COEN241 Cloud Computing
HW3
Name: Yu Bi

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

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

```
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
```

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::a8b0:8fff:fea6:f1f7  prefixlen 64  scopeid 0x20<link>
        ether aa:b0:8f:a6:f1:f7  txqueuelen 1000  (Ethernet)
        RX packets 71  bytes 5474 (5.4 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
```

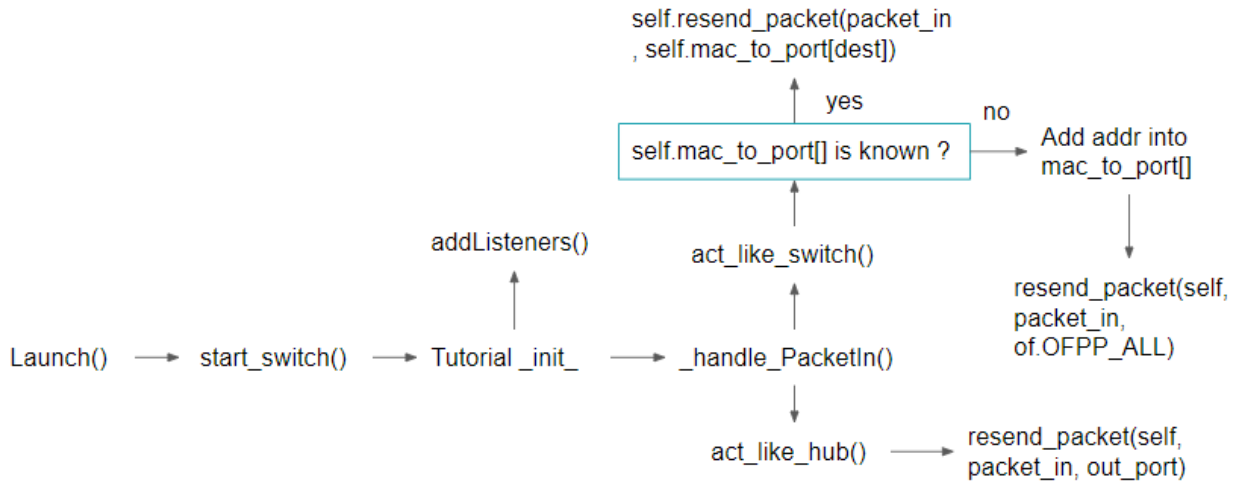
connect Mininet to POX

```
root@8b3827f08c82:~/pox# ./pox.py log.level --DEBUG misc.of_tutori
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-5.19.0-35-generic-x86_64-with-Ubuntu-
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:[00-00-00-00-00-01 4] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-01 4]
INFO:openflow.of_01:[00-00-00-00-00-06 5] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-06 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:[00-00-00-00-00-02 7] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-02 7]
INFO:openflow.of_01:[00-00-00-00-00-05 8] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-05 8]
DEBUG:openflow.of_01:1 connection aborted
```

```
[sudo] password for test:
root@8b3827f08c82:~# mn --custom binary_tree.py --controller remote --topo binar
y_tree
*** Error setting resource limits. Mininet's performance may be affected.
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Connecting to remote controller at 127.0.0.1:6633
*** 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>
```

Task 2:

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?



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 = 1.382 ms
        - h1 ping h8: average = 4.715 ms
    b. What is the minimum and maximum ping you have observed?
        - h1 ping h2: min = 0.943 ms, max = 2.209 ms
        - h1 ping h8: min = 3.673 ms, max = 49.359 ms
    c. What is the difference, and why?

The ping for h1 ping h8 is much higher than h1 ping h2. Because the packet needs to go through much more switches from h1 to h8 which needs more time.

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

a. What is “iperf” used for?

“Iperf” is a network tool, used for testing the bandwidth between the first hub and second hub. It can create TCP and UDP data streams and measure the throughput of a network that is carrying them.

b. What is the throughput for each case?

```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['14.9 Mbits/sec', '15.9 Mbits/sec']
mininet> iperf h1 h8
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['5.43 Mbits/sec', '6.04 Mbits/sec']
```

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

The throughput between h1 and h2 is much higher than the throughput between h1 and h8. The S3 is only connected with h1 and h2 hubs, so all the bandwidth can be utilized for these two. But for h1 and h2, there are several switches and these switches are connected to more hubs/switches. So their bandwidth is used by more links and transitions. That is why the throughput between h1 and h8 is only ⅓ of h1 and h2.

4. Which of the switches observe traffic?

All the switches observe traffic.

Try h1 ping h2 once, the connections are printed in the terminal with all switches to all other switches.

```
connection:  [00-00-00-00-00-03 5]lenth: 98
connection:  [00-00-00-00-00-02 6]lenth: 98
connection:  [00-00-00-00-00-02 6]lenth: 98
connection:  [00-00-00-00-00-03 5]lenth: 98
connection:  [00-00-00-00-00-03 5]lenth: 98
connection:  [00-00-00-00-00-04 2]lenth: 98
connection:  [00-00-00-00-00-04 2]lenth: 98
connection:  [00-00-00-00-00-04 2]lenth: 98
connection:  [00-00-00-00-00-04 2]lenth: 98
connection:  [00-00-00-00-00-01 3]lenth: 98
connection:  [00-00-00-00-00-01 3]lenth: 98
connection:  [00-00-00-00-00-01 3]lenth: 98
connection:  [00-00-00-00-00-01 3]lenth: 98
connection:  [00-00-00-00-00-02 6]lenth: 98
connection:  [00-00-00-00-00-05 7]lenth: 98
connection:  [00-00-00-00-00-05 7]lenth: 98
connection:  [00-00-00-00-00-05 7]lenth: 98
connection:  [00-00-00-00-00-05 7]lenth: 98
connection:  [00-00-00-00-00-05 7]lenth: 98
connection:  [00-00-00-00-00-05 7]lenth: 98
connection:  [00-00-00-00-00-05 7]lenth: 98
connection:  [00-00-00-00-00-05 7]lenth: 98
connection:  [00-00-00-00-00-05 7]lenth: 98
connection:  [00-00-00-00-00-07 1]lenth: 98
connection:  [00-00-00-00-00-07 1]lenth: 98
connection:  [00-00-00-00-00-07 1]lenth: 98
connection:  [00-00-00-00-00-07 1]lenth: 98
connection:  [00-00-00-00-00-07 1]lenth: 98
connection:  [00-00-00-00-00-07 1]lenth: 98
connection:  [00-00-00-00-00-07 1]lenth: 98
connection:  [00-00-00-00-00-07 1]lenth: 98
connection:  [00-00-00-00-00-07 1]lenth: 98
connection:  [00-00-00-00-00-07 1]lenth: 98
connection:  [00-00-00-00-00-07 1]lenth: 98
connection:  [00-00-00-00-00-04 2]lenth: 98
connection:  [00-00-00-00-00-01 3]lenth: 98
connection:  [00-00-00-00-00-06 4]lenth: 98
connection:  [00-00-00-00-00-06 4]lenth: 98
connection:  [00-00-00-00-00-06 4]lenth: 98
```

Task 3:

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).

There is s3 between h1 and h2. When h1 ping h2, the packet arrives at one of the in_port in s3, mac_to_port will learn whether this address is exited(packet_src). If not, add it to the mac_to_port list. Then check the packet_dst, if it is existed, send to this associated out_port. Otherwise send to all.

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?

- h1 ping h2: average = 1.478 ms
- h1 ping h8: average = 5.124 ms

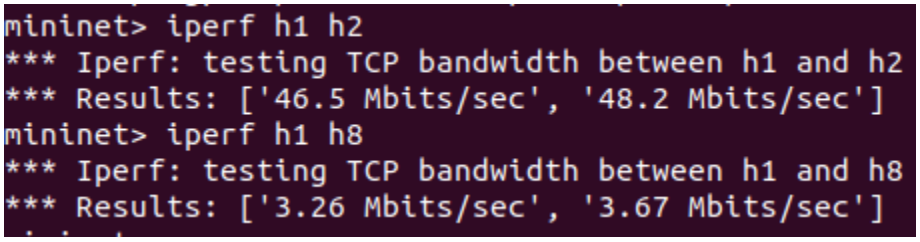
b. What is the minimum and maximum ping you have observed?

- h1 ping h2: min = 1.194 ms, max = 1.888 ms
- h1 ping h8: min = 4.396 ms, max = 6.343 ms

c. Any difference from Task 2 and why do you think there is a change if there is?
The average value is almost the same, but the max value is lower than task 2. After the first ping, it learned all the ports and addresses, so the terminal kept printing “destination known. Only send message to it.” That means after MAC learning, switches will only send to the right one, which will reduce the time needed. Also this will reduce the needs of flooding.

3. Q.3 Run “iperf h1 h8” 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?
The throughput for h1 and h2 is much higher than the result from task 2, while the throughput for h1 and h8 is a bit lower than task 2. When the act_like_switch is active, switches don’t need to broadcast the packets to all ports, so that reduces the conjunction.