Homework-3(COEN-241)

Task-1

Questions

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

Output of nodes:

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

Output of nets:

```
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
```

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::cce9:41ff:fe4b:d431  prefixlen 64  scopeid 0x20<link>
        ether ce:e9:41:4b:d4:31  txqueuelen 1000  (Ethernet)
        RX packets 15424  bytes 17142100 (17.1 MB)
        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

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?

   The controller initializes with the below function call:

   launch() -> start_switch(event) -> __init__(self,connection)

   When the packet arrives the below graph is followed depending upon the code:

   _handle_PacketIn(self,event) -> act_like_hub(self,packet,packet_in) -> resend_packet(self,packet_in, out_port):

   _handle_PacketIn(self,event) -> act_like_switch(self,packet,packet_in) -> resend_packet(self,packet_in, out_port):

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 h2

  Average time is 2.5 ms

```
mininet> h1 ping -c100 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=5.42 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=2.57 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=2.71 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=2.48 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=2.64 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=2.46 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=2.61 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=3.72 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=2.20 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=2.24 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=2.20 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=1.58 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=2.86 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=2.53 ms
```

- h1 ping h8

  Average time is 14ms

```
mininet> h1 ping -c100  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=24.0 ms
64 bytes from 10.0.0.8: icmp_seq=2 ttl=64 time=9.91 ms
64 bytes from 10.0.0.8: icmp_seq=3 ttl=64 time=11.9 ms
64 bytes from 10.0.0.8: icmp_seq=4 ttl=64 time=8.21 ms
64 bytes from 10.0.0.8: icmp_seq=5 ttl=64 time=8.52 ms
64 bytes from 10.0.0.8: icmp_seq=6 ttl=64 time=14.7 ms
64 bytes from 10.0.0.8: icmp_seq=7 ttl=64 time=7.98 ms
64 bytes from 10.0.0.8: icmp_seq=8 ttl=64 time=6.58 ms
64 bytes from 10.0.0.8: icmp_seq=9 ttl=64 time=8.04 ms
64 bytes from 10.0.0.8: icmp_seq=10 ttl=64 time=8.54 ms
64 bytes from 10.0.0.8: icmp_seq=11 ttl=64 time=6.73 ms
64 bytes from 10.0.0.8: icmp_seq=12 ttl=64 time=7.64 ms
64 bytes from 10.0.0.8: icmp_seq=13 ttl=64 time=9.26 ms
64 bytes from 10.0.0.8: icmp_seq=14 ttl=64 time=9.45 ms
64 bytes from 10.0.0.8: icmp_seq=15 ttl=64 time=10.5 ms
```

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

- h1 ping h2
  Minimum: 1.42ms
  Maximum: 5.42ms

- h1 ping h8
  Minimum: 5.55 ms
  Maximum: 24.0 ms

c. What is the difference, and why?

Minimum ping difference: 4.13 ms
Maximum ping difference: 18.58 ms

In case of h1 and h2 the ping time is lower because they are connected to the same switch. And in case of h1 and h8 they are not connected to the same switch , so they have to travel more switches hence the ping time is higher.

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

a. What is “iperf” used for?

Iperf is a open-source command-line tool used to diagnose the network speed issues by measuring maximum network throughput a server can handle. It generally helps to measure the bandwidth for the network performance and the quality of the network.

b. What is the throughput for each case?

- Iperf h1 h2

```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['17.7 Mbits/sec', '20.5 Mbits/sec']
```

- Iperf h1 h8

```
mininet> iperf h1 h8
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['4.00 Mbits/sec', '4.60 Mbits/sec']
```

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

Difference in throughput: 13.7 Mbits/sec, 15.9 Mbits/sec

As we have seen that the packet travel is less in case of h1 h2 as compared to h1 h8, it has a higher throughput because more packets can be transmitted.

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).

To check the traffic in the connection I have added the below print statement to the code of of_tutorial file:

print(f”switch: {event.connection} | source address: {packet.src}, destination address: {packet.dst} ”)

After adding this, I ran the command iperf h1 h2 and iperf h1 h8 in mininet. Almost all of the switches observed traffic. For iperf h1 h8, s1, s2, s3, s4, s5, s6, s7, all switches observes traffic because of its structure in binary_tree.py.

And in case of h1 h2 , s3 switch observes most of the traffic because of its connection.

Task-3

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).

   When h1 pings h2, it will look into the mac_to_port dictionary to check if the particular port exists or not. If yes, then it uses resend_packet() to send the packet.

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

      Average time: 4.12 ms

      H1 ping -c100 h8

      Average time: 10.5 ms

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

      H1 ping h2

Maximum time: 6.51 ms

Minimum time: 1.70 ms

H1 ping h8

Maximum time: 27.5 ms

Minimum time: 7.12 ms

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

Yes there is a difference in average time in task 1 and task2.

For h1 ping h2, there is a slight increase in average ping time.

For h1 ping h8, there is a decrease in average ping time. As mac_to_port is established so packets will be sent directly to the destination as compared to flooding the packets. Hence there is a slight decrease in average ping time.

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

Command: iperf h1 h2

Result : [‘4.18 Mbits/sec’, ‘5.00 Mbits/sec’]

Command: iperf h1 h8

Result : [‘3.00 Mbits/sec’, ‘3.68 Mbits/sec’]