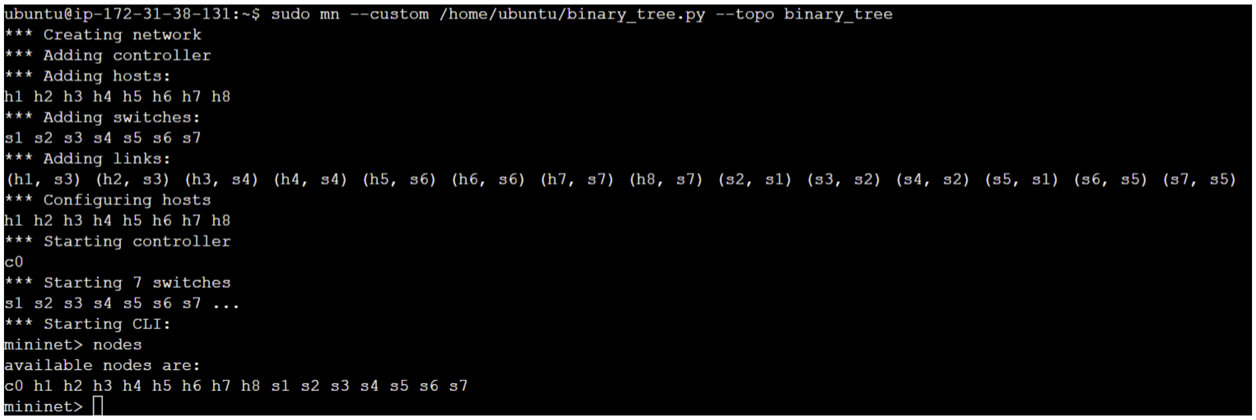
HW-3

Nityanand Pujari – W1650422

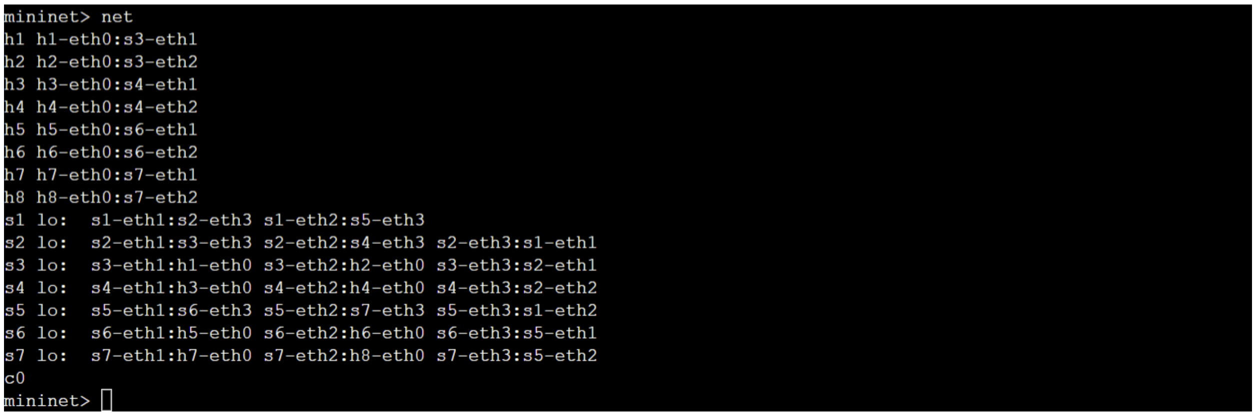
Task 1 : Defining custom topologies

1. What is the output of "nodes" and "net"?
   a. nodes



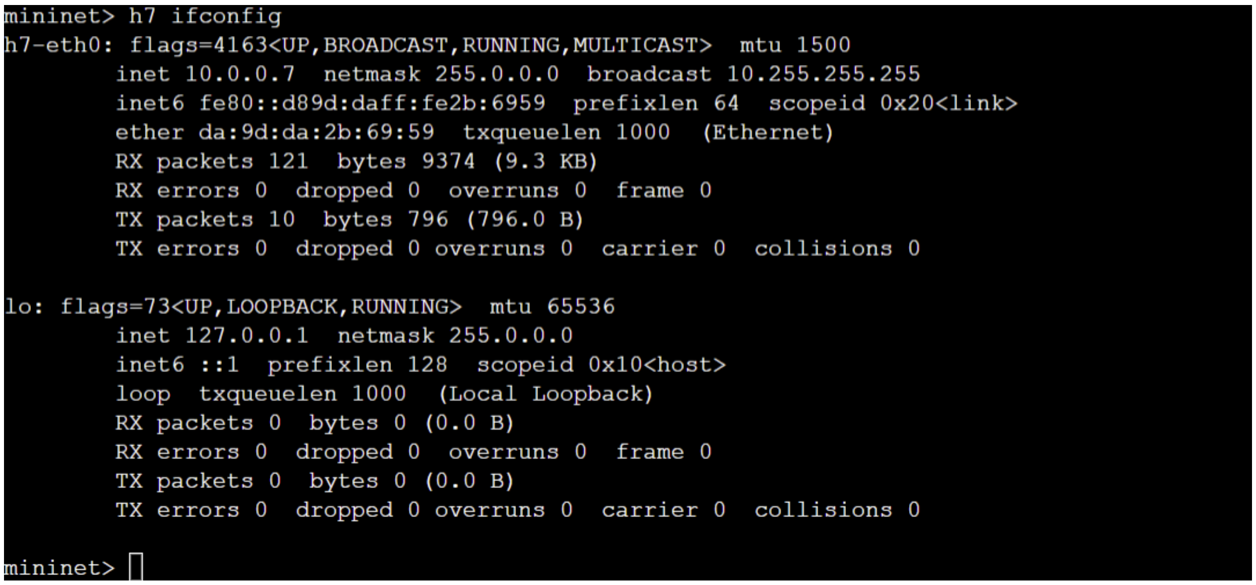
```
ubuntu@ip-172-31-38-131:~$ sudo mn --custom /home/ubuntu/binary_tree.py --topo binary_tree
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7
*** Adding links:
(h1, s3) (h2, s3) (h3, s4) (h4, s4) (h5, s6) (h6, s6) (h7, s7) (h8, s7) (s2, s1) (s3, s2) (s4, s2) (s5, s1) (s6, s5) (s7, s5)
*** 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> nodes
available nodes are:
c0 h1 h2 h3 h4 h5 h6 h7 h8 s1 s2 s3 s4 s5 s6 s7
mininet> 
```

   b. 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> 
```

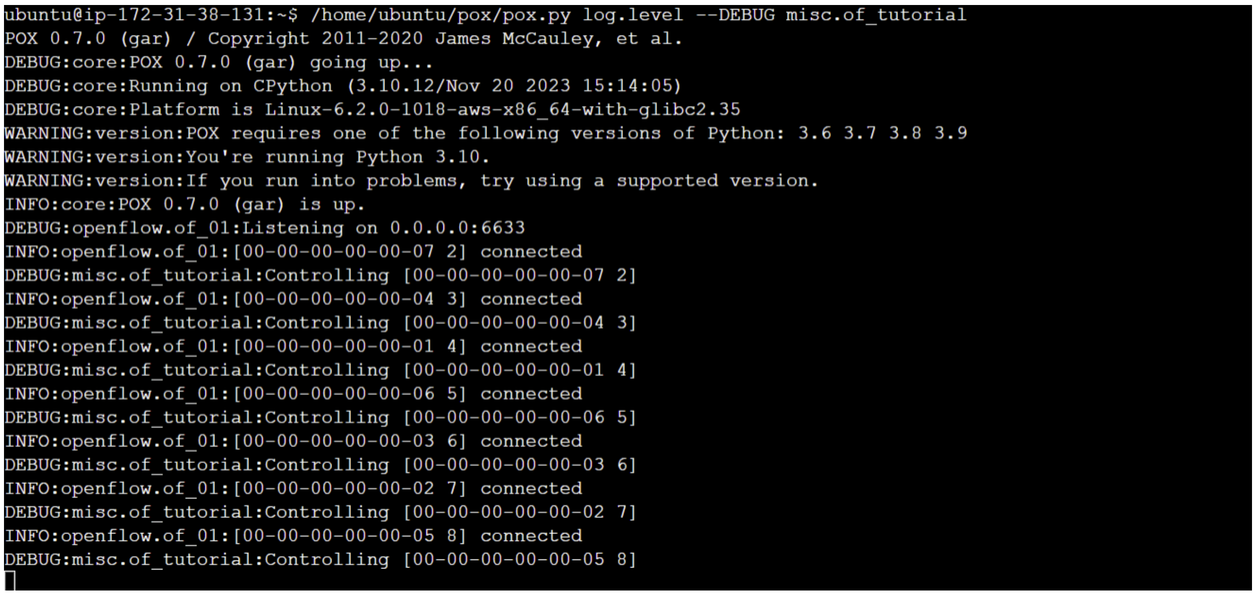
2. What is the output of "h7 ifconfig" ?


```
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::d89d:daff:fe2b:6959  prefixlen 64  scopeid 0x20<link>
        ether da:9d:da:2b:69:59  txqueuelen 1000  (Ethernet)
        RX packets 121  bytes 9374 (9.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 10  bytes 796 (796.0 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> 
```

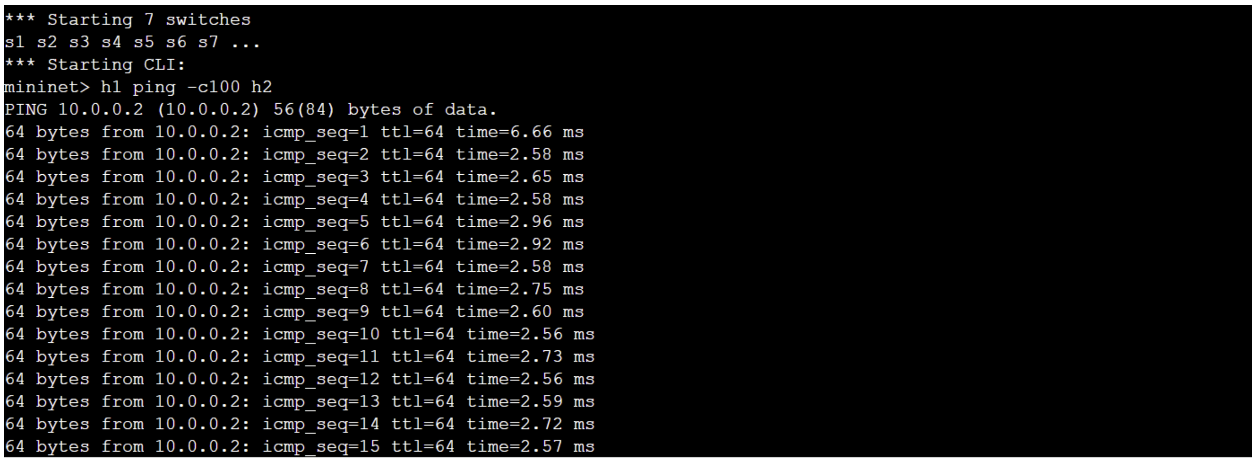
Task 2 : Analyze the “of_tutorial” controller

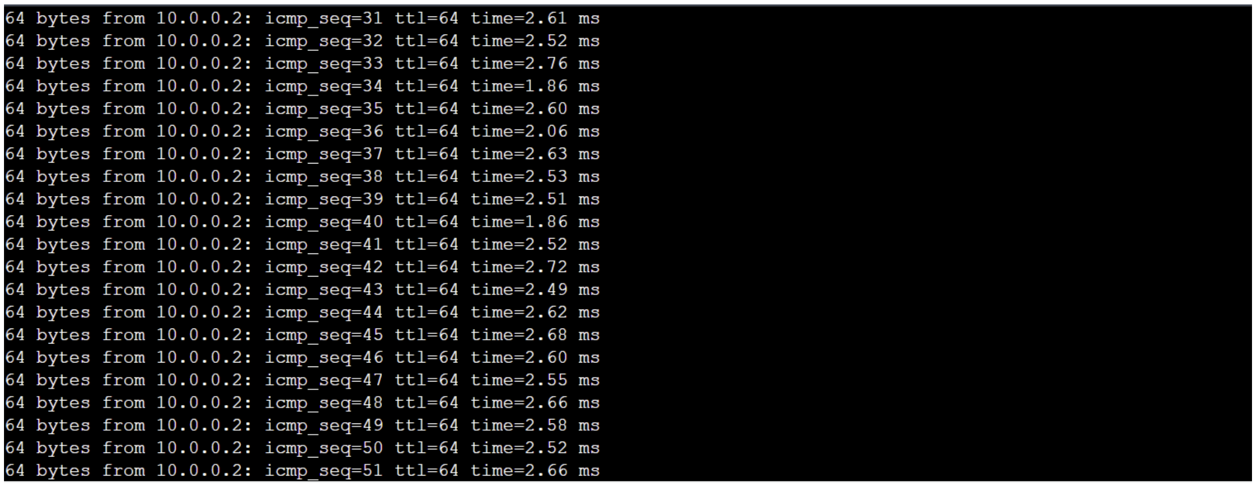
1. Draw the function call graph of this controller. For example, once a packet comes to the controller, which function is the first to be called, which one is the second, and so forth?

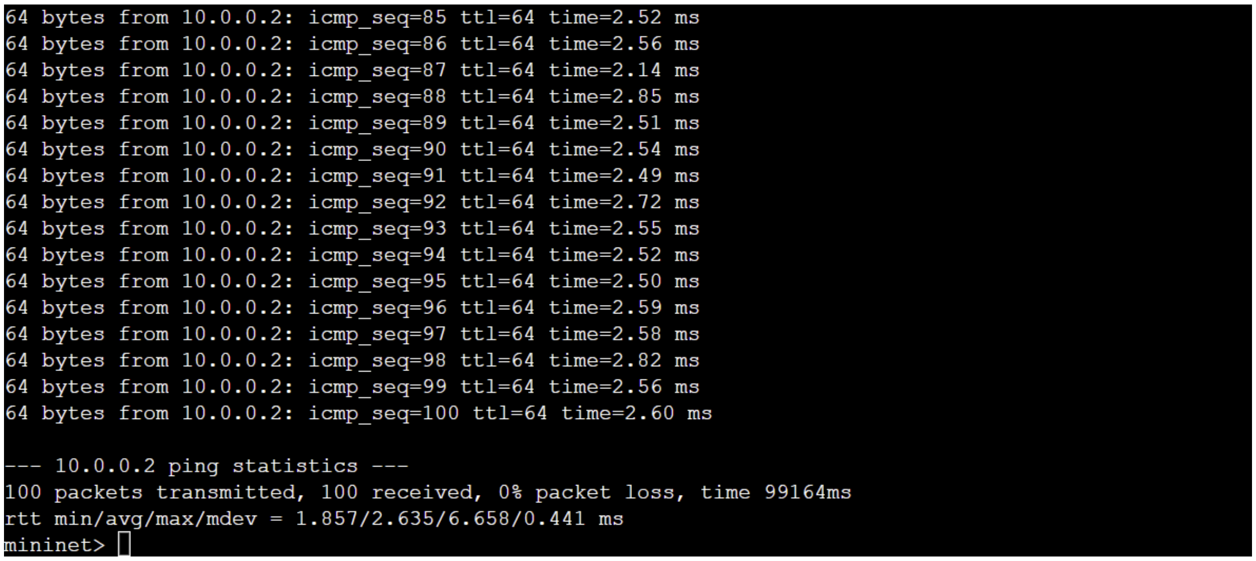
The process begins by initiating the POX listener using the command `./pox.py log.level --DEBUG misc.of_tutorial`. Executing this command activates the 'start switch', which triggers the `_handle_PacketIn()` method on the switch. This method, in turn, invokes the `act_like_hub()` function. The `act_like_hub()` function mimics the behavior of a hub by forwarding packets to all ports except the one it was received on. Subsequently, the `resend_packet()` method is employed, which appends a packet to the message payload and executes an action on it. Through this action, the switch is directed to forward the packet to a designated port. The sequence of function calls in the controller can be outlined as follows: `start switch -> _handle_PacketIn() -> act_like_hub() -> resend_packet() -> send(message)`.


```
ubuntu@ip-172-31-38-131:~$ /home/ubuntu/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 (3.10.12/Nov 20 2023 15:14:05)
DEBUG:core:Platform is Linux-6.2.0-1018-aws-x86_64-with-glibc2.35
WARNING:version:POX requires one of the following versions of Python: 3.6 3.7 3.8 3.9
WARNING:version:You're running Python 3.10.
WARNING:version:If you run into problems, try using a supported version.
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]
```

2. **h1 ping -c100 h2**

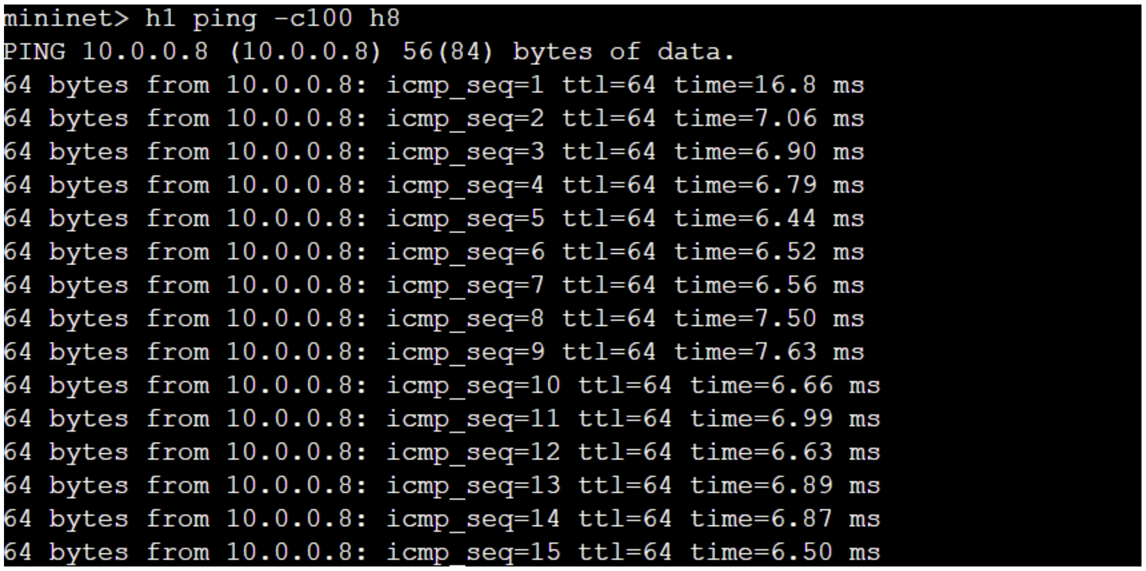

```
*** Starting 7 switches
s1 s2 s3 s4 s5 s6 s7 ...
*** Starting CLI:
mininet> h1 ping -c100 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=6.66 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=2.58 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=2.65 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=2.58 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=2.96 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=2.92 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=2.58 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=2.75 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=2.60 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=2.56 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=2.73 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=2.56 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=2.59 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=2.72 ms
64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=2.57 ms
```


```
64 bytes from 10.0.0.2: icmp_seq=31 ttl=64 time=2.61 ms
64 bytes from 10.0.0.2: icmp_seq=32 ttl=64 time=2.52 ms
64 bytes from 10.0.0.2: icmp_seq=33 ttl=64 time=2.76 ms
64 bytes from 10.0.0.2: icmp_seq=34 ttl=64 time=1.86 ms
64 bytes from 10.0.0.2: icmp_seq=35 ttl=64 time=2.60 ms
64 bytes from 10.0.0.2: icmp_seq=36 ttl=64 time=2.06 ms
64 bytes from 10.0.0.2: icmp_seq=37 ttl=64 time=2.63 ms
64 bytes from 10.0.0.2: icmp_seq=38 ttl=64 time=2.53 ms
64 bytes from 10.0.0.2: icmp_seq=39 ttl=64 time=2.51 ms
64 bytes from 10.0.0.2: icmp_seq=40 ttl=64 time=1.86 ms
64 bytes from 10.0.0.2: icmp_seq=41 ttl=64 time=2.52 ms
64 bytes from 10.0.0.2: icmp_seq=42 ttl=64 time=2.72 ms
64 bytes from 10.0.0.2: icmp_seq=43 ttl=64 time=2.49 ms
64 bytes from 10.0.0.2: icmp_seq=44 ttl=64 time=2.62 ms
64 bytes from 10.0.0.2: icmp_seq=45 ttl=64 time=2.68 ms
64 bytes from 10.0.0.2: icmp_seq=46 ttl=64 time=2.60 ms
64 bytes from 10.0.0.2: icmp_seq=47 ttl=64 time=2.55 ms
64 bytes from 10.0.0.2: icmp_seq=48 ttl=64 time=2.66 ms
64 bytes from 10.0.0.2: icmp_seq=49 ttl=64 time=2.58 ms
64 bytes from 10.0.0.2: icmp_seq=50 ttl=64 time=2.52 ms
64 bytes from 10.0.0.2: icmp_seq=51 ttl=64 time=2.66 ms
```

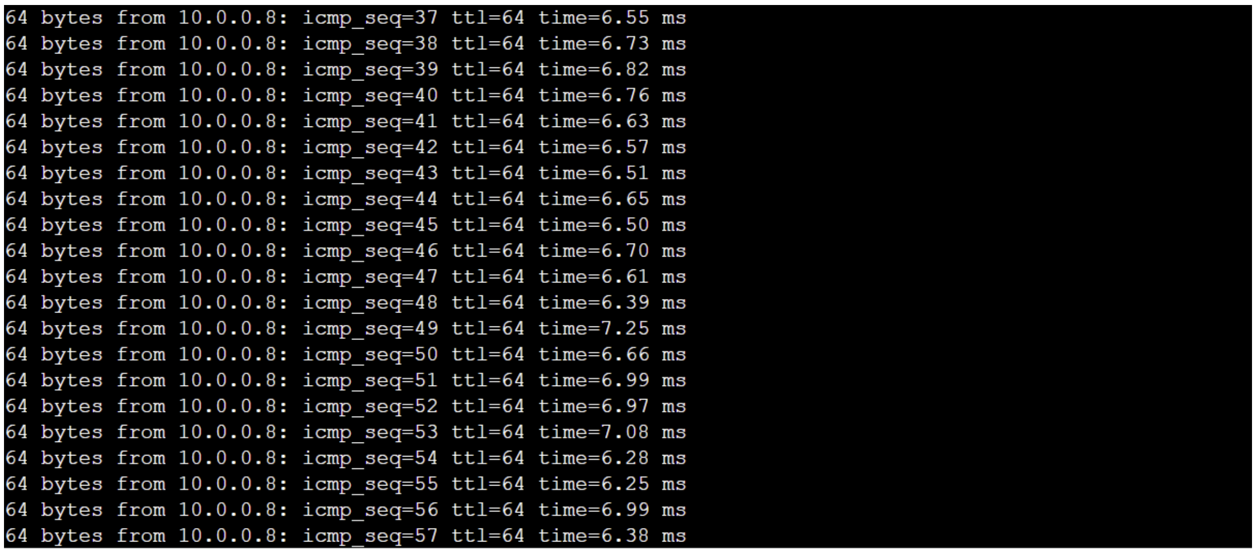


```
64 bytes from 10.0.0.2: icmp_seq=85 ttl=64 time=2.52 ms
64 bytes from 10.0.0.2: icmp_seq=86 ttl=64 time=2.56 ms
64 bytes from 10.0.0.2: icmp_seq=87 ttl=64 time=2.14 ms
64 bytes from 10.0.0.2: icmp_seq=88 ttl=64 time=2.85 ms
64 bytes from 10.0.0.2: icmp_seq=89 ttl=64 time=2.51 ms
64 bytes from 10.0.0.2: icmp_seq=90 ttl=64 time=2.54 ms
64 bytes from 10.0.0.2: icmp_seq=91 ttl=64 time=2.49 ms
64 bytes from 10.0.0.2: icmp_seq=92 ttl=64 time=2.72 ms
64 bytes from 10.0.0.2: icmp_seq=93 ttl=64 time=2.55 ms
64 bytes from 10.0.0.2: icmp_seq=94 ttl=64 time=2.52 ms
64 bytes from 10.0.0.2: icmp_seq=95 ttl=64 time=2.50 ms
64 bytes from 10.0.0.2: icmp_seq=96 ttl=64 time=2.59 ms
64 bytes from 10.0.0.2: icmp_seq=97 ttl=64 time=2.58 ms
64 bytes from 10.0.0.2: icmp_seq=98 ttl=64 time=2.82 ms
64 bytes from 10.0.0.2: icmp_seq=99 ttl=64 time=2.56 ms
64 bytes from 10.0.0.2: icmp_seq=100 ttl=64 time=2.60 ms

--- 10.0.0.2 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99164ms
rtt min/avg/max/mdev = 1.857/2.635/6.658/0.441 ms
mininet> 
```

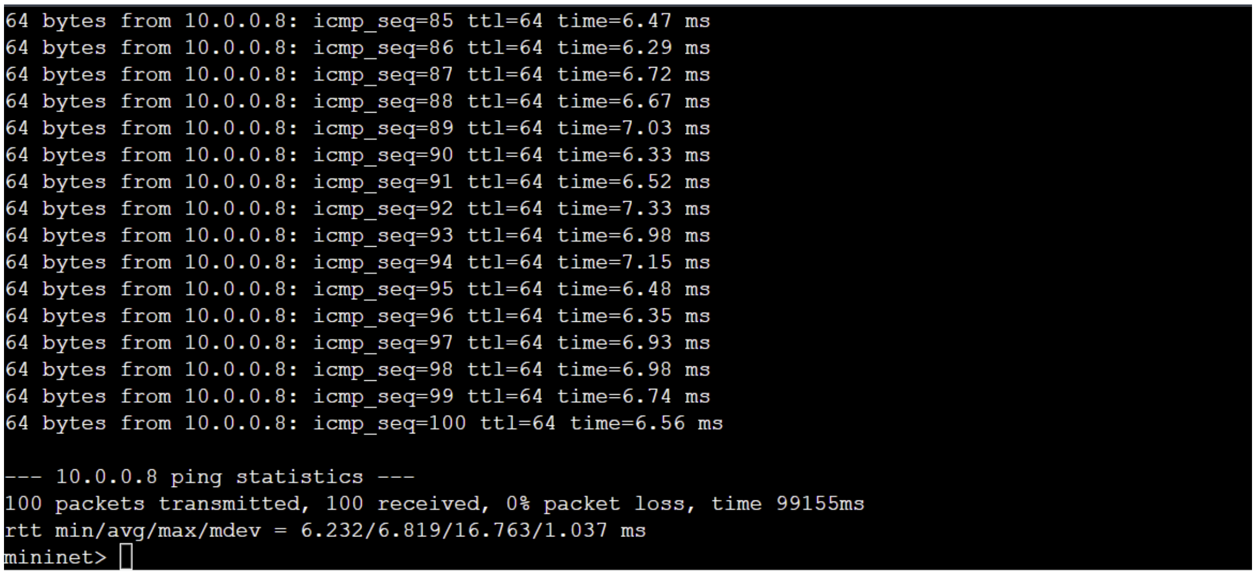
h1 ping -c100 h8



```
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=16.8 ms
64 bytes from 10.0.0.8: icmp_seq=2 ttl=64 time=7.06 ms
64 bytes from 10.0.0.8: icmp_seq=3 ttl=64 time=6.90 ms
64 bytes from 10.0.0.8: icmp_seq=4 ttl=64 time=6.79 ms
64 bytes from 10.0.0.8: icmp_seq=5 ttl=64 time=6.44 ms
64 bytes from 10.0.0.8: icmp_seq=6 ttl=64 time=6.52 ms
64 bytes from 10.0.0.8: icmp_seq=7 ttl=64 time=6.56 ms
64 bytes from 10.0.0.8: icmp_seq=8 ttl=64 time=7.50 ms
64 bytes from 10.0.0.8: icmp_seq=9 ttl=64 time=7.63 ms
64 bytes from 10.0.0.8: icmp_seq=10 ttl=64 time=6.66 ms
64 bytes from 10.0.0.8: icmp_seq=11 ttl=64 time=6.99 ms
64 bytes from 10.0.0.8: icmp_seq=12 ttl=64 time=6.63 ms
64 bytes from 10.0.0.8: icmp_seq=13 ttl=64 time=6.89 ms
64 bytes from 10.0.0.8: icmp_seq=14 ttl=64 time=6.87 ms
64 bytes from 10.0.0.8: icmp_seq=15 ttl=64 time=6.50 ms
```



```
64 bytes from 10.0.0.8: icmp_seq=37 ttl=64 time=6.55 ms
64 bytes from 10.0.0.8: icmp_seq=38 ttl=64 time=6.73 ms
64 bytes from 10.0.0.8: icmp_seq=39 ttl=64 time=6.82 ms
64 bytes from 10.0.0.8: icmp_seq=40 ttl=64 time=6.76 ms
64 bytes from 10.0.0.8: icmp_seq=41 ttl=64 time=6.63 ms
64 bytes from 10.0.0.8: icmp_seq=42 ttl=64 time=6.57 ms
64 bytes from 10.0.0.8: icmp_seq=43 ttl=64 time=6.51 ms
64 bytes from 10.0.0.8: icmp_seq=44 ttl=64 time=6.65 ms
64 bytes from 10.0.0.8: icmp_seq=45 ttl=64 time=6.50 ms
64 bytes from 10.0.0.8: icmp_seq=46 ttl=64 time=6.70 ms
64 bytes from 10.0.0.8: icmp_seq=47 ttl=64 time=6.61 ms
64 bytes from 10.0.0.8: icmp_seq=48 ttl=64 time=6.39 ms
64 bytes from 10.0.0.8: icmp_seq=49 ttl=64 time=7.25 ms
64 bytes from 10.0.0.8: icmp_seq=50 ttl=64 time=6.66 ms
64 bytes from 10.0.0.8: icmp_seq=51 ttl=64 time=6.99 ms
64 bytes from 10.0.0.8: icmp_seq=52 ttl=64 time=6.97 ms
64 bytes from 10.0.0.8: icmp_seq=53 ttl=64 time=7.08 ms
64 bytes from 10.0.0.8: icmp_seq=54 ttl=64 time=6.28 ms
64 bytes from 10.0.0.8: icmp_seq=55 ttl=64 time=6.25 ms
64 bytes from 10.0.0.8: icmp_seq=56 ttl=64 time=6.99 ms
64 bytes from 10.0.0.8: icmp_seq=57 ttl=64 time=6.38 ms
```



```
64 bytes from 10.0.0.8: icmp_seq=85 ttl=64 time=6.47 ms
64 bytes from 10.0.0.8: icmp_seq=86 ttl=64 time=6.29 ms
64 bytes from 10.0.0.8: icmp_seq=87 ttl=64 time=6.72 ms
64 bytes from 10.0.0.8: icmp_seq=88 ttl=64 time=6.67 ms
64 bytes from 10.0.0.8: icmp_seq=89 ttl=64 time=7.03 ms
64 bytes from 10.0.0.8: icmp_seq=90 ttl=64 time=6.33 ms
64 bytes from 10.0.0.8: icmp_seq=91 ttl=64 time=6.52 ms
64 bytes from 10.0.0.8: icmp_seq=92 ttl=64 time=7.33 ms
64 bytes from 10.0.0.8: icmp_seq=93 ttl=64 time=6.98 ms
64 bytes from 10.0.0.8: icmp_seq=94 ttl=64 time=7.15 ms
64 bytes from 10.0.0.8: icmp_seq=95 ttl=64 time=6.48 ms
64 bytes from 10.0.0.8: icmp_seq=96 ttl=64 time=6.35 ms
64 bytes from 10.0.0.8: icmp_seq=97 ttl=64 time=6.93 ms
64 bytes from 10.0.0.8: icmp_seq=98 ttl=64 time=6.98 ms
64 bytes from 10.0.0.8: icmp_seq=99 ttl=64 time=6.74 ms
64 bytes from 10.0.0.8: icmp_seq=100 ttl=64 time=6.56 ms

--- 10.0.0.8 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99155ms
rtt min/avg/max/mdev = 6.232/6.819/16.763/1.037 ms
mininet> 
```

a) How long does it take (on average) to ping for each case?

	Average Ping
h1 ping -c100 h2	2.6ms
h1 ping -c100 h8	6.81ms

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

	Minimum Ping	Maximum Ping
h1 ping -c100 h2	1.8ms	6.658ms
h1 ping -c100 h8	6.232ms	16.763ms

c) What is the difference, and why?

Host **h1** is connected to switch **s3**, which connects to **s2**, and then up to the root switch **s1**.

Host **h2** is also connected to switch **s3** and follows the same path up to **s1** as **h1**.

Host **h8** is connected to switch **s7**, which connects to **s5**, and then up to the root switch **s1**.

When **h1** pings **h2**, the packets go from **h1** to **s3** and then directly to **h2**, which involves just one switch hop.

However, when **h1** pings **h8**, the packets travel a longer path:

From **h1** to **s3**

From **s3** to **s2**

From **s2** to **s1** (the root switch)

From **s1** to **s5**

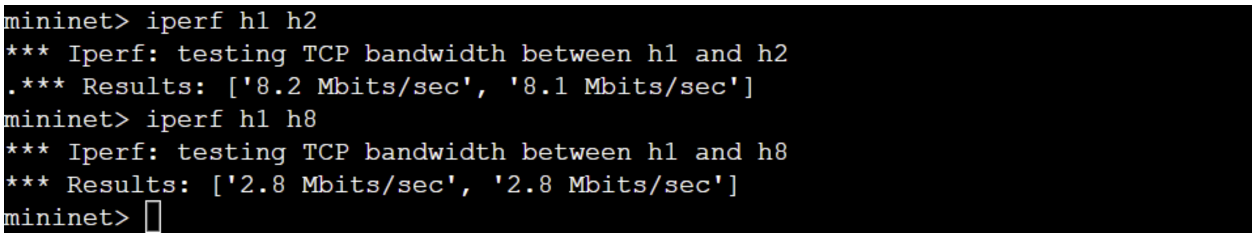
From **s5** to **s7**

Finally to **h8**

This path involves traversing five switches, compared to the single switch hop between **h1** and **h2**. Each additional switch introduces a small delay due to processing time (although typically microseconds in a real network, it's more pronounced in simulated environments). Additionally, each link can introduce its own delay based on the simulated conditions or the underlying performance of the system running the simulation.

This difference in path lengths is likely the reason for the higher latency observed when pinging from **h1** to **h8** compared to pinging from **h1** to **h2**. The increased number of hops results in more processing delay, queueing time, and potential variability in transmission, leading to the higher average, minimum, and maximum latency values observed for **h1** to **h8**.

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



```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
.*** Results: ['8.2 Mbits/sec', '8.1 Mbits/sec']
mininet> iperf h1 h8
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['2.8 Mbits/sec', '2.8 Mbits/sec']
mininet> 
```

a) What is "iperf" used for?

Ans→ iperf is an open-source free program that helps administrators to determine the bandwidth for line quality and network performance. It is used to calculate the amount of data transfer between any two nodes on a network line.

b) What is the throughput for each case?

mininet> iperf h1 h2

i.) *** Iperf: testing TCP bandwidth between h1 and h2

.*** Results: ['8.2 Mbits/sec', '8.1 Mbits/sec']

ii.) mininet> iperf h1 h8

*** Iperf: testing TCP bandwidth between h1 and h8

*** Results: ['2.8 Mbits/sec', '2.8 Mbits/sec']

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

→ The throughput between h1 and h2 exceeds that between h1 and h8 because the data travels a shorter distance with fewer intermediate nodes, resulting in reduced latency and network congestion. Consequently, data packets reach their destination more quickly and efficiently in the h1 to h2 scenario, leading to greater throughput compared to the h1 to h8 connection.

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

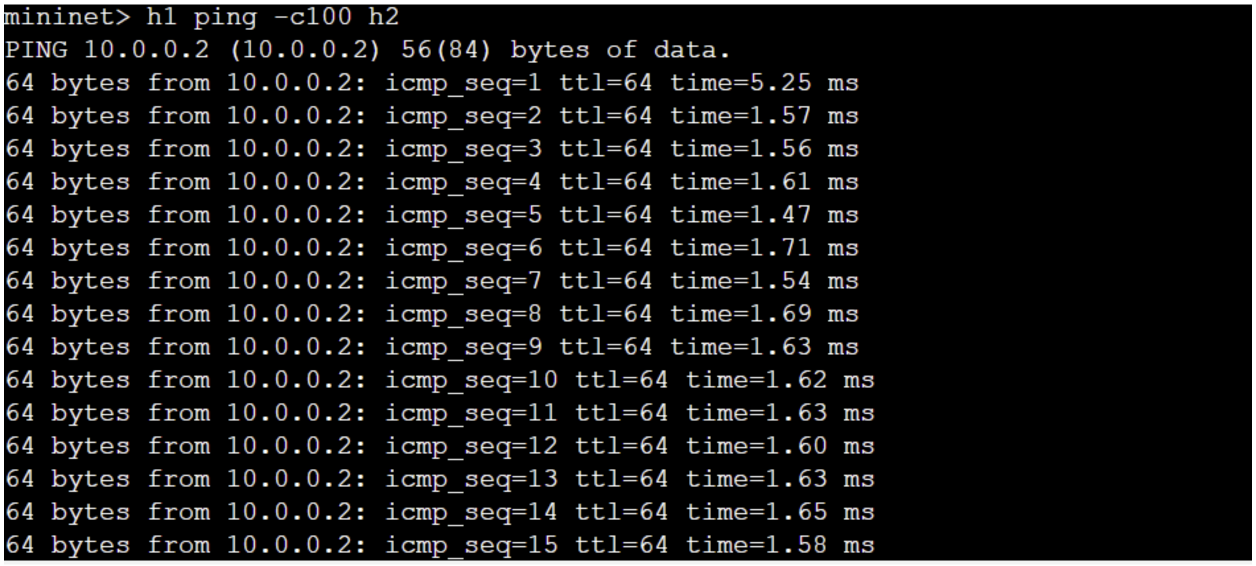
→ We can probe the info that can help us observe the traffic by adding loggers like log.info("Switch observing traffic : percent s" (self.connection)) to line 107 of the "of_tutorial" file of the controller. We can decipher from this that all switches are able to monitor and observe traffic when they are overloaded with packets. The event listener fuction _handle_PacketIn() is called whenever a packet is received.

Task 3 : MAC Learning Controller

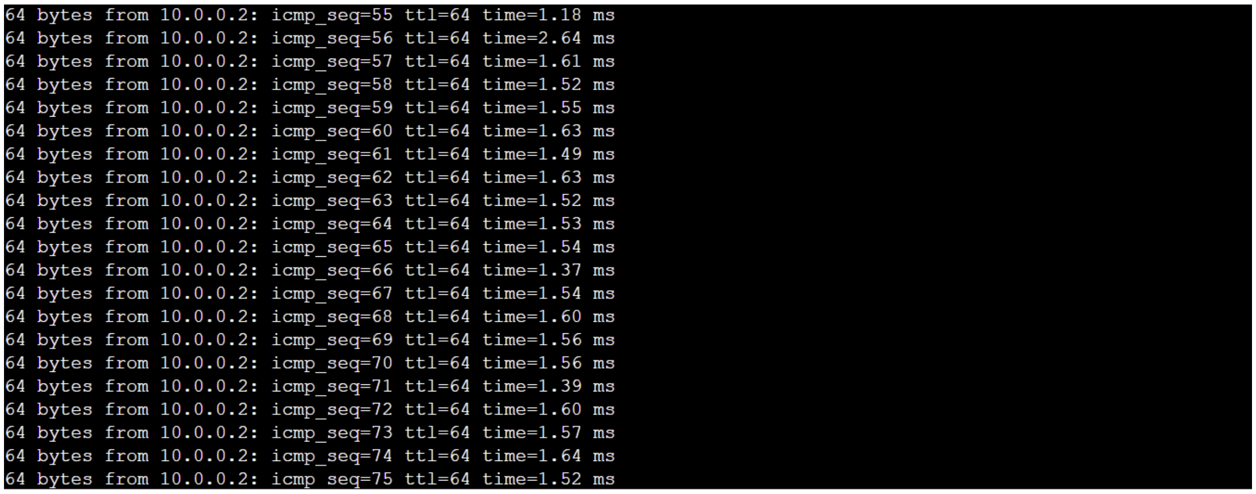
1. Describe how the above code works, such as how the "MAC to Port" map is established. You could use a 'ping' example to describe the establishment process (e.g., h1 ping h2).

Ans→ The `act_like_switch()` function establishes a mapping of MAC addresses to their corresponding ports. When a message needs to be sent, the controller, equipped with this mapping, can quickly identify the appropriate port for a given MAC address. This accelerates the delivery process for messages to known addresses since the destination port is already recognized. If the destination MAC address is not recognized, `act_like_switch()` defaults to broadcasting the packet to all ports. By reducing the likelihood of such broadcast storms, the MAC Learning Controller enhances both the efficiency of ping responses and overall network throughput.

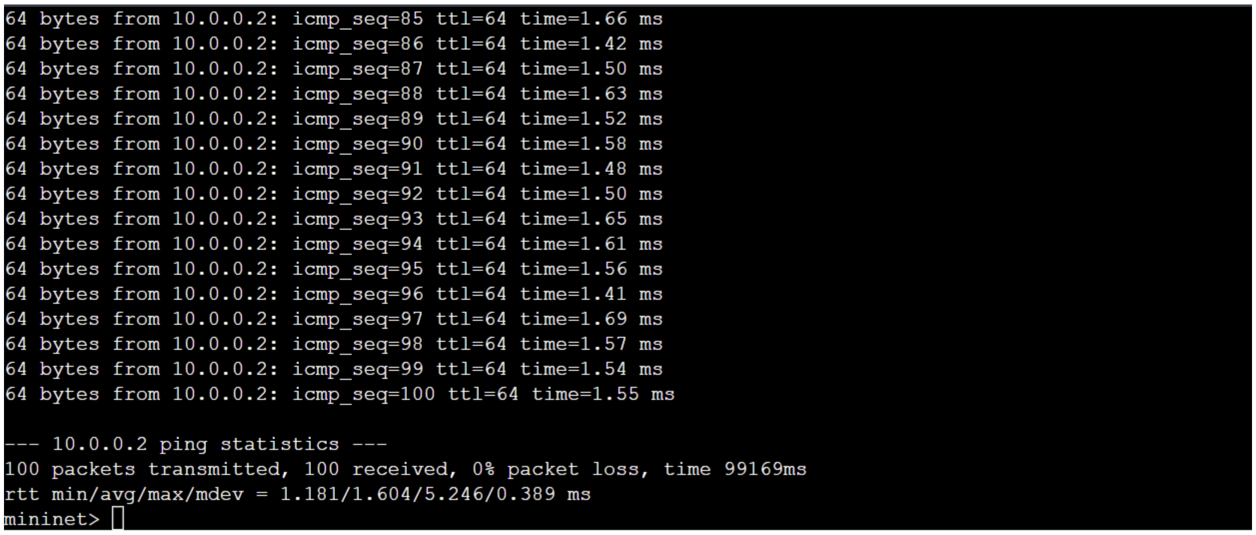
2. h1 ping -c100 h2



```
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=5.25 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=1.57 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=1.56 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=1.61 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=1.47 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=1.71 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=1.54 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=1.69 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=1.63 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=1.62 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=1.63 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=1.60 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=1.63 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=1.65 ms
64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=1.58 ms
```

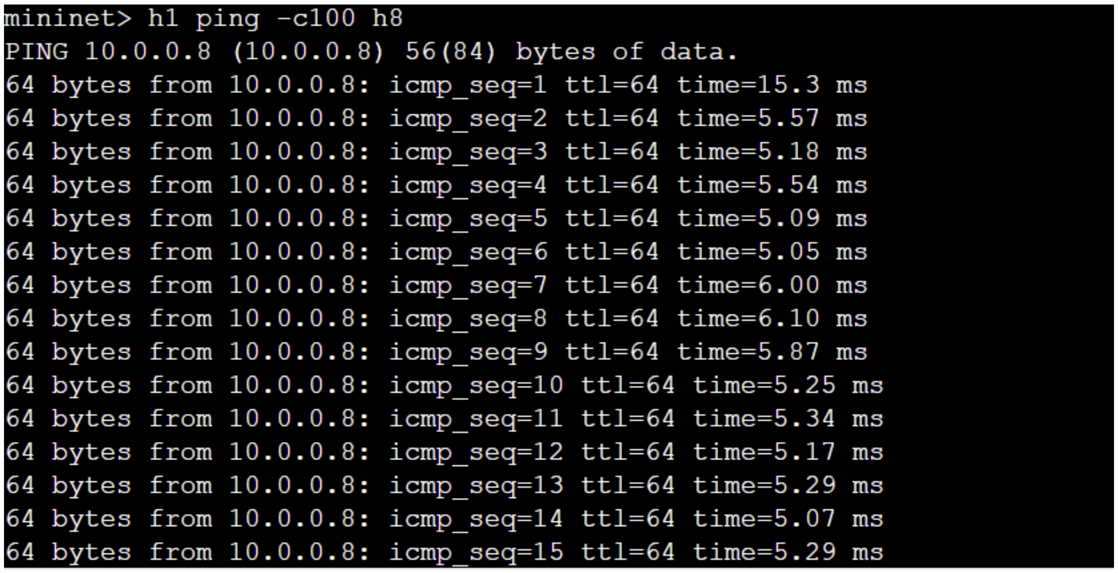


```
64 bytes from 10.0.0.2: icmp_seq=55 ttl=64 time=1.18 ms
64 bytes from 10.0.0.2: icmp_seq=56 ttl=64 time=2.64 ms
64 bytes from 10.0.0.2: icmp_seq=57 ttl=64 time=1.61 ms
64 bytes from 10.0.0.2: icmp_seq=58 ttl=64 time=1.52 ms
64 bytes from 10.0.0.2: icmp_seq=59 ttl=64 time=1.55 ms
64 bytes from 10.0.0.2: icmp_seq=60 ttl=64 time=1.63 ms
64 bytes from 10.0.0.2: icmp_seq=61 ttl=64 time=1.49 ms
64 bytes from 10.0.0.2: icmp_seq=62 ttl=64 time=1.63 ms
64 bytes from 10.0.0.2: icmp_seq=63 ttl=64 time=1.52 ms
64 bytes from 10.0.0.2: icmp_seq=64 ttl=64 time=1.53 ms
64 bytes from 10.0.0.2: icmp_seq=65 ttl=64 time=1.54 ms
64 bytes from 10.0.0.2: icmp_seq=66 ttl=64 time=1.37 ms
64 bytes from 10.0.0.2: icmp_seq=67 ttl=64 time=1.54 ms
64 bytes from 10.0.0.2: icmp_seq=68 ttl=64 time=1.60 ms
64 bytes from 10.0.0.2: icmp_seq=69 ttl=64 time=1.56 ms
64 bytes from 10.0.0.2: icmp_seq=70 ttl=64 time=1.56 ms
64 bytes from 10.0.0.2: icmp_seq=71 ttl=64 time=1.39 ms
64 bytes from 10.0.0.2: icmp_seq=72 ttl=64 time=1.60 ms
64 bytes from 10.0.0.2: icmp_seq=73 ttl=64 time=1.57 ms
64 bytes from 10.0.0.2: icmp_seq=74 ttl=64 time=1.64 ms
64 bytes from 10.0.0.2: icmp_seq=75 ttl=64 time=1.52 ms
```

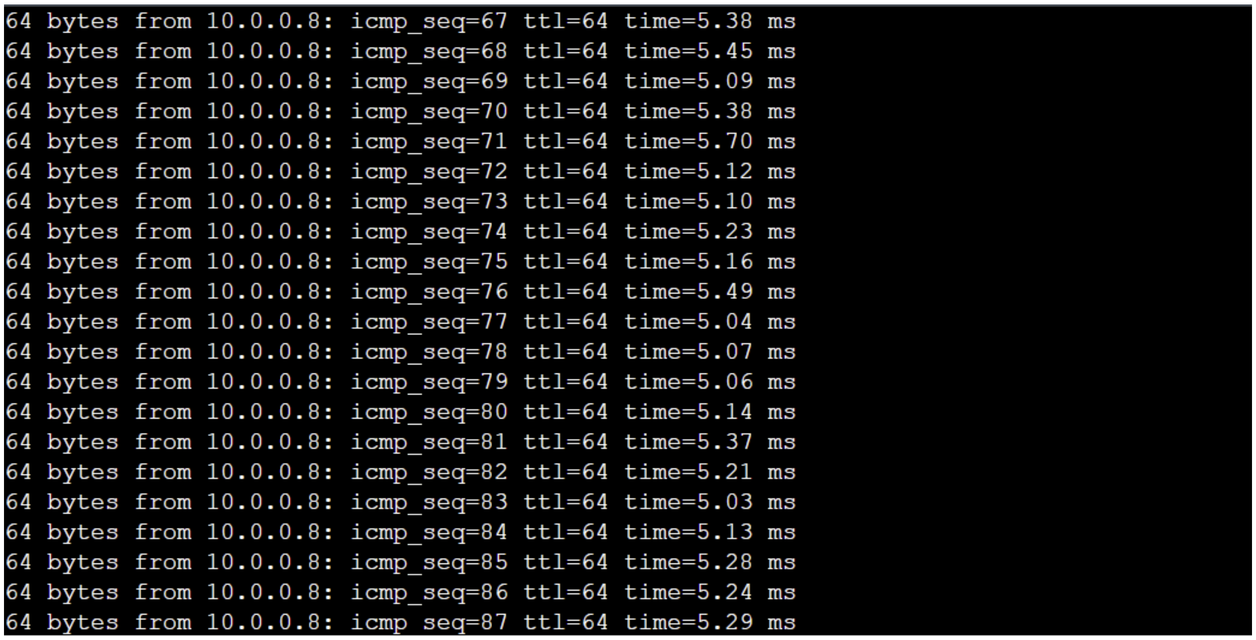


```
64 bytes from 10.0.0.2: icmp_seq=85 ttl=64 time=1.66 ms
64 bytes from 10.0.0.2: icmp_seq=86 ttl=64 time=1.42 ms
64 bytes from 10.0.0.2: icmp_seq=87 ttl=64 time=1.50 ms
64 bytes from 10.0.0.2: icmp_seq=88 ttl=64 time=1.63 ms
64 bytes from 10.0.0.2: icmp_seq=89 ttl=64 time=1.52 ms
64 bytes from 10.0.0.2: icmp_seq=90 ttl=64 time=1.58 ms
64 bytes from 10.0.0.2: icmp_seq=91 ttl=64 time=1.48 ms
64 bytes from 10.0.0.2: icmp_seq=92 ttl=64 time=1.50 ms
64 bytes from 10.0.0.2: icmp_seq=93 ttl=64 time=1.65 ms
64 bytes from 10.0.0.2: icmp_seq=94 ttl=64 time=1.61 ms
64 bytes from 10.0.0.2: icmp_seq=95 ttl=64 time=1.56 ms
64 bytes from 10.0.0.2: icmp_seq=96 ttl=64 time=1.41 ms
64 bytes from 10.0.0.2: icmp_seq=97 ttl=64 time=1.69 ms
64 bytes from 10.0.0.2: icmp_seq=98 ttl=64 time=1.57 ms
64 bytes from 10.0.0.2: icmp_seq=99 ttl=64 time=1.54 ms
64 bytes from 10.0.0.2: icmp_seq=100 ttl=64 time=1.55 ms

--- 10.0.0.2 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99169ms
rtt min/avg/max/mdev = 1.181/1.604/5.246/0.389 ms
mininet> 
```

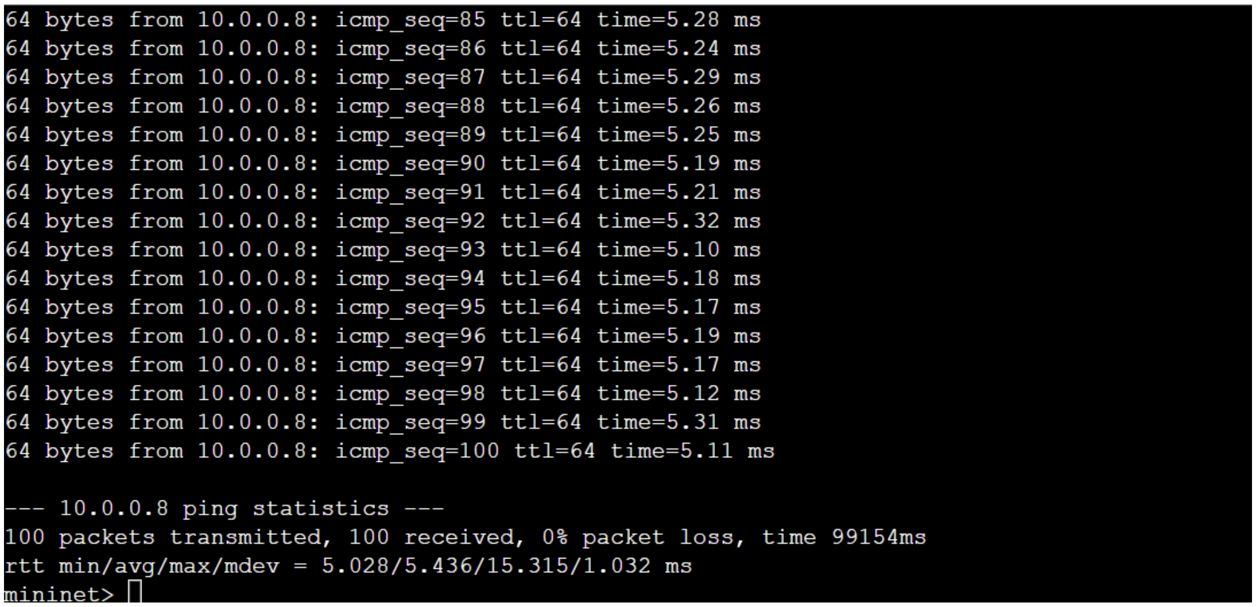
h1 ping -c100 h8



```
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=15.3 ms
64 bytes from 10.0.0.8: icmp_seq=2 ttl=64 time=5.57 ms
64 bytes from 10.0.0.8: icmp_seq=3 ttl=64 time=5.18 ms
64 bytes from 10.0.0.8: icmp_seq=4 ttl=64 time=5.54 ms
64 bytes from 10.0.0.8: icmp_seq=5 ttl=64 time=5.09 ms
64 bytes from 10.0.0.8: icmp_seq=6 ttl=64 time=5.05 ms
64 bytes from 10.0.0.8: icmp_seq=7 ttl=64 time=6.00 ms
64 bytes from 10.0.0.8: icmp_seq=8 ttl=64 time=6.10 ms
64 bytes from 10.0.0.8: icmp_seq=9 ttl=64 time=5.87 ms
64 bytes from 10.0.0.8: icmp_seq=10 ttl=64 time=5.25 ms
64 bytes from 10.0.0.8: icmp_seq=11 ttl=64 time=5.34 ms
64 bytes from 10.0.0.8: icmp_seq=12 ttl=64 time=5.17 ms
64 bytes from 10.0.0.8: icmp_seq=13 ttl=64 time=5.29 ms
64 bytes from 10.0.0.8: icmp_seq=14 ttl=64 time=5.07 ms
64 bytes from 10.0.0.8: icmp_seq=15 ttl=64 time=5.29 ms
```



```
64 bytes from 10.0.0.8: icmp_seq=67 ttl=64 time=5.38 ms
64 bytes from 10.0.0.8: icmp_seq=68 ttl=64 time=5.45 ms
64 bytes from 10.0.0.8: icmp_seq=69 ttl=64 time=5.09 ms
64 bytes from 10.0.0.8: icmp_seq=70 ttl=64 time=5.38 ms
64 bytes from 10.0.0.8: icmp_seq=71 ttl=64 time=5.70 ms
64 bytes from 10.0.0.8: icmp_seq=72 ttl=64 time=5.12 ms
64 bytes from 10.0.0.8: icmp_seq=73 ttl=64 time=5.10 ms
64 bytes from 10.0.0.8: icmp_seq=74 ttl=64 time=5.23 ms
64 bytes from 10.0.0.8: icmp_seq=75 ttl=64 time=5.16 ms
64 bytes from 10.0.0.8: icmp_seq=76 ttl=64 time=5.49 ms
64 bytes from 10.0.0.8: icmp_seq=77 ttl=64 time=5.04 ms
64 bytes from 10.0.0.8: icmp_seq=78 ttl=64 time=5.07 ms
64 bytes from 10.0.0.8: icmp_seq=79 ttl=64 time=5.06 ms
64 bytes from 10.0.0.8: icmp_seq=80 ttl=64 time=5.14 ms
64 bytes from 10.0.0.8: icmp_seq=81 ttl=64 time=5.37 ms
64 bytes from 10.0.0.8: icmp_seq=82 ttl=64 time=5.21 ms
64 bytes from 10.0.0.8: icmp_seq=83 ttl=64 time=5.03 ms
64 bytes from 10.0.0.8: icmp_seq=84 ttl=64 time=5.13 ms
64 bytes from 10.0.0.8: icmp_seq=85 ttl=64 time=5.28 ms
64 bytes from 10.0.0.8: icmp_seq=86 ttl=64 time=5.24 ms
64 bytes from 10.0.0.8: icmp_seq=87 ttl=64 time=5.29 ms
```


```
64 bytes from 10.0.0.8: icmp_seq=85 ttl=64 time=5.28 ms
64 bytes from 10.0.0.8: icmp_seq=86 ttl=64 time=5.24 ms
64 bytes from 10.0.0.8: icmp_seq=87 ttl=64 time=5.29 ms
64 bytes from 10.0.0.8: icmp_seq=88 ttl=64 time=5.26 ms
64 bytes from 10.0.0.8: icmp_seq=89 ttl=64 time=5.25 ms
64 bytes from 10.0.0.8: icmp_seq=90 ttl=64 time=5.19 ms
64 bytes from 10.0.0.8: icmp_seq=91 ttl=64 time=5.21 ms
64 bytes from 10.0.0.8: icmp_seq=92 ttl=64 time=5.32 ms
64 bytes from 10.0.0.8: icmp_seq=93 ttl=64 time=5.10 ms
64 bytes from 10.0.0.8: icmp_seq=94 ttl=64 time=5.18 ms
64 bytes from 10.0.0.8: icmp_seq=95 ttl=64 time=5.17 ms
64 bytes from 10.0.0.8: icmp_seq=96 ttl=64 time=5.19 ms
64 bytes from 10.0.0.8: icmp_seq=97 ttl=64 time=5.17 ms
64 bytes from 10.0.0.8: icmp_seq=98 ttl=64 time=5.12 ms
64 bytes from 10.0.0.8: icmp_seq=99 ttl=64 time=5.31 ms
64 bytes from 10.0.0.8: icmp_seq=100 ttl=64 time=5.11 ms

--- 10.0.0.8 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99154ms
rtt min/avg/max/mdev = 5.028/5.436/15.315/1.032 ms
mininet> 
```

a) How long does it take (on average) to ping for each case?

	Average Ping
h1 ping -c100 h2	1.6ms
h1 ping -c100 h8	5.4ms

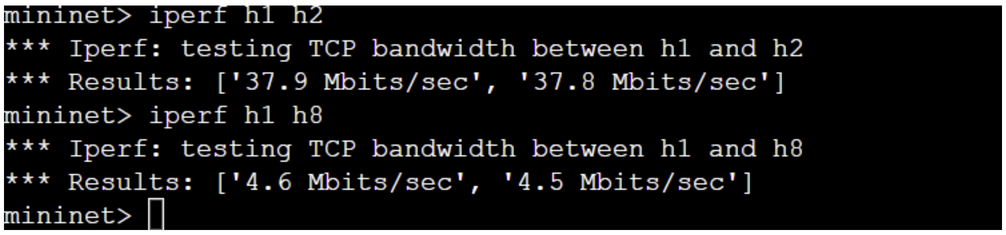
b)What is the minimum and maximum ping you have observed?

	Minimum Ping	Maximum Ping
h1 ping -c100 h2	1.2ms	5.2ms
h1 ping -c100 h8	5.08ms	15.315ms

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

Ans→ The duration required to execute operations in Task 3 is shorter than in Task 2, albeit the discrepancy is marginal. The variance in ping times can be attributed to the switches in Task 3 utilizing the "mac to port" mapping to forward packets. After an address's initial encounter is logged in the mapping, subsequent packets destined for that address are processed more swiftly due to the pre-established knowledge of the corresponding port and reduced network congestion.

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



```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['37.9 Mbits/sec', '37.8 Mbits/sec']
mininet> iperf h1 h8
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['4.6 Mbits/sec', '4.5 Mbits/sec']
mininet> 
```

a) What is the throughput for each case?

```
mininet> iperf h1 h2

*** Iperf: testing TCP bandwidth between h1 and h2

*** Results: ['37.9 Mbits/sec', '37.8 Mbits/sec']

mininet> iperf h1 h8

*** Iperf: testing TCP bandwidth between h1 and h8

*** Results: ['4.6 Mbits/sec', '4.5 Mbits/sec']
```

b) What is the difference and explain the reasons for the difference.

Ans→ Task 3 exhibits an improvement in throughput compared to Task 2 across both scenarios, largely due to the efficiency introduced by the "mac to port" mapping, which decreases network congestion. Furthermore, because the switches have pre-learned the addresses and their corresponding ports from initial transmissions, no packet flooding is necessary on subsequent communications. This prevents the switches from becoming overloaded with excessive forwarding requests. For transmissions from h1 to h2, Task 3 shows a notable increase in performance relative to Task 2, which can be attributed to the alleviated congestion. In contrast, the enhancement from h1 to h8 is less pronounced, potentially due to packet loss and the greater distance packets must travel.