Sourabh Deshmukh
W1648445

Homework Assignment 3

Task 1 QnA

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

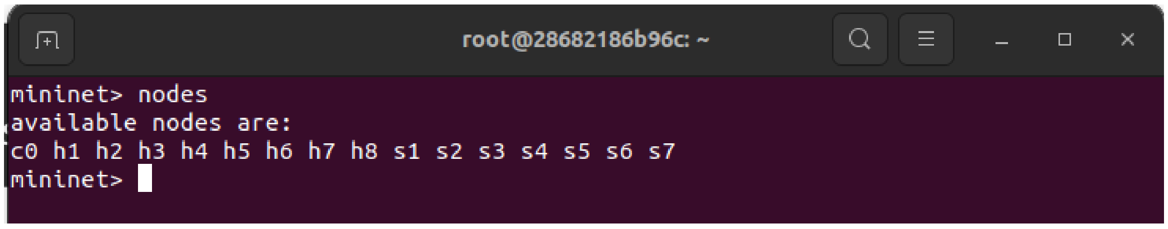
Output of command nodes

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

Output of command net

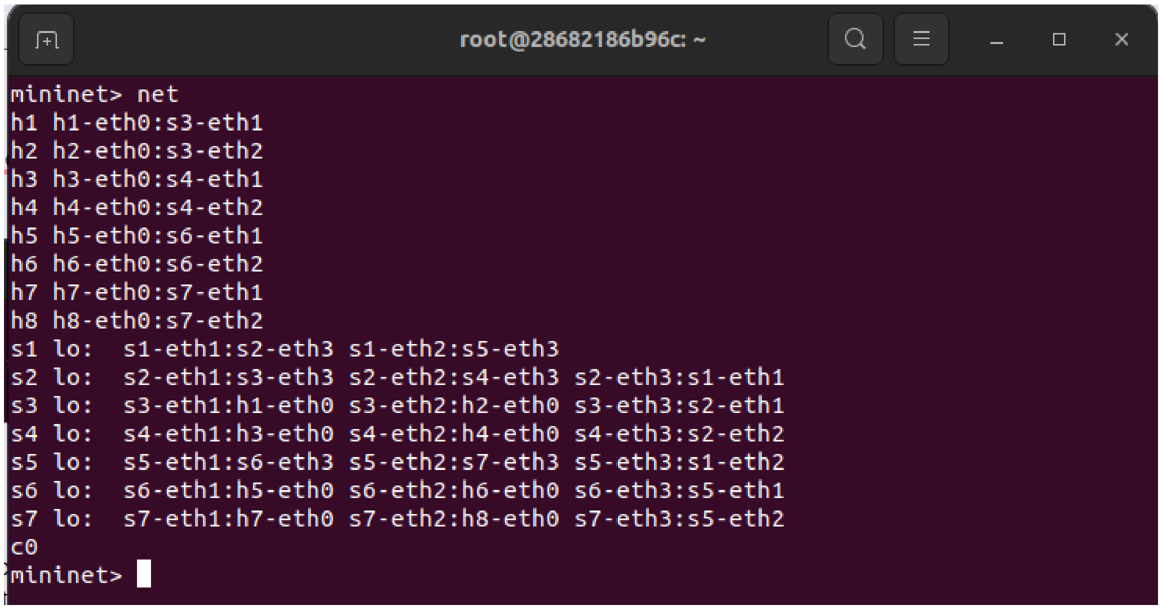
```
mininet> net
h1 h1-eth0:s3-eth1
h2 h2-eth0:s3-eth2
h3 h3-eth0:s4-eth1
h4 h4-eth0:s4-eth2
h5 h5-eth0:s6-eth1
h6 h6-eth0:s6-eth2
h7 h7-eth0:s7-eth1
h8 h8-eth0:s7-eth2
s1 lo:  s1-eth1:s2-eth3 s1-eth2:s5-eth3
s2 lo:  s2-eth1:s3-eth3 s2-eth2:s4-eth3 s2-eth3:s1-eth1
s3 lo:  s3-eth1:h1-eth0 s3-eth2:h2-eth0 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
```

Screenshot of command nodes



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

Screenshot of command net

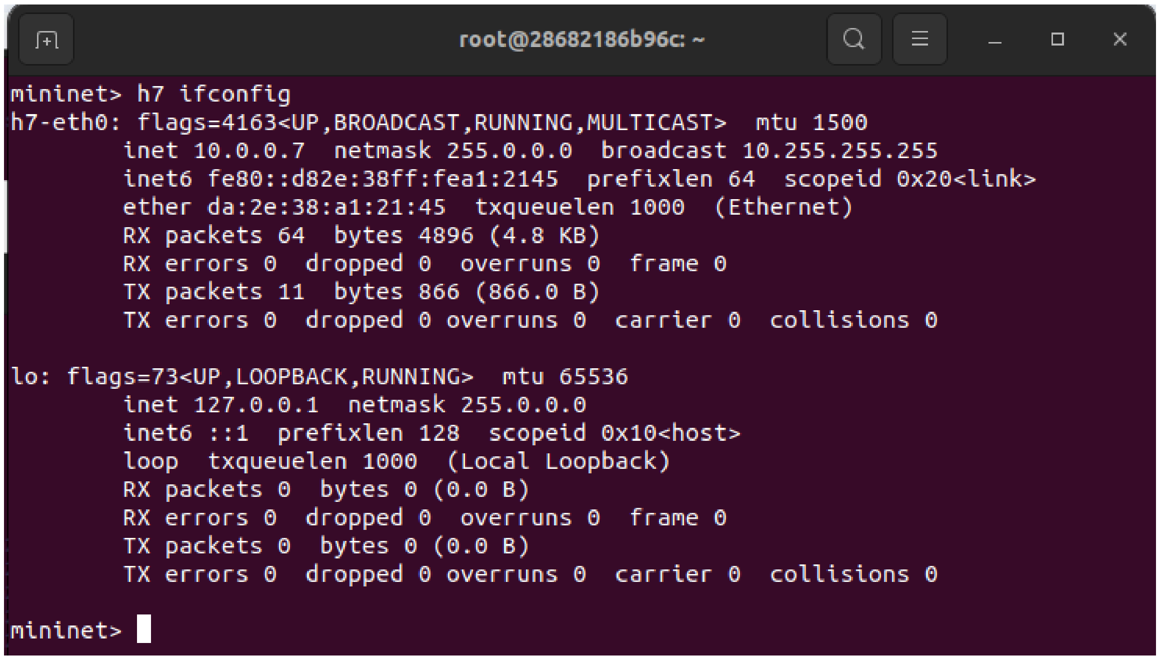


```
mininet> net
h1 h1-eth0:s3-eth1
h2 h2-eth0:s3-eth2
h3 h3-eth0:s4-eth1
h4 h4-eth0:s4-eth2
h5 h5-eth0:s6-eth1
h6 h6-eth0:s6-eth2
h7 h7-eth0:s7-eth1
h8 h8-eth0:s7-eth2
s1 lo:  s1-eth1:s2-eth3 s1-eth2:s5-eth3
s2 lo:  s2-eth1:s3-eth3 s2-eth2:s4-eth3 s2-eth3:s1-eth1
s3 lo:  s3-eth1:h1-eth0 s3-eth2:h2-eth0 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”
   Output of command 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::d82e:38ff:fea1:2145  prefixlen 64  scopeid
0x20<link>
        ether da:2e:38:a1:21:45  txqueuelen 1000  (Ethernet)
        RX packets 64  bytes 4896 (4.8 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
```

Screenshot of command h7 ifconfig



root@28682186b96c: ~

```
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::d82e:38ff:fea1:2145  prefixlen 64  scopeid 0x20<link>
        ether da:2e:38:a1:21:45  txqueuelen 1000  (Ethernet)
        RX packets 64  bytes 4896 (4.8 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

mininet>
```

Task 2 QnA

3. 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?

```
To begin, launch the POX listener by executing the command ./pox.py log.level --DEBUG misc.of_tutorial, which activates the start switch. Following this initialization, the _handle_PacketIn() method is invoked by the start switch to manage incoming packet messages from the switch. Subsequently, the _handle_PacketIn() function triggers the act_like_hub() function. The act_like_hub() function is designed to mimic a hub environment by broadcasting packets to all ports except the incoming port. Afterward, the resend_packet() method is invoked. This method appends a packet to the message data and executes an action on it. It instructs the switch to retransmit the packet to a specified port via this message. The sequence of operations is depicted as follows
start switch: _handle_PacketIn() → act_like_hub() → resend_packet() → send(msg)
```

Sourabh Deshmukh
W1648445

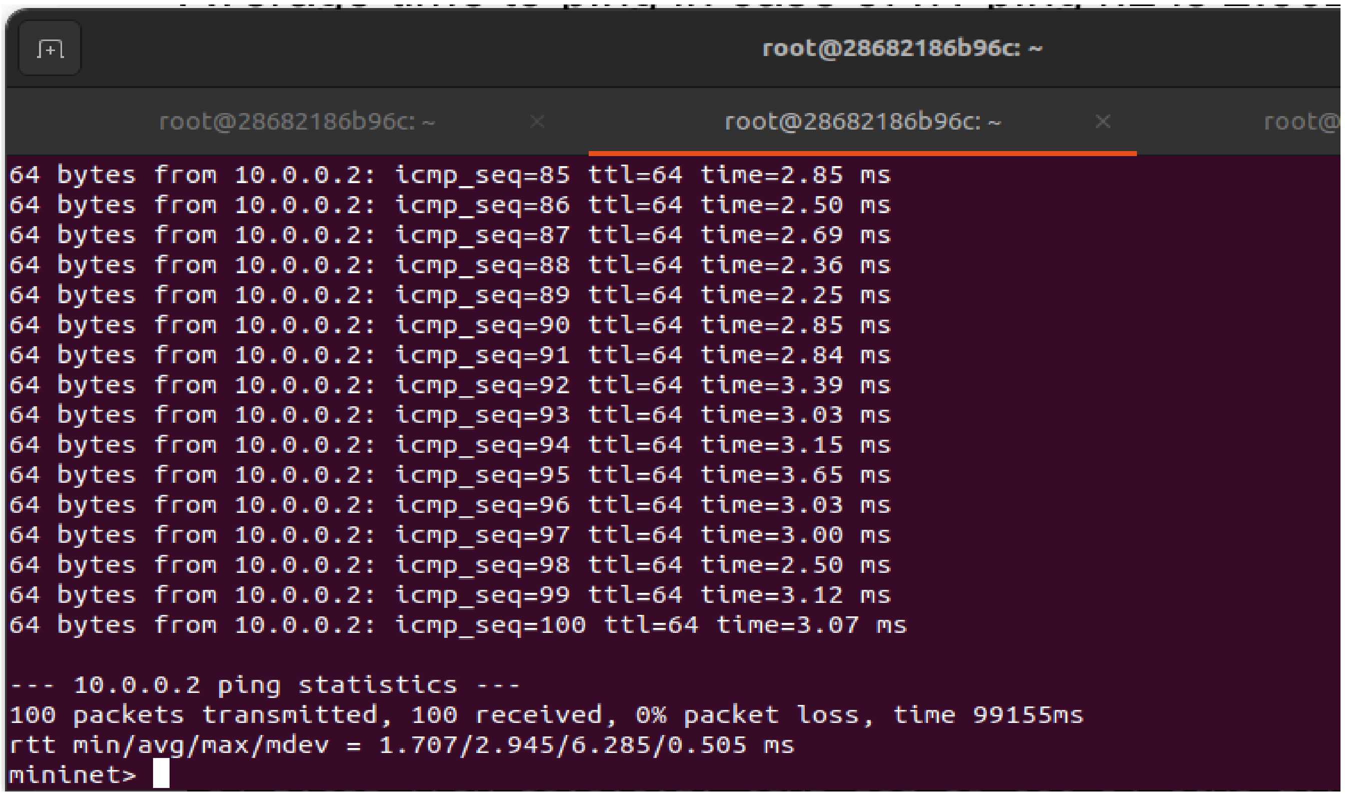
4. Have h1 ping h2, and h1 ping h8 for 100 times (e.g., h1 ping -c100 p2).

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

```
mininet> h1 ping -c100 h2
```

The average time to ping in the case of h1 ping h2 with 100 packets is 2.945ms

Screenshot



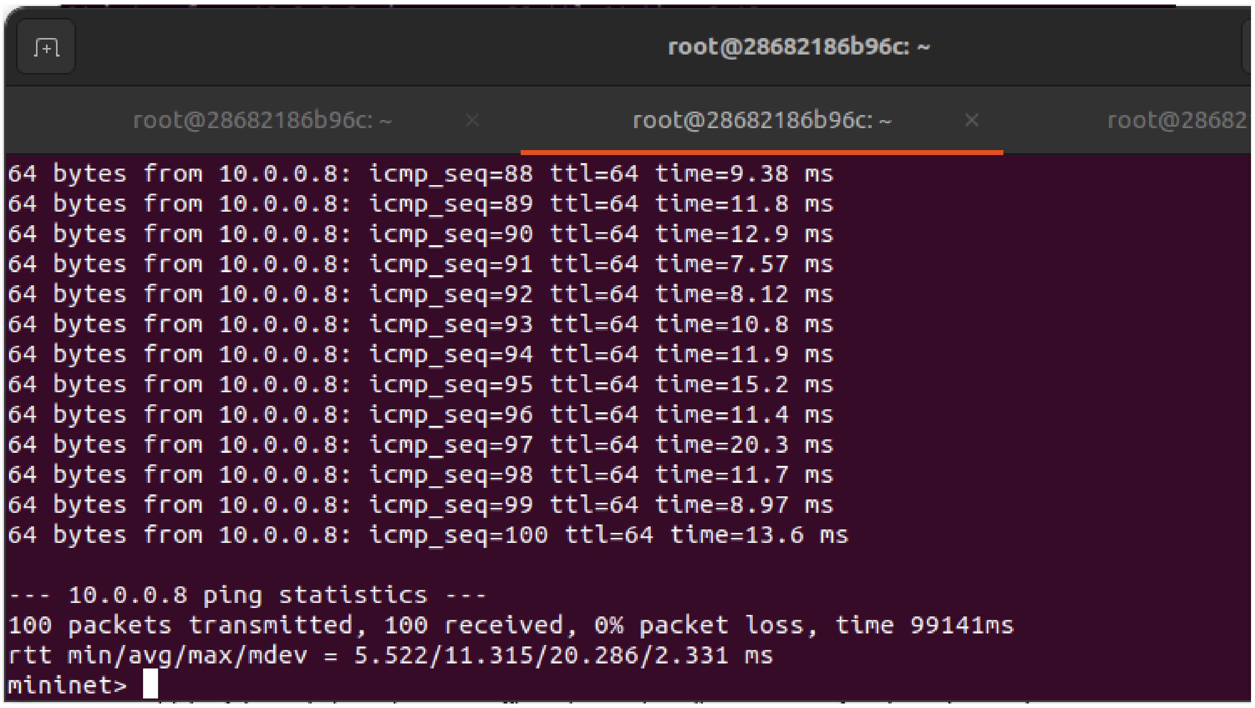
```
64 bytes from 10.0.0.2: icmp_seq=85 ttl=64 time=2.85 ms
64 bytes from 10.0.0.2: icmp_seq=86 ttl=64 time=2.50 ms
64 bytes from 10.0.0.2: icmp_seq=87 ttl=64 time=2.69 ms
64 bytes from 10.0.0.2: icmp_seq=88 ttl=64 time=2.36 ms
64 bytes from 10.0.0.2: icmp_seq=89 ttl=64 time=2.25 ms
64 bytes from 10.0.0.2: icmp_seq=90 ttl=64 time=2.85 ms
64 bytes from 10.0.0.2: icmp_seq=91 ttl=64 time=2.84 ms
64 bytes from 10.0.0.2: icmp_seq=92 ttl=64 time=3.39 ms
64 bytes from 10.0.0.2: icmp_seq=93 ttl=64 time=3.03 ms
64 bytes from 10.0.0.2: icmp_seq=94 ttl=64 time=3.15 ms
64 bytes from 10.0.0.2: icmp_seq=95 ttl=64 time=3.65 ms
64 bytes from 10.0.0.2: icmp_seq=96 ttl=64 time=3.03 ms
64 bytes from 10.0.0.2: icmp_seq=97 ttl=64 time=3.00 ms
64 bytes from 10.0.0.2: icmp_seq=98 ttl=64 time=2.50 ms
64 bytes from 10.0.0.2: icmp_seq=99 ttl=64 time=3.12 ms
64 bytes from 10.0.0.2: icmp_seq=100 ttl=64 time=3.07 ms

--- 10.0.0.2 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99155ms
rtt min/avg/max/mdev = 1.707/2.945/6.285/0.505 ms
mininet> 
```

```
mininet> h1 ping -c100 h8
```

The average time to ping in the case of h1 ping h8 with 100 packets is 11.315ms.

Sourabh Deshmukh
W1648445

Screenshot



```
64 bytes from 10.0.0.8: icmp_seq=88 ttl=64 time=9.38 ms
64 bytes from 10.0.0.8: icmp_seq=89 ttl=64 time=11.8 ms
64 bytes from 10.0.0.8: icmp_seq=90 ttl=64 time=12.9 ms
64 bytes from 10.0.0.8: icmp_seq=91 ttl=64 time=7.57 ms
64 bytes from 10.0.0.8: icmp_seq=92 ttl=64 time=8.12 ms
64 bytes from 10.0.0.8: icmp_seq=93 ttl=64 time=10.8 ms
64 bytes from 10.0.0.8: icmp_seq=94 ttl=64 time=11.9 ms
64 bytes from 10.0.0.8: icmp_seq=95 ttl=64 time=15.2 ms
64 bytes from 10.0.0.8: icmp_seq=96 ttl=64 time=11.4 ms
64 bytes from 10.0.0.8: icmp_seq=97 ttl=64 time=20.3 ms
64 bytes from 10.0.0.8: icmp_seq=98 ttl=64 time=11.7 ms
64 bytes from 10.0.0.8: icmp_seq=99 ttl=64 time=8.97 ms
64 bytes from 10.0.0.8: icmp_seq=100 ttl=64 time=13.6 ms

--- 10.0.0.8 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99141ms
rtt min/avg/max/mdev = 5.522/11.315/20.286/2.331 ms
mininet>
```

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

Minimum ping observed

```
For h1 ping -c100 h2 - 1.707ms
For h1 ping -c100 h8 - 5.522ms
```

Maximum ping observed

```
For h1 ping -c100 h2 - 6.285ms
For h1 ping -c100 h8 - 20.286ms
```

c. What is the difference, and why?

```
The ping time between h1 and h8 (11.315ms, as shown in the initial screenshot) is greater than that between h1 and h2 (2.945ms, as displayed in the screenshot). This disparity arises from the differing network paths taken by the packets.
```

```
To reach h8 from h1, the packets traverse through switches s3, s2, s1, s5, and s7. Conversely, there's only one switch between h1 and h2. Consequently, the ping time is prolonged in the latter scenario, specifically for the route between h1 and h8.
```

5. Run “iperf h1 h2” and “iperf h1 h8”
   a. What is “iperf” used for?’

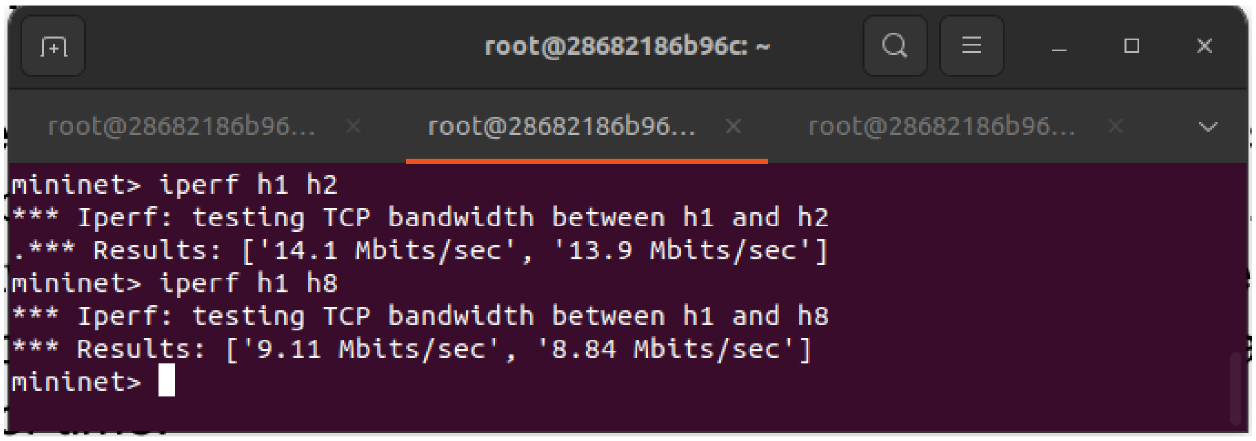
```
iPerf, is an Internet Performance Working Group. It is an open-source utility for conducting speed tests and assessing network performance. Its functionality is straightforward: it generates TCP and UDP streams to transmit traffic from one host to another, subsequently providing users with reports detailing the maximum bandwidth achieved. This capability enables users to ascertain network throughput and identify peak bandwidth capabilities.
```

   b. What is the throughput for each case?

```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
.*** Results: ['14.1 Mbits/sec', '13.9 Mbits/sec']
```

```
mininet> iperf h1 h8
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['9.11 Mbits/sec', '8.84 Mbits/sec']
```

Screenshot



```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
.*** Results: ['14.1 Mbits/sec', '13.9 Mbits/sec']
mininet> iperf h1 h8
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['9.11 Mbits/sec', '8.84 Mbits/sec']
mininet> 
```

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

```
The difference in performance is attributed to differences in network topology and latency levels. Fewer intermediary stops between h1 and h2 contribute to lower latency and reduced congestion, allowing for smoother data flow and higher throughput. Contrarily, the greater number of stops between h1 and h8 introduces more latency and potential congestion spots, thus hindering data transfer speed.
```

6. Which of the switches observes traffic? Please describe your way of observing such traffic on switches (e.g., adding some functions in the “of_tutorial” controller).

```
To identify which switches are monitoring traffic, we can implement a method within the of_tutorial controller. By adding the line log.info("Switch observing traffic: %s" % (self.connection)) to line number 137 of the controller, we can log information about switch activity. This allows us to determine that all switches are indeed monitoring traffic, especially during periods of high packet load. Also, the event listener function _handle_PacketIn is triggered whenever a switch receives a packet, providing additional insight into traffic observation mechanisms.
```

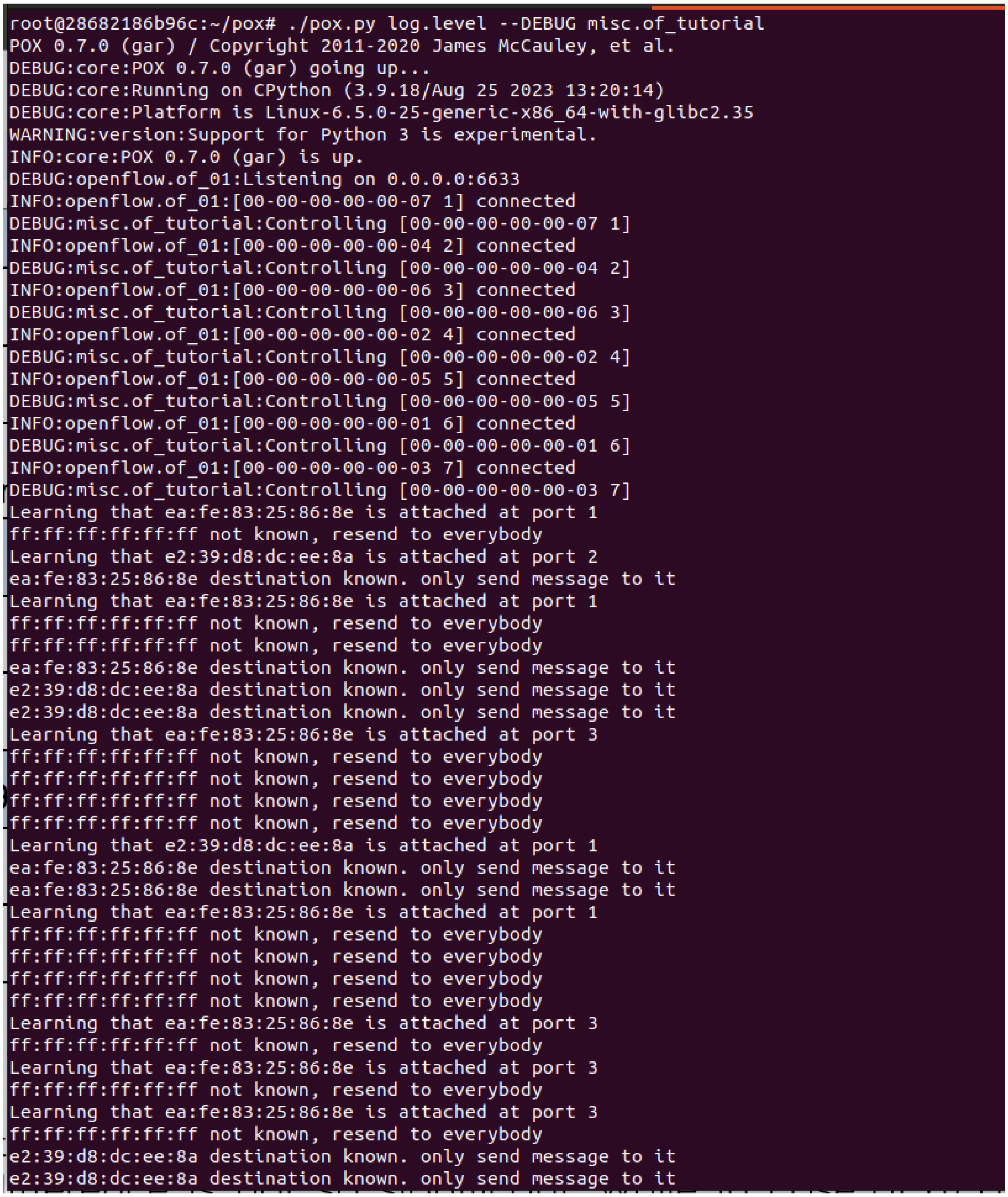
Task 3 QnA

7. 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).

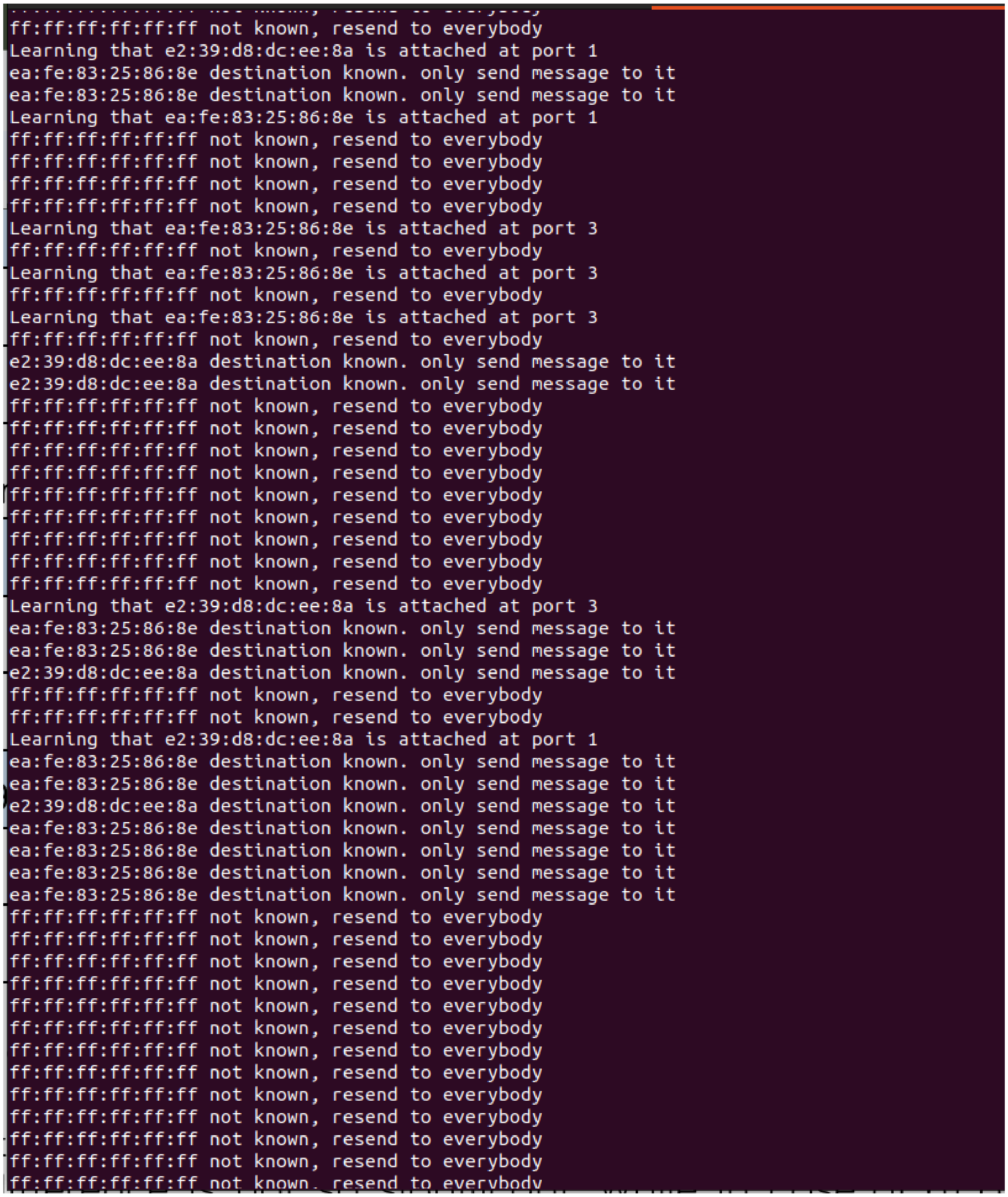
```
Within our code, the act-like-switch function serves a crucial role in identifying the whereabouts of MAC addresses. Therefore, when a MAC address is identified as the intended recipient of a message, the controller is capable of efficiently associating this MAC address with a specific port, simplifying the routing process. This optimization notably enhances the controller's efficiency in transmitting packets to addresses already within its knowledge, as packets can be swiftly directed to their designated ports. Where the destination MAC address remains unknown, the function resorts to flooding the packet to all possible destinations. With the reduction in the frequency of flooding, courtesy of the MAC Learning Controller, overall network performance experiences significant improvements in terms of both ping times and throughput.
```

Screenshots

Sourabh Deshmukh
W1648445

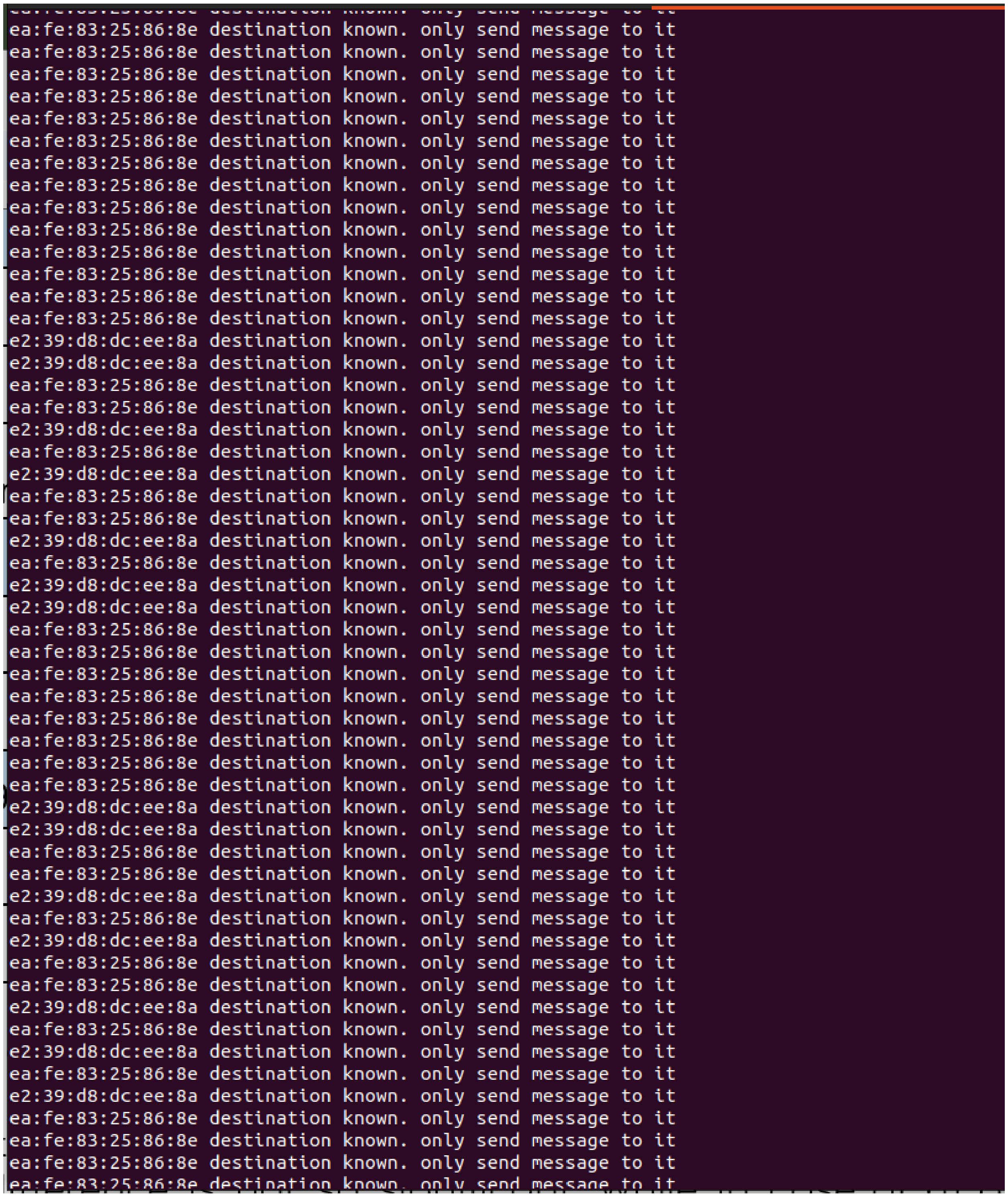


```
root@28682186b96c:~/pox# ./pox.py log.level --DEBUG misc.of_tutorial
POX 0.7.0 (gar) / Copyright 2011-2020 James McCauley, et al.
DEBUG:core:POX 0.7.0 (gar) going up...
DEBUG:core:Running on CPython (3.9.18/Aug 25 2023 13:20:14)
DEBUG:core:Platform is Linux-6.5.0-25-generic-x86_64-with-glibc2.35
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 1] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-07 1]
INFO:openflow.of_01:[00-00-00-00-00-04 2] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-04 2]
INFO:openflow.of_01:[00-00-00-00-00-06 3] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-06 3]
INFO:openflow.of_01:[00-00-00-00-00-02 4] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-02 4]
INFO:openflow.of_01:[00-00-00-00-00-05 5] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-05 5]
INFO:openflow.of_01:[00-00-00-00-00-01 6] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-01 6]
INFO:openflow.of_01:[00-00-00-00-00-03 7] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-03 7]
Learning that ea:fe:83:25:86:8e is attached at port 1
ff:ff:ff:ff:ff:ff not known, resend to everybody
Learning that e2:39:d8:dc:ee:8a is attached at port 2
ea:fe:83:25:86:8e destination known. only send message to it
Learning that ea:fe:83:25:86:8e is attached at port 1
ff:ff:ff:ff:ff:ff not known, resend to everybody
ff:ff:ff:ff:ff:ff not known, resend to everybody
ea:fe:83:25:86:8e destination known. only send message to it
e2:39:d8:dc:ee:8a destination known. only send message to it
e2:39:d8:dc:ee:8a destination known. only send message to it
Learning that ea:fe:83:25:86:8e is attached at port 3
ff:ff:ff:ff:ff:ff not known, resend to everybody
ff:ff:ff:ff:ff:ff not known, resend to everybody
ff:ff:ff:ff:ff:ff not known, resend to everybody
ff:ff:ff:ff:ff:ff not known, resend to everybody
Learning that e2:39:d8:dc:ee:8a is attached at port 1
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
Learning that ea:fe:83:25:86:8e is attached at port 1
ff:ff:ff:ff:ff:ff not known, resend to everybody
ff:ff:ff:ff:ff:ff not known, resend to everybody
ff:ff:ff:ff:ff:ff not known, resend to everybody
ff:ff:ff:ff:ff:ff not known, resend to everybody
Learning that ea:fe:83:25:86:8e is attached at port 3
ff:ff:ff:ff:ff:ff not known, resend to everybody
Learning that ea:fe:83:25:86:8e is attached at port 3
ff:ff:ff:ff:ff:ff not known, resend to everybody
Learning that ea:fe:83:25:86:8e is attached at port 3
ff:ff:ff:ff:ff:ff not known, resend to everybody
e2:39:d8:dc:ee:8a destination known. only send message to it
e2:39:d8:dc:ee:8a destination known. only send message to it
```



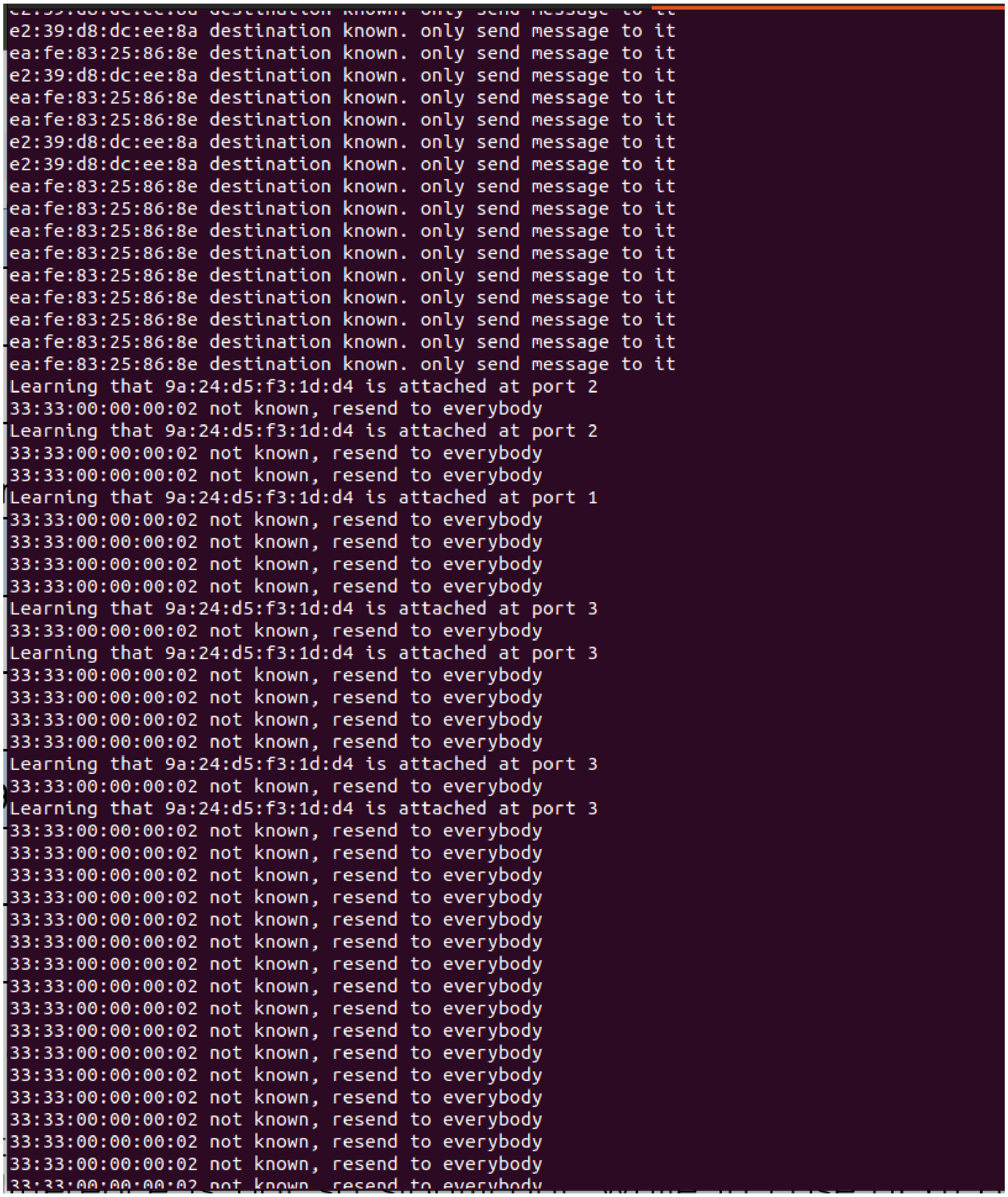
```
ff:ff:ff:ff:ff:ff not known, resend to everybody
Learning that e2:39:d8:dc:ee:8a is attached at port 1
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
Learning that ea:fe:83:25:86:8e is attached at port 1
ff:ff:ff:ff:ff:ff not known, resend to everybody
ff:ff:ff:ff:ff:ff not known, resend to everybody
ff:ff:ff:ff:ff:ff not known, resend to everybody
ff:ff:ff:ff:ff:ff not known, resend to everybody
Learning that ea:fe:83:25:86:8e is attached at port 3
ff:ff:ff:ff:ff:ff not known, resend to everybody
Learning that ea:fe:83:25:86:8e is attached at port 3
ff:ff:ff:ff:ff:ff not known, resend to everybody
Learning that ea:fe:83:25:86:8e is attached at port 3
ff:ff:ff:ff:ff:ff not known, resend to everybody
e2:39:d8:dc:ee:8a destination known. only send message to it
e2:39:d8:dc:ee:8a destination known. only send message to it
ff:ff:ff:ff:ff:ff not known, resend to everybody
ff:ff:ff:ff:ff:ff not known, resend to everybody
ff:ff:ff:ff:ff:ff not known, resend to everybody
ff:ff:ff:ff:ff:ff not known, resend to everybody
ff:ff:ff:ff:ff:ff not known, resend to everybody
ff:ff:ff:ff:ff:ff not known, resend to everybody
ff:ff:ff:ff:ff:ff not known, resend to everybody
ff:ff:ff:ff:ff:ff not known, resend to everybody
ff:ff:ff:ff:ff:ff not known, resend to everybody
Learning that e2:39:d8:dc:ee:8a is attached at port 3
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
e2:39:d8:dc:ee:8a destination known. only send message to it
ff:ff:ff:ff:ff:ff not known, resend to everybody
ff:ff:ff:ff:ff:ff not known, resend to everybody
Learning that e2:39:d8:dc:ee:8a is attached at port 1
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
e2:39:d8:dc:ee:8a destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ff:ff:ff:ff:ff:ff not known, resend to everybody
ff:ff:ff:ff:ff:ff not known, resend to everybody
ff:ff:ff:ff:ff:ff not known, resend to everybody
ff:ff:ff:ff:ff:ff not known, resend to everybody
ff:ff:ff:ff:ff:ff not known, resend to everybody
ff:ff:ff:ff:ff:ff not known, resend to everybody
ff:ff:ff:ff:ff:ff not known, resend to everybody
ff:ff:ff:ff:ff:ff not known, resend to everybody
ff:ff:ff:ff:ff:ff not known, resend to everybody
ff:ff:ff:ff:ff:ff not known, resend to everybody
ff:ff:ff:ff:ff:ff not known, resend to everybody
ff:ff:ff:ff:ff:ff not known, resend to everybody
ff:ff:ff:ff:ff:ff not known, resend to everybody
```

Sourabh Deshmukh
W1648445



```
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
e2:39:d8:dc:ee:8a destination known. only send message to it
e2:39:d8:dc:ee:8a destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
e2:39:d8:dc:ee:8a destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
e2:39:d8:dc:ee:8a destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
e2:39:d8:dc:ee:8a destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
e2:39:d8:dc:ee:8a destination known. only send message to it
e2:39:d8:dc:ee:8a destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
e2:39:d8:dc:ee:8a destination known. only send message to it
e2:39:d8:dc:ee:8a destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
e2:39:d8:dc:ee:8a destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
e2:39:d8:dc:ee:8a destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
e2:39:d8:dc:ee:8a destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
e2:39:d8:dc:ee:8a destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
e2:39:d8:dc:ee:8a destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
```

Sourabh Deshmukh
W1648445



```
e2:39:d8:dc:ee:8a destination known. only send message to it
e2:39:d8:dc:ee:8a destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
e2:39:d8:dc:ee:8a destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
e2:39:d8:dc:ee:8a destination known. only send message to it
e2:39:d8:dc:ee:8a destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
ea:fe:83:25:86:8e destination known. only send message to it
Learning that 9a:24:d5:f3:1d:d4 is attached at port 2
33:33:00:00:00:02 not known, resend to everybody
Learning that 9a:24:d5:f3:1d:d4 is attached at port 2
33:33:00:00:00:02 not known, resend to everybody
33:33:00:00:00:02 not known, resend to everybody
Learning that 9a:24:d5:f3:1d:d4 is attached at port 1
33:33:00:00:00:02 not known, resend to everybody
33:33:00:00:00:02 not known, resend to everybody
33:33:00:00:00:02 not known, resend to everybody
33:33:00:00:00:02 not known, resend to everybody
Learning that 9a:24:d5:f3:1d:d4 is attached at port 3
33:33:00:00:00:02 not known, resend to everybody
Learning that 9a:24:d5:f3:1d:d4 is attached at port 3
33:33:00:00:00:02 not known, resend to everybody
33:33:00:00:00:02 not known, resend to everybody
33:33:00:00:00:02 not known, resend to everybody
33:33:00:00:00:02 not known, resend to everybody
Learning that 9a:24:d5:f3:1d:d4 is attached at port 3
33:33:00:00:00:02 not known, resend to everybody
Learning that 9a:24:d5:f3:1d:d4 is attached at port 3
33:33:00:00:00:02 not known, resend to everybody
33:33:00:00:00:02 not known, resend to everybody
33:33:00:00:00:02 not known, resend to everybody
33:33:00:00:00:02 not known, resend to everybody
33:33:00:00:00:02 not known, resend to everybody
33:33:00:00:00:02 not known, resend to everybody
33:33:00:00:00:02 not known, resend to everybody
33:33:00:00:00:02 not known, resend to everybody
33:33:00:00:00:02 not known, resend to everybody
33:33:00:00:00:02 not known, resend to everybody
33:33:00:00:00:02 not known, resend to everybody
33:33:00:00:00:02 not known, resend to everybody
33:33:00:00:00:02 not known, resend to everybody
33:33:00:00:00:02 not known, resend to everybody
33:33:00:00:00:02 not known, resend to everybody
33:33:00:00:00:02 not known, resend to everybody
33:33:00:00:00:02 not known, resend to everybody
```

Sourabh Deshmukh
W1648445

8. (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?

      For h1 ping -c100 h2

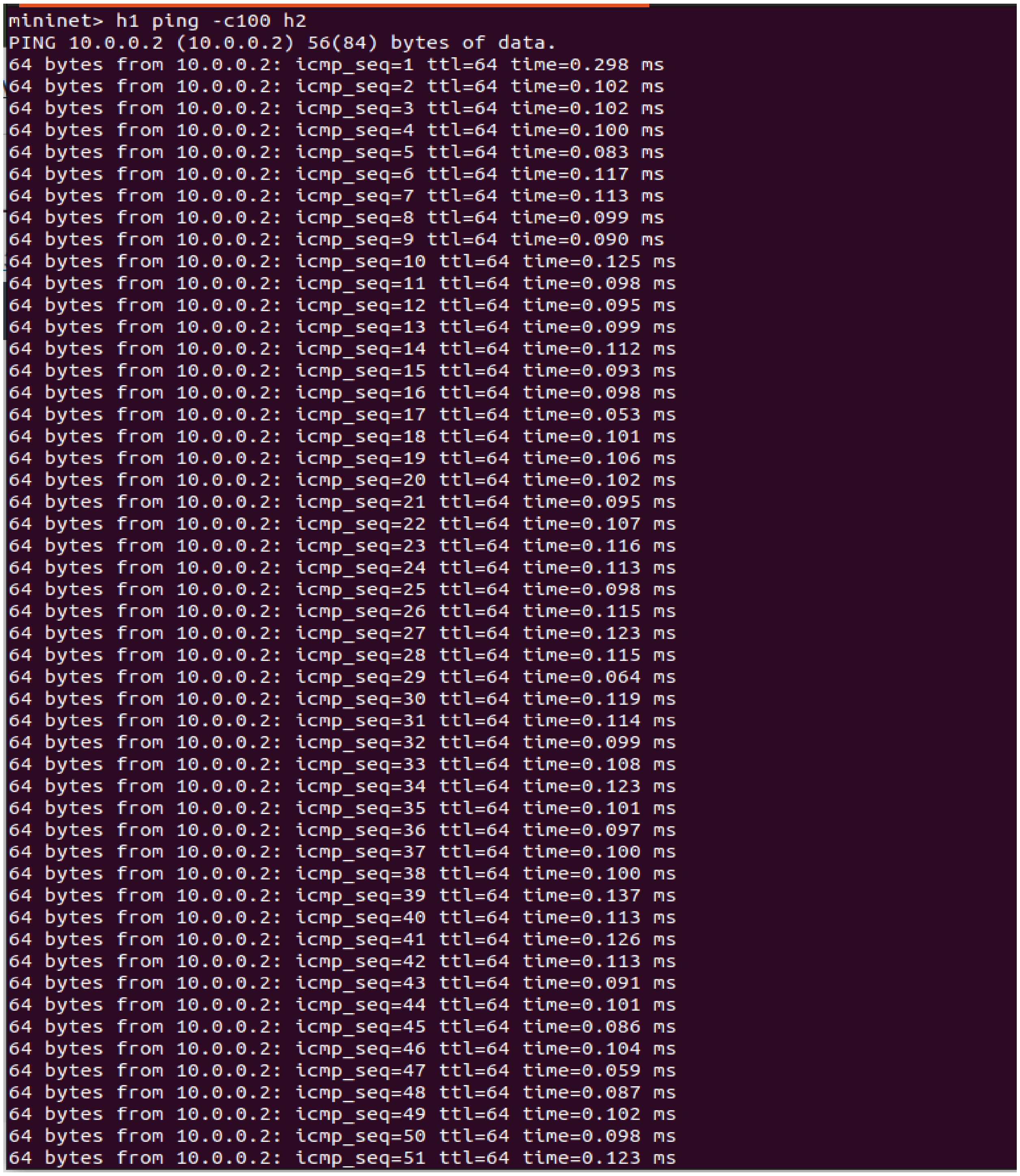
      The Average time to ping in the case of h1 ping h2 is 0.104 ms

      For h1 ping -c100 h8

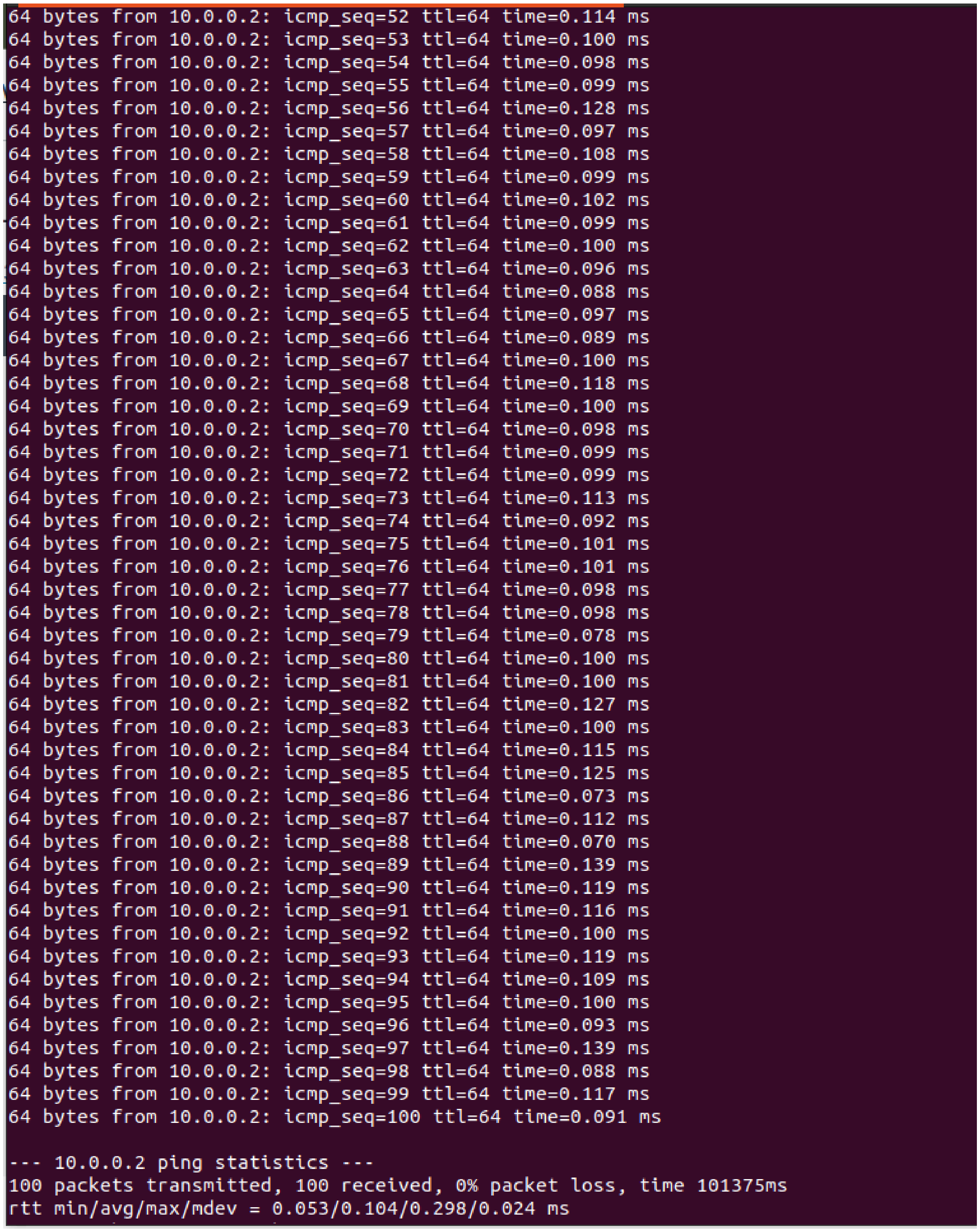
      The average time to ping in the case of h1 ping h8 is 0.324 ms

Sourabh Deshmukh
W1648445

Screenshot for h1 ping -c100 h2

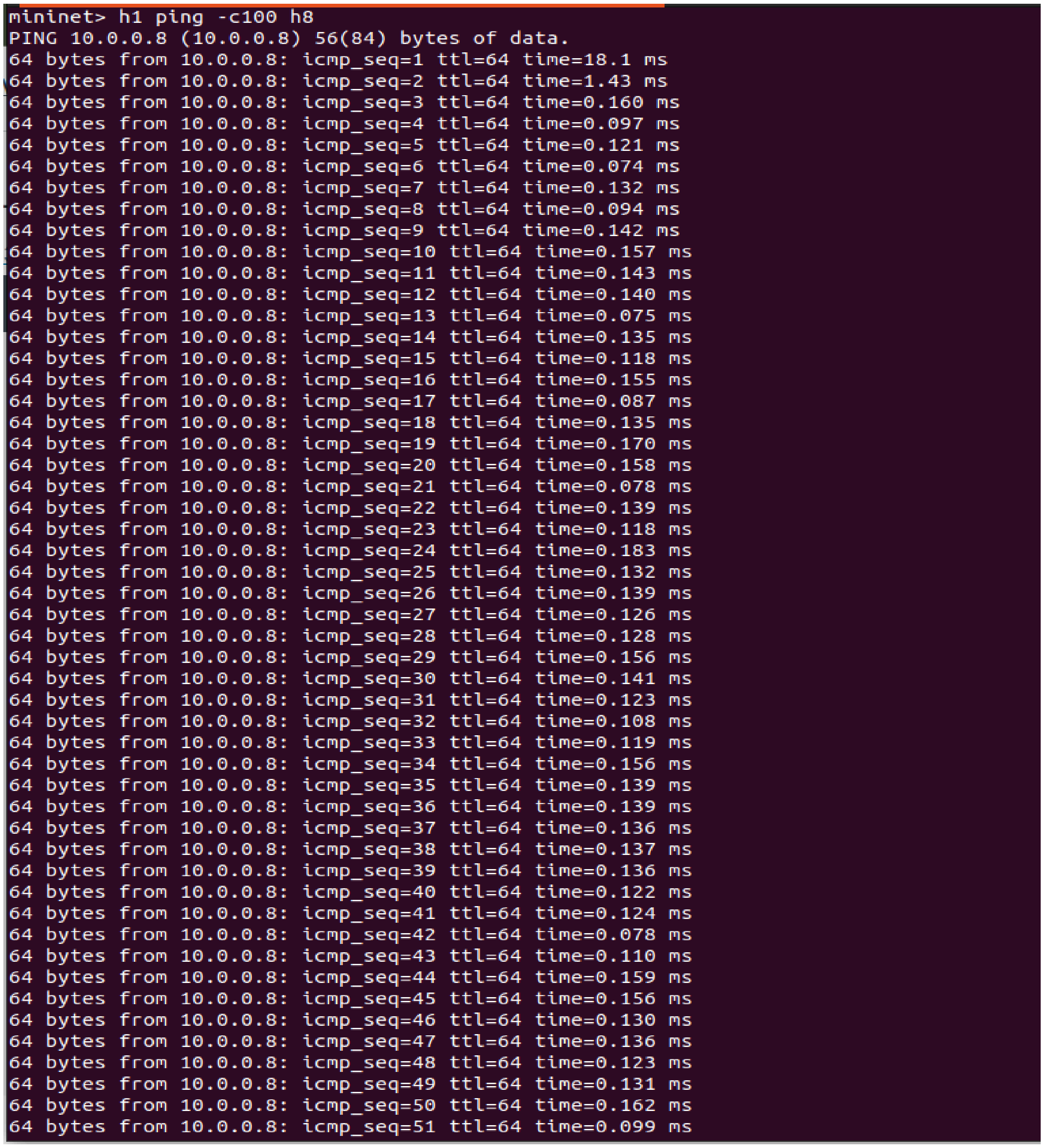


```
mininet> h1 ping -c100 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.298 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.102 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.102 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.100 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.083 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.117 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.113 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.099 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.090 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.125 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=0.098 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=0.095 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=0.099 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=0.112 ms
64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=0.093 ms
64 bytes from 10.0.0.2: icmp_seq=16 ttl=64 time=0.098 ms
64 bytes from 10.0.0.2: icmp_seq=17 ttl=64 time=0.053 ms
64 bytes from 10.0.0.2: icmp_seq=18 ttl=64 time=0.101 ms
64 bytes from 10.0.0.2: icmp_seq=19 ttl=64 time=0.106 ms
64 bytes from 10.0.0.2: icmp_seq=20 ttl=64 time=0.102 ms
64 bytes from 10.0.0.2: icmp_seq=21 ttl=64 time=0.095 ms
64 bytes from 10.0.0.2: icmp_seq=22 ttl=64 time=0.107 ms
64 bytes from 10.0.0.2: icmp_seq=23 ttl=64 time=0.116 ms
64 bytes from 10.0.0.2: icmp_seq=24 ttl=64 time=0.113 ms
64 bytes from 10.0.0.2: icmp_seq=25 ttl=64 time=0.098 ms
64 bytes from 10.0.0.2: icmp_seq=26 ttl=64 time=0.115 ms
64 bytes from 10.0.0.2: icmp_seq=27 ttl=64 time=0.123 ms
64 bytes from 10.0.0.2: icmp_seq=28 ttl=64 time=0.115 ms
64 bytes from 10.0.0.2: icmp_seq=29 ttl=64 time=0.064 ms
64 bytes from 10.0.0.2: icmp_seq=30 ttl=64 time=0.119 ms
64 bytes from 10.0.0.2: icmp_seq=31 ttl=64 time=0.114 ms
64 bytes from 10.0.0.2: icmp_seq=32 ttl=64 time=0.099 ms
64 bytes from 10.0.0.2: icmp_seq=33 ttl=64 time=0.108 ms
64 bytes from 10.0.0.2: icmp_seq=34 ttl=64 time=0.123 ms
64 bytes from 10.0.0.2: icmp_seq=35 ttl=64 time=0.101 ms
64 bytes from 10.0.0.2: icmp_seq=36 ttl=64 time=0.097 ms
64 bytes from 10.0.0.2: icmp_seq=37 ttl=64 time=0.100 ms
64 bytes from 10.0.0.2: icmp_seq=38 ttl=64 time=0.100 ms
64 bytes from 10.0.0.2: icmp_seq=39 ttl=64 time=0.137 ms
64 bytes from 10.0.0.2: icmp_seq=40 ttl=64 time=0.113 ms
64 bytes from 10.0.0.2: icmp_seq=41 ttl=64 time=0.126 ms
64 bytes from 10.0.0.2: icmp_seq=42 ttl=64 time=0.113 ms
64 bytes from 10.0.0.2: icmp_seq=43 ttl=64 time=0.091 ms
64 bytes from 10.0.0.2: icmp_seq=44 ttl=64 time=0.101 ms
64 bytes from 10.0.0.2: icmp_seq=45 ttl=64 time=0.086 ms
64 bytes from 10.0.0.2: icmp_seq=46 ttl=64 time=0.104 ms
64 bytes from 10.0.0.2: icmp_seq=47 ttl=64 time=0.059 ms
64 bytes from 10.0.0.2: icmp_seq=48 ttl=64 time=0.087 ms
64 bytes from 10.0.0.2: icmp_seq=49 ttl=64 time=0.102 ms
64 bytes from 10.0.0.2: icmp_seq=50 ttl=64 time=0.098 ms
64 bytes from 10.0.0.2: icmp_seq=51 ttl=64 time=0.123 ms
```

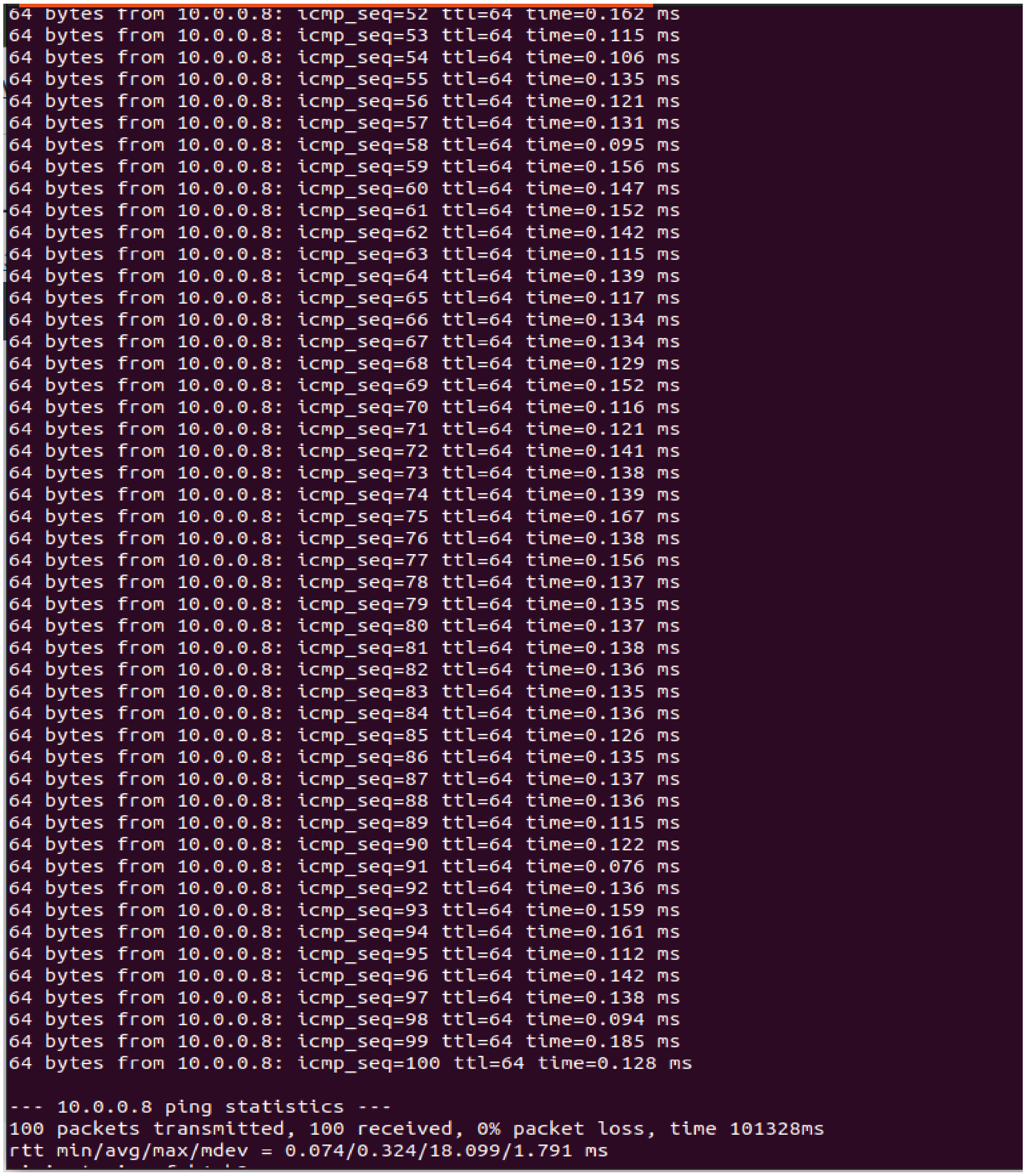


```
64 bytes from 10.0.0.2: icmp_seq=52 ttl=64 time=0.114 ms
64 bytes from 10.0.0.2: icmp_seq=53 ttl=64 time=0.100 ms
64 bytes from 10.0.0.2: icmp_seq=54 ttl=64 time=0.098 ms
64 bytes from 10.0.0.2: icmp_seq=55 ttl=64 time=0.099 ms
64 bytes from 10.0.0.2: icmp_seq=56 ttl=64 time=0.128 ms
64 bytes from 10.0.0.2: icmp_seq=57 ttl=64 time=0.097 ms
64 bytes from 10.0.0.2: icmp_seq=58 ttl=64 time=0.108 ms
64 bytes from 10.0.0.2: icmp_seq=59 ttl=64 time=0.099 ms
64 bytes from 10.0.0.2: icmp_seq=60 ttl=64 time=0.102 ms
64 bytes from 10.0.0.2: icmp_seq=61 ttl=64 time=0.099 ms
64 bytes from 10.0.0.2: icmp_seq=62 ttl=64 time=0.100 ms
64 bytes from 10.0.0.2: icmp_seq=63 ttl=64 time=0.096 ms
64 bytes from 10.0.0.2: icmp_seq=64 ttl=64 time=0.088 ms
64 bytes from 10.0.0.2: icmp_seq=65 ttl=64 time=0.097 ms
64 bytes from 10.0.0.2: icmp_seq=66 ttl=64 time=0.089 ms
64 bytes from 10.0.0.2: icmp_seq=67 ttl=64 time=0.100 ms
64 bytes from 10.0.0.2: icmp_seq=68 ttl=64 time=0.118 ms
64 bytes from 10.0.0.2: icmp_seq=69 ttl=64 time=0.100 ms
64 bytes from 10.0.0.2: icmp_seq=70 ttl=64 time=0.098 ms
64 bytes from 10.0.0.2: icmp_seq=71 ttl=64 time=0.099 ms
64 bytes from 10.0.0.2: icmp_seq=72 ttl=64 time=0.099 ms
64 bytes from 10.0.0.2: icmp_seq=73 ttl=64 time=0.113 ms
64 bytes from 10.0.0.2: icmp_seq=74 ttl=64 time=0.092 ms
64 bytes from 10.0.0.2: icmp_seq=75 ttl=64 time=0.101 ms
64 bytes from 10.0.0.2: icmp_seq=76 ttl=64 time=0.101 ms
64 bytes from 10.0.0.2: icmp_seq=77 ttl=64 time=0.098 ms
64 bytes from 10.0.0.2: icmp_seq=78 ttl=64 time=0.098 ms
64 bytes from 10.0.0.2: icmp_seq=79 ttl=64 time=0.078 ms
64 bytes from 10.0.0.2: icmp_seq=80 ttl=64 time=0.100 ms
64 bytes from 10.0.0.2: icmp_seq=81 ttl=64 time=0.100 ms
64 bytes from 10.0.0.2: icmp_seq=82 ttl=64 time=0.127 ms
64 bytes from 10.0.0.2: icmp_seq=83 ttl=64 time=0.100 ms
64 bytes from 10.0.0.2: icmp_seq=84 ttl=64 time=0.115 ms
64 bytes from 10.0.0.2: icmp_seq=85 ttl=64 time=0.125 ms
64 bytes from 10.0.0.2: icmp_seq=86 ttl=64 time=0.073 ms
64 bytes from 10.0.0.2: icmp_seq=87 ttl=64 time=0.112 ms
64 bytes from 10.0.0.2: icmp_seq=88 ttl=64 time=0.070 ms
64 bytes from 10.0.0.2: icmp_seq=89 ttl=64 time=0.139 ms
64 bytes from 10.0.0.2: icmp_seq=90 ttl=64 time=0.119 ms
64 bytes from 10.0.0.2: icmp_seq=91 ttl=64 time=0.116 ms
64 bytes from 10.0.0.2: icmp_seq=92 ttl=64 time=0.100 ms
64 bytes from 10.0.0.2: icmp_seq=93 ttl=64 time=0.119 ms
64 bytes from 10.0.0.2: icmp_seq=94 ttl=64 time=0.109 ms
64 bytes from 10.0.0.2: icmp_seq=95 ttl=64 time=0.100 ms
64 bytes from 10.0.0.2: icmp_seq=96 ttl=64 time=0.093 ms
64 bytes from 10.0.0.2: icmp_seq=97 ttl=64 time=0.139 ms
64 bytes from 10.0.0.2: icmp_seq=98 ttl=64 time=0.088 ms
64 bytes from 10.0.0.2: icmp_seq=99 ttl=64 time=0.117 ms
64 bytes from 10.0.0.2: icmp_seq=100 ttl=64 time=0.091 ms

--- 10.0.0.2 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 101375ms
rtt min/avg/max/mdev = 0.053/0.104/0.298/0.024 ms
```

Screenshot for 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=18.1 ms
64 bytes from 10.0.0.8: icmp_seq=2 ttl=64 time=1.43 ms
64 bytes from 10.0.0.8: icmp_seq=3 ttl=64 time=0.160 ms
64 bytes from 10.0.0.8: icmp_seq=4 ttl=64 time=0.097 ms
64 bytes from 10.0.0.8: icmp_seq=5 ttl=64 time=0.121 ms
64 bytes from 10.0.0.8: icmp_seq=6 ttl=64 time=0.074 ms
64 bytes from 10.0.0.8: icmp_seq=7 ttl=64 time=0.132 ms
64 bytes from 10.0.0.8: icmp_seq=8 ttl=64 time=0.094 ms
64 bytes from 10.0.0.8: icmp_seq=9 ttl=64 time=0.142 ms
64 bytes from 10.0.0.8: icmp_seq=10 ttl=64 time=0.157 ms
64 bytes from 10.0.0.8: icmp_seq=11 ttl=64 time=0.143 ms
64 bytes from 10.0.0.8: icmp_seq=12 ttl=64 time=0.140 ms
64 bytes from 10.0.0.8: icmp_seq=13 ttl=64 time=0.075 ms
64 bytes from 10.0.0.8: icmp_seq=14 ttl=64 time=0.135 ms
64 bytes from 10.0.0.8: icmp_seq=15 ttl=64 time=0.118 ms
64 bytes from 10.0.0.8: icmp_seq=16 ttl=64 time=0.155 ms
64 bytes from 10.0.0.8: icmp_seq=17 ttl=64 time=0.087 ms
64 bytes from 10.0.0.8: icmp_seq=18 ttl=64 time=0.135 ms
64 bytes from 10.0.0.8: icmp_seq=19 ttl=64 time=0.170 ms
64 bytes from 10.0.0.8: icmp_seq=20 ttl=64 time=0.158 ms
64 bytes from 10.0.0.8: icmp_seq=21 ttl=64 time=0.078 ms
64 bytes from 10.0.0.8: icmp_seq=22 ttl=64 time=0.139 ms
64 bytes from 10.0.0.8: icmp_seq=23 ttl=64 time=0.118 ms
64 bytes from 10.0.0.8: icmp_seq=24 ttl=64 time=0.183 ms
64 bytes from 10.0.0.8: icmp_seq=25 ttl=64 time=0.132 ms
64 bytes from 10.0.0.8: icmp_seq=26 ttl=64 time=0.139 ms
64 bytes from 10.0.0.8: icmp_seq=27 ttl=64 time=0.126 ms
64 bytes from 10.0.0.8: icmp_seq=28 ttl=64 time=0.128 ms
64 bytes from 10.0.0.8: icmp_seq=29 ttl=64 time=0.156 ms
64 bytes from 10.0.0.8: icmp_seq=30 ttl=64 time=0.141 ms
64 bytes from 10.0.0.8: icmp_seq=31 ttl=64 time=0.123 ms
64 bytes from 10.0.0.8: icmp_seq=32 ttl=64 time=0.108 ms
64 bytes from 10.0.0.8: icmp_seq=33 ttl=64 time=0.119 ms
64 bytes from 10.0.0.8: icmp_seq=34 ttl=64 time=0.156 ms
64 bytes from 10.0.0.8: icmp_seq=35 ttl=64 time=0.139 ms
64 bytes from 10.0.0.8: icmp_seq=36 ttl=64 time=0.139 ms
64 bytes from 10.0.0.8: icmp_seq=37 ttl=64 time=0.136 ms
64 bytes from 10.0.0.8: icmp_seq=38 ttl=64 time=0.137 ms
64 bytes from 10.0.0.8: icmp_seq=39 ttl=64 time=0.136 ms
64 bytes from 10.0.0.8: icmp_seq=40 ttl=64 time=0.122 ms
64 bytes from 10.0.0.8: icmp_seq=41 ttl=64 time=0.124 ms
64 bytes from 10.0.0.8: icmp_seq=42 ttl=64 time=0.078 ms
64 bytes from 10.0.0.8: icmp_seq=43 ttl=64 time=0.110 ms
64 bytes from 10.0.0.8: icmp_seq=44 ttl=64 time=0.159 ms
64 bytes from 10.0.0.8: icmp_seq=45 ttl=64 time=0.156 ms
64 bytes from 10.0.0.8: icmp_seq=46 ttl=64 time=0.130 ms
64 bytes from 10.0.0.8: icmp_seq=47 ttl=64 time=0.136 ms
64 bytes from 10.0.0.8: icmp_seq=48 ttl=64 time=0.123 ms
64 bytes from 10.0.0.8: icmp_seq=49 ttl=64 time=0.131 ms
64 bytes from 10.0.0.8: icmp_seq=50 ttl=64 time=0.162 ms
64 bytes from 10.0.0.8: icmp_seq=51 ttl=64 time=0.099 ms
```



```
64 bytes from 10.0.0.8: icmp_seq=52 ttl=64 time=0.162 ms
64 bytes from 10.0.0.8: icmp_seq=53 ttl=64 time=0.115 ms
64 bytes from 10.0.0.8: icmp_seq=54 ttl=64 time=0.106 ms
64 bytes from 10.0.0.8: icmp_seq=55 ttl=64 time=0.135 ms
64 bytes from 10.0.0.8: icmp_seq=56 ttl=64 time=0.121 ms
64 bytes from 10.0.0.8: icmp_seq=57 ttl=64 time=0.131 ms
64 bytes from 10.0.0.8: icmp_seq=58 ttl=64 time=0.095 ms
64 bytes from 10.0.0.8: icmp_seq=59 ttl=64 time=0.156 ms
64 bytes from 10.0.0.8: icmp_seq=60 ttl=64 time=0.147 ms
64 bytes from 10.0.0.8: icmp_seq=61 ttl=64 time=0.152 ms
64 bytes from 10.0.0.8: icmp_seq=62 ttl=64 time=0.142 ms
64 bytes from 10.0.0.8: icmp_seq=63 ttl=64 time=0.115 ms
64 bytes from 10.0.0.8: icmp_seq=64 ttl=64 time=0.139 ms
64 bytes from 10.0.0.8: icmp_seq=65 ttl=64 time=0.117 ms
64 bytes from 10.0.0.8: icmp_seq=66 ttl=64 time=0.134 ms
64 bytes from 10.0.0.8: icmp_seq=67 ttl=64 time=0.134 ms
64 bytes from 10.0.0.8: icmp_seq=68 ttl=64 time=0.129 ms
64 bytes from 10.0.0.8: icmp_seq=69 ttl=64 time=0.152 ms
64 bytes from 10.0.0.8: icmp_seq=70 ttl=64 time=0.116 ms
64 bytes from 10.0.0.8: icmp_seq=71 ttl=64 time=0.121 ms
64 bytes from 10.0.0.8: icmp_seq=72 ttl=64 time=0.141 ms
64 bytes from 10.0.0.8: icmp_seq=73 ttl=64 time=0.138 ms
64 bytes from 10.0.0.8: icmp_seq=74 ttl=64 time=0.139 ms
64 bytes from 10.0.0.8: icmp_seq=75 ttl=64 time=0.167 ms
64 bytes from 10.0.0.8: icmp_seq=76 ttl=64 time=0.138 ms
64 bytes from 10.0.0.8: icmp_seq=77 ttl=64 time=0.156 ms
64 bytes from 10.0.0.8: icmp_seq=78 ttl=64 time=0.137 ms
64 bytes from 10.0.0.8: icmp_seq=79 ttl=64 time=0.135 ms
64 bytes from 10.0.0.8: icmp_seq=80 ttl=64 time=0.137 ms
64 bytes from 10.0.0.8: icmp_seq=81 ttl=64 time=0.138 ms
64 bytes from 10.0.0.8: icmp_seq=82 ttl=64 time=0.136 ms
64 bytes from 10.0.0.8: icmp_seq=83 ttl=64 time=0.135 ms
64 bytes from 10.0.0.8: icmp_seq=84 ttl=64 time=0.136 ms
64 bytes from 10.0.0.8: icmp_seq=85 ttl=64 time=0.126 ms
64 bytes from 10.0.0.8: icmp_seq=86 ttl=64 time=0.135 ms
64 bytes from 10.0.0.8: icmp_seq=87 ttl=64 time=0.137 ms
64 bytes from 10.0.0.8: icmp_seq=88 ttl=64 time=0.136 ms
64 bytes from 10.0.0.8: icmp_seq=89 ttl=64 time=0.115 ms
64 bytes from 10.0.0.8: icmp_seq=90 ttl=64 time=0.122 ms
64 bytes from 10.0.0.8: icmp_seq=91 ttl=64 time=0.076 ms
64 bytes from 10.0.0.8: icmp_seq=92 ttl=64 time=0.136 ms
64 bytes from 10.0.0.8: icmp_seq=93 ttl=64 time=0.159 ms
64 bytes from 10.0.0.8: icmp_seq=94 ttl=64 time=0.161 ms
64 bytes from 10.0.0.8: icmp_seq=95 ttl=64 time=0.112 ms
64 bytes from 10.0.0.8: icmp_seq=96 ttl=64 time=0.142 ms
64 bytes from 10.0.0.8: icmp_seq=97 ttl=64 time=0.138 ms
64 bytes from 10.0.0.8: icmp_seq=98 ttl=64 time=0.094 ms
64 bytes from 10.0.0.8: icmp_seq=99 ttl=64 time=0.185 ms
64 bytes from 10.0.0.8: icmp_seq=100 ttl=64 time=0.128 ms

--- 10.0.0.8 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 101328ms
rtt min/avg/max/mdev = 0.074/0.324/18.099/1.791 ms
```

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

Minimum ping Observed -

```
For h1 ping -c100 h2 - 0.053ms
For h1 ping -c100 h8 - 0.074ms
```

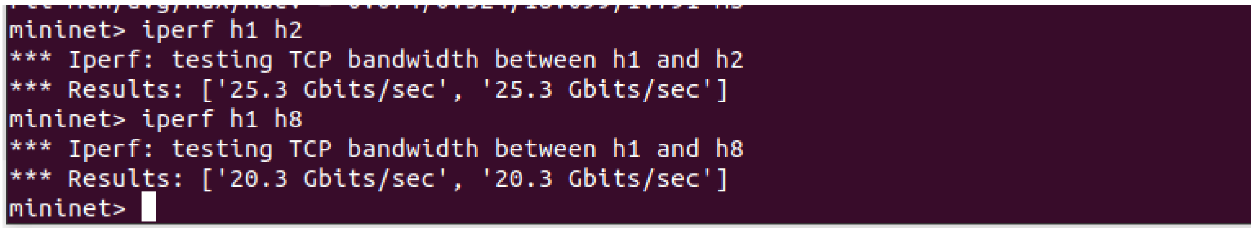
Maximum ping Observed -

```
For h1 ping -c100 h2 - 0.298ms
For h1 ping -c100 h8 - 18.099ms
```

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

```
Task 3 displays a slightly shorter duration compared to Task 2
when evaluating the ping time between h1 and h2, even though the
discrepancy is minor. However, the ping time disparity between h1
and h8 is notably significant due to the extensive routing
involved. In Task 3, where only the initial few packets are
subjected to flooding, it becomes evident that this task operates
notably quicker or demonstrates reduced ping times. This
efficiency arises from the fact that once the destination MAC
address is identified within the "Mac to port" mapping, subsequent
packets are promptly directed to their designated ports without
the need for flooding. Consequently, future pings experience
considerable acceleration owing to diminished network congestion.
```

9. Q.3 Run "iperf h1 h2" and "iperf h1 h8".
   a. What is the throughput for each case?



```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['25.3 Gbits/sec', '25.3 Gbits/sec']
mininet> iperf h1 h8
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['20.3 Gbits/sec', '20.3 Gbits/sec']
mininet>
```

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

```
In both scenarios, Task 3 demonstrates excellent throughput compared to
Task 2. This can be attributed to the reduced network congestion inherent
in Task 3. As the "Mac to port" mapping has already learned all the
necessary port associations, there is no need for packet flooding, thereby
alleviating the burden on switches. The implementation of pre-computed and
learned routes, coupled with controller updates, is evident in the
significant throughput enhancements observed between Task 2 and Task 3 for
h1 and h2. Contrarily, the throughput improvement is less pronounced for h1
and h8 due to the higher number of hops and increased likelihood of packet
dropping along the route.
```