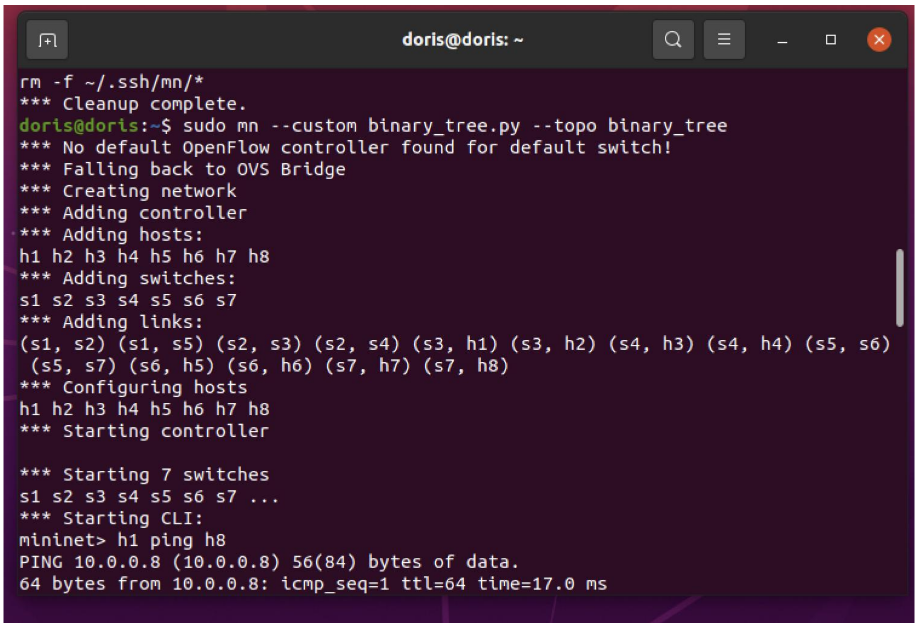
Chuandi Zhou

HW3 Report

Task 1:



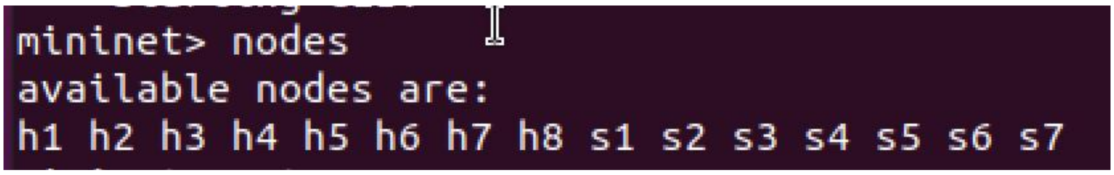
```
rm -f ~/.ssh/mn/*
*** Cleanup complete.
doris@doris:~$ sudo mn --custom binary_tree.py --topo binary_tree
*** No default OpenFlow controller found for default switch!
*** Falling back to OVS Bridge
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
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=17.0 ms
```

Questions

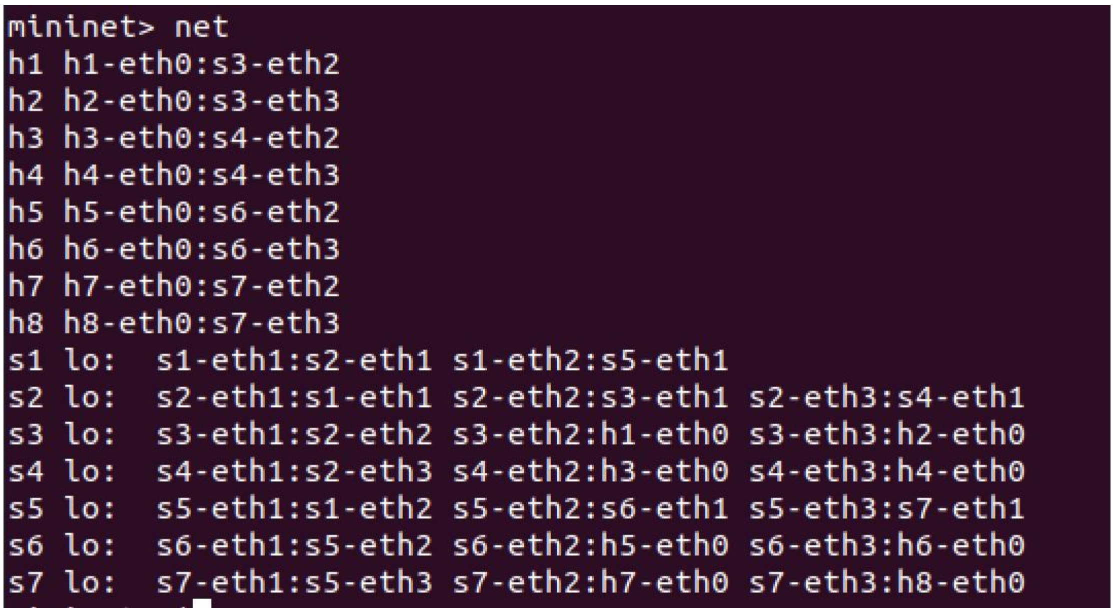
1. What is the output of “nodes” and “net”

nodes



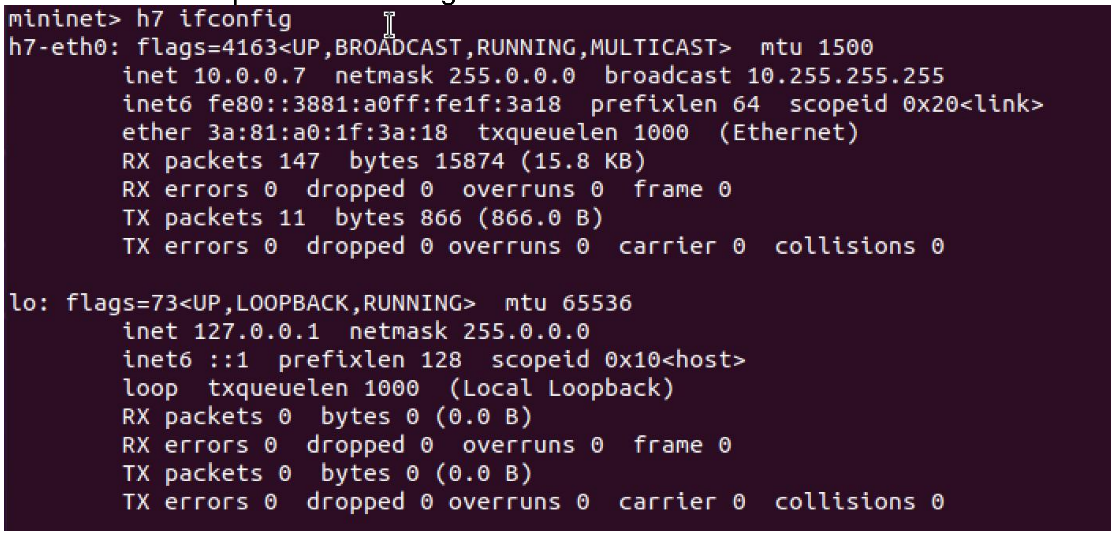
```
mininet> nodes
available nodes are:
h1 h2 h3 h4 h5 h6 h7 h8 s1 s2 s3 s4 s5 s6 s7
```

net



```
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
```

2. What is the output of “h7 ifconfig”

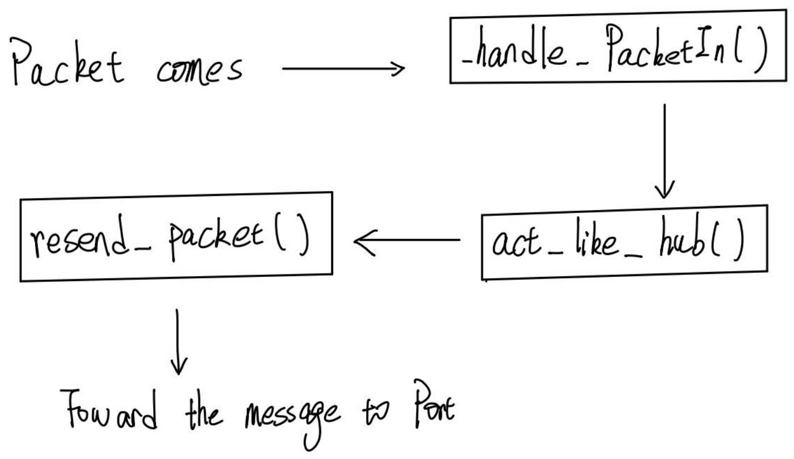


```
mininet> h7 ifconfig
h7-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.7  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::3881:a0ff:fe1f:3a18  prefixlen 64  scopeid 0x20<link>
        ether 3a:81:a0:1f:3a:18  txqueuelen 1000  (Ethernet)
        RX packets 147  bytes 15874 (15.8 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
```

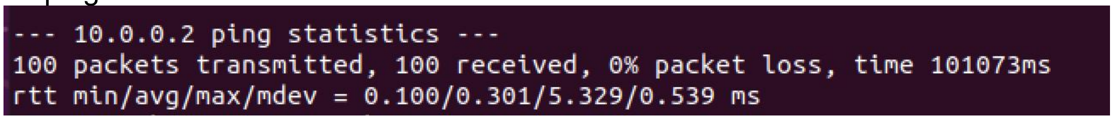
Task 2:

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?



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?

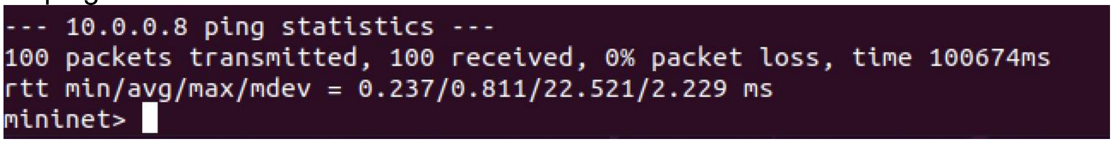
h1 ping -c100 h2



```
--- 10.0.0.2 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 101073ms
rtt min/avg/max/mdev = 0.100/0.301/5.329/0.539 ms
```

avg:0.301

h1 ping -c100 h8



```
--- 10.0.0.8 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 100674ms
rtt min/avg/max/mdev = 0.237/0.811/22.521/2.229 ms
mininet>
```

avg:0.811

b. What is the minimum and maximum ping you have observed?

h1 ping h2:

min:0.100 max:5.329

h1 ping h8:

min:0.237 max:22.521

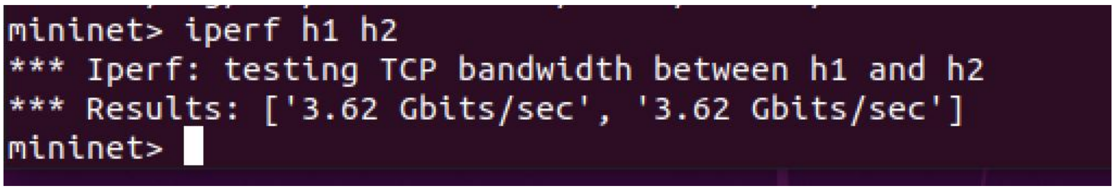
c. What is the difference, and why?

Clearly, h1 ping h8 is slower than h1 ping h2, because packets go through more switches for h1 ping h8 than h1 ping h2.

3. Run “iperf h1 h2” and “iperf h1 h8”
a. What is “iperf” used for?

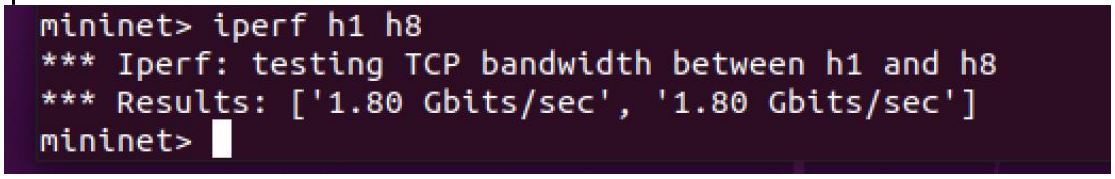
To test the bandwidth between any two hosts in order to evaluate the network performance.

b. What is the throughput for each case?
iperf h1 h2



```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['3.62 Gbits/sec', '3.62 Gbits/sec']
mininet> 
```

iperf h1 h8



```
mininet> iperf h1 h8
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['1.80 Gbits/sec', '1.80 Gbits/sec']
mininet> 
```

c. What is the difference, and explain the reasons for the difference.

Throughput is different because of the distance between the hosts. The more distance between hosts, the more congestion to the network and the lesser throughput.

For h1 h2, throughput is more because there is 1 switch between them. And for h1 h8, there are 3 switches between them, leading to lesser 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).

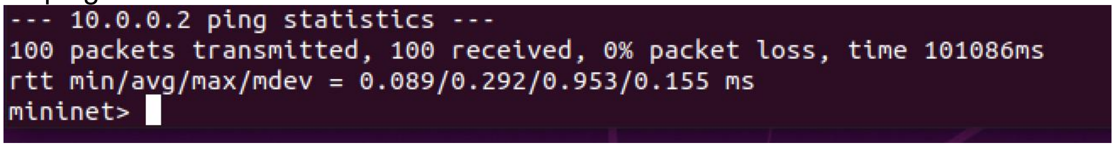
Switch s1, s2, s3, s5 and s7 . I can use function _handle_PacketIn(), which handles packets in messages from the switch.

Task 3:

1. Describe how the above code works, such as how the "MAC to Port" map is established.

In of_tutorial.py, function act_like_switch() and dictionary mac_to_port are in responsible for routing all the MAC addresses to a particular port number. It will flood all the packets to all destinations if the destination is not known, and it will be added to the dictionary for future use once the right destination is found. If the same request is received next time, it just need to be found in the map_to_port and sent.

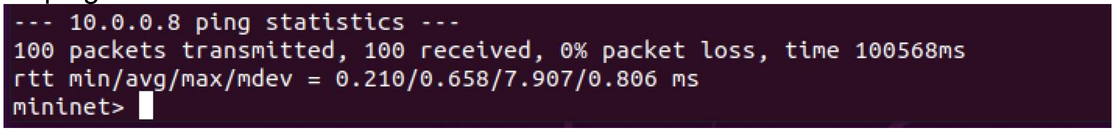
2. **(Comment out all prints before doing this experiment)** Have h1 ping h2, and h1 ping
h8 for 100 times (e.g., h1 ping -c100 p2).
a. How long did it take (on average) to ping for each case?
h1 ping -c100 h2



```
--- 10.0.0.2 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 101086ms
rtt min/avg/max/mdev = 0.089/0.292/0.953/0.155 ms
mininet> 
```

avg:0.292

h1 ping -c100 h8



```
--- 10.0.0.8 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 100568ms
rtt min/avg/max/mdev = 0.210/0.658/7.907/0.806 ms
mininet> 
```

avg:0.658

b. What is the minimum and maximum ping you have observed?

h1 ping h2:

min:0.089 max:0.953

h1 ping h8:

min:0.210 max:7.907

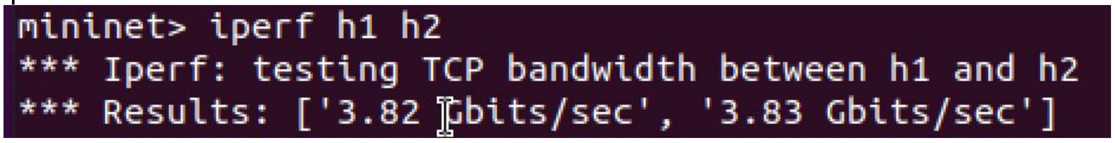
c. Any difference from Task 2 and why do you think there is a change if there is?

Faster than ping in Task2. Because now the controller can use function mapping in mac_to_port () to get MAC addresses and send them to the appropriate destination port.

3. Q.3 Run "iperf h1 h2" and "iperf h1 h8".

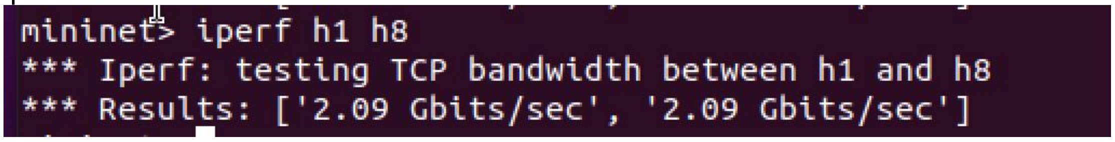
a. What is the throughput for each case?

iperf h1 h2



```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['3.82 Gbits/sec', '3.83 Gbits/sec']
```

iperf h1 h8



```
mininet> iperf h1 h8
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['2.09 Gbits/sec', '2.09 Gbits/sec']
```

b. What is the difference from Task 2 and why do you think there is a change if there is?

The throughput in both cases is higher than throughout in Task 2. Because MAC learning is implemented, which enable appropriate mapping of MAC addresses to destination port, so congestion is reduced, and throughput is increased.