Task1

Questions:

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

nodes:

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

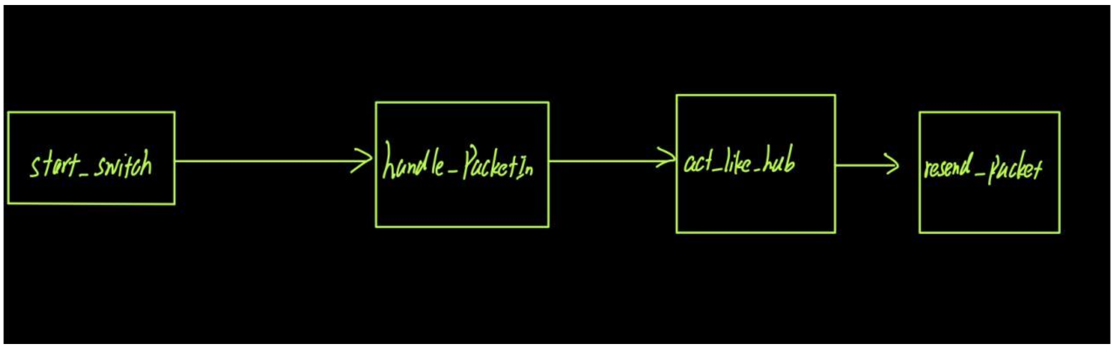
net:

```
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
```

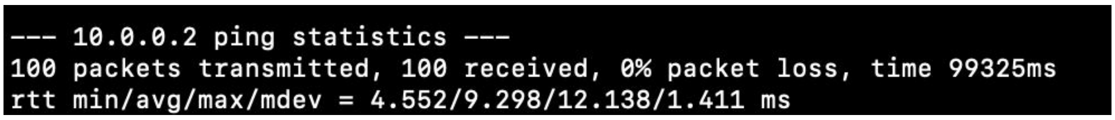
2. What is the output of “h7 ifconfig”

```
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::b066:80ff:fece:e6dc  prefixlen 64  scopeid 
0x20<link>
        ether b2:66:80:ce:e6:dc  txqueuelen 1000  (Ethernet)
        RX packets 142  bytes 10740 (10.7 KB)
        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
```

Task2
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?

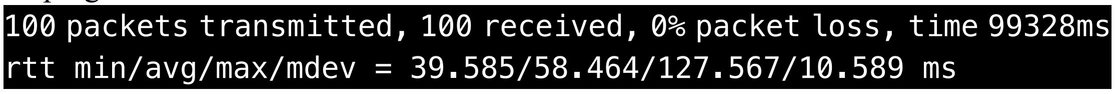


2. Have h1 ping h2, and h1 ping h8 for 100 times (e.g., h1 ping -c100 p2).
a. How long does it take (on average) to ping for each case?
h1 ping -c100 h2



```
--- 10.0.0.2 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99325ms
rtt min/avg/max/mdev = 4.552/9.298/12.138/1.411 ms
```

Avg time = 9.298ms
h1 ping -c100 h8



```
100 packets transmitted, 100 received, 0% packet loss, time 99328ms
rtt min/avg/max/mdev = 39.585/58.464/127.567/10.589 ms
```

Avg time = 58.464ms

b. What is the minimum and maximum ping you have observed?
h1 ping -c100 h2:
Min: 4.552ms   Max:12.138ms
h1 ping -c100 h8
Min: 39.585ms   Max: 127.567ms

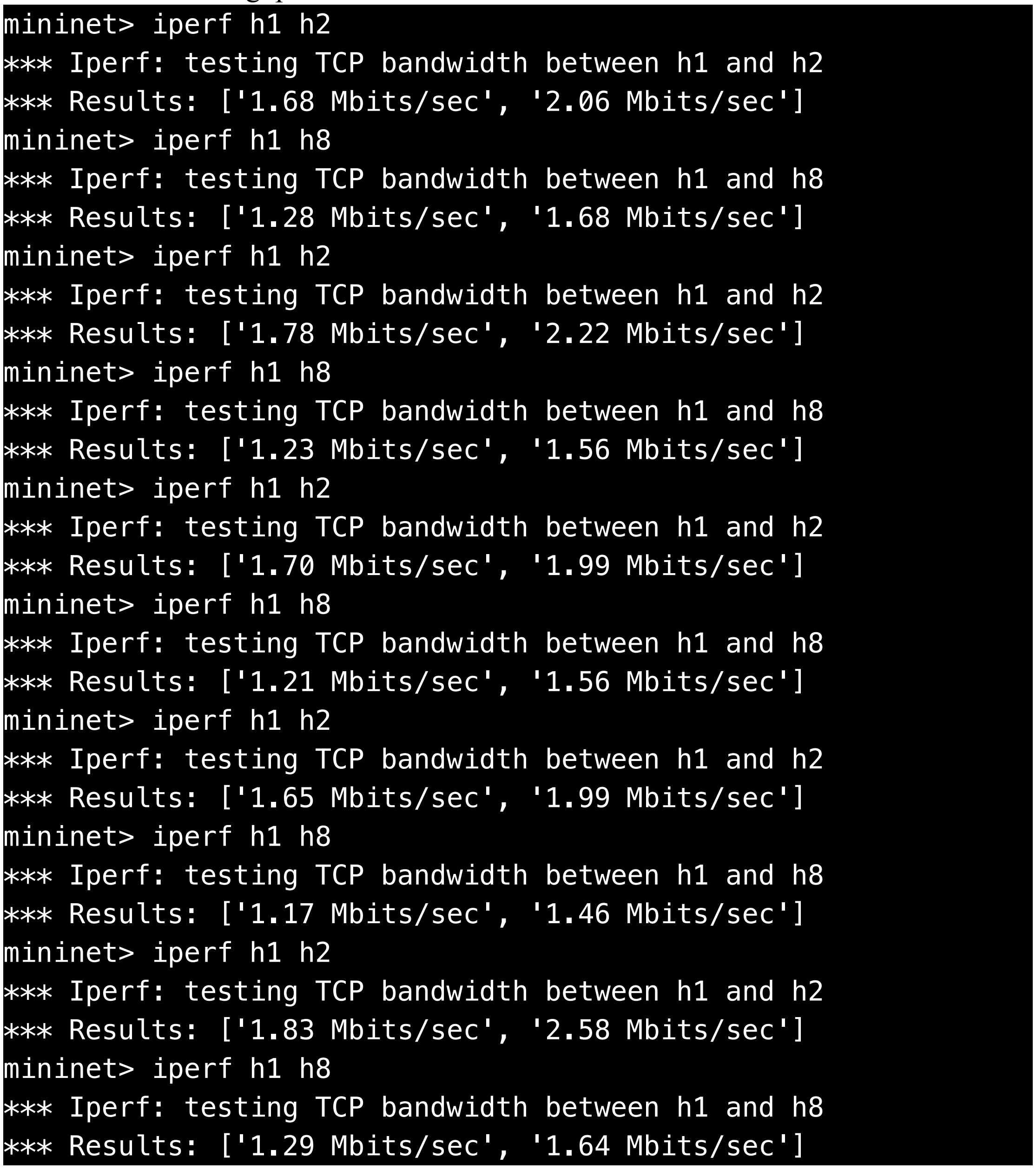
c. What is the difference, and why?
The transmission time between h1 and h8 is bigger than that between h1 and h2.
Reason:
The distance(links) between h1 and h8 is longer than that between h1 and h2. So the transmission time between h1 and h8 is bigger than that between h1 and h2. Additionally, hub directly transmits the package to all other ports, that is, the transmission line is shared. Then the occurrence of conflicts will increase and transmission time will increase when the distance is longer.

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

a. What is “iperf” used for?

Iperf is a commonly used network testing tool that 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: ['1.68 Mbits/sec', '2.06 Mbits/sec']
mininet> iperf h1 h8
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['1.28 Mbits/sec', '1.68 Mbits/sec']
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['1.78 Mbits/sec', '2.22 Mbits/sec']
mininet> iperf h1 h8
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['1.23 Mbits/sec', '1.56 Mbits/sec']
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['1.70 Mbits/sec', '1.99 Mbits/sec']
mininet> iperf h1 h8
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['1.21 Mbits/sec', '1.56 Mbits/sec']
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['1.65 Mbits/sec', '1.99 Mbits/sec']
mininet> iperf h1 h8
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['1.17 Mbits/sec', '1.46 Mbits/sec']
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['1.83 Mbits/sec', '2.58 Mbits/sec']
mininet> iperf h1 h8
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['1.29 Mbits/sec', '1.64 Mbits/sec']
```

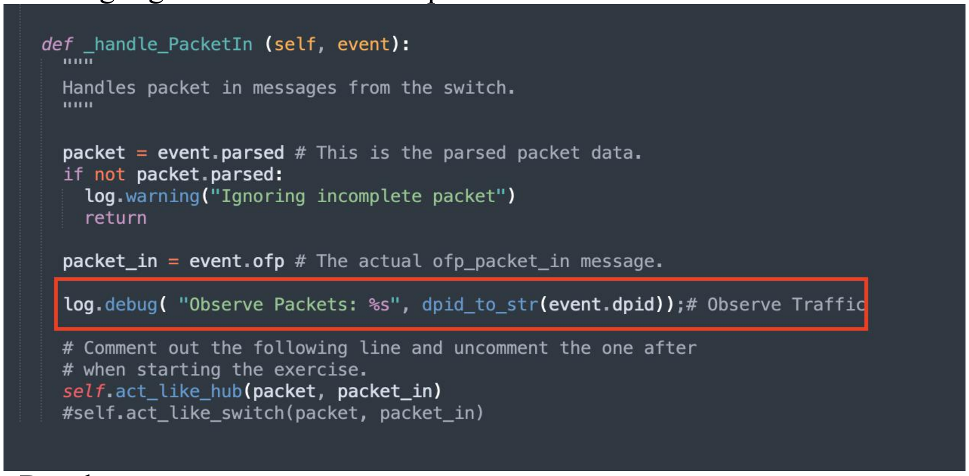
Avg bandwidth between h1 and h2: 1.73 Mbits/sec, 2.17 Mbits/sec

Avg bandwidth between h1 and h8: 1.24 Mbits/sec, 1.58 Mbits/sec

c. What is the difference, and explain the reasons for the difference.
The throughput of **iperf h1 h8** is similar to **iperf h1 h2,** but the bandwidth of **iperf h1 h8** is slightly smaller than that of **iperf h1 h2.**
Reason: hub directly transmits the package to all other ports, that is, the transmission line is shared.When the distance between two hosts is longer and all lines are shared, the occurrence of transmission collisions between the two hosts will increases. Then the bandwidth between two hosts will decrease.

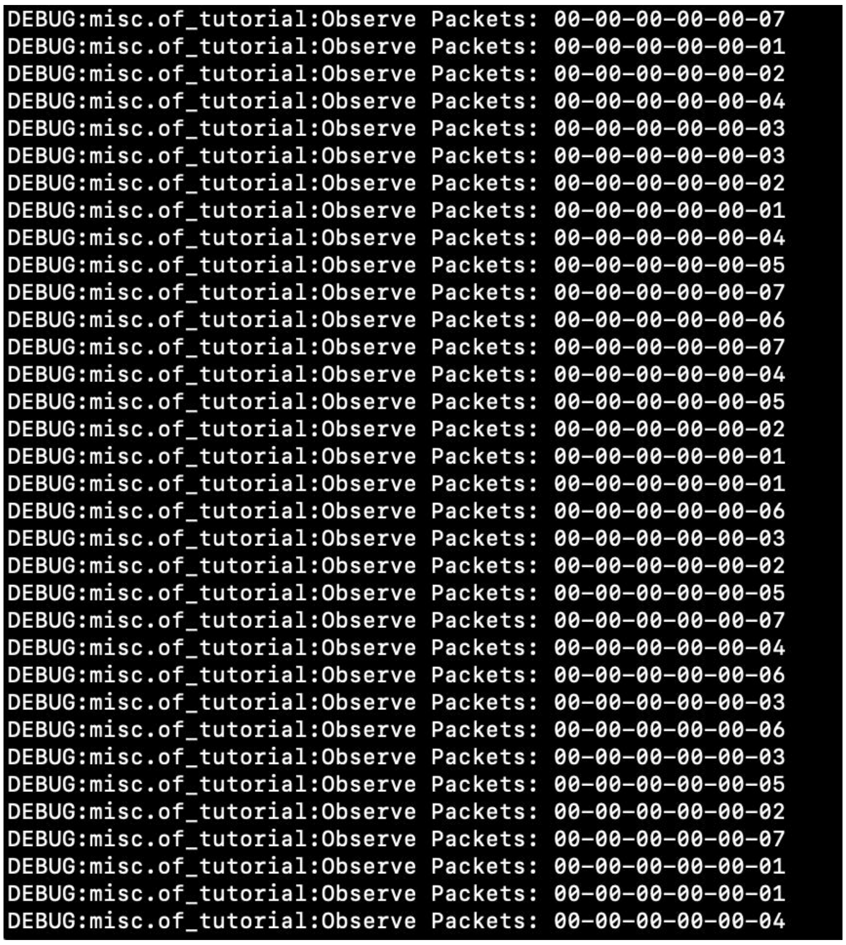
4. Which of the switches observe traffic? Please describe your way for observing such traffic on switches (e.g., adding some functions in the "of_tutorial" controller).

Adding log function to observe packets:



```
def _handle_PacketIn (self, event):
  """
  Handles packet in messages from the switch.
  """

  packet = event.parsed # This is the parsed packet data.
  if not packet.parsed:
    log.warning("Ignoring incomplete packet")
    return

  packet_in = event.ofp # The actual ofp_packet_in message.

  log.debug( "Observe Packets: %s", dpid_to_str(event.dpid));# Observe Traffic

  # Comment out the following line and uncomment the one after
  # when starting the exercise.
  self.act_like_hub(packet, packet_in)
  #self.act_like_switch(packet, packet_in)
```

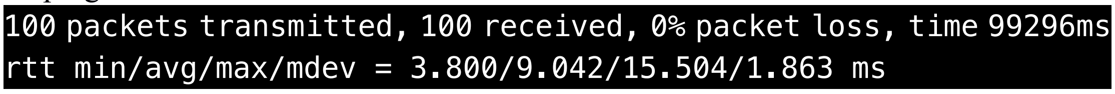
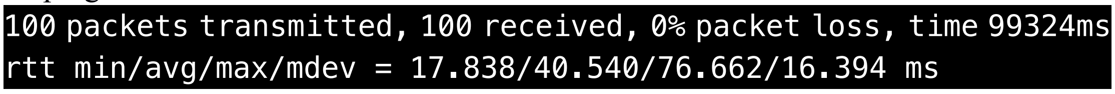
Result:



```
DEBUG:misc.of_tutorial:Observe Packets: 00-00-00-00-00-07
DEBUG:misc.of_tutorial:Observe Packets: 00-00-00-00-00-01
DEBUG:misc.of_tutorial:Observe Packets: 00-00-00-00-00-02
DEBUG:misc.of_tutorial:Observe Packets: 00-00-00-00-00-04
DEBUG:misc.of_tutorial:Observe Packets: 00-00-00-00-00-03
DEBUG:misc.of_tutorial:Observe Packets: 00-00-00-00-00-03
DEBUG:misc.of_tutorial:Observe Packets: 00-00-00-00-00-02
DEBUG:misc.of_tutorial:Observe Packets: 00-00-00-00-00-01
DEBUG:misc.of_tutorial:Observe Packets: 00-00-00-00-00-04
DEBUG:misc.of_tutorial:Observe Packets: 00-00-00-00-00-05
DEBUG:misc.of_tutorial:Observe Packets: 00-00-00-00-00-07
DEBUG:misc.of_tutorial:Observe Packets: 00-00-00-00-00-06
DEBUG:misc.of_tutorial:Observe Packets: 00-00-00-00-00-07
DEBUG:misc.of_tutorial:Observe Packets: 00-00-00-00-00-04
DEBUG:misc.of_tutorial:Observe Packets: 00-00-00-00-00-05
DEBUG:misc.of_tutorial:Observe Packets: 00-00-00-00-00-02
DEBUG:misc.of_tutorial:Observe Packets: 00-00-00-00-00-01
DEBUG:misc.of_tutorial:Observe Packets: 00-00-00-00-00-01
DEBUG:misc.of_tutorial:Observe Packets: 00-00-00-00-00-06
DEBUG:misc.of_tutorial:Observe Packets: 00-00-00-00-00-03
DEBUG:misc.of_tutorial:Observe Packets: 00-00-00-00-00-02
DEBUG:misc.of_tutorial:Observe Packets: 00-00-00-00-00-05
DEBUG:misc.of_tutorial:Observe Packets: 00-00-00-00-00-07
DEBUG:misc.of_tutorial:Observe Packets: 00-00-00-00-00-04
DEBUG:misc.of_tutorial:Observe Packets: 00-00-00-00-00-06
DEBUG:misc.of_tutorial:Observe Packets: 00-00-00-00-00-03
DEBUG:misc.of_tutorial:Observe Packets: 00-00-00-00-00-06
DEBUG:misc.of_tutorial:Observe Packets: 00-00-00-00-00-03
DEBUG:misc.of_tutorial:Observe Packets: 00-00-00-00-00-05
DEBUG:misc.of_tutorial:Observe Packets: 00-00-00-00-00-02
DEBUG:misc.of_tutorial:Observe Packets: 00-00-00-00-00-07
DEBUG:misc.of_tutorial:Observe Packets: 00-00-00-00-00-01
DEBUG:misc.of_tutorial:Observe Packets: 00-00-00-00-00-01
DEBUG:misc.of_tutorial:Observe Packets: 00-00-00-00-00-04
```

Task3
Questions:
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).
① If the target port is unknown: learn from the source packet. Learning that source packet is attached at port (packet_in.in_port)
② If the target port is known: send the packet to the port;
③ Other: Send to all other ports
E.g: h1 ping h2
If the port of h2 is unknown, just learn the port of h2 from packet.
if the port of h2 is known, just send packet to the port.
else send packet to all ports.

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

```
100 packets transmitted, 100 received, 0% packet loss, time 99296ms
rtt min/avg/max/mdev = 3.800/9.042/15.504/1.863 ms
```
Avg: 9.042ms
h1 ping -c100 h8

```
100 packets transmitted, 100 received, 0% packet loss, time 99324ms
rtt min/avg/max/mdev = 17.838/40.540/76.662/16.394 ms
```
Avg: 40.540 ms

b. What is the minimum and maximum ping you have observed?
h1 ping -c100 h2
Min: 3.800 ms Max: 15.504ms
h1 ping -c100 h8
Min: 17.838 ms Max: 76.662ms

c. Any difference from Task 2 and why do you think there is a change if there is?
The transfer time is slightly smaller when act_like_switch function is used than when act_like_hub function is used.

The Hub directly transmits the package to all other ports, that is, the transmission line is shared. However, the Switch can select the target port, which greatly reduces the occurrence of conflicts and provides an exclusive line for both parties.

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

a. What is the throughput for each case?

iperf h1 h2 : Results: ['5.88 Mbits/sec', '7.17 Mbits/sec']

iperf h1 h8: Results: ['1.81 Mbits/sec', '2.27 Mbits/sec']

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

Difference: Experiments show that the bandwidth of the switch is larger than that of the hub.

Reason:

The Hub directly transmits the package to all other ports, that is, the transmission line is shared. However, the Switch can select the target port, which greatly reduces the occurrence of conflicts and provides an exclusive line for both parties. For example, for virtual hosts h1, h2, and h3 in mininet, if h1 wants to ping h2, the hub will send packets to h2 and h3, while the switch will only send packets to h2.