

HW3 Report

Task 1: Defining custom topologies

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

Ans:

Output of nodes:

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

Output of net:

```
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
mininet> 
```

2. What is the output of “h7 ifconfig”

Ans:

```
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::18b2:34ff:fe12:f0c prefixlen 64 scopeid 0x20<link>
    ether 1a:b2:34:12:0f:0c txqueuelen 1000 (Ethernet)
    RX packets 75 bytes 5574 (5.5 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

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:

After turning on the POX Listener by running the command “./pox.py log.level --DEBUG misc.of_tutorial”, it turns on the start switch. The start switch then calls the method “_handle_PacketIn()” which is responsible for handling the packet in messages from the switch. The method “_handle_PacketIn()” triggers the act_like_hub() function. act_like_hub() function simulates a hub environment by sending the packets to all the ports except the input port. Now, the resend_packet() function is called. The resend_packet() function adds a packet to the message data and performs action on it. This message instructs the switch to resend the packet to a specified port. The entire flow looks like:

start switch → _handle_PacketIn() → act_like_hub() → resend_packet() → send(message)

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?

Ans:

Average time for h1 ping -c100 h2 is: 3.954 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=2.12 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=2.83 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=3.99 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=3.66 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=4.32 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=3.86 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=4.42 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=5.23 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=5.12 ms
^C
--- 10.0.0.2 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8035ms
rtt min/avg/max/mdev = 2.121/3.954/5.230/0.947 ms
```

Average time for h1 ping -c100 h8 is: 24.556 ms

```

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=99.0 ms
64 bytes from 10.0.0.8: icmp_seq=2 ttl=64 time=15.5 ms
64 bytes from 10.0.0.8: icmp_seq=3 ttl=64 time=16.0 ms
64 bytes from 10.0.0.8: icmp_seq=4 ttl=64 time=20.8 ms
64 bytes from 10.0.0.8: icmp_seq=5 ttl=64 time=29.2 ms
64 bytes from 10.0.0.8: icmp_seq=6 ttl=64 time=19.3 ms
64 bytes from 10.0.0.8: icmp_seq=7 ttl=64 time=24.6 ms
64 bytes from 10.0.0.8: icmp_seq=8 ttl=64 time=20.3 ms
64 bytes from 10.0.0.8: icmp_seq=9 ttl=64 time=22.1 ms
64 bytes from 10.0.0.8: icmp_seq=10 ttl=64 time=16.4 ms
64 bytes from 10.0.0.8: icmp_seq=11 ttl=64 time=11.8 ms
64 bytes from 10.0.0.8: icmp_seq=12 ttl=64 time=13.0 ms
64 bytes from 10.0.0.8: icmp_seq=13 ttl=64 time=10.6 ms
^C
--- 10.0.0.8 ping statistics ---
13 packets transmitted, 13 received, 0% packet loss, time 12092ms
rtt min/avg/max/mdev = 10.606/24.556/99.041/22.085 ms
mininet>

```

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

Ans:

h1 ping -c100 h2

Min: 2.121 ms

Max: 5.230 ms

h1 ping -c100 h8

Min: 10.606 ms

Max: 99.041 ms

c. What is the difference, and why?

Ans:

The ping time for h1 to h2 is less than h1 to h8 because h1 is connected to h2 only via one switch while h1 is connected to h8 via s3, s2, s1, s5, s7 switches. h1 has to traverse through 5 switches to reach h8 and h1 has to travel only one switch to reach h2. Hence, there is a difference between the ping times.

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

a. What is “iperf” used for?

Ans:

“iperf” is a network testing tool that assists network administrators to measure the maximum achievable bandwidth between two network endpoints. It can be used to test both TCP and UDP network protocols. It sends the data from one node to another and measures the amount of time it takes for the data to be transmitted and received. With this information, it calculates the maximum achievable bandwidth.

b. What is the throughput for each case?

Ans:

Below screenshot shows the throughput:

```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['9.80 Mbits/sec', '10.8 Mbits/sec']
mininet> iperf h1 h8
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['4.33 Mbits/sec', '4.78 Mbits/sec']
mininet> 
```

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

Ans:

Throughput is higher in case of h1 and h2 as there is less network congestion and h1 has to travel through only one switch. Hence, it can send data much faster.

Throughput is lower in case of h1 and h8 as there are more hops and hence can cause network congestion. Also, h1 has to travel via s3, s2, s1, s5, s7 switches to h8 causing the decrease in the speed by which the data is sent.

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:

The method `_handle_PacketIn()` of the `of_tutorial.py` file is invoked whenever a packet is received. And hence adding the below line in the `_handle_PacketIn()` method helps us inspect the traffic and shows us that all the switches monitor the traffic.

```
log.info("Switch observing traffic: %s" % (self.connection))
```

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:

By adding the given code, `act_like_switch()` method allows us to determine where MAC addresses are located. When it is determined that a MAC address is the one to which a sender desires to send a message, the controller can conveniently map it to a port. This also helps the controller's speed when sending packets to addresses that are previously known because the packet is simply delivered to that well-known port. However, if the destination address is unknown, it simply sends the packet to all destinations (aka flooding). The ping times and throughput are improved by the MAC Learning Controller by minimizing the flooding.

```

root@7ea2a161a991:~/pox# ./pox.py log.level --DEBUG misc.of_tutorial
POX 0.7.0 (gar) / Copyright 2011-2020 James McCauley, et al.
DEBUG:core:POX 0.7.0 (gar) going up...
DEBUG:core:Running on CPython (3.6.9/Feb 28 2023 09:55:20)
DEBUG:core:Platform is Linux-5.15.0-67-generic-x86_64-with-Ubuntu-18.04-bionic
WARNING:version:Support for Python 3 is experimental.
INFO:core:POX 0.7.0 (gar) is up.
DEBUG:openflow.of_01:Listening on 0.0.0.0:6633
INFO:openflow.of_01:[00-00-00-00-00-07 2] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-07 2]
INFO:openflow.of_01:[00-00-00-00-00-04 3] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-04 3]
INFO:openflow.of_01:[00-00-00-00-00-06 4] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-06 4]
INFO:openflow.of_01:[00-00-00-00-00-01 5] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-01 5]
INFO:openflow.of_01:[00-00-00-00-00-03 6] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-03 6]
INFO:openflow.of_01:[00-00-00-00-00-05 7] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-05 7]
INFO:openflow.of_01:[00-00-00-00-00-02 8] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-02 8]
Learning that ee:5e:f5:f0:11:3a is attached at port 1
Destination: 33:33:00:00:00:16 unknown. Resend to all
Learning that ee:5e:f5:f0:11:3a is attached at port 1
Destination: 33:33:00:00:00:16 unknown. Resend to all
Learning that ee:5e:f5:f0:11:3a is attached at port 3
Destination: 33:33:00:00:00:16 unknown. Resend to all
Learning that ee:5e:f5:f0:11:3a is attached at port 2
Destination: 33:33:00:00:00:16 unknown. Resend to all
Destination: 33:33:ff:f0:11:3a unknown. Resend to all
Learning that ee:5e:f5:f0:11:3a is attached at port 3
Destination: 33:33:00:00:00:16 unknown. Resend to all
Learning that ee:5e:f5:f0:11:3a is attached at port 3
Destination: 33:33:00:00:00:16 unknown. Resend to all
Learning that ee:5e:f5:f0:11:3a is attached at port 3
Destination: 33:33:00:00:00:16 unknown. Resend to all
Destination: 33:33:ff:f0:11:3a unknown. Resend to all
Destination: 33:33:ff:f0:11:3a unknown. Resend to all
Destination: 33:33:ff:f0:11:3a unknown. Resend to all
Learning that 0e:e0:23:9d:05:21 is attached at port 2
Destination: 33:33:ff:9d:05:21 unknown. Resend to all
Learning that e6:0b:b8:2a:2b:f7 is attached at port 1
Destination: 33:33:ff:2a:2b:f7 unknown. Resend to all
Destination: 33:33:ff:f0:11:3a unknown. Resend to all
Learning that 0e:e0:23:9d:05:21 is attached at port 2
Destination: 33:33:ff:9d:05:21 unknown. Resend to all
Destination: 33:33:ff:f0:11:3a unknown. Resend to all
Destination: 33:33:ff:f0:11:3a unknown. Resend to all
Learning that e6:0b:b8:2a:2b:f7 is attached at port 1
Destination: 33:33:ff:2a:2b:f7 unknown. Resend to all

```

```
root@7ea2a161a991: ~/pox
Destination: 33:33:ff:2a:2b:f7 unknown. Resend to all
Learning that e6:0b:b8:2a:2b:f7 is attached at port 1
Destination: 33:33:ff:2a:2b:f7 unknown. Resend to all
Learning that 56:30:75:b5:44:b7 is attached at port 2
Destination: 33:33:ff:b5:44:b7 unknown. Resend to all
Learning that e6:0b:b8:2a:2b:f7 is attached at port 3
Destination: 33:33:ff:2a:2b:f7 unknown. Resend to all
Learning that 0e:e0:23:9d:05:21 is attached at port 3
Destination: 33:33:ff:9d:05:21 unknown. Resend to all
Learning that 56:30:75:b5:44:b7 is attached at port 1
Destination: 33:33:ff:b5:44:b7 unknown. Resend to all
Learning that e6:0b:b8:2a:2b:f7 is attached at port 3
Destination: 33:33:ff:2a:2b:f7 unknown. Resend to all
Learning that 0e:e0:23:9d:05:21 is attached at port 3
Destination: 33:33:ff:9d:05:21 unknown. Resend to all
Learning that e6:0b:b8:2a:2b:f7 is attached at port 3
Destination: 33:33:ff:2a:2b:f7 unknown. Resend to all
Learning that 0e:e0:23:9d:05:21 is attached at port 3
Destination: 33:33:ff:9d:05:21 unknown. Resend to all
Learning that 56:30:75:b5:44:b7 is attached at port 3
Destination: 33:33:ff:b5:44:b7 unknown. Resend to all
Learning that 56:30:75:b5:44:b7 is attached at port 2
Destination: 33:33:ff:b5:44:b7 unknown. Resend to all
Destination: 33:33:00:00:00:16 unknown. Resend to all
Learning that 56:30:75:b5:44:b7 is attached at port 3
Destination: 33:33:ff:b5:44:b7 unknown. Resend to all
Destination: 33:33:00:00:00:16 unknown. Resend to all
Learning that 56:30:75:b5:44:b7 is attached at port 3
Destination: 33:33:ff:b5:44:b7 unknown. Resend to all
Learning that 56:30:75:b5:44:b7 is attached at port 3
Destination: 33:33:ff:b5:44:b7 unknown. Resend to all
Destination: 33:33:00:00:00:16 unknown. Resend to all
Destination: 33:33:00:00:00:16 unknown. Resend to all
Learning that a2:79:99:6c:02:97 is attached at port 1
Destination: 33:33:00:00:00:16 unknown. Resend to all
Destination: 33:33:00:00:00:16 unknown. Resend to all
Learning that a2:79:99:6c:02:97 is attached at port 2
Destination: 33:33:00:00:00:16 unknown. Resend to all
Destination: 33:33:00:00:00:16 unknown. Resend to all
Destination: 33:33:00:00:00:16 unknown. Resend to all
Learning that 26:25:12:13:e3:1a is attached at port 2
Destination: 33:33:00:00:00:16 unknown. Resend to all
Destination: 33:33:00:00:00:02 unknown. Resend to all
Learning that a2:79:99:6c:02:97 is attached at port 1
Destination: 33:33:00:00:00:16 unknown. Resend to all
Learning that a2:79:99:6c:02:97 is attached at port 3
Destination: 33:33:00:00:00:16 unknown. Resend to all
Learning that 26:25:12:13:e3:1a is attached at port 2
Destination: 33:33:00:00:00:16 unknown. Resend to all
Destination: 33:33:00:00:00:02 unknown. Resend to all
Learning that 26:25:12:13:e3:1a is attached at port 1
Destination: 33:33:00:00:00:16 unknown. Resend to all
```

```
root@7ea2a161a991: ~/pox
Destination: 33:33:00:00:00:16 unknown. Resend to all
Learning that a2:79:99:6c:02:97 is attached at port 3
Destination: 33:33:00:00:00:16 unknown. Resend to all
Destination: 33:33:00:00:00:02 unknown. Resend to all
Destination: 33:33:00:00:00:02 unknown. Resend to all
Learning that 26:25:12:13:e3:1a is attached at port 3
Destination: 33:33:00:00:00:16 unknown. Resend to all
Learning that a2:79:99:6c:02:97 is attached at port 3
Destination: 33:33:00:00:00:16 unknown. Resend to all
Learning that a2:79:99:6c:02:97 is attached at port 3
Destination: 33:33:00:00:00:16 unknown. Resend to all
Learning that 26:25:12:13:e3:1a is attached at port 3
Destination: 33:33:00:00:00:16 unknown. Resend to all
Learning that 26:25:12:13:e3:1a is attached at port 3
Destination: 33:33:00:00:00:16 unknown. Resend to all
Destination: 33:33:00:00:00:16 unknown. Resend to all
Destination: 33:33:00:00:00:02 unknown. Resend to all
Destination: 33:33:00:00:00:02 unknown. Resend to all
Destination: 33:33:00:00:00:02 unknown. Resend to all
Destination: 33:33:00:00:00:16 unknown. Resend to all
Destination: 33:33:00:00:00:16 unknown. Resend to all
Destination: 33:33:00:00:00:16 unknown. Resend to all
Destination: 33:33:00:00:00:16 unknown. Resend to all
Destination: 33:33:00:00:00:16 unknown. Resend to all
Destination: 33:33:00:00:00:16 unknown. Resend to all
Learning that c2:34:91:c3:99:50 is attached at port 2
Destination: 33:33:00:00:00:16 unknown. Resend to all
Destination: 33:33:00:00:00:02 unknown. Resend to all
Learning that c2:34:91:c3:99:50 is attached at port 1
Destination: 33:33:00:00:00:16 unknown. Resend to all
Destination: 33:33:00:00:00:02 unknown. Resend to all
Learning that c2:34:91:c3:99:50 is attached at port 3
Destination: 33:33:00:00:00:16 unknown. Resend to all
Destination: 33:33:00:00:00:02 unknown. Resend to all
Learning that c2:34:91:c3:99:50 is attached at port 1
Destination: 33:33:00:00:00:16 unknown. Resend to all
Destination: 33:33:00:00:00:02 unknown. Resend to all
Learning that c2:34:91:c3:99:50 is attached at port 3
Destination: 33:33:00:00:00:16 unknown. Resend to all
Destination: 33:33:00:00:00:02 unknown. Resend to all
Learning that c2:34:91:c3:99:50 is attached at port 3
Destination: 33:33:00:00:00:16 unknown. Resend to all
Destination: 33:33:00:00:00:02 unknown. Resend to all
Destination: 33:33:00:00:00:16 unknown. Resend to all
Destination: 33:33:00:00:00:16 unknown. Resend to all
Destination: 33:33:00:00:00:02 unknown. Resend to all
```

```
root@7ea2a161a991: ~/pox
```

[illegible]

[illegible]


```
root@364e16c6b1fd: ~  
*** Unknown command: h1 ping -c100 h2  
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=2.18 ms  
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=1.40 ms  
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=1.35 ms  
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=1.79 ms  
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=3.85 ms  
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=1.71 ms  
^C  
--- 10.0.0.2 ping statistics ---  
6 packets transmitted, 6 received, 0% packet loss, time 5021ms  
rtt min/avg/max/mdev = 1.357/2.050/3.858/0.853 ms
```

The average ping time for h1 ping -c100 h8 is: 12.864 ms

```
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=26.2 ms  
64 bytes from 10.0.0.8: icmp_seq=2 ttl=64 time=11.0 ms  
64 bytes from 10.0.0.8: icmp_seq=3 ttl=64 time=7.45 ms  
64 bytes from 10.0.0.8: icmp_seq=4 ttl=64 time=6.63 ms  
^C  
--- 10.0.0.8 ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 3010ms  
rtt min/avg/max/mdev = 6.637/12.864/26.298/7.933 ms  
mininet> █
```

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

Ans:

h1 ping -c100 h2

Min: 1.357 ms

Max: 3.858 ms

h1 ping -c100 h8

Min: 6.637 ms

Max: 26.298 ms

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

Ans:

In this case, because of using a switch instead of a hub, the controller takes less time.

The time difference between using a switch and hub, however not substantial, can cause a huge impact in the real time applications. Task 3 has less ping time because

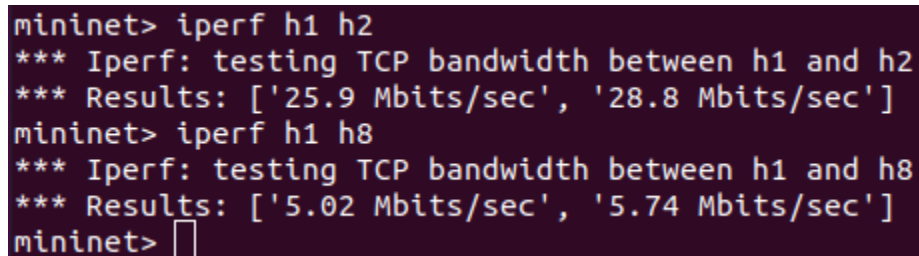
flooding takes place only for the first few packets. The switches will resend the packet to the appropriate port that is mapped in the "mac to port" mapping, once the target MAC address has been discovered. As a result, Task 3's future pings are executed much quicker.

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

a. What is the throughput for each case?

Ans:

Below screenshot shows the throughput:



```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['25.9 Mbits/sec', '28.8 Mbits/sec']
mininet> iperf h1 h8
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['5.02 Mbits/sec', '5.74 Mbits/sec']
mininet> 
```

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

Ans:

As we can see in the above screenshot, the throughput for task 3 is higher than the throughput for task 2. This is because after the "MAC_to_port" map has learned all the ports, there will be less packet flooding and hence less network congestion.

There is significant improvement in the throughputs, in case of h1 and h2, when task 2 and task 3 are compared, this is because the routes are pre computed and trained with changes in the controller.

There is no significant difference in the throughput, in case of h1 and h8, when task 2 and task 3 are compared, this is because the number of hops are more and also packet dropping is considered.