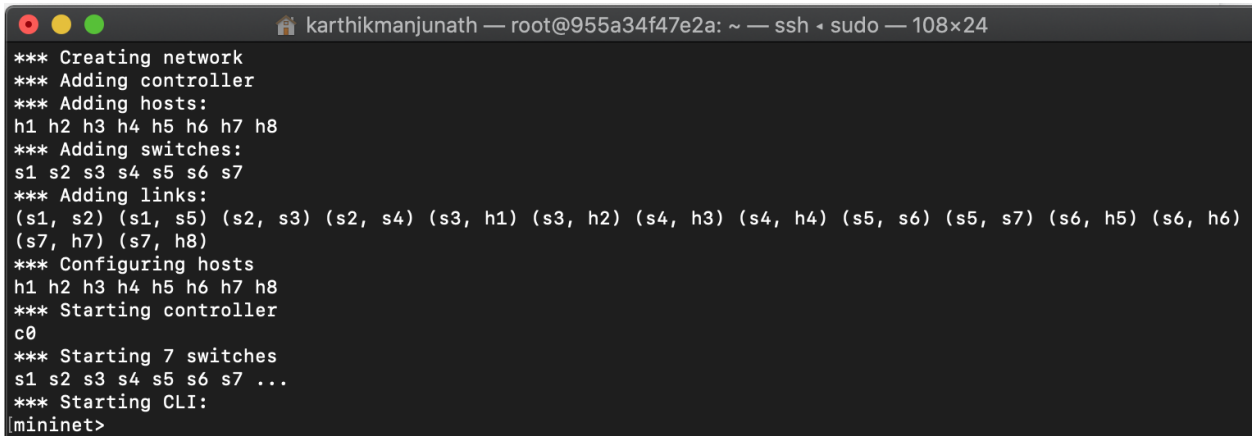
Karthik Cheernalli Manjunath
00001587860

COEN241 - Cloud Computing

Homework #3

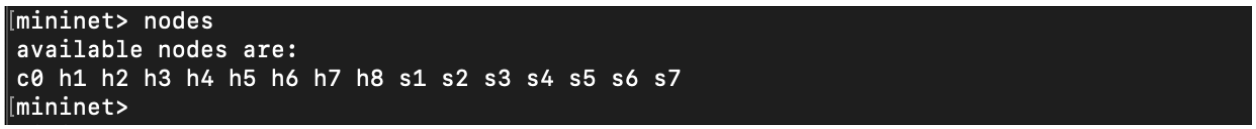
Task 1: Defining custom topologies

> $ mn --custom binary_tree.py --topo binary_tree



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

> nodes



This will list out all the hosts, switches and controllers as associated with the topology.

> 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>
```

This will display all the links associated with the topology

2. What is the output of “h7 ifconfig” ?
Ans.

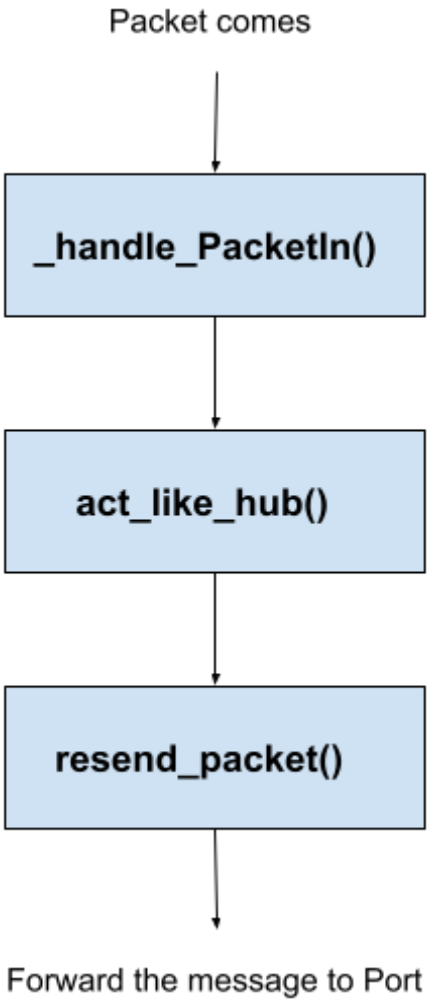
> h7 ifconfig

```
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::ac7d:ffff:fe8d:8e65  prefixlen 64  scopeid 0x20<link>
        ether ae:7d:ff:8d:8e:65  txqueuelen 1000  (Ethernet)
        RX packets 69  bytes 5342 (5.3 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

mininet>
```

Task 2: Install POX

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 ?

Ans. This is the function call graph of this controller.
First it will call _handle_PacketIn() function, then act_like_hub() function followed by resend_packet() function



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?
Ans.

> h1 ping -c100 h2

```
--- 10.0.0.2 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99171ms
rtt min/avg/max/mdev = 1.492/3.495/17.762/2.859 ms
mininet>
```

> h1 ping -c100 h8

```
--- 10.0.0.8 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99121ms
rtt min/avg/max/mdev = 7.736/15.485/64.786/10.826 ms
mininet>
```

a. Average time taken for ping in each case
> h1 to h2 : **3.495 ms**
> h1 to h8 : **15.485 ms**

b. Minimum time taken for ping in each case
> h1 to h2 : **1.492 ms**
> h1 to h8 : **7.736 ms**

Maximum time taken for ping in each case
> h1 to h2 : **17.762 ms**
> h1 to h8 : **64.786 ms**

c. The difference between h1 ping h2 is relatively faster than h1 ping h8 because of the number of connections between each host.
h1 ping h2: h1 -> s3 -> h2
h1 ping h8: h1 -> s3 -> s2 -> s1 -> s5 -> s7 -> h8
Here, as per the topology, there is only a switch between h1 and h2 but between h1 and h8 there are multiple (three) switches which will bring some congestion. This is the reason for the delay in the pings.
So there is an avg delay of around **12 ms** between the pings h1 -> h2 and h1 -> h8.

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.

Ans.
a. The “iperf” is used to test the bandwidth between any two hosts. This is basically used for network performance evaluation.

b. Throughput for each case is as below

> iperf h1 h2

```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['5.00 Mbits/sec', '5.98 Mbits/sec']
mininet>
```

> iperf h1 h8

```
mininet> iperf h1 h8
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['2.00 Mbits/sec', '2.31 Mbits/sec']
mininet>
```

c. There is a difference in the throughput because of the distance between the hosts. The increase in the distance between hosts will bring some congestion to the network and thus lesser throughput.
As observed from above (b), throughput is more in case of h1 -> h2 because there is only 1 switch between them. On the other hand, for the case h1 -> h8 there are 3 switches between them which is leading to a 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).
Ans. As per the given conditions, h1 -> h2 and h1 -> h8. Switches s1, s2, s3, s5 and s7 observe the most traffic. This can be observed from the _handle_PacketIn() where it handles packets in messages from the switch.

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).

Ans.
In of_tutorial.py, now the act_like_switch function is in charge of routing all the MAC addresses to a particular port number with the help of a dictionary mac_to_port. If the destination is not known, it will simply flood all the packets to all destinations and once found the right destination it will be added to the dictionary for future use. Next time when the same request is received it just looks up the dictionary map_to_port and sends accordingly. This is what we have described as MAC learning and this will ensure lesser network congestion and thus higher throughput.

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?

Ans.
> h1 ping -c100 h2

```
--- 10.0.0.2 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99162ms
rtt min/avg/max/mdev = 1.384/2.300/5.884/0.756 ms
mininet>
```

> h1 ping -c100 h8

```
--- 10.0.0.8 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99161ms
rtt min/avg/max/mdev = 7.631/12.171/77.132/8.388 ms
mininet>
```

a. Average time taken for ping in each case

> h1 to h2 : **2.300 ms**

> h1 to h8 : **12.171 ms**

b. Minimum time taken for ping in each case

> h1 to h2 : **1.384 ms**
> h1 to h8 : **7.631 ms**

Maximum time taken for ping in each case

> h1 to h2 : **5.884 ms**
> h1 to h8 : **77.132 ms**

c. As per the given values above, the pings in Task 2 (invoking act_like_hub) were relatively slower than pings in Task 3 (invoking act_like_switch). This change is more significant in h1 -> h8 as compared to h1 -> h2 because of the increased number of switches in the path. We are observing this change because now the controller is able to get MAC addresses and send them to the appropriate destination port using the mapping in mac_to_port (MAC learning) as 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?
Ans.

a.
> iperf h1 h2

```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['6.58 Mbits/sec', '7.75 Mbits/sec']
mininet>
```

> iperf h1 h8

```
mininet> iperf h1 h8
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['2.53 Mbits/sec', '3.05 Mbits/sec']
mininet>
```

b. The throughput in both the cases is higher as compared to Task 2. This is because of the implementation of the MAC learning which has enabled appropriate mapping of MAC addresses to destination port leading to reduced congestion and thereby increasing throughput. Refer to Task 2 Question (3b) for the relative details.