CSEN 241 – CLOUD COMPUTING

HOMEWORK 3

NAME: ADITYA SHRIVASTAVA

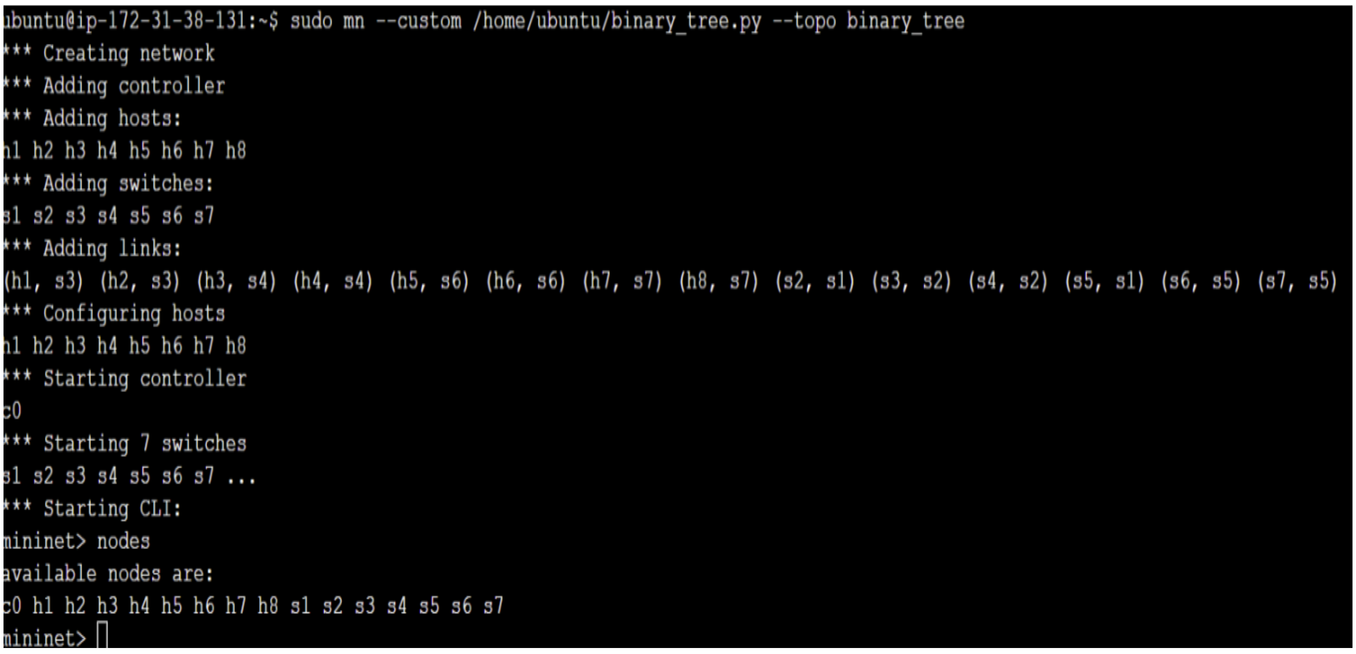
SCU ID: W1648524

Github Link for HW:
https://github.com/AdityaShrivastava30/CSEN241_Cloud_Computing/tree/master/Homeworks/HW3

SOLUTION TASK 1: Defining custom topologies:

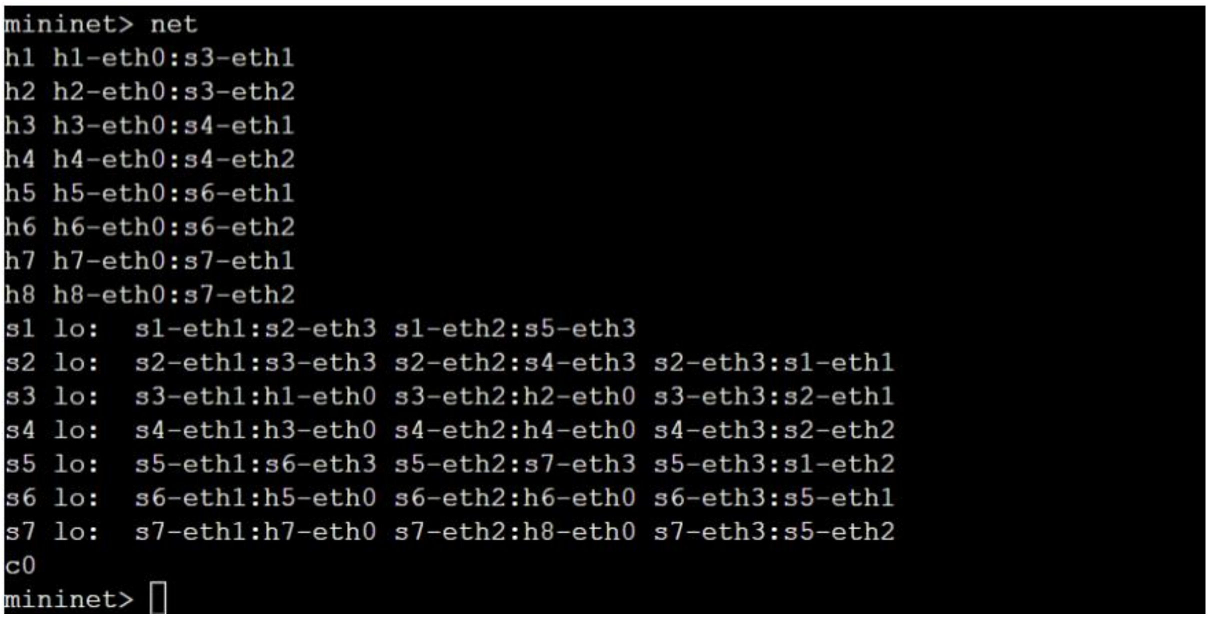
Q1. What is the output of “nodes” and “net” ?

a. Nodes



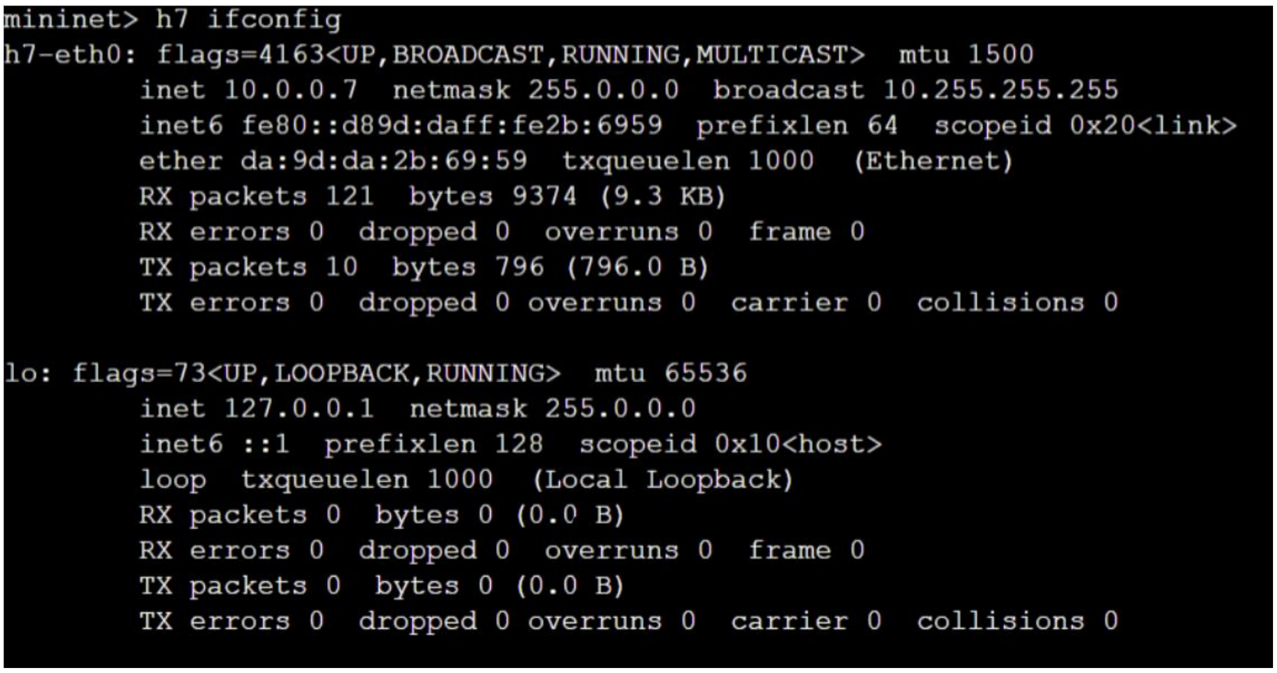
```
ubuntu@ip-172-31-38-131:~$ sudo mn --custom /home/ubuntu/binary_tree.py --topo binary_tree
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7
*** Adding links:
(h1, s3) (h2, s3) (h3, s4) (h4, s4) (h5, s6) (h6, s6) (h7, s7)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

Q2. What is the output of "h7 ifconfig" ?



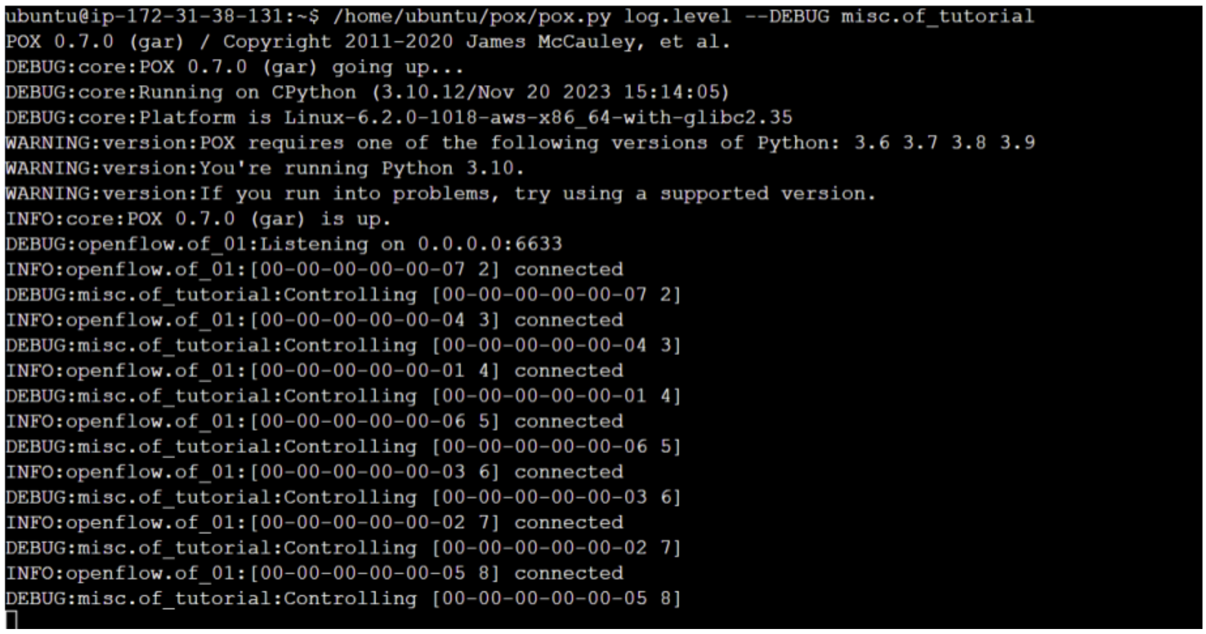
```
mininet> h7 ifconfig
h7-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.7  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::d89d:daff:fe2b:6959  prefixlen 64  scopeid 0x20<link>
        ether da:9d:da:2b:69:59  txqueuelen 1000  (Ethernet)
        RX packets 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
```

SOLUTION TASK 2: Analyze the "of_tutorial" controller

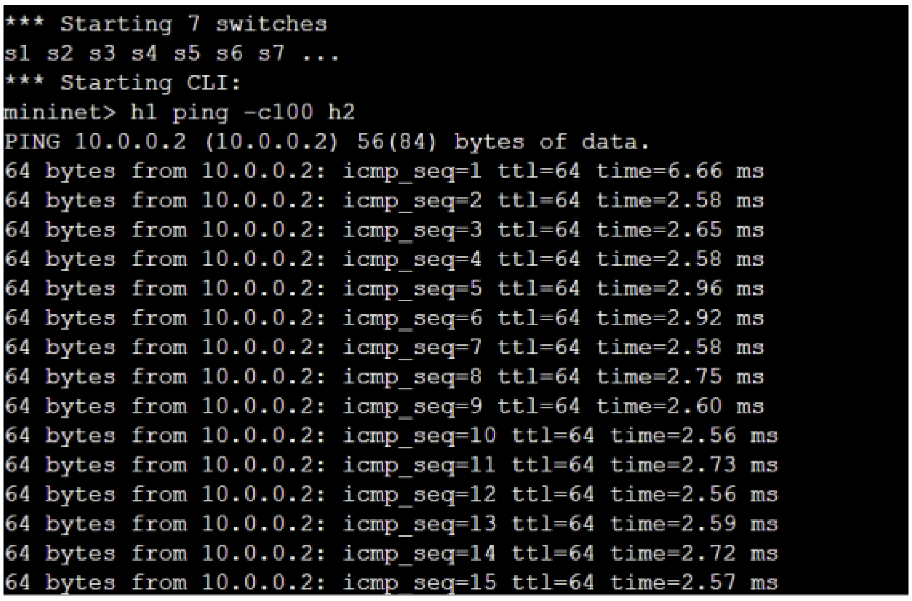
Q1. Draw the function call graph of this controller. For example, once a packet comes to the controller, which function is the first to be called, which one is the second, and so forth?

Ans: To start the procedure, run `./pox.py log.level --DEBUG misc.of_tutorial to start the POX listener. By running this command, the switch's `_handle_PacketIn()} function is triggered, activating the'start switch'. In turn, this technique calls the `act_like_hub()} function. The `act_like_hub()` function forwards packets to all ports other than the one they were received on in order to simulate the behavior of a hub. The `resend_packet()` method is then used, appending a packet to the message payload and performing an action on it. The switch is instructed to route the packet to a specified port by means of this action. The controller's function calls go like this: start switch -> _handle_PacketIn() -> act_like_hub() -> resend_packet() -> send(message).

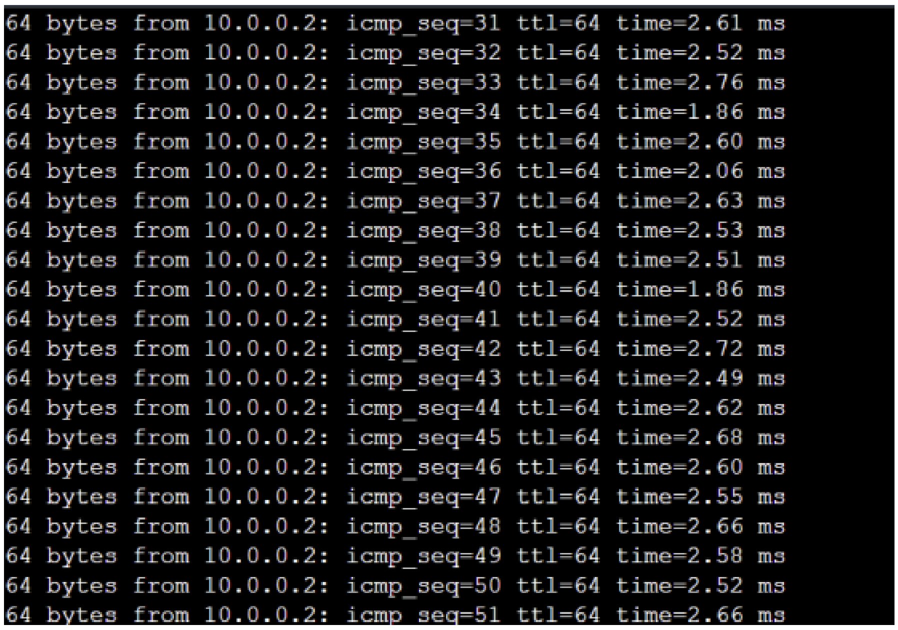


```
ubuntu@ip-172-31-38-131:~$ /home/ubuntu/pox/pox.py log.level --DEBUG misc.of_tutorial
POX 0.7.0 (gar) / Copyright 2011-2020 James McCauley, et al.
DEBUG:core:POX 0.7.0 (gar) going up...
DEBUG:core:Running on CPython (3.10.12/Nov 20 2023 15:14:05)
DEBUG:core:Platform is Linux-6.2.0-1018-aws-x86_64-with-glibc2.35
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]
```

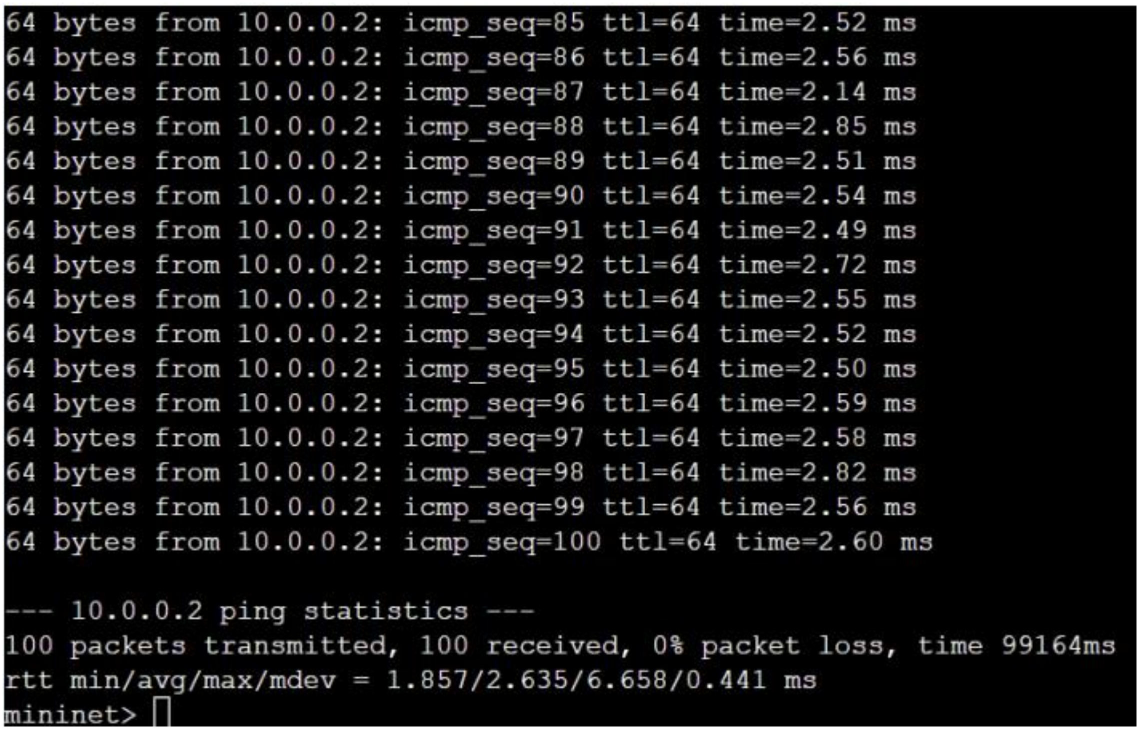
Q2. **h1 ping -c100 h2**



```
*** 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
```



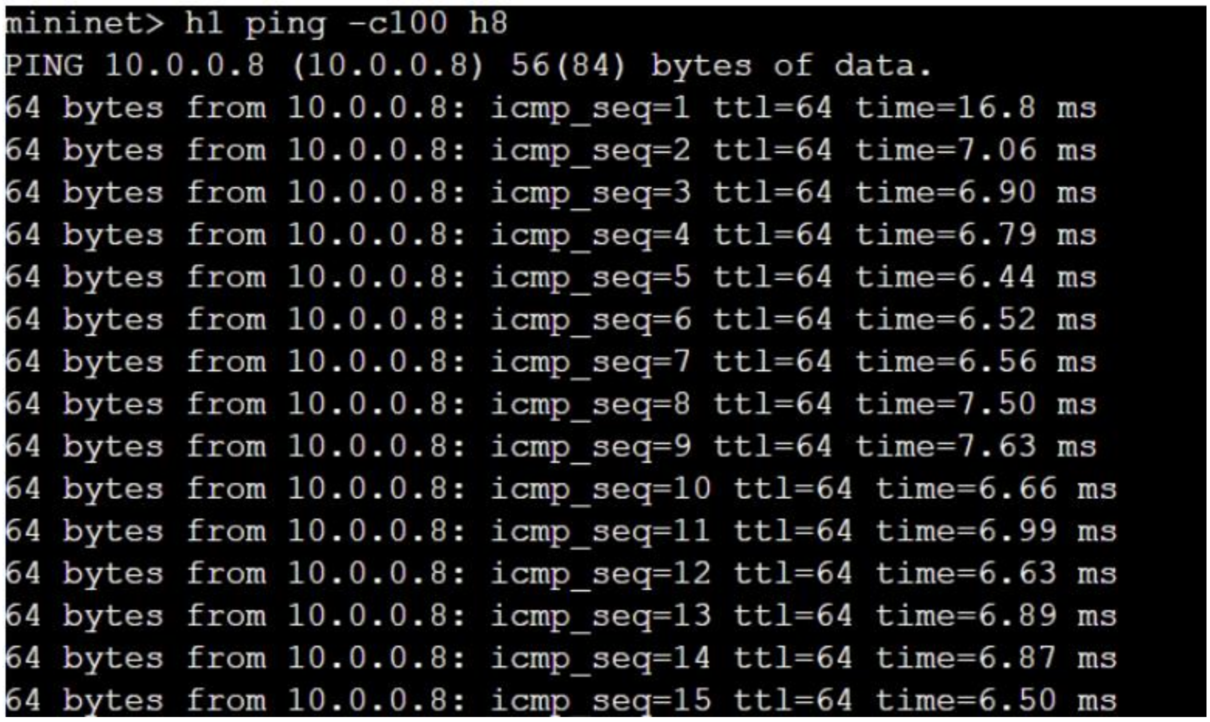
```
64 bytes from 10.0.0.2: icmp_seq=31 ttl=64 time=2.61 ms
64 bytes from 10.0.0.2: icmp_seq=32 ttl=64 time=2.52 ms
64 bytes from 10.0.0.2: icmp_seq=33 ttl=64 time=2.76 ms
64 bytes from 10.0.0.2: icmp_seq=34 ttl=64 time=1.86 ms
64 bytes from 10.0.0.2: icmp_seq=35 ttl=64 time=2.60 ms
64 bytes from 10.0.0.2: 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
```



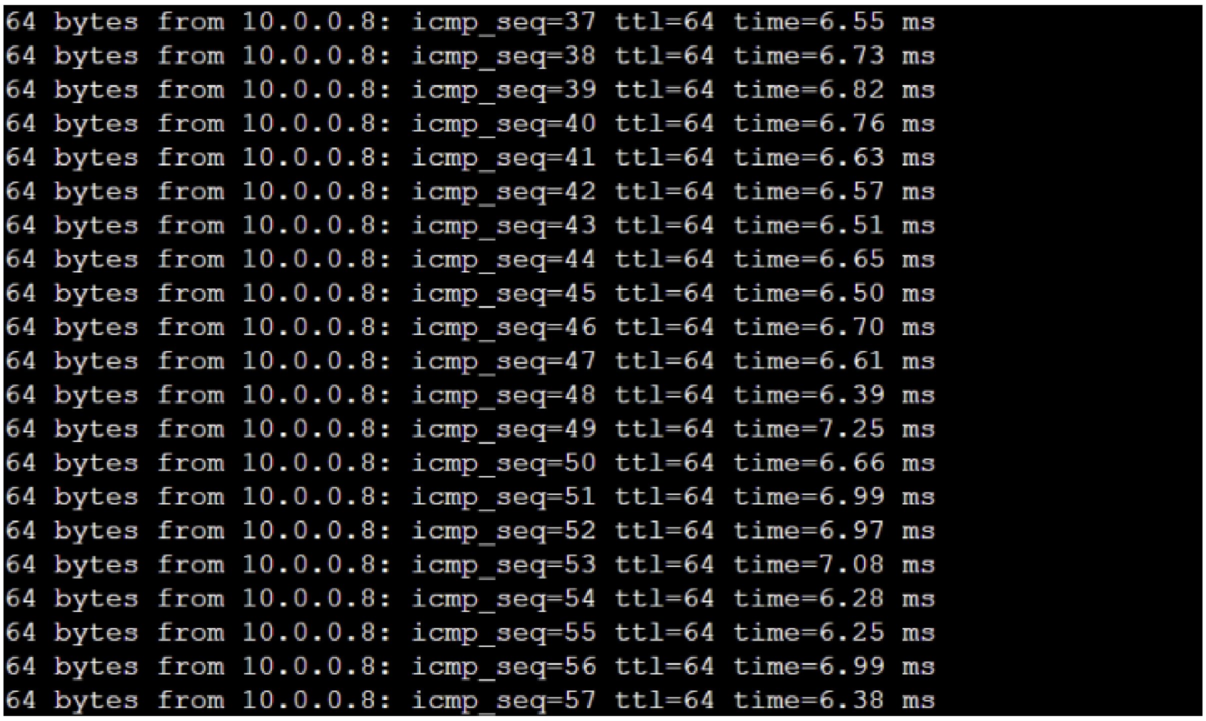
```
64 bytes from 10.0.0.2: icmp_seq=85 ttl=64 time=2.52 ms
64 bytes from 10.0.0.2: icmp_seq=86 ttl=64 time=2.56 ms
64 bytes from 10.0.0.2: icmp_seq=87 ttl=64 time=2.14 ms
64 bytes from 10.0.0.2: icmp_seq=88 ttl=64 time=2.85 ms
64 bytes from 10.0.0.2: icmp_seq=89 ttl=64 time=2.51 ms
64 bytes from 10.0.0.2: 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> 
```

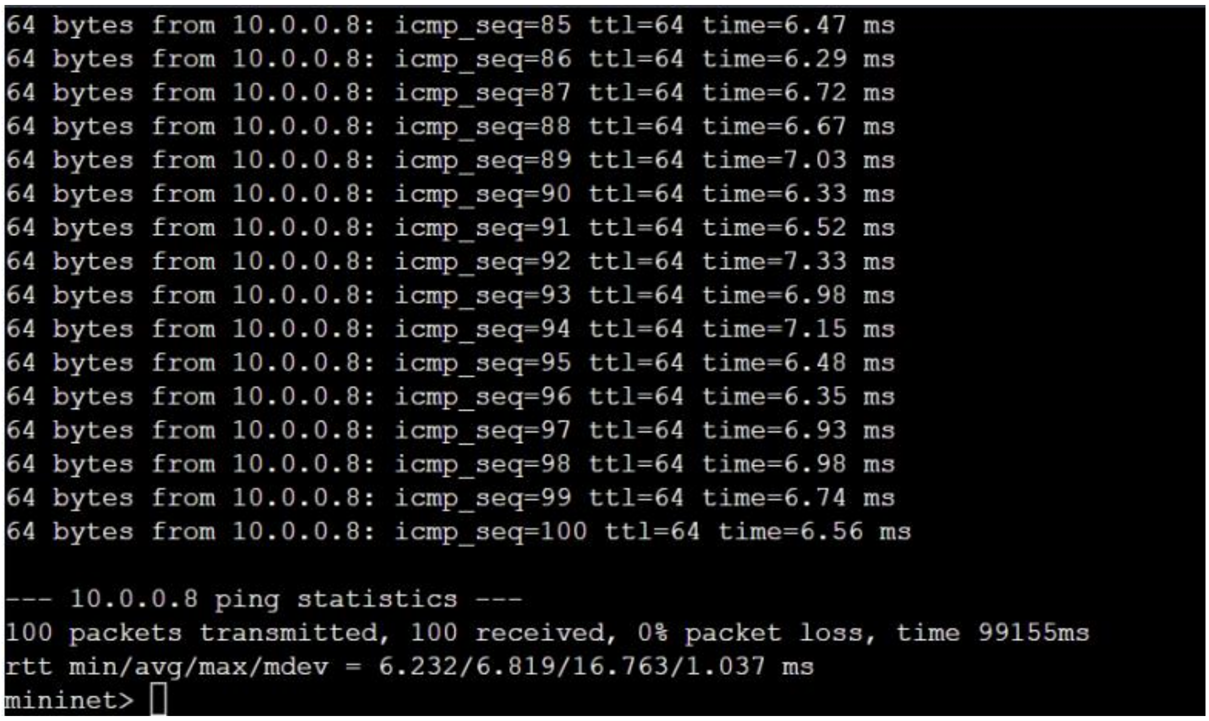
h1 ping -c100 h8



```
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?

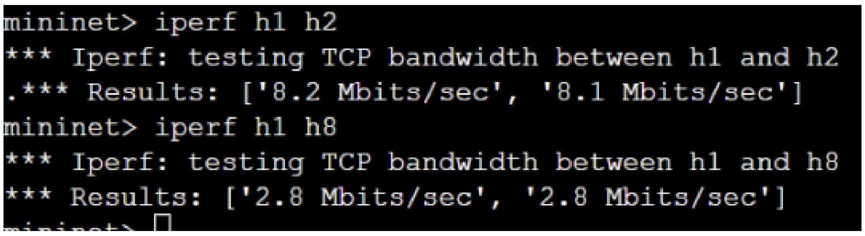
Ans c) Host h1 is linked to switch s3, which is linked to s2, and then to switch s1, the root switch. Host h2 is linked to switch s3 as well and takes the same route as h1 up to s1.
Host h8 is linked to switch s7, which is linked to switch s5, and lastly, to switch s1, the root switch. The packets travel from h1 to s3 and then straight to h2 when h1 pings h2, requiring only one switch hop. Nevertheless, the packets take a longer route when h1 pings h8:
From h1 to s3

Moving from s3 to s2
The root switch is made from s2 to s1.

From s1 to s5
S5 to S7
And lastly, h8

Whereas the single switch hop between h1 and h2 only requires transiting five switches on this path. Processing time causes a slight delay with each new switch; in a real network, this delay is usually microseconds, but in simulated systems, it is more noticeable. Furthermore, depending on the simulated conditions or the underlying performance of the system that is doing the simulation, each link may impose its own latency.
The increased latency seen while pinging from h1 to h8 as opposed to h1 to h2 is probably caused by this difference in path lengths. For h1 to h8, the greater average, minimum, and maximum latency values are caused by extra processing delay, queueing time, and possible transmission fluctuation as a result of the increasing number of hops.

Q3: 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']
```

a) What is “iperf” used for?

Ans) An open-source, free tool called iperf assists network managers in figuring out bandwidth requirements for line quality and overall network performance. It is employed to determine how much data is transferred over a network line between any two nodes.

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.

Ans c) Because the data travels a shorter distance with fewer intermediate nodes, resulting in lower delay and network congestion, the throughput between h1 and h2 exceeds that between h1 and h8. As a result, in the h1 to h2 scenario, data packets arrive at their destination faster and more effectively, resulting in higher throughput than in the h1 to h8 link.

Q4: 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).

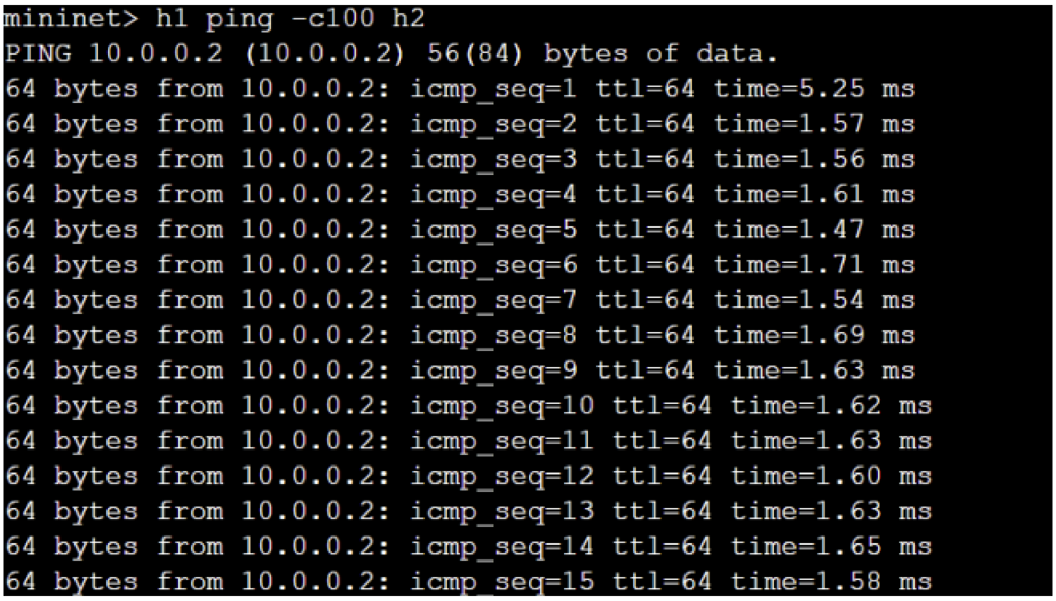
Ans -> By adding loggers such as log.info ("Switch observing traffic: percent s" (self.connection)) to line 107 of the controller's "of_tutorial" file, we can probe the information that can assist us observe the traffic. This indicates to us that even when switches are overburdened with packets, they can still monitor and observe traffic. Every time a packet is received, the event listener function _handle_PacketIn() is invoked.

SOLUTION TASK 3: MAC Learning Controller

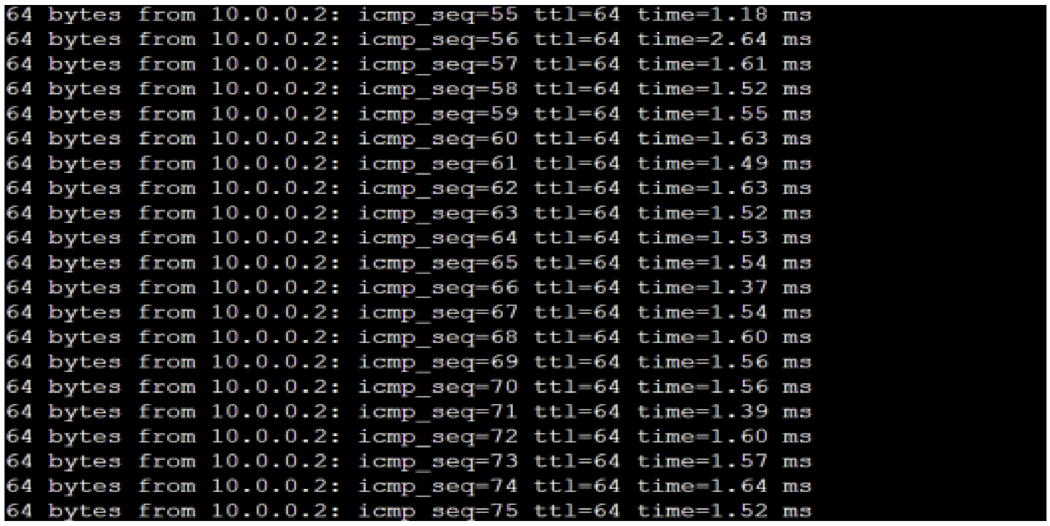
Q1: Describe how the above code works, such as how the "MAC to Port" map is established. You could use a 'ping' example to describe the establishment process (e.g., h1 ping h2).

Ans) The `act_like_switch()} method creates a mapping between MAC addresses and ports. With this mapping, the controller can rapidly determine which port is appropriate for a given MAC address when a message needs to be sent. Due to the destination port's prior recognition, this expedites the transmission of messages to known addresses. Broadcasting the packet to all ports is the default behavior of `act_like_switch()} in case the destination MAC address is not recognized. Throughput on the network as a whole and ping response efficiency are both improved by the MAC Learning Controller by decreasing the probability of these broadcast storms.

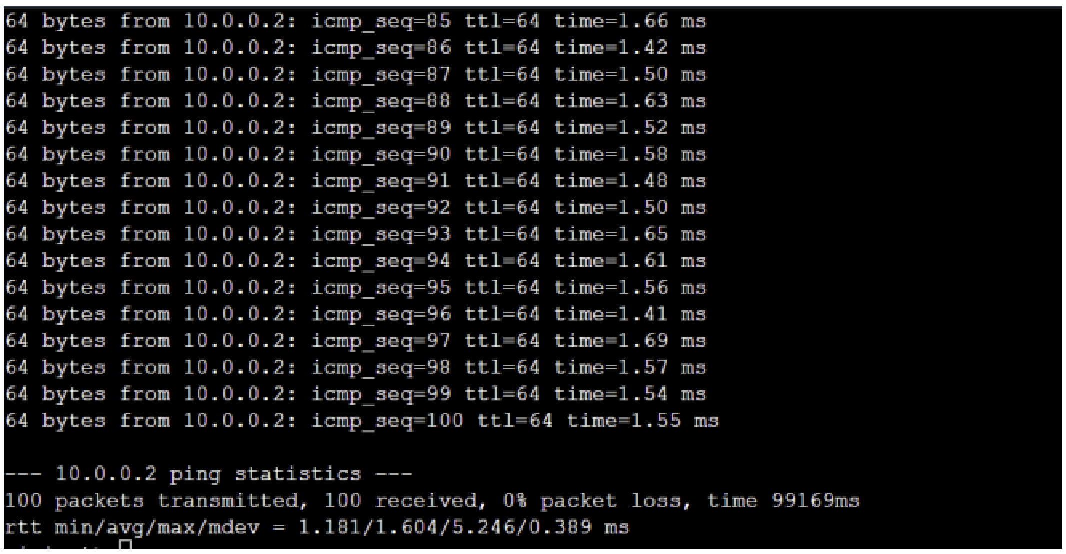
Q2: h1 ping -c100 h2



```
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
```



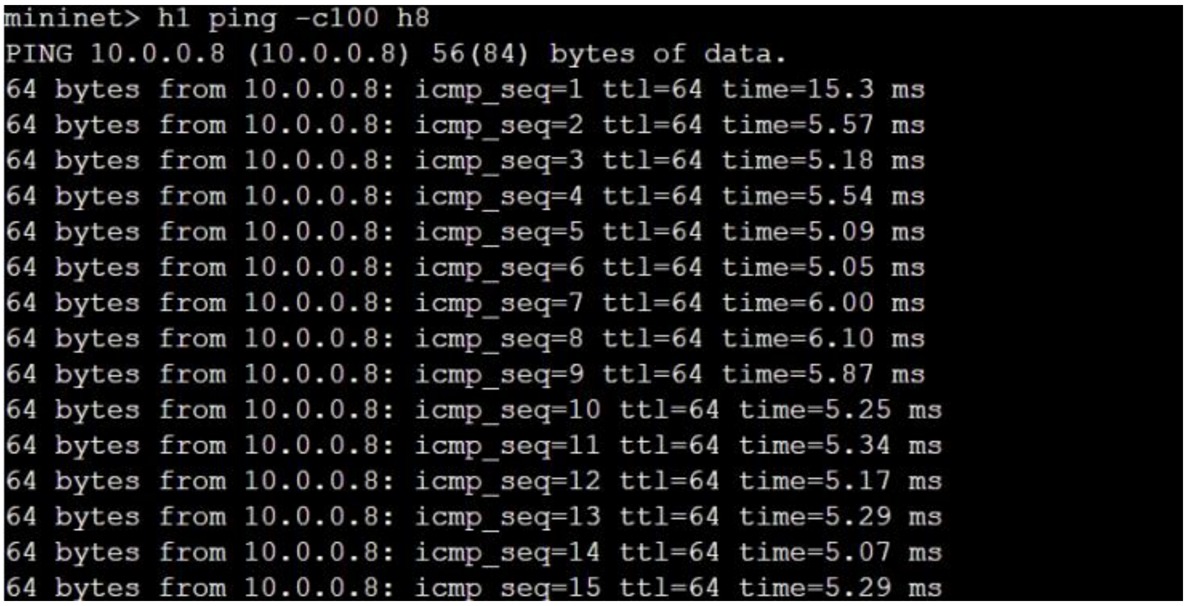
```
64 bytes from 10.0.0.2: icmp_seq=55 ttl=64 time=1.18 ms
64 bytes from 10.0.0.2: icmp_seq=56 ttl=64 time=2.64 ms
64 bytes from 10.0.0.2: icmp_seq=57 ttl=64 time=1.61 ms
64 bytes from 10.0.0.2: icmp_seq=58 ttl=64 time=1.52 ms
64 bytes from 10.0.0.2: icmp_seq=59 ttl=64 time=1.55 ms
64 bytes from 10.0.0.2: 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
```



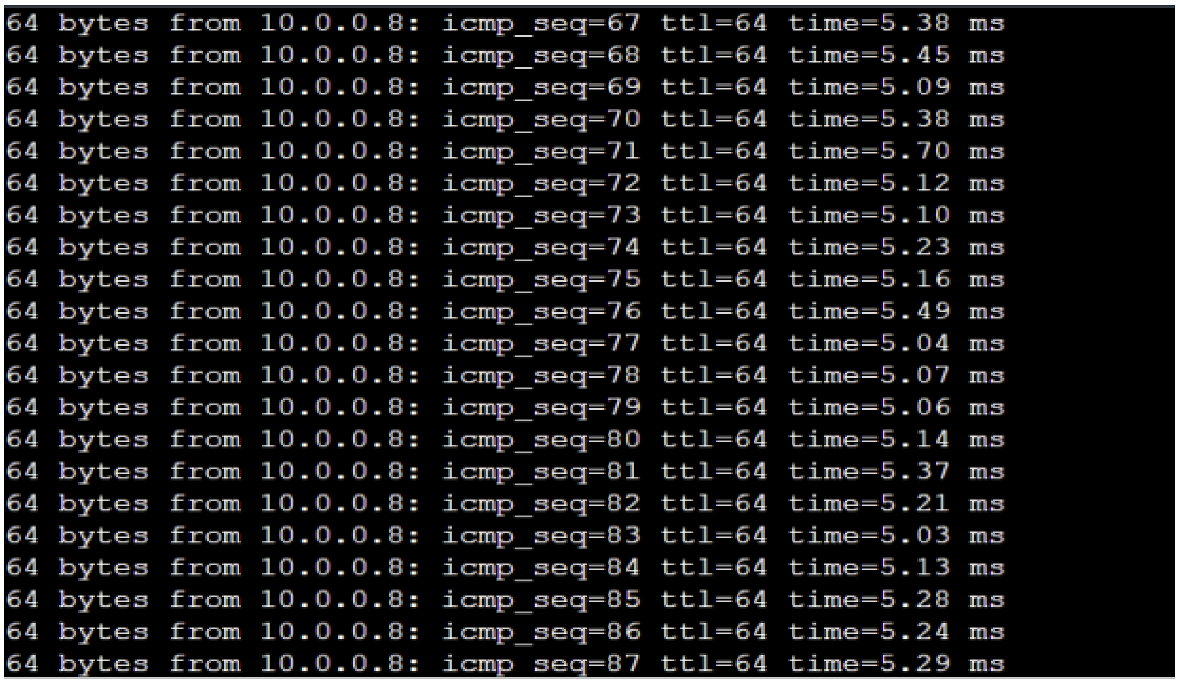
```
64 bytes from 10.0.0.2: icmp_seq=85 ttl=64 time=1.66 ms
64 bytes from 10.0.0.2: icmp_seq=86 ttl=64 time=1.42 ms
64 bytes from 10.0.0.2: icmp_seq=87 ttl=64 time=1.50 ms
64 bytes from 10.0.0.2: icmp_seq=88 ttl=64 time=1.63 ms
64 bytes from 10.0.0.2: icmp_seq=89 ttl=64 time=1.52 ms
64 bytes from 10.0.0.2: 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
```

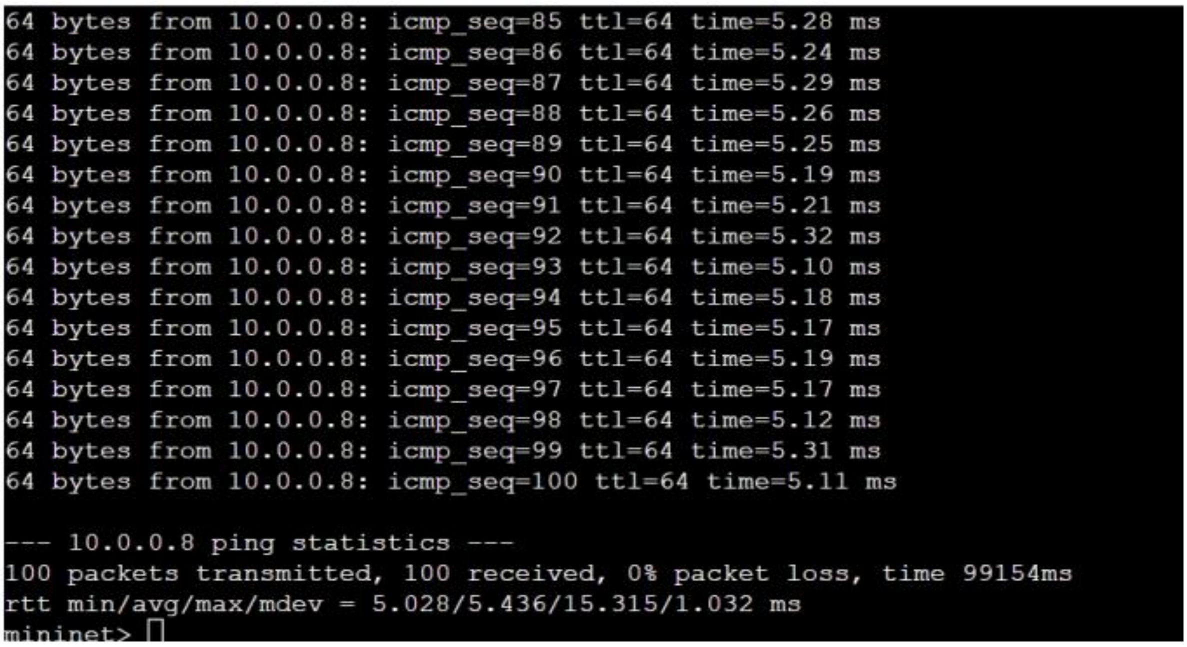
h1 ping -c100 h8



```
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

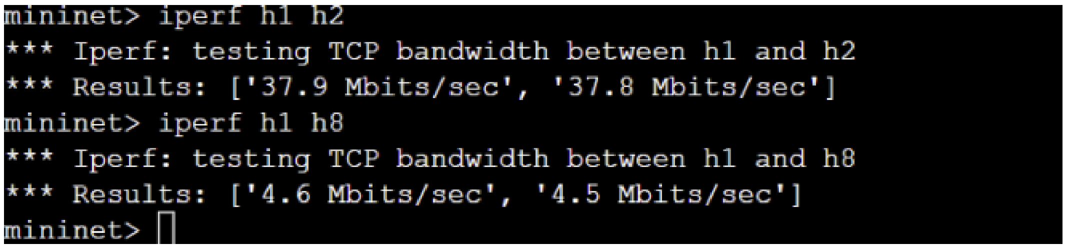
b) What is the minimum and maximum ping you have observed?

	Minimum Ping	Maximum Ping
h1 ping -c 100 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) Even if the difference is small, Task 3 takes less time to complete operations than Task 2. The switches in Task 3 are using the "mac to port" mapping to forward packets, which explains the variation in ping times. Following the logging of an address's first encounter in the mapping, packets intended for that address are processed faster because the matching port is already known, resulting in less network congestion.

Q3: 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) In both cases, Task 3's throughput is higher than Task 2's, mostly as a result of the "mac to port" mapping's increased efficiency and reduced network congestion. Furthermore, on subsequent contacts, there is no need for packet flooding because the switches already know the addresses and the ports that correspond to them from the first broadcasts. By doing this, too many forwarding requests are kept from overloading the switches. Task 3 performs noticeably better than Task 2 for transmissions from h1 to h2, which is likely due to the reduced congestion. The improvement from h1 to h8, however, is less noticeable. This could be because of packet loss and the longer distance that packets need to travel.