

COEN 241 HW 3 Mininet & OpenFlow

Student Name: Mingyang Wu

Student ID: 00001628984

Task 1: Defining custom topologies

Questions

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

Output of “nodes”:

```
root@COEN241:/home/wmy# ls
binary_tree.py  Downloads  mininet  oftest  pox  Templates
Desktop         faasd     Music    openflow  Public  Videos
Documents       h1        oflops   Pictures  snap
root@COEN241:/home/wmy# mn --custom binary_tree.py --topo binary_tree
*** 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
c0
*** Starting 7 switches
s1 s2 s3 s4 s5 s6 s7 ...
*** Starting CLI:
mininet> nodes
available nodes are:
c0 h1 h2 h3 h4 h5 h6 h7 h8 s1 s2 s3 s4 s5 s6 s7
mininet>
```

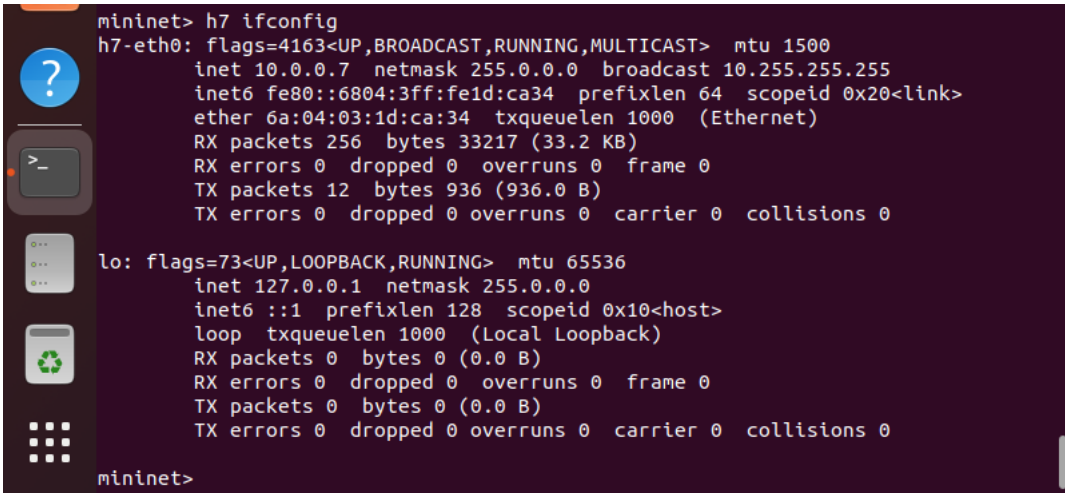
It shows all the controllers, hosts, and switches

Output of “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
c0
mininet>
```

It shows all the links between the nodes

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::6804:3ff:fe1d:ca34  prefixlen 64  scopeid 0x20<link>
    ether 6a:04:03:1d:ca:34  txqueuelen 1000  (Ethernet)
    RX packets 256  bytes 33217 (33.2 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

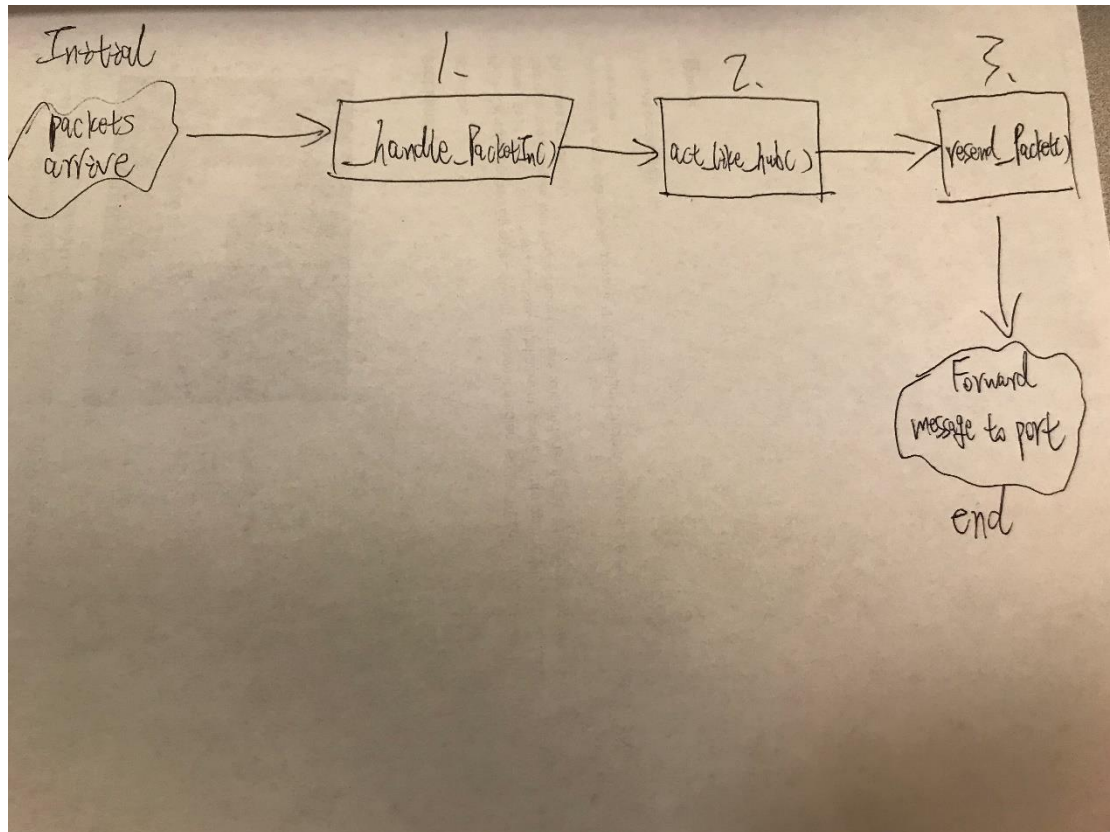
mininet>
```

Task 2: Analyze the "of_tutorial" controller

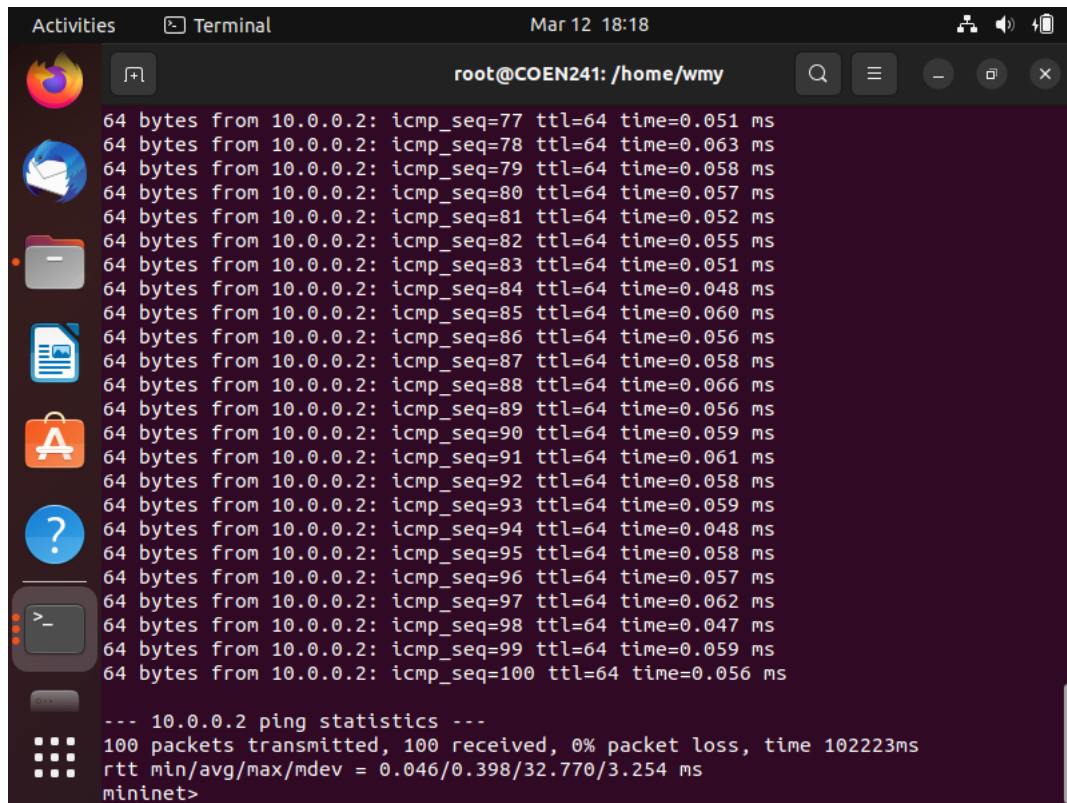
Questions

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?

When a packet comes to the controller:



2. Have h1 ping h2, and h1 ping h8 for 100 times (e.g., h1 ping -c100 p2).
For h1 ping h2:

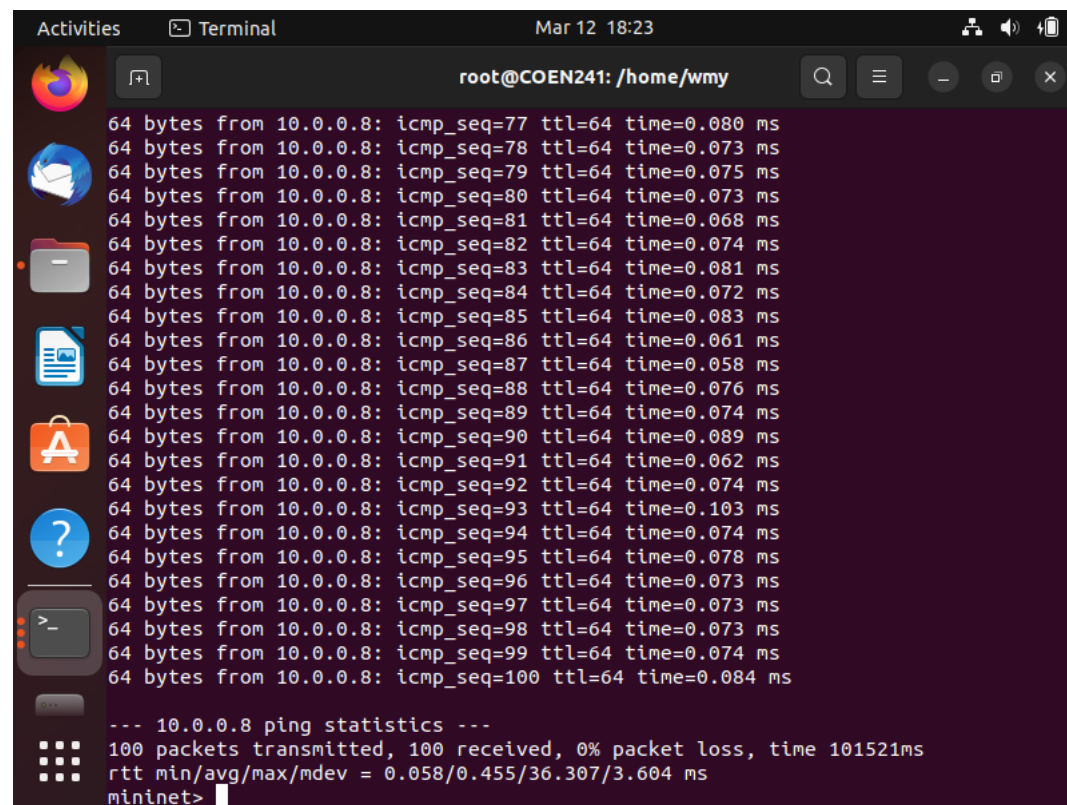


A terminal window titled "Terminal" with the date "Mar 12 18:18" and the user "root@COEN241: /home/wmy". The terminal displays the output of a ping command. It shows 100 successful ping attempts from 10.0.0.2 to 10.0.0.2, each with a 64-byte payload, TTL of 64, and a time between 0.046 ms and 0.062 ms. Below the ping results, it shows the ping statistics: 100 packets transmitted, 100 received, 0% packet loss, and a total time of 102223 ms. The RTT statistics are: min/avg/max/mdev = 0.046/0.398/32.770/3.254 ms. The prompt "mininet>" is visible at the bottom.

```
root@COEN241: /home/wmy
64 bytes from 10.0.0.2: icmp_seq=77 ttl=64 time=0.051 ms
64 bytes from 10.0.0.2: icmp_seq=78 ttl=64 time=0.063 ms
64 bytes from 10.0.0.2: icmp_seq=79 ttl=64 time=0.058 ms
64 bytes from 10.0.0.2: icmp_seq=80 ttl=64 time=0.057 ms
64 bytes from 10.0.0.2: icmp_seq=81 ttl=64 time=0.052 ms
64 bytes from 10.0.0.2: icmp_seq=82 ttl=64 time=0.055 ms
64 bytes from 10.0.0.2: icmp_seq=83 ttl=64 time=0.051 ms
64 bytes from 10.0.0.2: icmp_seq=84 ttl=64 time=0.048 ms
64 bytes from 10.0.0.2: icmp_seq=85 ttl=64 time=0.060 ms
64 bytes from 10.0.0.2: icmp_seq=86 ttl=64 time=0.056 ms
64 bytes from 10.0.0.2: icmp_seq=87 ttl=64 time=0.058 ms
64 bytes from 10.0.0.2: icmp_seq=88 ttl=64 time=0.066 ms
64 bytes from 10.0.0.2: icmp_seq=89 ttl=64 time=0.056 ms
64 bytes from 10.0.0.2: icmp_seq=90 ttl=64 time=0.059 ms
64 bytes from 10.0.0.2: icmp_seq=91 ttl=64 time=0.061 ms
64 bytes from 10.0.0.2: icmp_seq=92 ttl=64 time=0.058 ms
64 bytes from 10.0.0.2: icmp_seq=93 ttl=64 time=0.059 ms
64 bytes from 10.0.0.2: icmp_seq=94 ttl=64 time=0.048 ms
64 bytes from 10.0.0.2: icmp_seq=95 ttl=64 time=0.058 ms
64 bytes from 10.0.0.2: icmp_seq=96 ttl=64 time=0.057 ms
64 bytes from 10.0.0.2: icmp_seq=97 ttl=64 time=0.062 ms
64 bytes from 10.0.0.2: icmp_seq=98 ttl=64 time=0.047 ms
64 bytes from 10.0.0.2: icmp_seq=99 ttl=64 time=0.059 ms
64 bytes from 10.0.0.2: icmp_seq=100 ttl=64 time=0.056 ms

--- 10.0.0.2 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 102223ms
rtt min/avg/max/mdev = 0.046/0.398/32.770/3.254 ms
mininet>
```

For h1 ping h8:



```
root@COEN241: /home/wmy
64 bytes from 10.0.0.8: icmp_seq=77 ttl=64 time=0.080 ms
64 bytes from 10.0.0.8: icmp_seq=78 ttl=64 time=0.073 ms
64 bytes from 10.0.0.8: icmp_seq=79 ttl=64 time=0.075 ms
64 bytes from 10.0.0.8: icmp_seq=80 ttl=64 time=0.073 ms
64 bytes from 10.0.0.8: icmp_seq=81 ttl=64 time=0.068 ms
64 bytes from 10.0.0.8: icmp_seq=82 ttl=64 time=0.074 ms
64 bytes from 10.0.0.8: icmp_seq=83 ttl=64 time=0.081 ms
64 bytes from 10.0.0.8: icmp_seq=84 ttl=64 time=0.072 ms
64 bytes from 10.0.0.8: icmp_seq=85 ttl=64 time=0.083 ms
64 bytes from 10.0.0.8: icmp_seq=86 ttl=64 time=0.061 ms
64 bytes from 10.0.0.8: icmp_seq=87 ttl=64 time=0.058 ms
64 bytes from 10.0.0.8: icmp_seq=88 ttl=64 time=0.076 ms
64 bytes from 10.0.0.8: icmp_seq=89 ttl=64 time=0.074 ms
64 bytes from 10.0.0.8: icmp_seq=90 ttl=64 time=0.089 ms
64 bytes from 10.0.0.8: icmp_seq=91 ttl=64 time=0.062 ms
64 bytes from 10.0.0.8: icmp_seq=92 ttl=64 time=0.074 ms
64 bytes from 10.0.0.8: icmp_seq=93 ttl=64 time=0.103 ms
64 bytes from 10.0.0.8: icmp_seq=94 ttl=64 time=0.074 ms
64 bytes from 10.0.0.8: icmp_seq=95 ttl=64 time=0.078 ms
64 bytes from 10.0.0.8: icmp_seq=96 ttl=64 time=0.073 ms
64 bytes from 10.0.0.8: icmp_seq=97 ttl=64 time=0.073 ms
64 bytes from 10.0.0.8: icmp_seq=98 ttl=64 time=0.073 ms
64 bytes from 10.0.0.8: icmp_seq=99 ttl=64 time=0.074 ms
64 bytes from 10.0.0.8: icmp_seq=100 ttl=64 time=0.084 ms

--- 10.0.0.8 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 101521ms
rtt min/avg/max/mdev = 0.058/0.455/36.307/3.604 ms
mininet>
```

a. How long does it take (on average) to ping for each case?

For h1 ping h2:

0.398ms

For h1 ping h8:

0.455ms

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

For h1 ping h2:

Max: 32.770ms Min: 0.046ms

For h1 ping h8:

Max: 36.307ms Min: 0.058ms

c. What is the difference, and why?

From the data we can see that the speed of h1 ping h2 is faster than h1 ping h8 because the number of switches for h1 ping h2 is less than h1 ping h8. In this case, there exists a congestion in h1 ping h8, which leads to a delay in ping.

3. Run “iperf h1 h2” and “iperf h1 h8”

a. What is “iperf” used for?

The command “iperf” is used for testing TCP bandwidth between two hosts.

b. What is the throughput for each case?

For iperf h1 h2, the results are:

5.07 Mbits/sec and 6.02 Mbits/sec

For iperf h1 h8, the results are:

2.11 Mbits/sec and 2.36 Mbits/sec

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

From the results, it showed that there is a difference between using iperf h1 h2 and iperf h1 h8, due to the distance between the hosts.

For example, it is very obvious that the switches between h1 and h8 are more than h1 and h2, in this case, the throughput for h1 & h8 is less than it for h1 & h2 because the packets were sent and received by every switch.

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).

All of the switches observe traffic.

To observe this, we can use the `_handle_PacketIn()` function and add a function to show the info of packets before using `act_like_hub()` function because `_handle_PacketIn()` is used when a node receives packet, so we can observe the traffic of switches.

Task 3: MAC Learning Controller

Questions

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).

The code in of_tutorial.py aims to let the switch to learn where to send packets for giving it the source MAC address to their destination ports. In this case, the switch can know where to send the packets to and reduce the network workload. From the code, if the port associated with the destination MAC of the packet is unknown, then the switch will flood the packet to every switch.

We can use a h1 ping h2 example to illustrate this:

1. h1 sends packets to s3
2. Then, s3 maps MAC to h1 and flood to s2 and h2 if the destination is unknown
3. h2 responds to s3 and maps the destination port

2. 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?

For h1 ping h2:

0.213ms

For h1 ping h8:

0.384ms

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

For h1 ping h2:

Max: 30.211ms Min: 0.025ms

For h1 ping h8:

Max: 34.134ms Min: 0.032ms

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

By observing the data we can recognize that the ping time is less than it in Task 2. I think the change may be the controller gets MAC address and send them to the specific destination ports in Task 3 rather than just flooding all packets in Task 2.

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

a. What is the throughput for each case?

For "iperf h1 h2":

6.32 Mbits/sec and 7.17 Mbits/sec

For "iperf h1 h8":

2.84 Mbits/sec and 3.24 Mbits/sec

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

From the data we can observe that the throughput is higher than it in Task 2. The change is because in Task 3, it used MAC address, which helps the switch to route to appropriate destination port and reduce the network congestion.