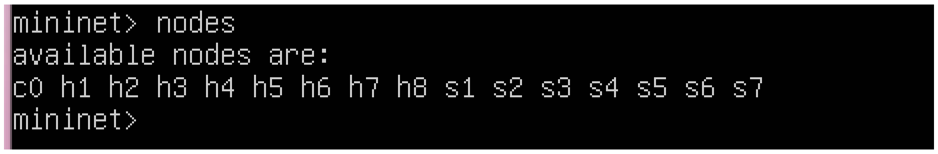
Homework 3 Report

Krithik Naralasetty

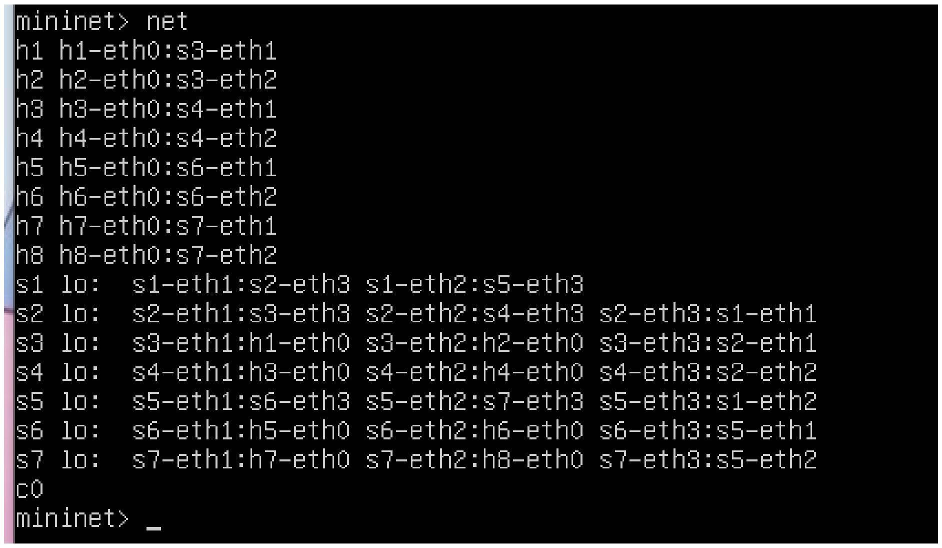
Task 1

Question 1: Output of “nodes” and “net”



```
mininet> nodes
available nodes are:
c0 h1 h2 h3 h4 h5 h6 h7 h8 s1 s2 s3 s4 s5 s6 s7
mininet>
```

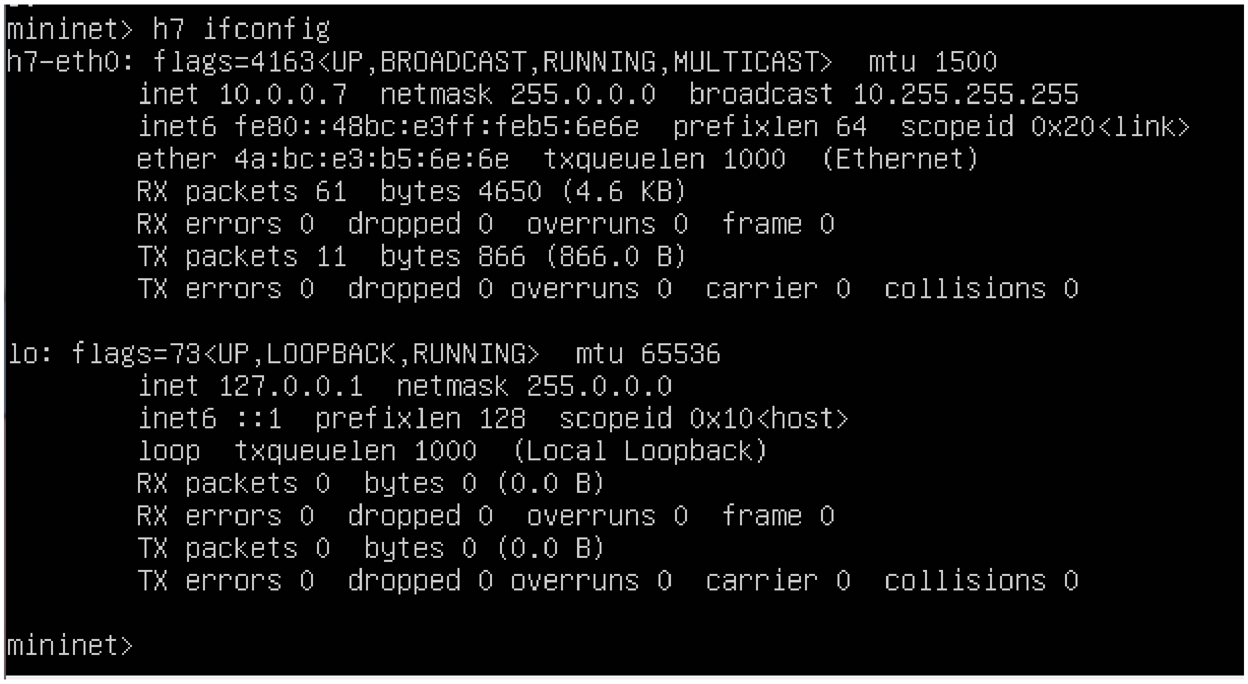
It basically lists out all the hosts, switches, and controllers assigned to the topology of devices.



```
mininet> net
h1 h1-eth0:s3-eth1
h2 h2-eth0:s3-eth2
h3 h3-eth0:s4-eth1
h4 h4-eth0:s4-eth2
h5 h5-eth0:s6-eth1
h6 h6-eth0:s6-eth2
h7 h7-eth0:s7-eth1
h8 h8-eth0:s7-eth2
s1 lo:  s1-eth1:s2-eth3 s1-eth2:s5-eth3
s2 lo:  s2-eth1:s3-eth3 s2-eth2:s4-eth3 s2-eth3:s1-eth1
s3 lo:  s3-eth1:h1-eth0 s3-eth2:h2-eth0 s3-eth3:s2-eth1
s4 lo:  s4-eth1:h3-eth0 s4-eth2:h4-eth0 s4-eth3:s2-eth2
s5 lo:  s5-eth1:s6-eth3 s5-eth2:s7-eth3 s5-eth3:s1-eth2
s6 lo:  s6-eth1:h5-eth0 s6-eth2:h6-eth0 s6-eth3:s5-eth1
s7 lo:  s7-eth1:h7-eth0 s7-eth2:h8-eth0 s7-eth3:s5-eth2
c0
mininet> _
```

Net gives us the physical network configuration (port connections) of the devices which were linked together

Question 2: Output of “h7 ifconfig”

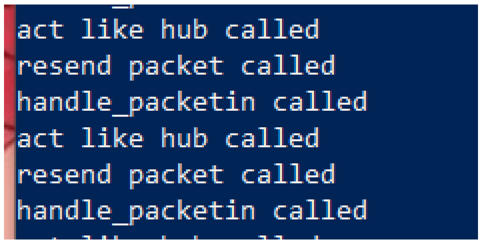


```
mininet> h7 ifconfig
h7-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.7  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::48bc:e3ff:feb5:6e6e  prefixlen 64  scopeid 0x20<link>
        ether 4a:bc:e3:b5:6e:6e  txqueuelen 1000  (Ethernet)
        RX packets 61  bytes 4650 (4.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 11  bytes 866 (866.0 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 1000  (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>
```

The interface configuration of h7 is shown

Task 2

Question 1: Function Call graph of the controller

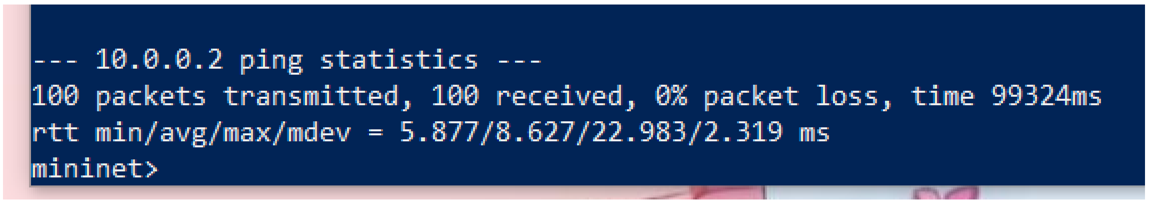


```
act like hub called
resend packet called
handle_packetin called
act like hub called
resend packet called
handle_packetin called
```

The function call graph of the controller looked like the above.
The first call was made to the “start switch” function, of course.
Then the tutorial call was called, along with the function in the order

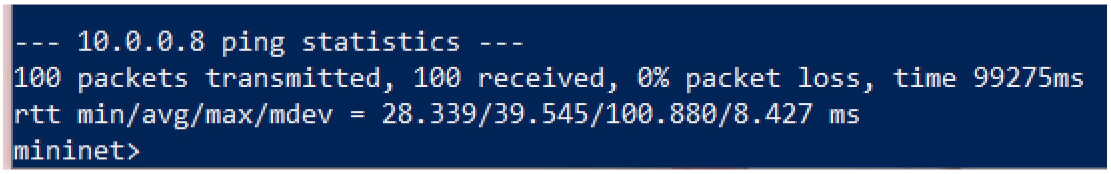
1. handle_packetIn function
2. act_like_hub function
3. Resend_packet function

Question 2: Ping h1 -> h2 and h1 -> h8. Stats



```
--- 10.0.0.2 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99324ms
rtt min/avg/max/mdev = 5.877/8.627/22.983/2.319 ms
mininet>
```

h1 ping h2

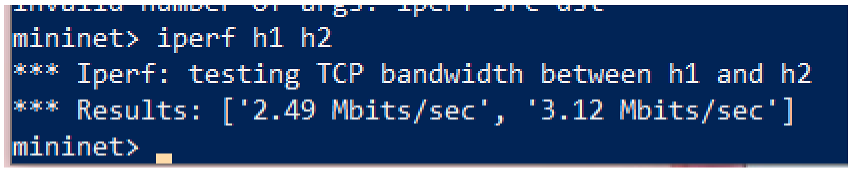


```
--- 10.0.0.8 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99275ms
rtt min/avg/max/mdev = 28.339/39.545/100.880/8.427 ms
mininet>
```

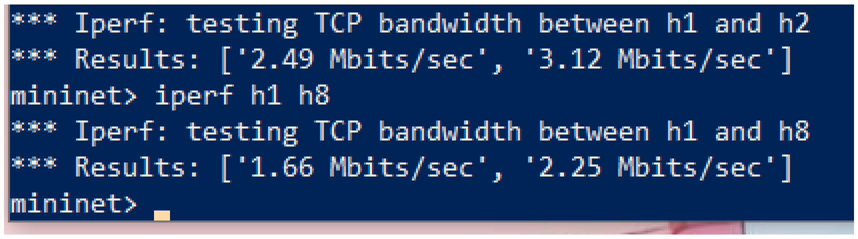
h1 ping h8

A. Pinging average for each case
   a. h1 ping h2: average is 8.627
   b. h1 ping h8:: average is 39.545
B. Minimum and Maximum ping observed
   a. h1 ping h2: minimum was 5.877 and maximum was 22.983
   b. h1 ping h8: minimum was 28.339 and maximum was 100.880
C. The difference
   a. h1 ping h2: was 17.106. The maximum RTT was because the forwarding path had to be established into the controller for the first time (it was observed on the first ping).
   b. h1 ping h8: was 72.541. The maximum RTT was because of the forwarding path was calculated at the first ping
   c. Explanation: The difference between h1 ping h2 (being fast) and h1 ping h8 (being slow) was because of the number of connections between each host. h1 was on the opposite side of h8 in the tree, but h1 and h2 had only a single switch between them. That was the reason for the delay in the RTTs of both the pings.

Question 3: iperf of h1->h2 and iperf h1->h8

A. "Iperf" is to measure the bandwidth between source and destination, for the network performance and quality of a network line
B. Throughput for each case is:
   a. 

```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['2.49 Mbits/sec', '3.12 Mbits/sec']
mininet>
```

b. 

```
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['2.49 Mbits/sec', '3.12 Mbits/sec']
mininet> iperf h1 h8
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['1.66 Mbits/sec', '2.25 Mbits/sec']
mininet>
```

C. Differences between the throughputs of the bandwidth are because of the distance between the hosts. The number of switches between h1 and h8 (3 switches) is far more compared to between h1 and h2 (1 switch) and that is why the bandwidth minimizes

Question 4:Which switches Observe Traffic

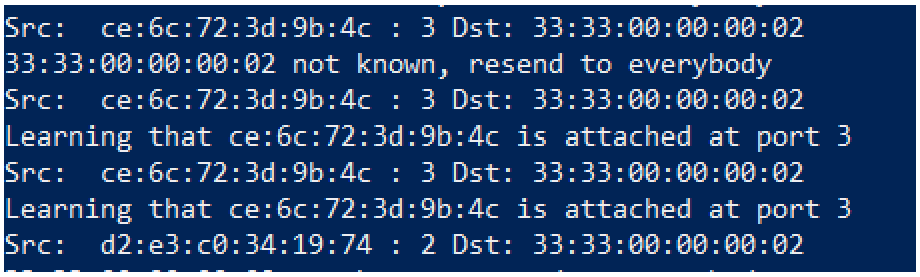
A. Switches that observe traffic
All the switches observed traffic because the network had to be flooded to search for the destination port whenever we try to ping between 2 hosts

B. How did I find it out?
The pox debug output shows the flooding of packets into different MAC addresses, if we add the print statement of “print(str(self.mac_to_port) + " checked")” in the act_like_hub function

Task 3

Question 1: Describe how the above code works

The code basically describes the action of building rules in the OpenFlow controller attached to the mininet. The code basically checks the controller if a rule was established between the given hosts. If it doesn’t exist, the packet is basically flooded over the entire network to try and find the destination and the port it is attached to, by using OFPP_ALL which basically floods all the physical ports except the input port.



```
Src:  ce:6c:72:3d:9b:4c : 3 Dst: 33:33:00:00:00:02
33:33:00:00:00:02 not known, resend to everybody
Src:  ce:6c:72:3d:9b:4c : 3 Dst: 33:33:00:00:00:02
Learning that ce:6c:72:3d:9b:4c is attached at port 3
Src:  ce:6c:72:3d:9b:4c : 3 Dst: 33:33:00:00:00:02
Learning that ce:6c:72:3d:9b:4c is attached at port 3
Src:  d2:e3:c0:34:19:74 : 2 Dst: 33:33:00:00:00:02
```

Question 2: Ping h1->h2 and ping h1->h8

A. How long did it take (on average) to ping for each case?

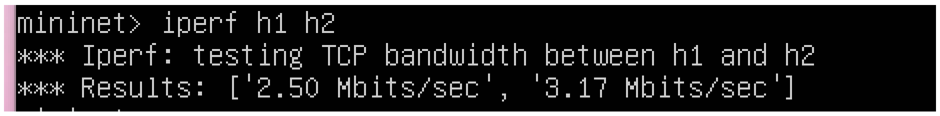
   a. For case of h1->h2 it took an average of 8.270
   b. For the case of h1->h8 it took an average of 37.917

B. What are the minimum and maximum ping you have observed?
   a. h1 ping h2: minimum was 5.799 and maximum was 33.881
   b. h1 ping h8: minimum was 28.721 and maximum was 95.079

C. Any difference from Task 2 and why do you think there is a change if there is?
   a. The pings were relatively faster than before the switches were learning (ie better than when they just acted as hubs).
   b. The maximum RTT for h1-h2 was larger because the controller had to introduce the rule.
   c. Explanation: The change in RTT was observed because the controller was able to distinguish the port numbers and built the forwarding rules. (Mac address of host was assigned to a port)

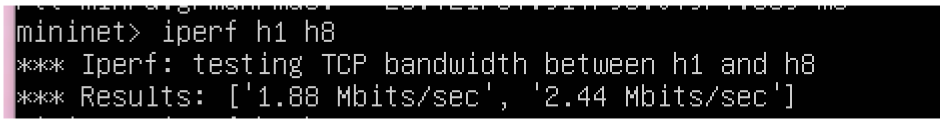
Question 3:Run iperf h1->h2 and iperf h1->h8

A. What is the throughput for each case?
   The throughput for each case is given below
   Iperf h1 h2 is below



```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['2.50 Mbits/sec', '3.17 Mbits/sec']
```

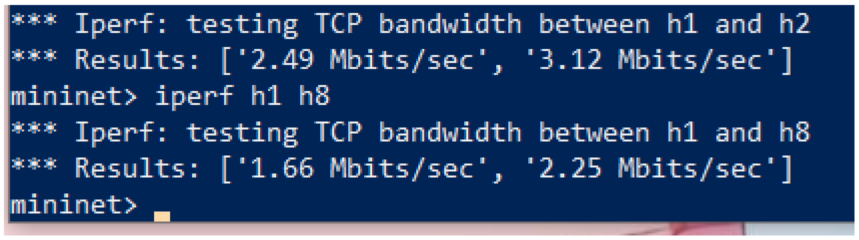
   Iperf h1 h8 is below



```
mininet> iperf h1 h8
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['1.88 Mbits/sec', '2.44 Mbits/sec']
```

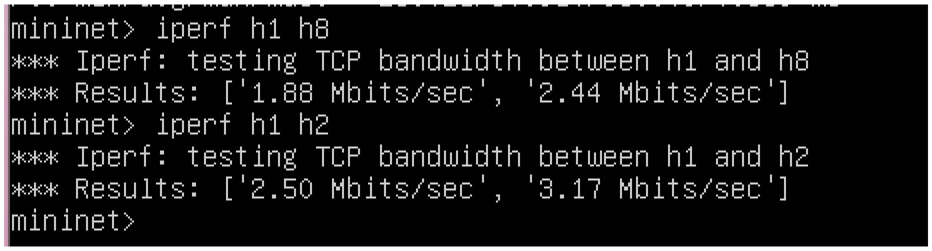
B. What is the difference from Task 2 and why do you think there is a change if there is?
   The observed change is that the throughput was higher than observed in Task 2. It was likely because of the implementation of MAC to Port rules in the controller of OpenFlow.

   Without learning



```
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['2.49 Mbits/sec', '3.12 Mbits/sec']
mininet> iperf h1 h8
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['1.66 Mbits/sec', '2.25 Mbits/sec']
mininet> 
```

   With learning



```
[illegible]
mininet> iperf h1 h8
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['1.88 Mbits/sec', '2.44 Mbits/sec']
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['2.50 Mbits/sec', '3.17 Mbits/sec']
mininet>
```