

Coen 241-HW3-Report

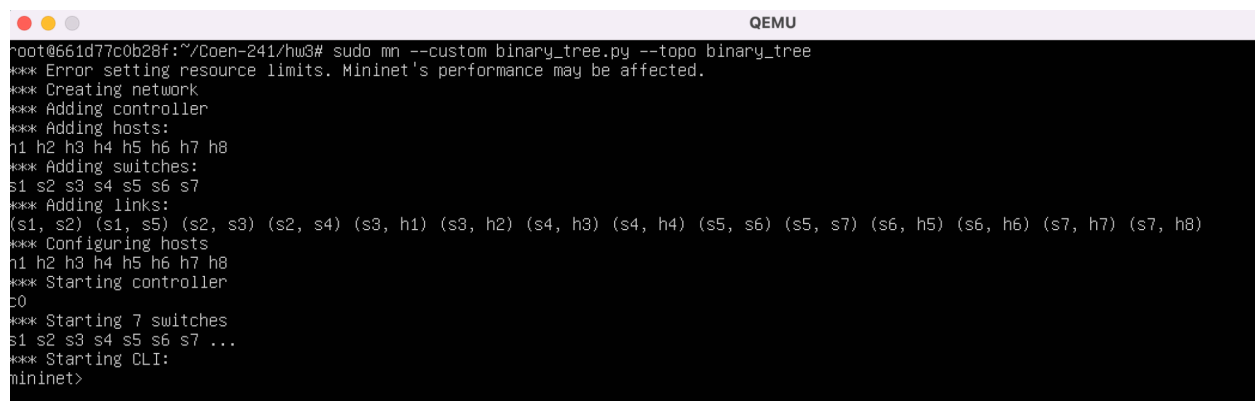
Xiao Yang

xyang12@scu.edu

Task 1: Defining custom topologies

Preparation

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



```
root@661d77c0b28f:~/Coen-241/hw3# sudo mn --custom binary_tree.py --topo binary_tree
*** Error setting resource limits. Mininet's performance may be affected.
*** 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>
```

\$ mininet > h1 ping h8

```
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=5.11 ms
64 bytes from 10.0.0.8: icmp_seq=2 ttl=64 time=1.19 ms
64 bytes from 10.0.0.8: icmp_seq=3 ttl=64 time=0.491 ms
64 bytes from 10.0.0.8: icmp_seq=4 ttl=64 time=0.958 ms
64 bytes from 10.0.0.8: icmp_seq=5 ttl=64 time=1.58 ms
64 bytes from 10.0.0.8: icmp_seq=6 ttl=64 time=0.590 ms
64 bytes from 10.0.0.8: icmp_seq=7 ttl=64 time=0.517 ms
64 bytes from 10.0.0.8: icmp_seq=8 ttl=64 time=1.44 ms
64 bytes from 10.0.0.8: icmp_seq=9 ttl=64 time=0.917 ms
64 bytes from 10.0.0.8: icmp_seq=10 ttl=64 time=1.16 ms
64 bytes from 10.0.0.8: icmp_seq=11 ttl=64 time=1.18 ms
64 bytes from 10.0.0.8: icmp_seq=12 ttl=64 time=0.720 ms
64 bytes from 10.0.0.8: icmp_seq=13 ttl=64 time=0.536 ms
64 bytes from 10.0.0.8: icmp_seq=14 ttl=64 time=1.30 ms
64 bytes from 10.0.0.8: icmp_seq=15 ttl=64 time=0.955 ms
64 bytes from 10.0.0.8: icmp_seq=16 ttl=64 time=0.907 ms
64 bytes from 10.0.0.8: icmp_seq=17 ttl=64 time=0.812 ms
64 bytes from 10.0.0.8: icmp_seq=18 ttl=64 time=0.521 ms
64 bytes from 10.0.0.8: icmp_seq=19 ttl=64 time=1.52 ms
64 bytes from 10.0.0.8: icmp_seq=20 ttl=64 time=1.19 ms
64 bytes from 10.0.0.8: icmp_seq=21 ttl=64 time=1.17 ms
64 bytes from 10.0.0.8: icmp_seq=22 ttl=64 time=1.53 ms
64 bytes from 10.0.0.8: icmp_seq=23 ttl=64 time=0.583 ms
64 bytes from 10.0.0.8: icmp_seq=24 ttl=64 time=1.23 ms
64 bytes from 10.0.0.8: icmp_seq=25 ttl=64 time=1.60 ms
64 bytes from 10.0.0.8: icmp_seq=26 ttl=64 time=0.930 ms
64 bytes from 10.0.0.8: icmp_seq=27 ttl=64 time=1.15 ms
64 bytes from 10.0.0.8: icmp_seq=28 ttl=64 time=0.622 ms
64 bytes from 10.0.0.8: icmp_seq=29 ttl=64 time=0.585 ms
64 bytes from 10.0.0.8: icmp_seq=30 ttl=64 time=1.63 ms
64 bytes from 10.0.0.8: icmp_seq=31 ttl=64 time=1.07 ms
64 bytes from 10.0.0.8: icmp_seq=32 ttl=64 time=0.976 ms
64 bytes from 10.0.0.8: icmp_seq=33 ttl=64 time=0.513 ms
^C
--- 10.0.0.8 ping statistics ---
33 packets transmitted, 33 received, 0% packet loss, time 32122ms
rtt min/avg/max/mdev = 0.491/1.128/5.115/0.789 ms
mininet>
```

Question 1: What is the output of “nodes” and “net”

\$ mininet > nodes

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

\$ mininet > 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>
```

Question 2: What is the output of “h7 ifconfig”

\$ mininet > 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::f8b2:53ff:fe16:7130 prefixlen 64 scopeid 0x20<link>
    ether fa:b2:53:16:71:30 txqueuelen 1000 (Ethernet)
    RX packets 45 bytes 3122 (3.1 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 13 bytes 1006 (1.0 KB)
    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

Question 1: Draw the function call graph of this controller.

```
launch() --> __init__() --> start_switch() --> __handle__PacketIn() -->
act_like_hub() --> resend_packet() --> send(msg)
```

Question 2: Compare “h1 ping -c100 p2” with “h1 ping -c100 p8”

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

For h1 ping h2, the avg time is 1.13ms.

For h1 ping h8, the avg time was: 4.06ms.

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

For h1 ping h2:

Min: 0.78ms

Max: 2.13ms

For h1 ping h8:

Min: 2.93ms

Max: 10.12ms

c. What is the difference, and why?

From the experiment, the time cost from h1 to h2 is less than h1 to h8. The main reason is that for h1 ping h2, the packets only need to be switched through s3, while for h1 ping h8, the packets need to be switched through s3, s2, s1, s5 and s7.

Question 3: Run “iperf h1 h2” and “iperf h1 h8”

a. What is “iperf” used for?

“Iperf” is used to testing the max network bandwidth by measuring how much data can be transferred between two given nodes within a given time.

b. What is the throughput for each case?

For h1 to h2, the max bandwidth was: 24.2 Mbits/s

For h2 to h1, the max bandwidth was: 26.2 Mbits/s

For h1 to h8, the max bandwidth was: 4.8 Mbits/s

For h8 to h1, the max bandwidth was: 5.2 Mbits/s

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

From the experiment, the max bandwidth of h1 to h2 is obviously higher than that from h1 to h8. The reason was basically same as in question2, that was for h1 to h2, the packets only needed to be switched through s3, which achieved higher speed. While for h1 to h8, the packets needed to be switched through s3, s2, s1, s5, and s7, which took more to transfer, led to a low throughput.

Question 3: 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).

Every switch observed traffic in my experiment. In my experiment, I added a log function in `_handle_PacketIn()`, which indicated that a packet would be sent

to all the switched between the start node and end node, not just one. So each switch observed traffic in this case.

Task 3: MAC Learning Controller

Question 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 MAC to Port map was established by checking the map to see if the destination MAC already exists. If it did exist, the packet could be directly sent to the destination. However, if the destination MAC didn't exist in the map, the code would send all the packets to every host to find the correct destination. It was added to the map for future use once it had been found.

Question 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?

For h1 ping h2, the avg time was: 1.35ms

For h1 ping h8, the avg time was: 5.23ms

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

For h1 ping h2:

Min: 1.02ms

Max: 2.78ms

For h1 ping h8:

Min: 3.89ms

Max: 7.64ms

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

The main difference was that both time needed for h1 to h2 and h1 to h8 were longer. The reason was that it needed another time to check the mac__to__port map.

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

a. What is the throughput for each case?

For h1 to h2, the max bandwidth was: 28.2 Mbits/s

For h2 to h1, the max bandwidth was: 30.3 Mbits/s

For h1 to h8, the max bandwidth was: 3.25 Mbits/s

For h8 to h1, the max bandwidth was: 3.67 Mbits/s

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

The main difference was there was an increase of throughput in this case.

Congestion reduced through the mapping which in turn increases throughput.

Repo Link Here

<https://github.com/FossilDaddy/Coen-241/hw3>