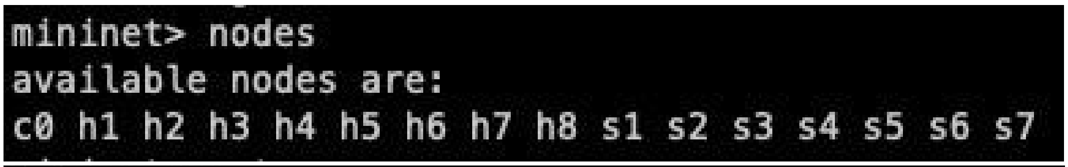
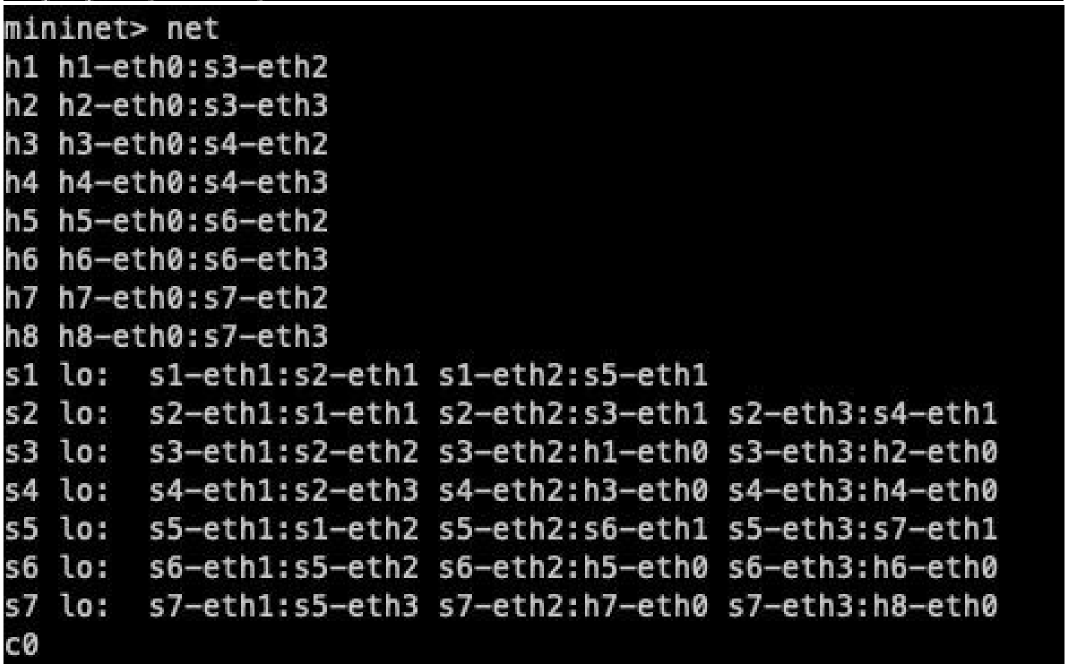
Task 1

Question 1:

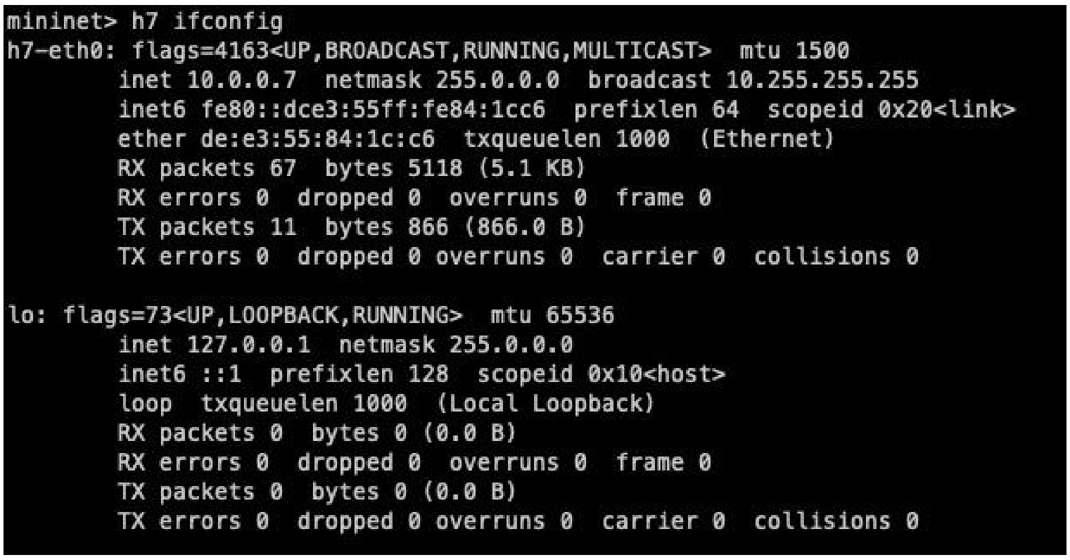


```
mininet> nodes
available nodes are:
c0 h1 h2 h3 h4 h5 h6 h7 h8 s1 s2 s3 s4 s5 s6 s7
```



```
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
```

Question 2:



```
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::dce3:55ff:fe84:1cc6  prefixlen 64  scopeid 0x20<link>
        ether de:e3:55:84:1c:c6  txqueuelen 1000  (Ethernet)
        RX packets 67  bytes 5118 (5.1 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
```

Task 2

Question 1:
launch() --> start_switch() --> Tutotial(event.connetion) --> __init__() -->_handle_PacketIn() --> act_like_hub() --> resend_packet()

Question 2:
(a)
h1 ping h2:
    average: 1.757 ms
h1 ping h8:
    average: 8.549 ms

(b)
h1 ping h2:
    max: 6.132 ms
    min: 1.577 ms
h1 ping h8:
    max: 19.377 ms
    min: 8.084 ms

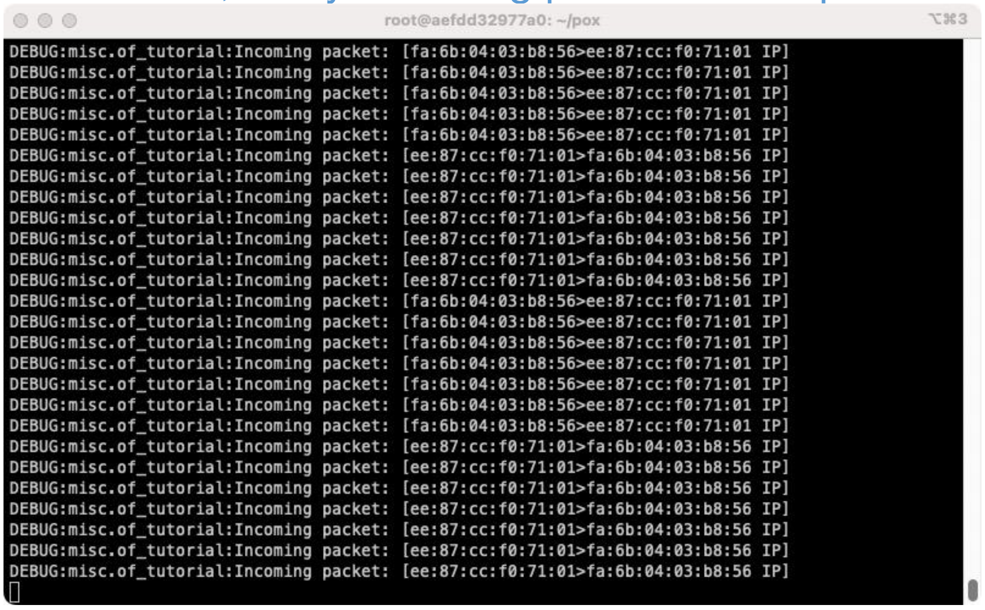
(c)
Overall, h1 ping h2 takes much less time than h1 ping h8. The reason is, in our custom topology, h1 only take one switch to h2, but takes 5 switches to h8.
Within each ping (h1 to h2 / h1 to h8), first ping take the most time, almost twice as the rest. Other pings takes pretty much the same time including the min time. The reason is, the first ping is a “Cold Start”, which mean it needs to figure out what path it needs to take to get to detination.

Question 3:

(a)
iperf is to test bandwith between two nodes
(b)
iperf h1 h2:
    ['16.3 Gbits/sec', '16.3 Gbits/sec']
iperf h1 h8:
    ['12.6 Gbits/sec', '12.6 Gbits/sec']
(c)
h1 to h2 take one switch while h1 to h8 takes 5 switches, so there will be more performance loss on the way. This explains why bandwith from h1 to h8 has smaller than h1 to h2

Question 4:
For h1 ping h2:
    s3 observe traffic
For h1 ping h8:
    s3, s2, s1, s5, s7 observe traffic
To observe traffic, add a line in of_tutorial.py:

```
class Tutorial():
    def _handle_PacketIn():
        log.debug(f"Incoming packet: {packet}")
```

In this case, every incoming packet will be printed.


```
DEBUG:misc.of_tutorial:Incoming packet: [fa:6b:04:03:b8:56>ee:87:cc:f0:71:01 IP]
DEBUG:misc.of_tutorial:Incoming packet: [fa:6b:04:03:b8:56>ee:87:cc:f0:71:01 IP]
DEBUG:misc.of_tutorial:Incoming packet: [fa:6b:04:03:b8:56>ee:87:cc:f0:71:01 IP]
DEBUG:misc.of_tutorial:Incoming packet: [fa:6b:04:03:b8:56>ee:87:cc:f0:71:01 IP]
DEBUG:misc.of_tutorial:Incoming packet: [fa:6b:04:03:b8:56>ee:87:cc:f0:71:01 IP]
DEBUG:misc.of_tutorial:Incoming packet: [ee:87:cc:f0:71:01>fa:6b:04:03:b8:56 IP]
DEBUG:misc.of_tutorial:Incoming packet: [ee:87:cc:f0:71:01>fa:6b:04:03:b8:56 IP]
DEBUG:misc.of_tutorial:Incoming packet: [ee:87:cc:f0:71:01>fa:6b:04:03:b8:56 IP]
DEBUG:misc.of_tutorial:Incoming packet: [ee:87:cc:f0:71:01>fa:6b:04:03:b8:56 IP]
DEBUG:misc.of_tutorial:Incoming packet: [ee:87:cc:f0:71:01>fa:6b:04:03:b8:56 IP]
DEBUG:misc.of_tutorial:Incoming packet: [ee:87:cc:f0:71:01>fa:6b:04:03:b8:56 IP]
DEBUG:misc.of_tutorial:Incoming packet: [ee:87:cc:f0:71:01>fa:6b:04:03:b8:56 IP]
DEBUG:misc.of_tutorial:Incoming packet: [fa:6b:04:03:b8:56>ee:87:cc:f0:71:01 IP]
DEBUG:misc.of_tutorial:Incoming packet: [fa:6b:04:03:b8:56>ee:87:cc:f0:71:01 IP]
DEBUG:misc.of_tutorial:Incoming packet: [fa:6b:04:03:b8:56>ee:87:cc:f0:71:01 IP]
DEBUG:misc.of_tutorial:Incoming packet: [fa:6b:04:03:b8:56>ee:87:cc:f0:71:01 IP]
DEBUG:misc.of_tutorial:Incoming packet: [fa:6b:04:03:b8:56>ee:87:cc:f0:71:01 IP]
DEBUG:misc.of_tutorial:Incoming packet: [fa:6b:04:03:b8:56>ee:87:cc:f0:71:01 IP]
DEBUG:misc.of_tutorial:Incoming packet: [fa:6b:04:03:b8:56>ee:87:cc:f0:71:01 IP]
DEBUG:misc.of_tutorial:Incoming packet: [ee:87:cc:f0:71:01>fa:6b:04:03:b8:56 IP]
DEBUG:misc.of_tutorial:Incoming packet: [ee:87:cc:f0:71:01>fa:6b:04:03:b8:56 IP]
DEBUG:misc.of_tutorial:Incoming packet: [ee:87:cc:f0:71:01>fa:6b:04:03:b8:56 IP]
DEBUG:misc.of_tutorial:Incoming packet: [ee:87:cc:f0:71:01>fa:6b:04:03:b8:56 IP]
DEBUG:misc.of_tutorial:Incoming packet: [ee:87:cc:f0:71:01>fa:6b:04:03:b8:56 IP]
DEBUG:misc.of_tutorial:Incoming packet: [ee:87:cc:f0:71:01>fa:6b:04:03:b8:56 IP]
DEBUG:misc.of_tutorial:Incoming packet: [ee:87:cc:f0:71:01>fa:6b:04:03:b8:56 IP]
```

Task 3

Question 1:
In of_tutorial.py, there is a Tutorial class, which has a mac_to_port dictionary to store mac to port key-value pairs, and the dictionary is empty when this class in instantiated. This program is functioning as a remote controller for our entire miniet network.

Every time, a packet comes in, the program will check if the source port and source mac is stored in mac_to_port dictionary, if no it will “learn”, which means to add this source mac-port key-value pair into mac_to_port dictionary.

Then, the porgram will check if destination mac has a related port stored in mac_to_port dictionation, if so, send to this port, if not send pacet to everybody.

Quetion 2:
(a)
h1 ping h2:
  avg: 1.656 ms
h1 ping h8:
  avg: 7.177 ms
(b)
h1 ping h2:
  min: 1.289 ms
  max: 2.825 ms
h1 ping h8:
  min: 6.636ms
  max: 15.673 ms
(c)
Yes, there is a change comparing to Task 2. For both case, min and avg stays almost the same, while the max

(always the first ping) has an obvious reduce in time. The reason causing this is, when a host pinging another and doesn’t know the other, it will communicate with the remote controller for instructions. In our case, we have a more smart controller, so it reduces time comparing to Task 2.

Quetion 3:

(a)

iperf h1 h2:

```
	['20.9 Mbits/sec', '23.2 Mbits/sec']
```

iperf h1 h8:

```
	['3.12 Mbits/sec', '3.71 Mbits/sec']
```

(b)

Bandwith in Task 3 is significantly lower than Task 2 in both cases. A possible reason could be, when calling iperf, h1 needs to consult remost controller, and the network between miniet and pox processes has much smaller bandwith comparing to miniet nodes talking to each other within miniet process.