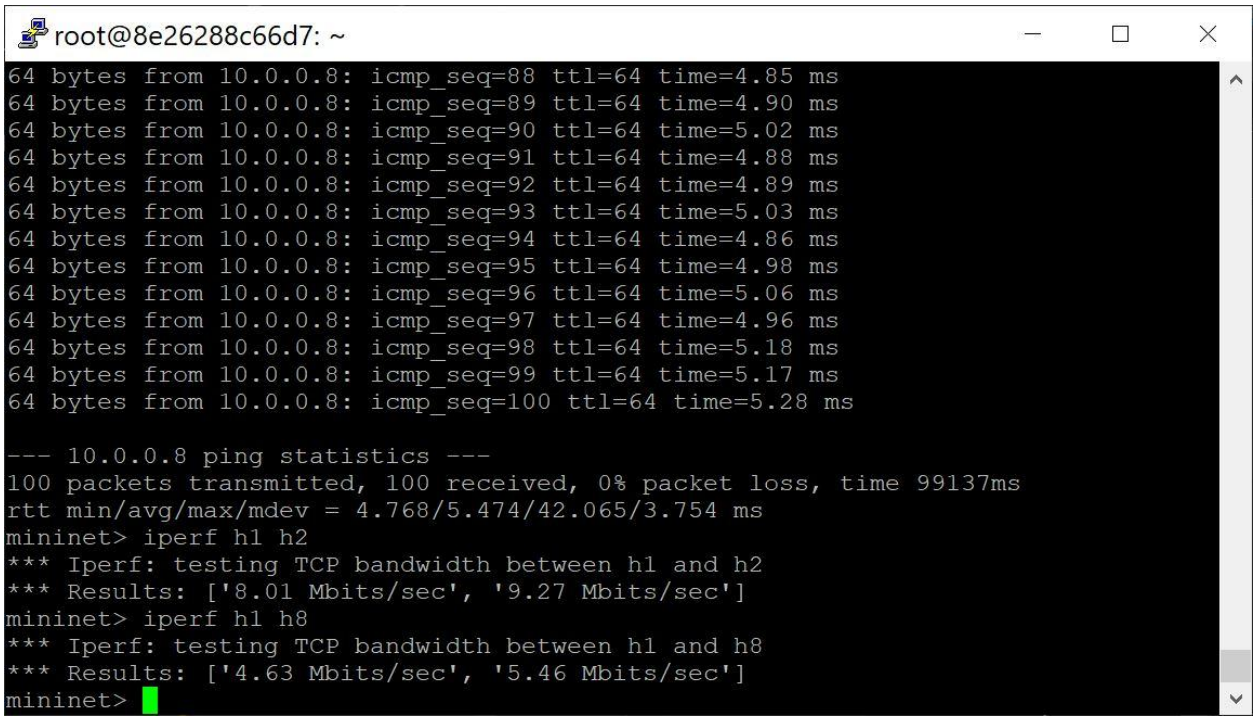


a. h1-h2: avg = 1.490 ms | h1-h8:avg = 5.454 ms

b. h1-h2: min = 1.097 ms, max = 3.195 ms | h1-h8: min 4.723 ms, max = 44.636 ms

c. The difference is h1-h2 takes less time than h1-h8. This is because for h1-h2, packets only need to go through switch s3, but for h1-h8, packets will have to go through switches s3, s2, s1, s5, and s7 so the ping takes longer.

3.



a. “iperf” tests the maximum network bandwidth by measuring how much data can be transferred between two nodes within a given time frame. This is also known as throughput.

b. h1-h2: 8.01 Mbits/sec, h2-h1: 9.27 Mbits/sec | h1-h8: 4.63 Mbits/sec, h8-h1: 5.46 Mbits/sec

c. The throughput for h1-h2 is larger than h1-h8. As h1-h2 only needs to go through one switch, it takes the packets a shorter time to be transferred, so more packets can be transferred within a given time frame. h1-h8 will have to go through switches s3, s2, s1, s5, and s7,

4.

```
mininet> h1 ping -c1 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=2.87 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 2.876/2.876/2.876/0.000 ms
mininet>
```

root@8e26288c66d7: ~/pox

```
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-04 3]
INFO:openflow.of_01:[00-00-00-00-00-06 4] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-06 4]
INFO:openflow.of_01:[00-00-00-00-00-03 5] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-03 5]
INFO:openflow.of_01:[00-00-00-00-00-05 6] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-05 6]
INFO:openflow.of_01:[00-00-00-00-00-02 7] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-02 7]
PACK IN: [00-00-00-00-00-03 5]
PACK IN: [00-00-00-00-00-03 5]
PACK IN: [00-00-00-00-00-02 7]
PACK IN: [00-00-00-00-00-02 7]
PACK IN: [00-00-00-00-00-01 2]
PACK IN: [00-00-00-00-00-01 2]
PACK IN: [00-00-00-00-00-04 3]
PACK IN: [00-00-00-00-00-04 3]
PACK IN: [00-00-00-00-00-05 6]
PACK IN: [00-00-00-00-00-07 1]
PACK IN: [00-00-00-00-00-06 4]
PACK IN: [00-00-00-00-00-05 6]
PACK IN: [00-00-00-00-00-07 1]
PACK IN: [00-00-00-00-00-06 4]
```

All of the switches observe traffic for h1-h2 and h1-h8. I put in ‘print(“PACK IN:”, self.connection)’ in the _handle_PacketIn function to see this.

Task 3

1.

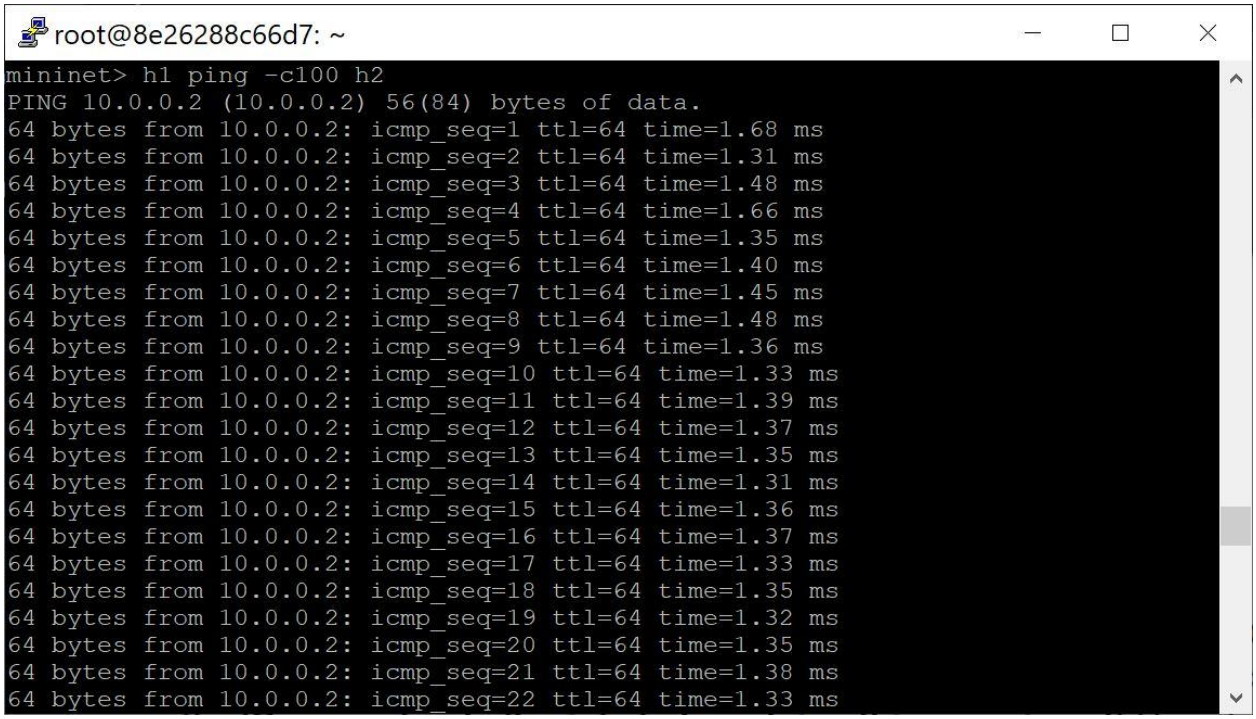
```
mininet> h1 ping -c2 h8
PING 10.0.0.8 (10.0.0.8) 56(84) bytes of data.
64 bytes from 10.0.0.8: icmp_seq=1 ttl=64 time=8.47 ms
64 bytes from 10.0.0.8: icmp_seq=2 ttl=64 time=7.15 ms

--- 10.0.0.8 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 7.154/7.814/8.474/0.660 ms
mininet>
```

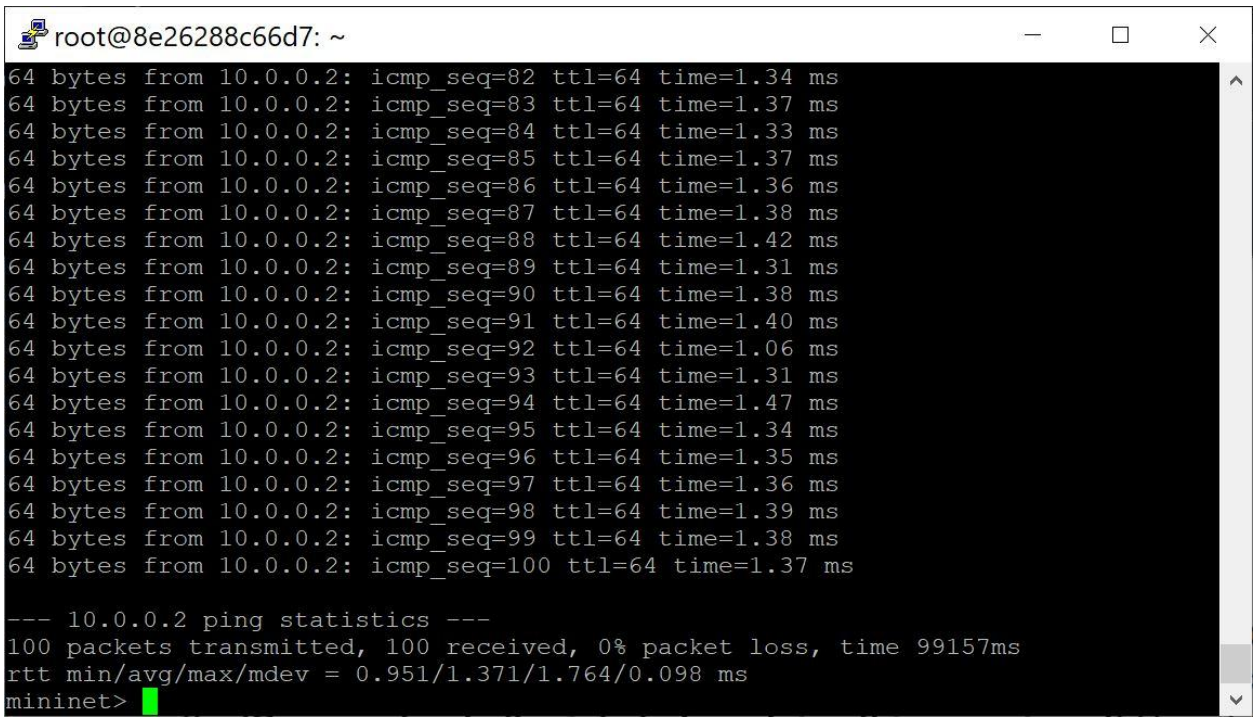
root@8e26288c66d7: ~/pox

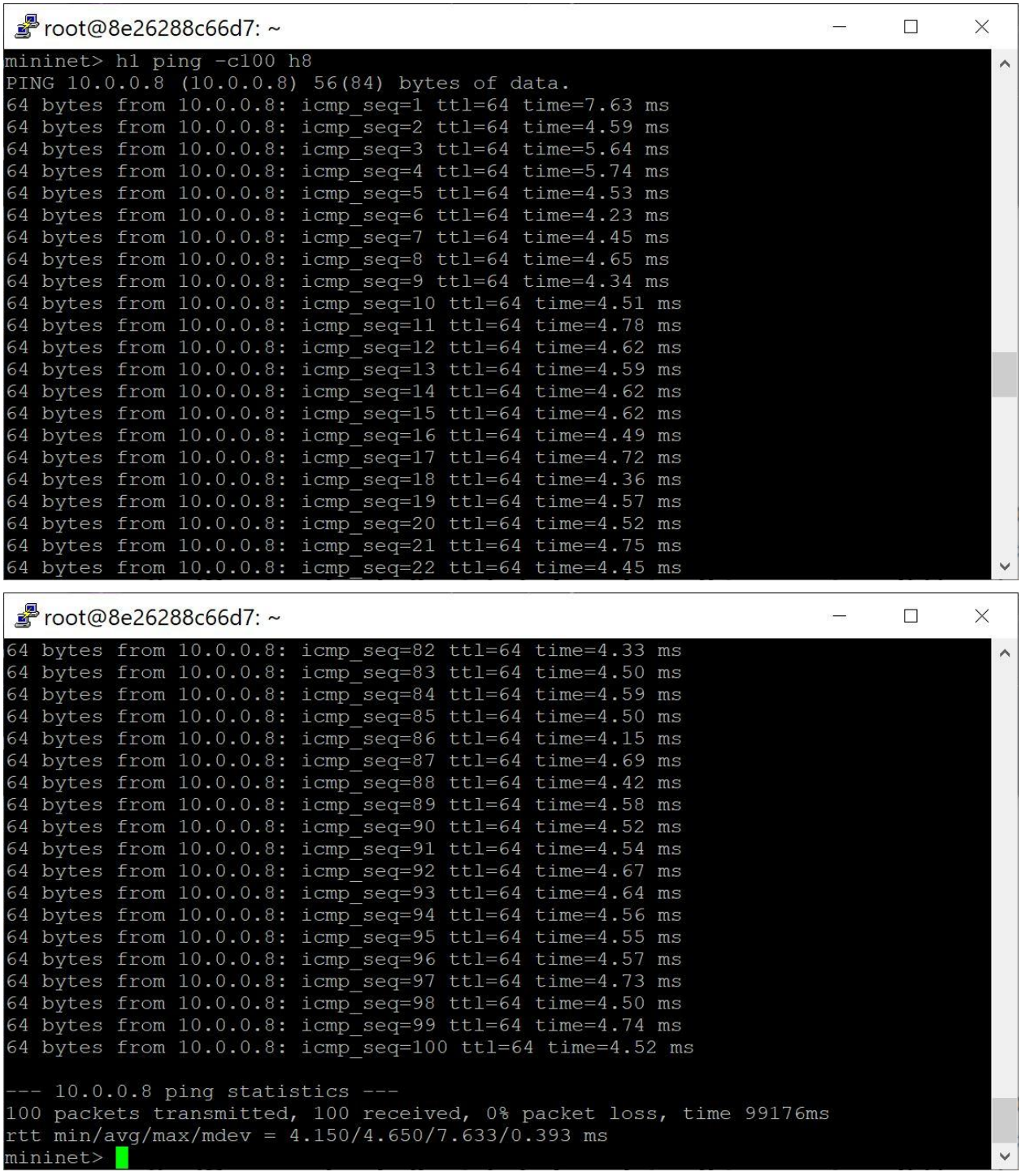
```
6a:90:69:72:de:67 not known, resend to everybody
Learning that da:80:ea:68:51:5b is attached at port 1
6a:90:69:72:de:67 not known, resend to everybody
Learning that da:80:ea:68:51:5b is attached at port 1
6a:90:69:72:de:67 not known, resend to everybody
Learning that 6a:90:69:72:de:67 is attached at port 3
da:80:ea:68:51:5b destination known. only send message to it
Learning that da:80:ea:68:51:5b is attached at port 1
6a:90:69:72:de:67 not known, resend to everybody
Learning that 6a:90:69:72:de:67 is attached at port 3
da:80:ea:68:51:5b destination known. only send message to it
Learning that 6a:90:69:72:de:67 is attached at port 2
da:80:ea:68:51:5b destination known. only send message to it
Learning that 6a:90:69:72:de:67 is attached at port 1
da:80:ea:68:51:5b destination known. only send message to it
Learning that 6a:90:69:72:de:67 is attached at port 1
da:80:ea:68:51:5b destination known. only send message to it
6a:90:69:72:de:67 destination known. only send message to it
6a:90:69:72:de:67 destination known. only send message to it
6a:90:69:72:de:67 destination known. only send message to it
6a:90:69:72:de:67 destination known. only send message to it
6a:90:69:72:de:67 destination known. only send message to it
da:80:ea:68:51:5b destination known. only send message to it
da:80:ea:68:51:5b destination known. only send message to it
```

The "MAC to Port" map is established by the Tutorial object for each switch storing the MAC address as key and the port as its value for the MAC addresses that are not known when a packet is received from said MAC address. This way, the switch can send packets directly to the ports it needs to send packets to if the packet destination MAC address and port is known.

2.


root@8e26288c66d7: ~
mininet> h1 ping -c100 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=1.68 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=1.31 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=1.48 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=1.66 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=1.35 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=1.40 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=1.45 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=1.48 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=1.36 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=1.33 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=1.39 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=1.37 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=1.35 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=1.31 ms
64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=1.36 ms
64 bytes from 10.0.0.2: icmp_seq=16 ttl=64 time=1.37 ms
64 bytes from 10.0.0.2: icmp_seq=17 ttl=64 time=1.33 ms
64 bytes from 10.0.0.2: icmp_seq=18 ttl=64 time=1.35 ms
64 bytes from 10.0.0.2: icmp_seq=19 ttl=64 time=1.32 ms
64 bytes from 10.0.0.2: icmp_seq=20 ttl=64 time=1.35 ms
64 bytes from 10.0.0.2: icmp_seq=21 ttl=64 time=1.38 ms
64 bytes from 10.0.0.2: icmp_seq=22 ttl=64 time=1.33 ms


root@8e26288c66d7: ~
64 bytes from 10.0.0.2: icmp_seq=82 ttl=64 time=1.34 ms
64 bytes from 10.0.0.2: icmp_seq=83 ttl=64 time=1.37 ms
64 bytes from 10.0.0.2: icmp_seq=84 ttl=64 time=1.33 ms
64 bytes from 10.0.0.2: icmp_seq=85 ttl=64 time=1.37 ms
64 bytes from 10.0.0.2: icmp_seq=86 ttl=64 time=1.36 ms
64 bytes from 10.0.0.2: icmp_seq=87 ttl=64 time=1.38 ms
64 bytes from 10.0.0.2: icmp_seq=88 ttl=64 time=1.42 ms
64 bytes from 10.0.0.2: icmp_seq=89 ttl=64 time=1.31 ms
64 bytes from 10.0.0.2: icmp_seq=90 ttl=64 time=1.38 ms
64 bytes from 10.0.0.2: icmp_seq=91 ttl=64 time=1.40 ms
64 bytes from 10.0.0.2: icmp_seq=92 ttl=64 time=1.06 ms
64 bytes from 10.0.0.2: icmp_seq=93 ttl=64 time=1.31 ms
64 bytes from 10.0.0.2: icmp_seq=94 ttl=64 time=1.47 ms
64 bytes from 10.0.0.2: icmp_seq=95 ttl=64 time=1.34 ms
64 bytes from 10.0.0.2: icmp_seq=96 ttl=64 time=1.35 ms
64 bytes from 10.0.0.2: icmp_seq=97 ttl=64 time=1.36 ms
64 bytes from 10.0.0.2: icmp_seq=98 ttl=64 time=1.39 ms
64 bytes from 10.0.0.2: icmp_seq=99 ttl=64 time=1.38 ms
64 bytes from 10.0.0.2: icmp_seq=100 ttl=64 time=1.37 ms
--- 10.0.0.2 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99157ms
rtt min/avg/max/mdev = 0.951/1.371/1.764/0.098 ms
mininet>

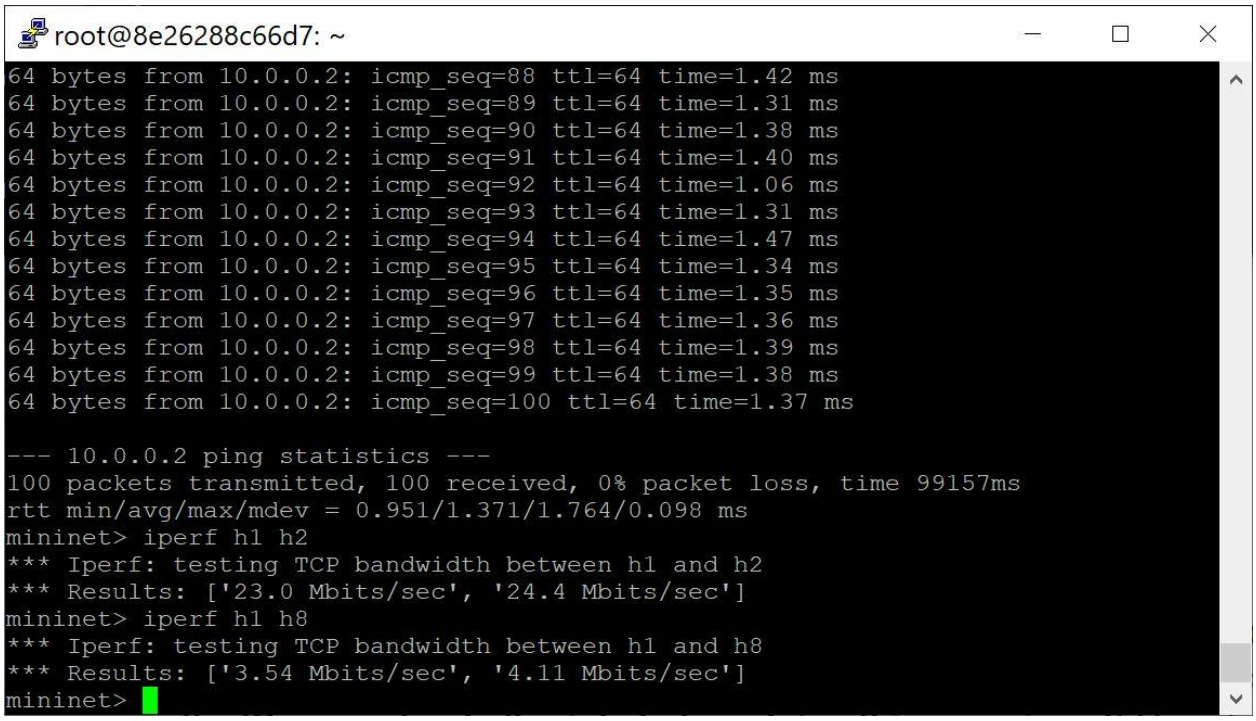


a. h1-h2: avg = 1.371 ms | h1-h8:avg = 4.650 ms

b. h1-h2: min = 0.951 ms, max = 1.764 ms | h1-h8: min 4.723 ms, max = 7.633 ms

c. The pings take a shorter time than in Task 2 because the switches know where to send their packets after the first packet flood. It does not have to send packets to every port and instead it can send the packets to the specific destination port.

3.



a. h1-h2: 23.0 Mbits/sec, h2-h1: 24.4 Mbits/sec | h1-h8: 3.54 Mbits/sec, h8-h1: 4.11 Mbits/sec

b. The throughput for h1-h2 is much larger as the destination port is known by the sender, so the data transfer is faster and more data can be sent within a given time frame. There is not much of a change in throughput for h1-h8. This could be because the network path still consists of five switches and the map might not have a big effect in the short “iperf” measurement time frame.