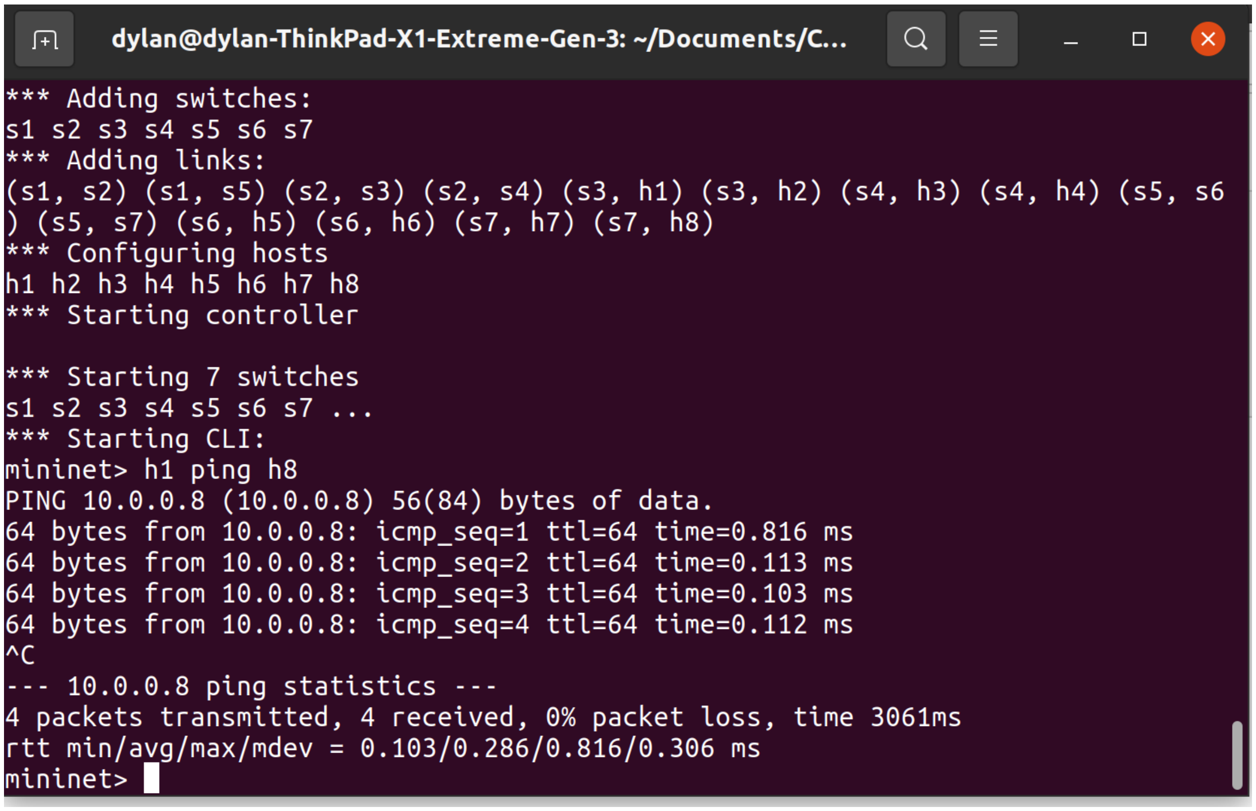
Dylan Hoover
COEN 241

HW 3 Report

Task 1: Defining custom topologies

$ sudo mn --custom binary_tree.py --topo binary_tree
mininet> h1 ping h8

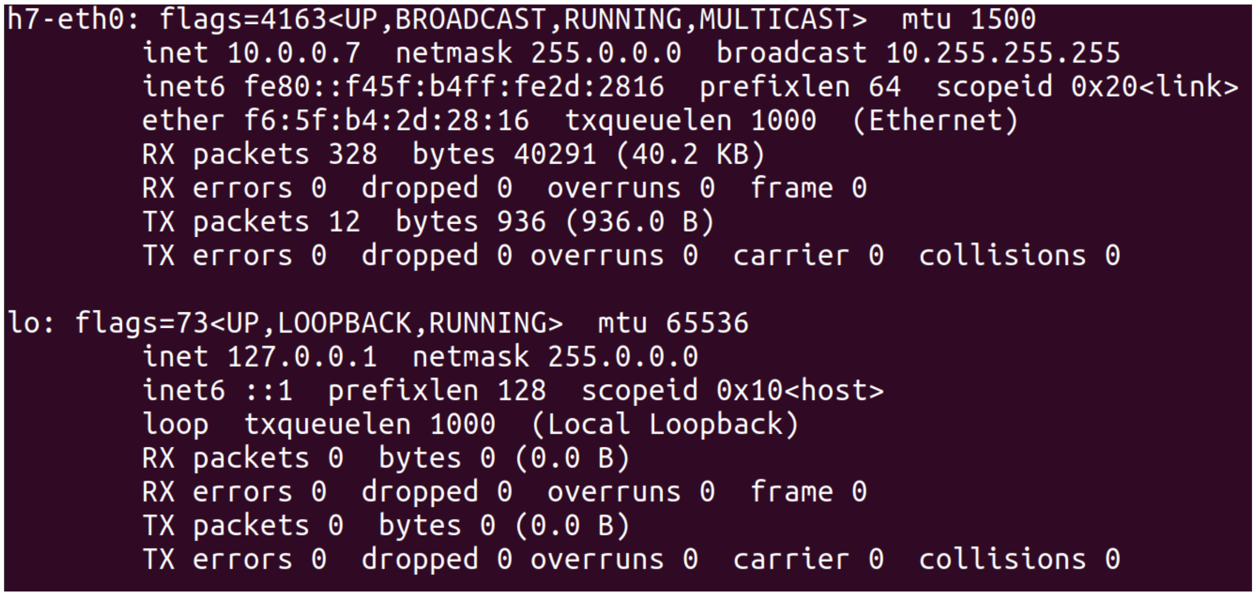


```
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7
*** Adding links:
(s1, s2) (s1, s5) (s2, s3) (s2, s4) (s3, h1) (s3, h2) (s4, h3) (s4, h4) (s5, s6
) (s5, s7) (s6, h5) (s6, h6) (s7, h7) (s7, h8)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller

*** Starting 7 switches
s1 s2 s3 s4 s5 s6 s7 ...
*** Starting CLI:
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=0.816 ms
64 bytes from 10.0.0.8: icmp_seq=2 ttl=64 time=0.113 ms
64 bytes from 10.0.0.8: icmp_seq=3 ttl=64 time=0.103 ms
64 bytes from 10.0.0.8: icmp_seq=4 ttl=64 time=0.112 ms
^C
--- 10.0.0.8 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3061ms
rtt min/avg/max/mdev = 0.103/0.286/0.816/0.306 ms
mininet>
```

What is the output of “nodes” and “net”:
available nodes are:
h1 h2 h3 h4 h5 h6 h7 h8 s1 s2 s3 s4 s5 s6 s7

What is the output of “h7 ifconfig”:



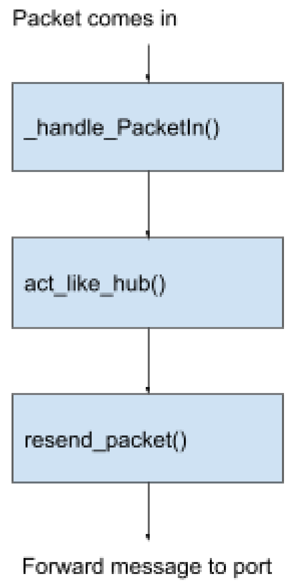
```
h7-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.7  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::f45f:b4ff:fe2d:2816  prefixlen 64  scopeid 0x20<link>
        ether f6:5f:b4:2d:28:16  txqueuelen 1000  (Ethernet)
        RX packets 328  bytes 40291 (40.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 12  bytes 936 (936.0 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
```

Task 2: Analyze the “of_tutorial’ controller

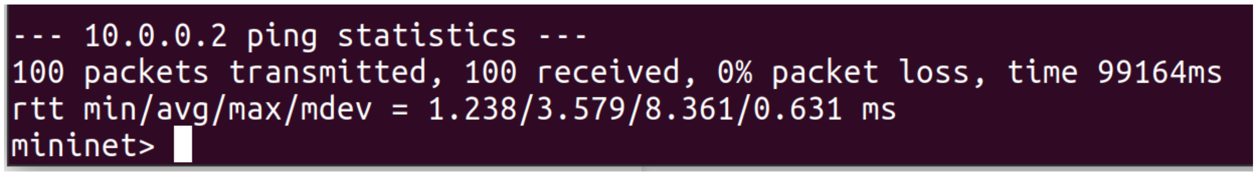
1. Draw the function call graph of this controller. For example, once a packet comes to the controller, which function is the first to be called, which one is the second, and so forth?

The function call for when a packet comes in goes in this order: _handle_Packetin() -> act_like_hub() -> resend_packet()



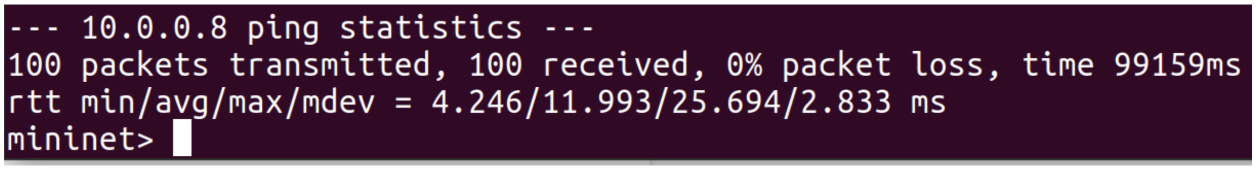
2. Have h1 ping h2, and h1 ping h8 for 100 times (e.g., h1 ping -c100 p2).
a. How long does it take (on average) to ping for each case?
b. What is the minimum and maximum ping you have observed?
c. What is the difference, and why?

>h1 ping -c100 h2



```
--- 10.0.0.2 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99164ms
rtt min/avg/max/mdev = 1.238/3.579/8.361/0.631 ms
mininet>
```

>h1 ping -c100 h8



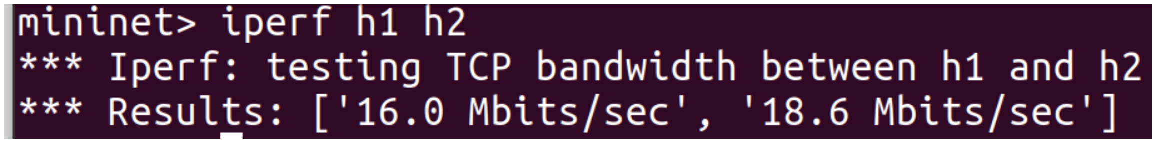
```
--- 10.0.0.8 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99159ms
rtt min/avg/max/mdev = 4.246/11.993/25.694/2.833 ms
mininet>
```

a. Average:
   h1 to h2: 3.579 ms
   h1 to h8: 11.993 ms
b. Minimum time
   h1 to h2: 1.238 ms
   h1 to h8 4.246 ms
   Maximum time
   h1 to h2: 8.361 ms
   h1 to h8: 25.694 ms
c. The difference between h1,h2 ping is faster than h1,h8 ping because of the number of connections between the hosts. There is only one switch between h1 and h2 whereas there are three switches between h1 and h8. More switches cause congestion and increase the time it takes to send packets.

3. Run “iperf h1 h2” and “iperf h1 h8”
a. What is “iperf” used for?
b. What is the throughput for each case?
c. What is the difference, and explain the reasons for the difference.

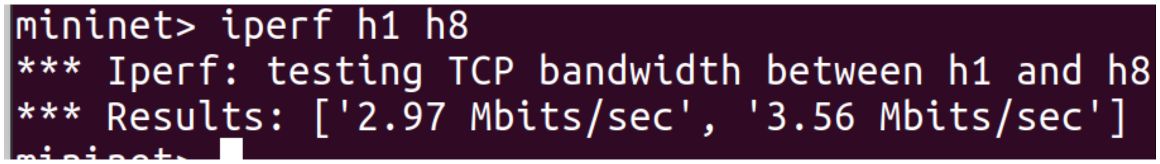
a. ‘iperf’ is used to test the bandwidth between two hosts
b. Throughput shown below:

> iperf h1 h2



```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['16.0 Mbits/sec', '18.6 Mbits/sec']
```

> iperf h1 h8



```
mininet> iperf h1 h8
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['2.97 Mbits/sec', '3.56 Mbits/sec']
```

c. There is a difference in throughput due to the distance between the hosts, thus the increased distance causes congestion to the network and less throughput.

4. Which of the switches observe traffic? Please describe your way for observing such traffic on switches (e.g., adding some functions in the "of_tutorial" controller).

When using h1->h2 and h1->h8 the switches that get used the most are s1, s2, s3, s5, and s7. This can be observed by using the _handle_PacketIn() function.

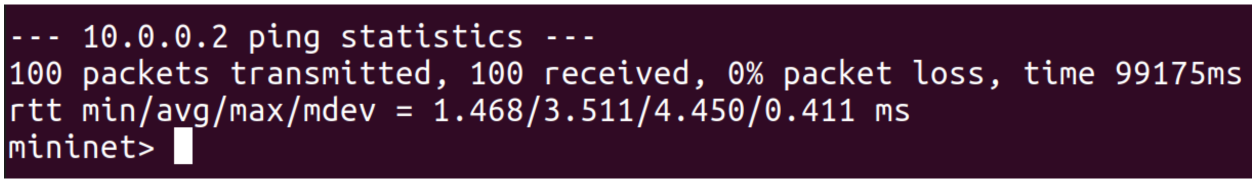
Task 3: MAC Learning Controller

1. Describe how the above code works, such as how the "MAC to Port" map is established. You could use a 'ping' example to describe the establishment process (e.g., h1 ping h2).

In of_tutorial the act_like_switch function is now in charge of routing the MAC addresses to a particular port number utilizing mac_to_port dictionary. If a destination is not known packets will be flooded to all destinations until the correct destination is found, that destination will then be added to the dictionary. The next time the destination will just be looked up in the dictionary, this is what we call mac learning

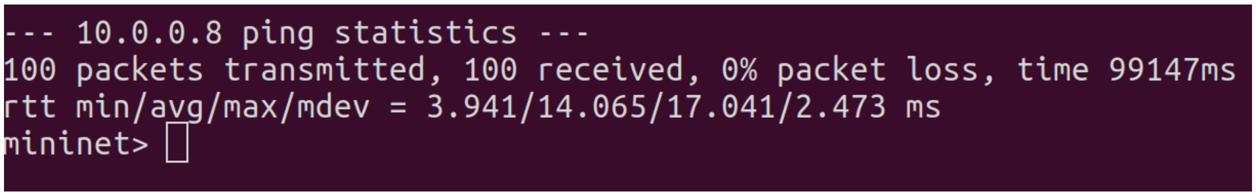
2. (Comment out all prints before doing this experiment) Have h1 ping h2, and h1 ping h8 for 100 times (e.g., h1 ping -c100 p2).
a. How long did it take (on average) to ping for each case?
b. What is the minimum and maximum ping you have observed?
c. Any difference from Task 2 and why do you think there is a change if there is?

>h1 ping -c100 h2



```
--- 10.0.0.2 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99175ms
rtt min/avg/max/mdev = 1.468/3.511/4.450/0.411 ms
mininet>
```

>h1 ping -c100 h8

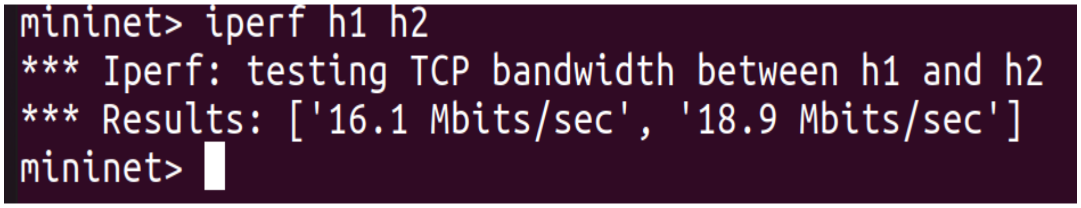


```
--- 10.0.0.8 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99147ms
rtt min/avg/max/mdev = 3.941/14.065/17.041/2.473 ms
mininet>
```

a. Average:
   h1->h2: 3.511
   h1->h8: 14.065
b. Minimum:
   h1->h2: 1.468
   h1->h8: 3.941
   Maximum:
   h1->h2: 4.450
   h1->h8: 17.041
c. Compared to task 2 the pings were relatively faster in task 3, this is especially noticeable in h1->h8. This change is due to the controller now knowing exactly where to send the packets based on the mac address opposed to flooding of packets in task 2.

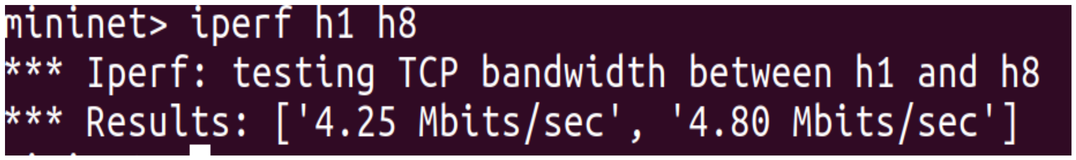
3. Q.3 Run "iperf h1 h2" and "iperf h1 h8".
a. What is the throughput for each case?
b. What is the difference from Task 2 and why do you think there is a change if there is?

a. Throughput:
   > iperf h1 h2



```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['16.1 Mbits/sec', '18.9 Mbits/sec']
mininet>
```

>iperf h1 h8



```
mininet> iperf h1 h8
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['4.25 Mbits/sec', '4.80 Mbits/sec']
```

b. The throughput in both cases is higher compared to task 2. This is due to the implementation of MAC learning which enabled mapping of the mac addresses so there was less congestion in the network.