Διαχείριση Δικτύων Βασισμένων στο Λογισμικό
6ο εργαστήριο: "Mininet walkthrough"

ΟΝΟΜΑΤΕΠΩΝΥΜΟ:	Νικολας Μαυροπουλος
Α.Μ.:	21865

Για να ξεκινήσετε χρειάζεστε ένα Ubuntu image (π.χ. από εδώ: https://ubuntu.com/download/desktop το Ubuntu 20.04.4 LTS) και ένα virtualisation system όπως το VirtualBox: https://www.virtualbox.org/wiki/Downloads.

Ανοίξτε το Ubuntu image μέσω VirtualBox ("New" → and use default options. Then "Start" and load the downloaded image. "Install Ubuntu" option (not "Try Ubuntu")!).

Επισκεφτείτε τον ιστότοπο http://mininet.org/download/ και ακολουθήστε τις οδηγίες για εγκατάσταση του Mininet.

Συστήνεται το Option 2: Native Installation from Source ή Option 3: Installation from Packages.

Option 2 commands summary (συστήνεται να ακολουθήσετε τις αναλυτικές οδηγίες από το http://mininet.org/download/):

```
git clone https://github.com/mininet/mininet
```

```
cd mininet
git tag  # list available versions
git checkout -b mininet-2.3.0 2.3.0  # or whatever version you wish to
install
cd ..
```

```
mininet/util/install.sh -a
```

```
sudo mn --switch ovsbr --test pingall
```

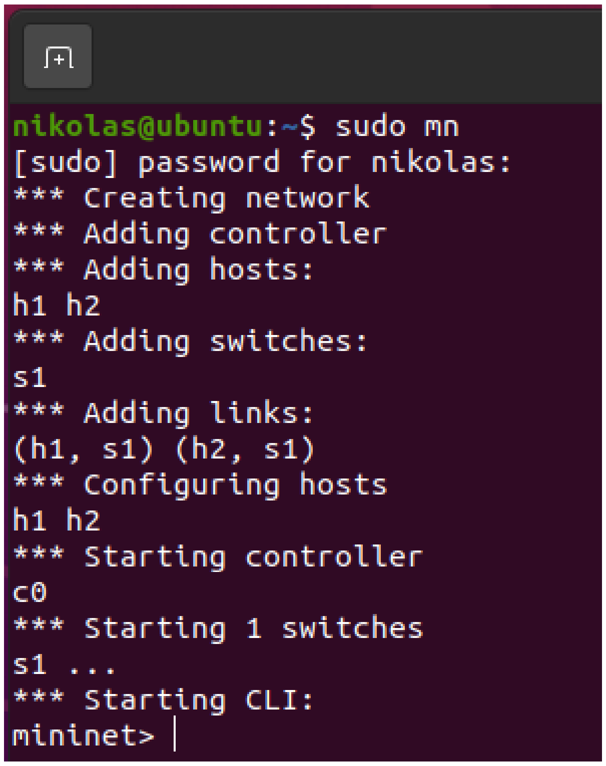
```
echo py sys.version | sudo mn -v output
```

Now you should have installed Mininet!
Στη συνέχεια, εγκαταστήστε τον POX SDN Controller σύμφωνα με τις οδηγίες:
https://noxrepo.github.io/pox-doc/html/

Now follow the Walkthrough to get familiar with Mininet commands and typical usage:
http://mininet.org/walkthrough/
While following the Walkthrough answer the following questions (which belong to the Walkthrough) by providing respective screenshots. Note that the Walkthrough provides details which will help you answer these questions.

Q1. Provide the output of the following command, to prove that you have successfully installed Mininet:

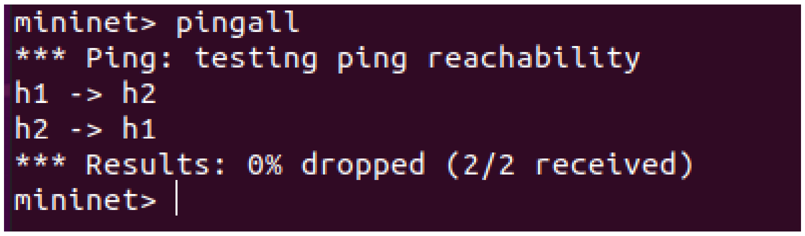
```
$ sudo mn
```



```
nikolas@ubuntu:~$ sudo mn
[sudo] password for nikolas:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

Q2. Verify that you can ping among all hosts:

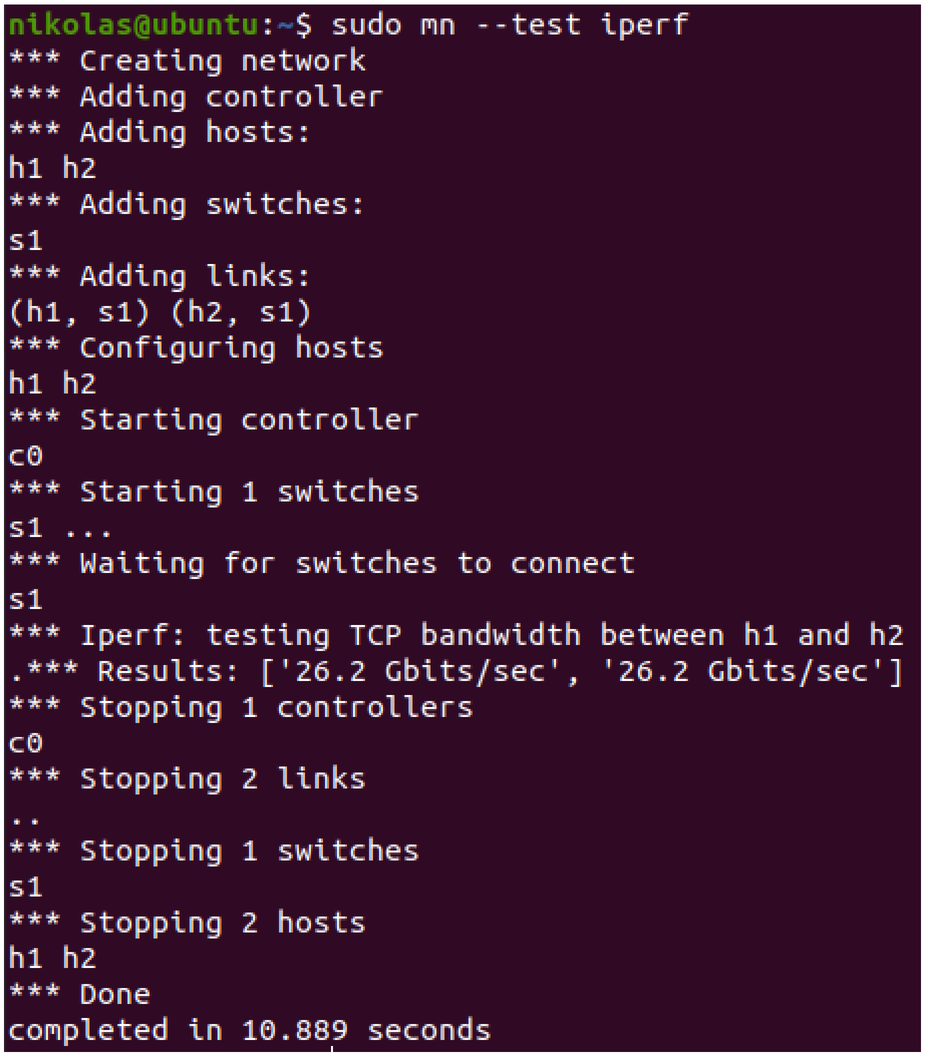
```
mininet> pingall
```



```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet>
```

Q3. Run a regression test:

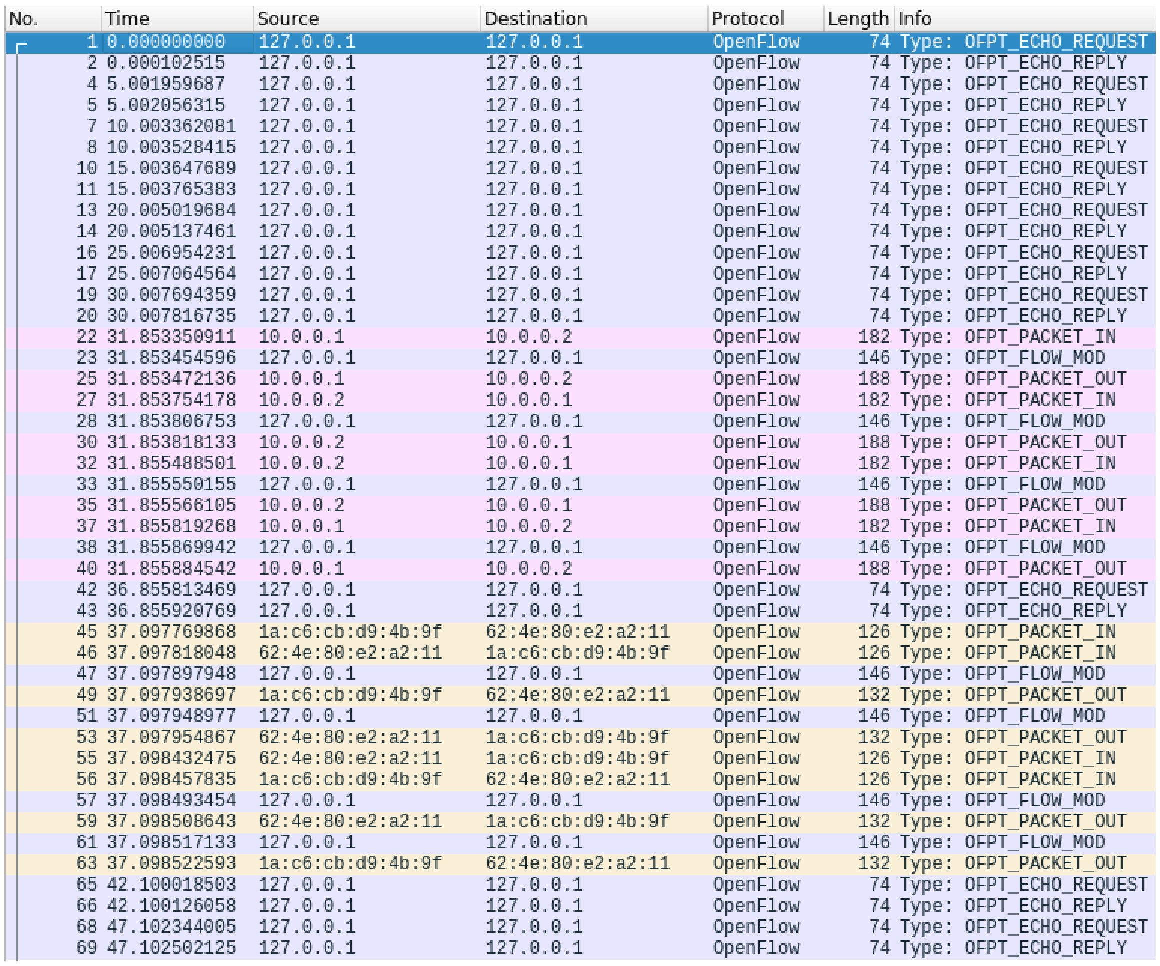
```
$ sudo mn --test iperf
```



```
nikolas@ubuntu:~$ sudo mn --test iperf
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Waiting for switches to connect
s1
*** Iperf: testing TCP bandwidth between h1 and h2
.*** Results: ['26.2 Gbits/sec', '26.2 Gbits/sec']
*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 10.889 seconds
```

Q4. In the Mininet walkthrough it is written that: "*You should see OpenFlow control traffic. The first host ARPs for the MAC address of the second, which causes a* `packet_in` *message to go to the controller. The controller then sends a* `packet_out` *message to flood the broadcast packet to other ports on the switch (in this example, the only other data port). The second host sees the ARP request and sends a reply. This reply goes to the controller, which sends it to the first host and pushes down a flow entry.*"

Add the respective screenshot from Wireshark.

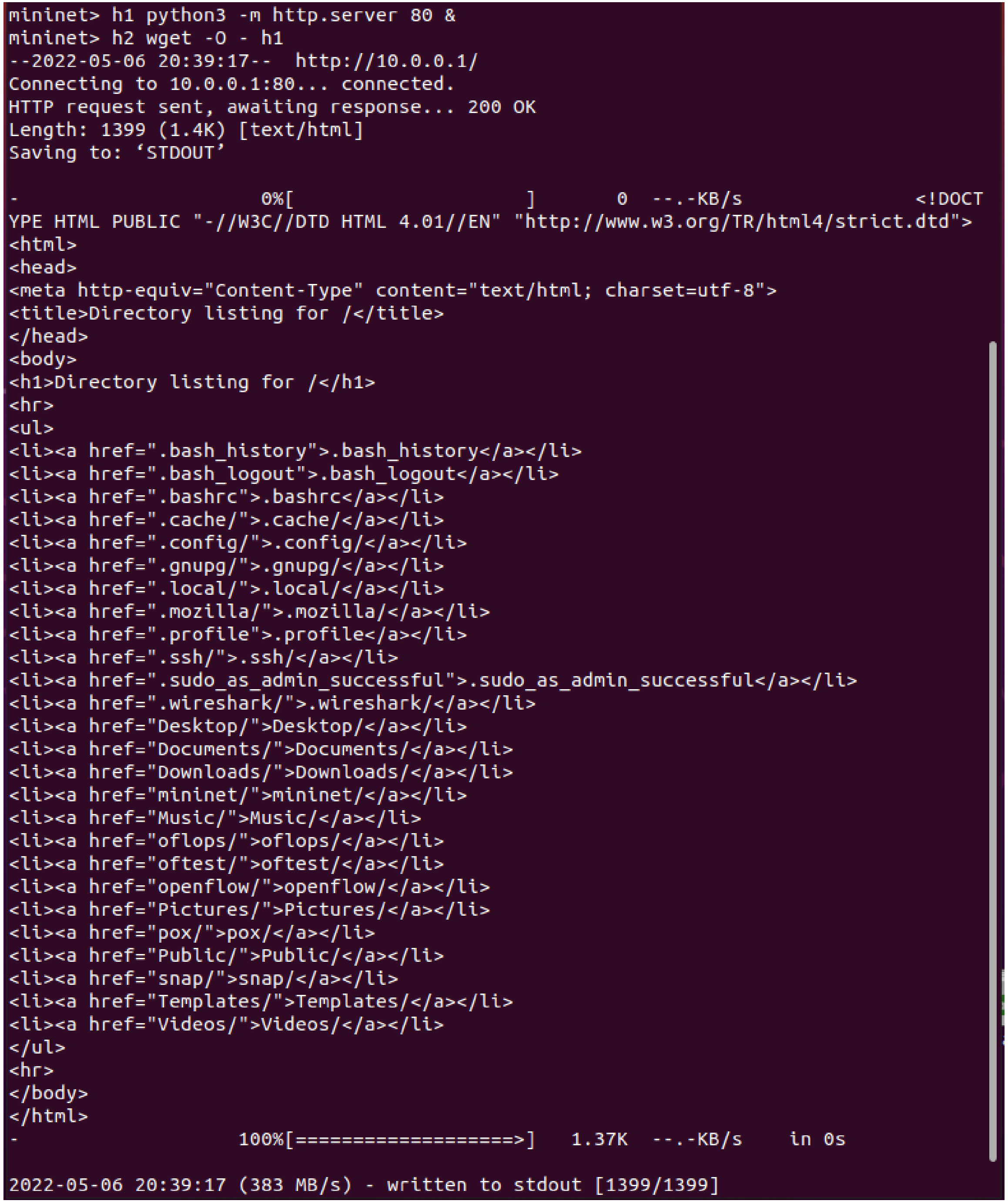


No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_ECHO_REQUEST
2	0.000102515	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_ECHO_REPLY
4	5.001959687	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_ECHO_REQUEST
5	5.002056315	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_ECHO_REPLY
7	10.003362081	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_ECHO_REQUEST
8	10.003528415	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_ECHO_REPLY
10	15.003647689	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_ECHO_REQUEST
11	15.003765383	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_ECHO_REPLY
13	20.005019684	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_ECHO_REQUEST
14	20.005137461	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_ECHO_REPLY
16	25.006954231	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_ECHO_REQUEST
17	25.007064564	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_ECHO_REPLY
19	30.007694359	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_ECHO_REQUEST
20	30.007816735	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_ECHO_REPLY
22	31.853350911	10.0.0.1	10.0.0.2	OpenFlow	182	Type: OFPT_PACKET_IN
23	31.853454596	127.0.0.1	127.0.0.1	OpenFlow	146	Type: OFPT_FLOW_MOD
25	31.853472136	10.0.0.1	10.0.0.2	OpenFlow	188	Type: OFPT_PACKET_OUT
27	31.853754178	10.0.0.2	10.0.0.1	OpenFlow	182	Type: OFPT_PACKET_IN
28	31.853806753	127.0.0.1	127.0.0.1	OpenFlow	146	Type: OFPT_FLOW_MOD
30	31.853818133	10.0.0.2	10.0.0.1	OpenFlow	188	Type: OFPT_PACKET_OUT
32	31.855488501	10.0.0.2	10.0.0.1	OpenFlow	182	Type: OFPT_PACKET_IN
33	31.855550155	127.0.0.1	127.0.0.1	OpenFlow	146	Type: OFPT_FLOW_MOD
35	31.855566105	10.0.0.2	10.0.0.1	OpenFlow	188	Type: OFPT_PACKET_OUT
37	31.855819268	10.0.0.1	10.0.0.2	OpenFlow	182	Type: OFPT_PACKET_IN
38	31.855869942	127.0.0.1	127.0.0.1	OpenFlow	146	Type: OFPT_FLOW_MOD
40	31.855884542	10.0.0.1	10.0.0.2	OpenFlow	188	Type: OFPT_PACKET_OUT
42	36.855813469	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_ECHO_REQUEST
43	36.855920769	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_ECHO_REPLY
45	37.097769868	1a:c6:cb:d9:4b:9f	62:4e:80:e2:a2:11	OpenFlow	126	Type: OFPT_PACKET_IN
46	37.097818048	62:4e:80:e2:a2:11	1a:c6:cb:d9:4b:9f	OpenFlow	126	Type: OFPT_PACKET_IN
47	37.097897948	127.0.0.1	127.0.0.1	OpenFlow	146	Type: OFPT_FLOW_MOD
49	37.097938697	1a:c6:cb:d9:4b:9f	62:4e:80:e2:a2:11	OpenFlow	132	Type: OFPT_PACKET_OUT
51	37.097948977	127.0.0.1	127.0.0.1	OpenFlow	146	Type: OFPT_FLOW_MOD
53	37.097954867	62:4e:80:e2:a2:11	1a:c6:cb:d9:4b:9f	OpenFlow	132	Type: OFPT_PACKET_OUT
55	37.098432475	62:4e:80:e2:a2:11	1a:c6:cb:d9:4b:9f	OpenFlow	126	Type: OFPT_PACKET_IN
56	37.098457835	1a:c6:cb:d9:4b:9f	62:4e:80:e2:a2:11	OpenFlow	126	Type: OFPT_PACKET_IN
57	37.098493454	127.0.0.1	127.0.0.1	OpenFlow	146	Type: OFPT_FLOW_MOD
59	37.098508643	62:4e:80:e2:a2:11	1a:c6:cb:d9:4b:9f	OpenFlow	132	Type: OFPT_PACKET_OUT
61	37.098517133	127.0.0.1	127.0.0.1	OpenFlow	146	Type: OFPT_FLOW_MOD
63	37.098522593	1a:c6:cb:d9:4b:9f	62:4e:80:e2:a2:11	OpenFlow	132	Type: OFPT_PACKET_OUT
65	42.100018503	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_ECHO_REQUEST
66	42.100126058	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_ECHO_REPLY
68	47.102344005	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_ECHO_REQUEST
69	47.102502125	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_ECHO_REPLY

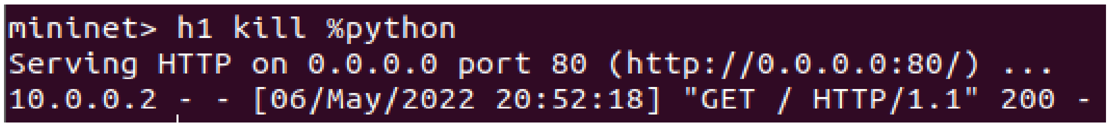
Q5. Create a simple HTTP server at h1 and connect h2 to this server.

```
mininet> h1 python -m http.server 80 &
mininet> h2 wget -O - h1
...
mininet> h1 kill %python
```

(* or "SimpleHTTPServer" instead of "http.server") if you get an error message!



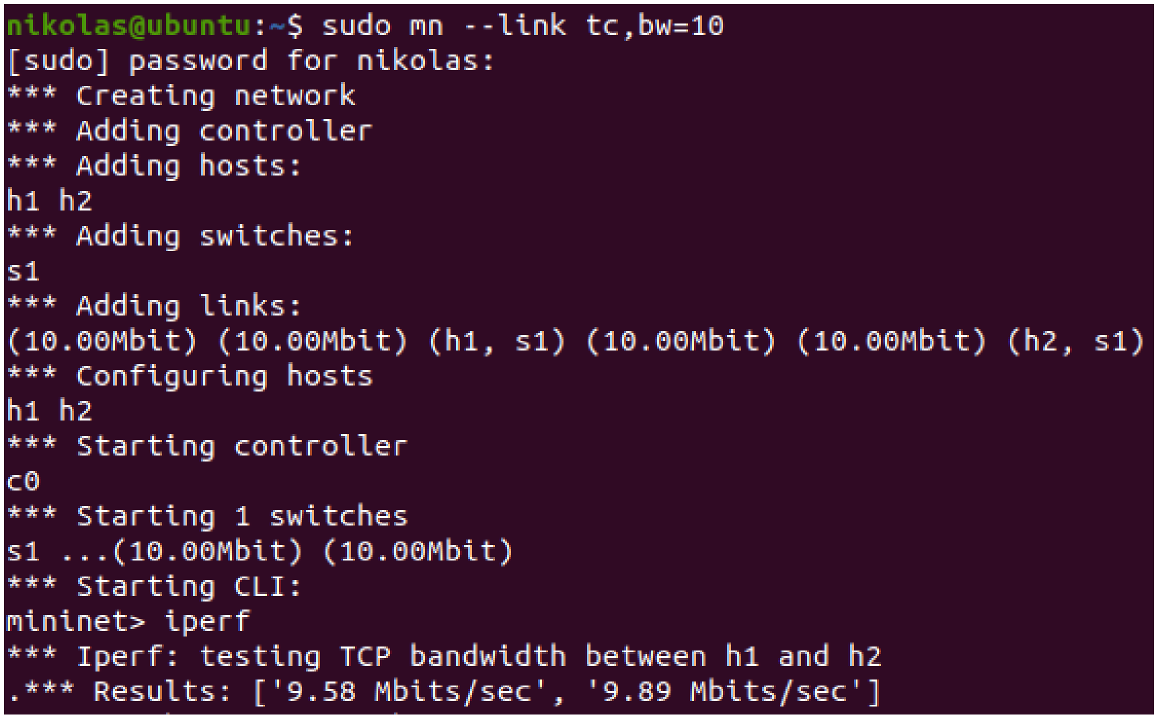
```
mininet> h1 python3 -m http.server 80 &
mininet> h2 wget -O - h1
--2022-05-06 20:39:17--  http://10.0.0.1/
Connecting to 10.0.0.1:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1399 (1.4K) [text/html]
Saving to: ‘STDOUT’

-                      0%[                    ]       0  --.-KB/s               <!DOCT
YPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Directory listing for /</title>
</head>
<body>
<h1>Directory listing for /</h1>
<hr>
<ul>
<li><a href=".bash_history">.bash_history</a></li>
<li><a href=".bash_logout">.bash_logout</a></li>
<li><a href=".bashrc">.bashrc</a></li>
<li><a href=".cache/">.cache/</a></li>
<li><a href=".config/">.config/</a></li>
<li><a href=".gnupg/">.gnupg/</a></li>
<li><a href=".local/">.local/</a></li>
<li><a href=".mozilla/">.mozilla/</a></li>
<li><a href=".profile">.profile</a></li>
<li><a href=".ssh/">.ssh/</a></li>
<li><a href=".sudo_as_admin_successful">.sudo_as_admin_successful</a></li>
<li><a href=".wireshark/">.wireshark/</a></li>
<li><a href="Desktop/">Desktop/</a></li>
<li><a href="Documents/">Documents/</a></li>
<li><a href="Downloads/">Downloads/</a></li>
<li><a href="mininet/">mininet/</a></li>
<li><a href="Music/">Music/</a></li>
<li><a href="oflops/">oflops/</a></li>
<li><a href="oftest/">oftest/</a></li>
<li><a href="openflow/">openflow/</a></li>
<li><a href="Pictures/">Pictures/</a></li>
<li><a href="pox/">pox/</a></li>
<li><a href="Public/">Public/</a></li>
<li><a href="snap/">snap/</a></li>
<li><a href="Templates/">Templates/</a></li>
<li><a href="Videos/">Videos/</a></li>
</ul>
<hr>
</body>
</html>
-                    100%[===================>]   1.37K  --.-KB/s    in 0s

2022-05-06 20:39:17 (383 MB/s) - written to stdout [1399/1399]
```



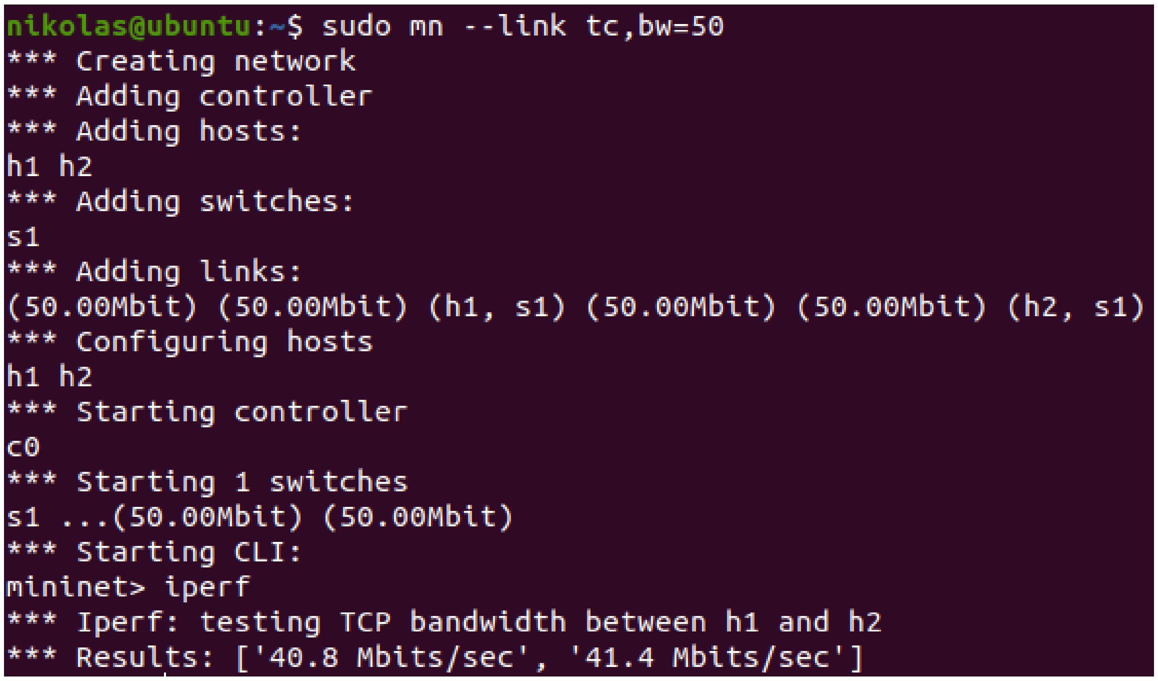
```
mininet> h1 kill %python
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.0.0.2 - - [06/May/2022 20:52:18] "GET / HTTP/1.1" 200 -
```

Q6. In the simplest topology conduct traffic control by setting bandwidth = 10Mbps – 50Mbps – 100Mbps and display the results using iperf.

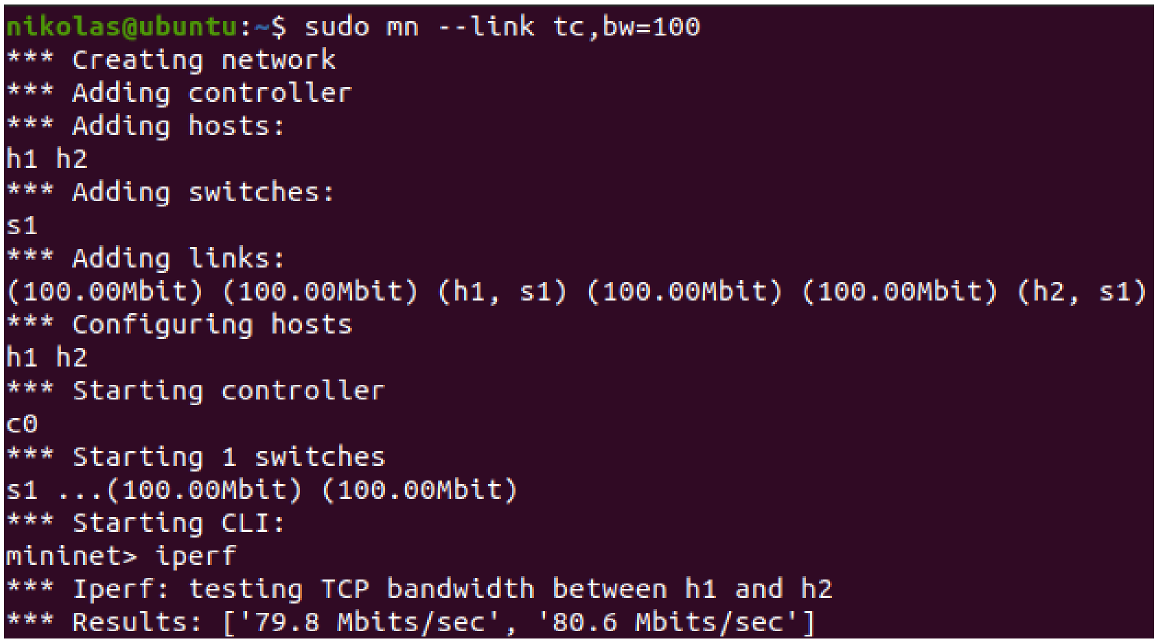
```
$ sudo mn --test iperf
```



```
nikolas@ubuntu:~$ sudo mn --link tc,bw=10
[sudo] password for nikolas:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(10.00Mbit) (10.00Mbit) (h1, s1) (10.00Mbit) (10.00Mbit) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...(10.00Mbit) (10.00Mbit)
*** Starting CLI:
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h2
.*** Results: ['9.58 Mbits/sec', '9.89 Mbits/sec']
```

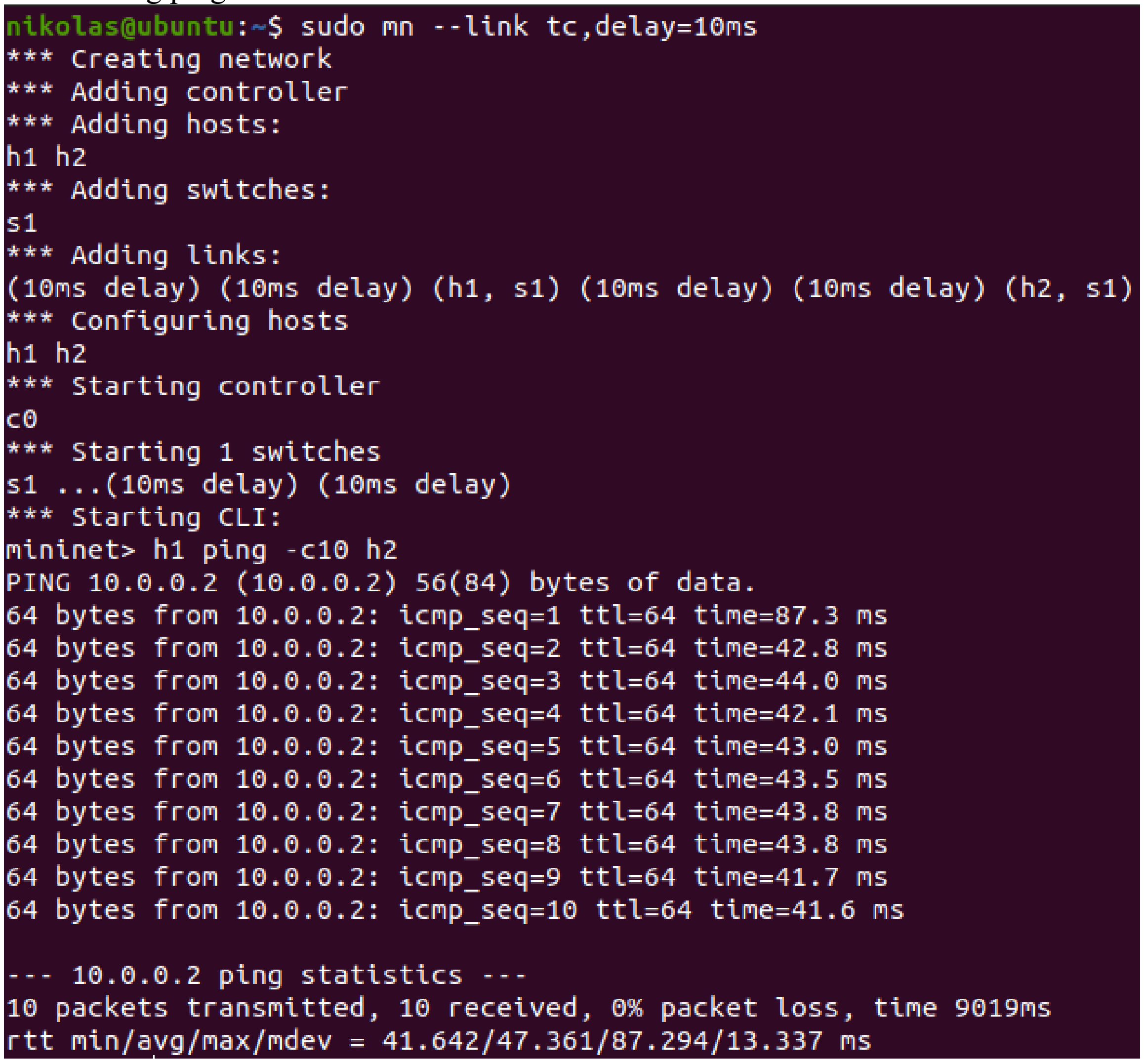


```
nikolas@ubuntu:~$ sudo mn --link tc,bw=50
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(50.00Mbit) (50.00Mbit) (h1, s1) (50.00Mbit) (50.00Mbit) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...(50.00Mbit) (50.00Mbit)
*** Starting CLI:
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['40.8 Mbits/sec', '41.4 Mbits/sec']
```

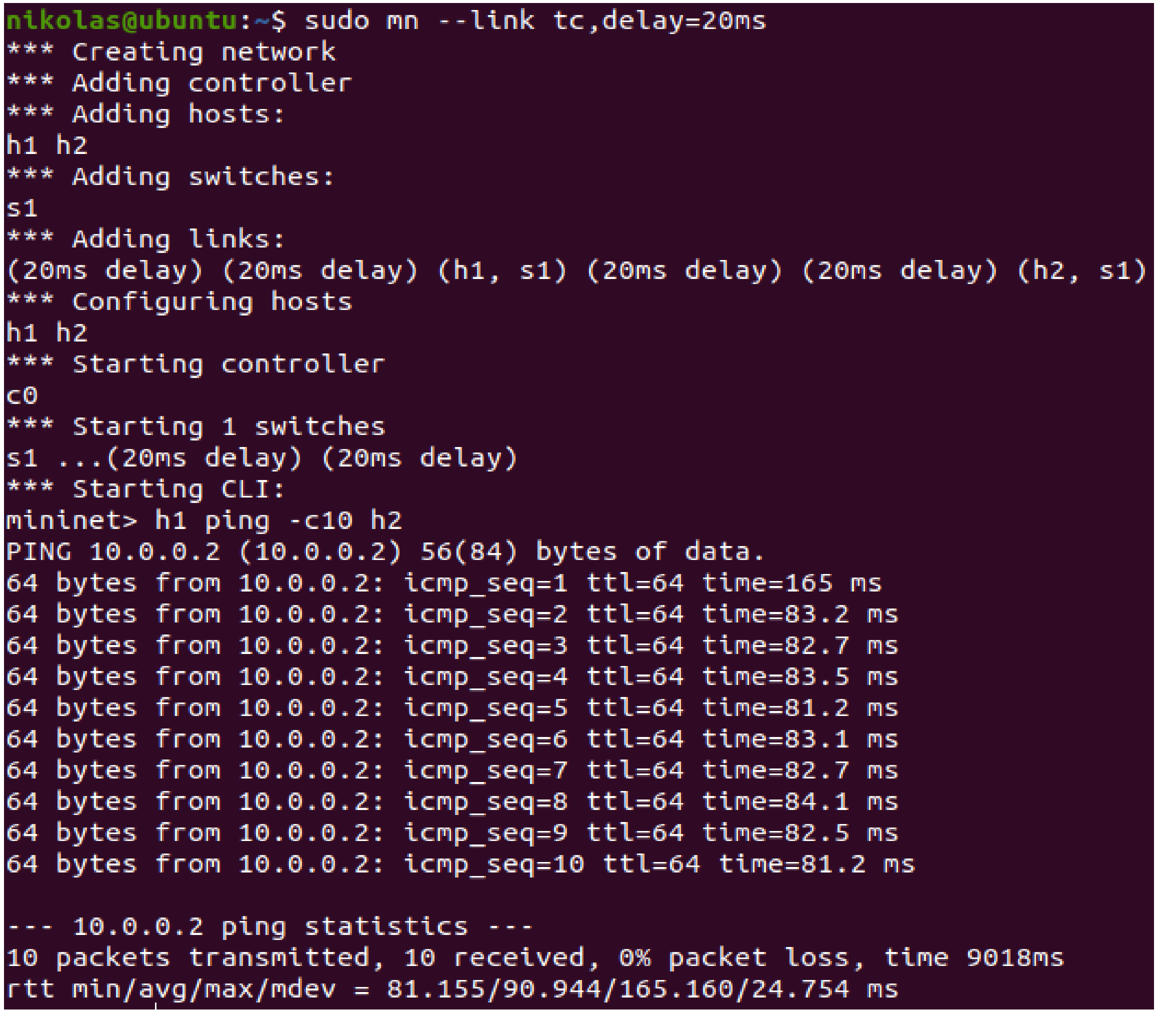


```
nikolas@ubuntu:~$ sudo mn --link tc,bw=100
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(100.00Mbit) (100.00Mbit) (h1, s1) (100.00Mbit) (100.00Mbit) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...(100.00Mbit) (100.00Mbit)
*** Starting CLI:
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['79.8 Mbits/sec', '80.6 Mbits/sec']
```

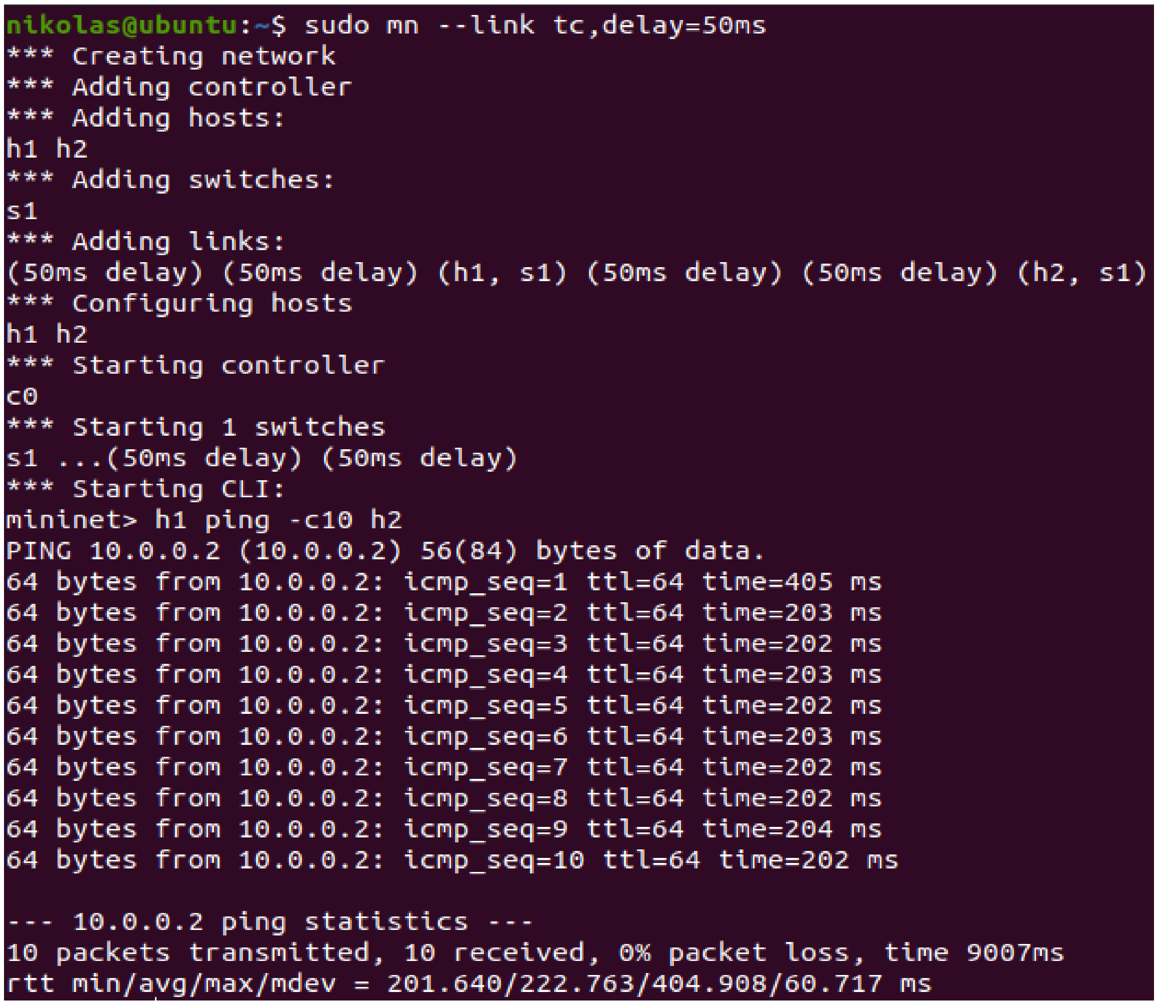
Then, control the delay of all links by setting delay = 10ms – 20ms – 50ms and log the results using ping.



```
nikolas@ubuntu:~$ sudo mn --link tc,delay=10ms
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(10ms delay) (10ms delay) (h1, s1) (10ms delay) (10ms delay) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...(10ms delay) (10ms delay)
*** Starting CLI:
mininet> h1 ping -c10 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=87.3 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=42.8 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=44.0 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=42.1 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=43.0 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=43.5 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=43.8 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=43.8 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=41.7 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=41.6 ms

--- 10.0.0.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9019ms
rtt min/avg/max/mdev = 41.642/47.361/87.294/13.337 ms
```



```
nikolas@ubuntu:~$ sudo mn --link tc,delay=20ms
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(20ms delay) (20ms delay) (h1, s1) (20ms delay) (20ms delay) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...(20ms delay) (20ms delay)
*** Starting CLI:
mininet> h1 ping -c10 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=165 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=83.2 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=82.7 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=83.5 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=81.2 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=83.1 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=82.7 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=84.1 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=82.5 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=81.2 ms

--- 10.0.0.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9018ms
rtt min/avg/max/mdev = 81.155/90.944/165.160/24.754 ms
```

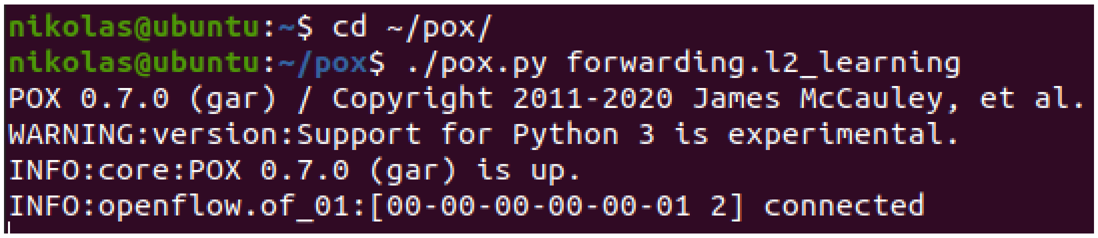


```
nikolas@ubuntu:~$ sudo mn --link tc,delay=50ms
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(50ms delay) (50ms delay) (h1, s1) (50ms delay) (50ms delay) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...(50ms delay) (50ms delay)
*** Starting CLI:
mininet> h1 ping -c10 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=405 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=203 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=202 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=203 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=202 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=203 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=202 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=202 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=204 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=202 ms

--- 10.0.0.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9007ms
rtt min/avg/max/mdev = 201.640/222.763/404.908/60.717 ms
```

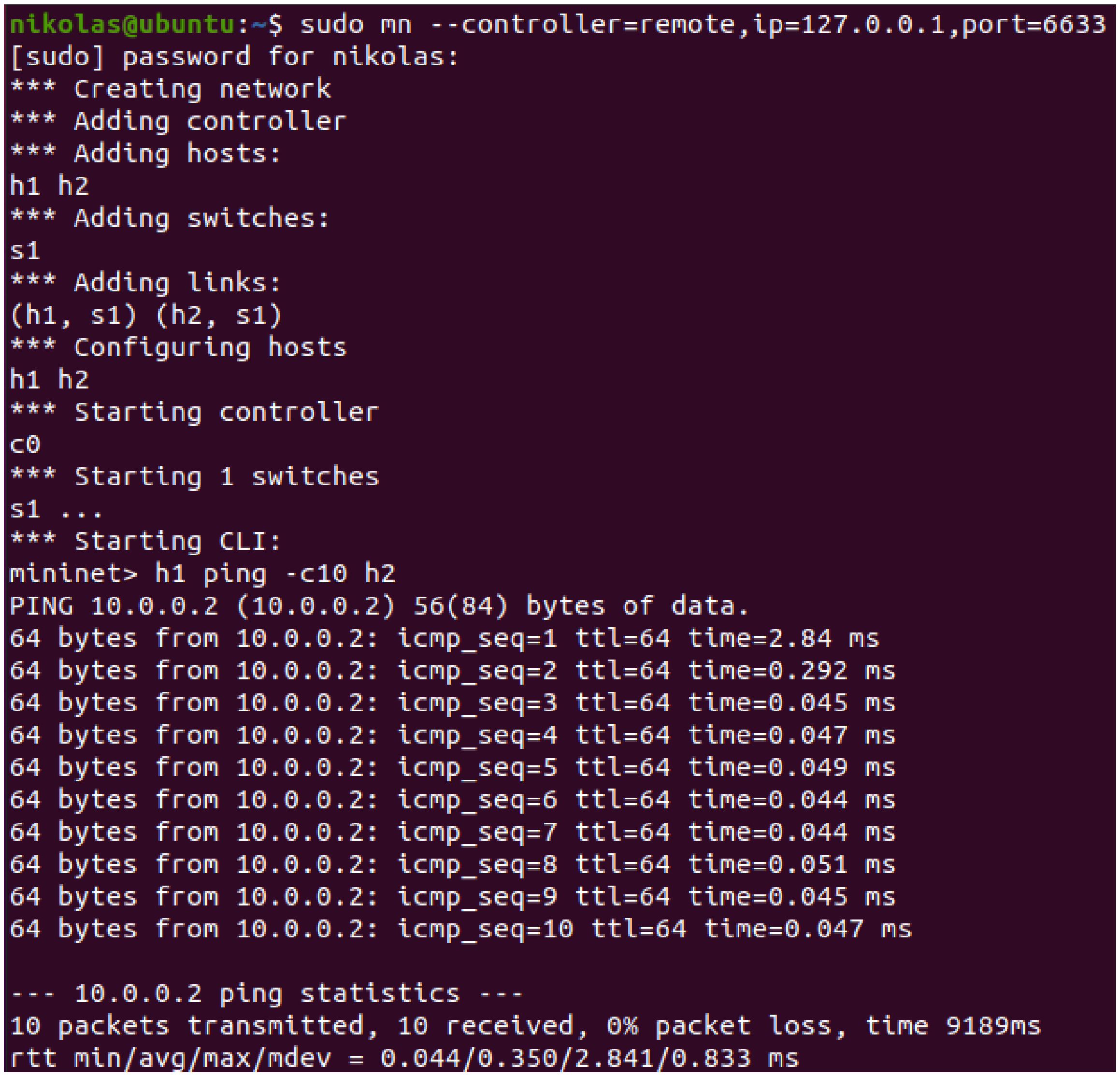
Q7. Prove the POX controller is running and is connected to Mininet:

```
$ sudo mn --controller=remote,ip=127.0.0.1,port=6633
```

```
$ cd ~/pox
$ ./pox.py forwarding.l2_learning
```



```
nikolas@ubuntu:~$ cd ~/pox/
nikolas@ubuntu:~/pox$ ./pox.py forwarding.l2_learning
POX 0.7.0 (gar) / Copyright 2011-2020 James McCauley, et al.
WARNING:version:Support for Python 3 is experimental.
INFO:core:POX 0.7.0 (gar) is up.
INFO:openflow.of_01:[00-00-00-00-00-01 2] connected
```

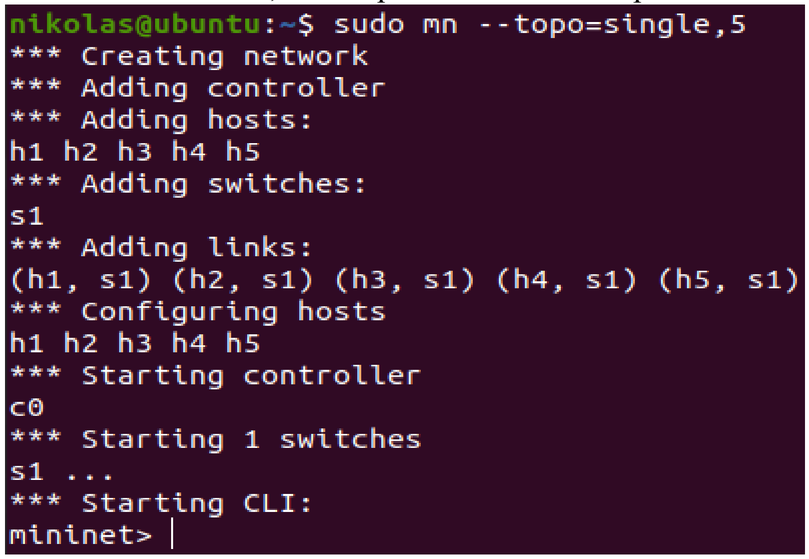


```
nikolas@ubuntu:~$ sudo mn --controller=remote,ip=127.0.0.1,port=6633
[sudo] password for nikolas:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> h1 ping -c10 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=2.84 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.292 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.045 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.047 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.049 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.044 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.044 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.051 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.045 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.047 ms

--- 10.0.0.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9189ms
rtt min/avg/max/mdev = 0.044/0.350/2.841/0.833 ms
```

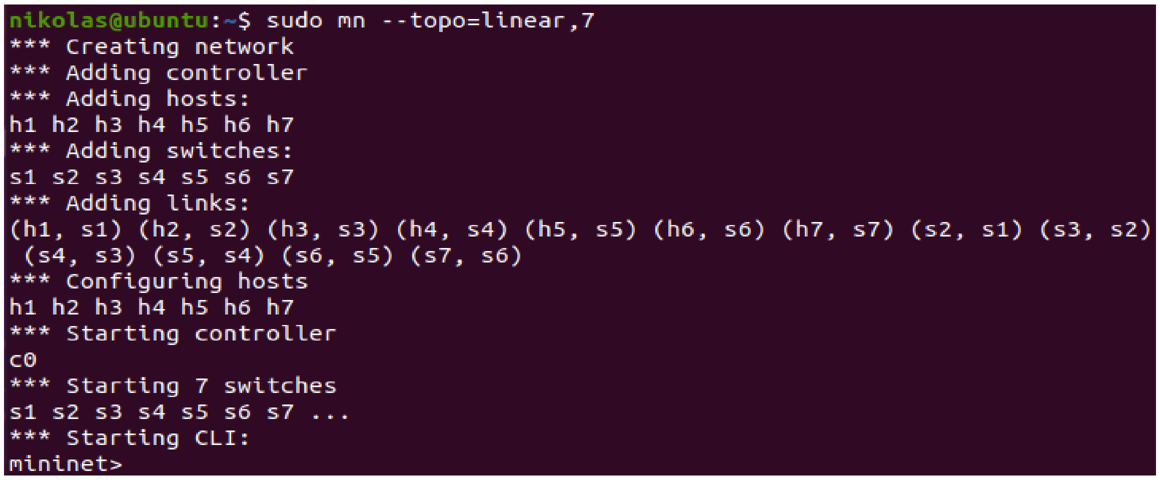
Q8. Using Mininet, create the following topologies:

- Single, with 5 hosts
- Linear, with 7 hosts
- Tree, with 27 hosts

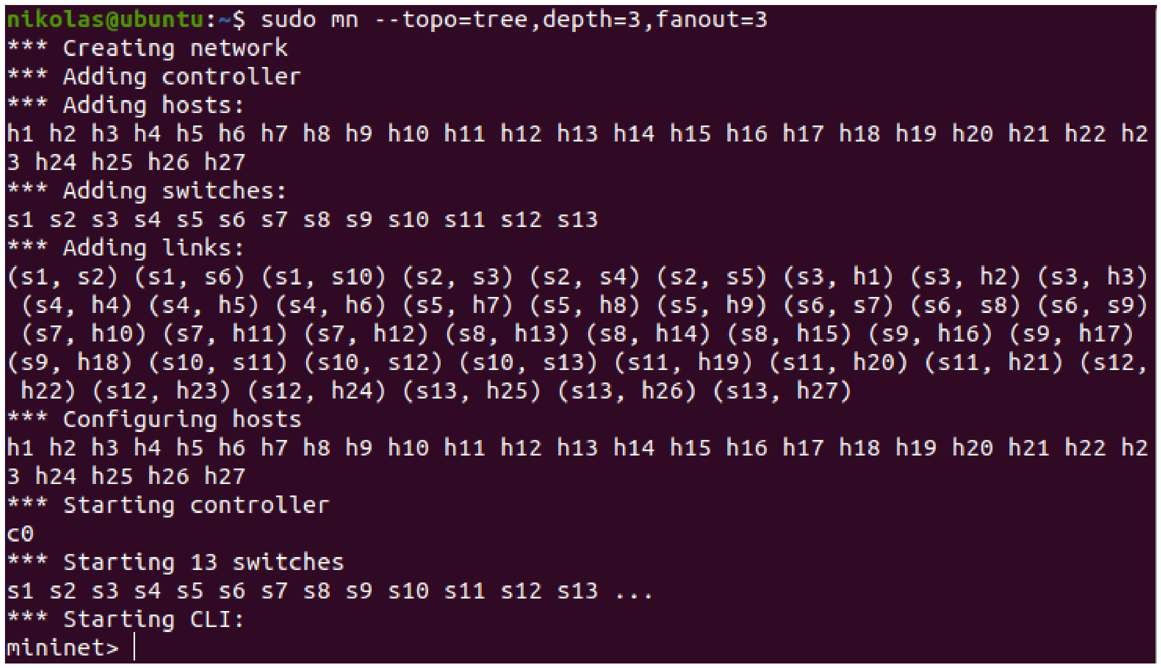
Show the commands, and respective terminal outputs.



```
nikolas@ubuntu:~$ sudo mn --topo=single,5
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1)
*** Configuring hosts
h1 h2 h3 h4 h5
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

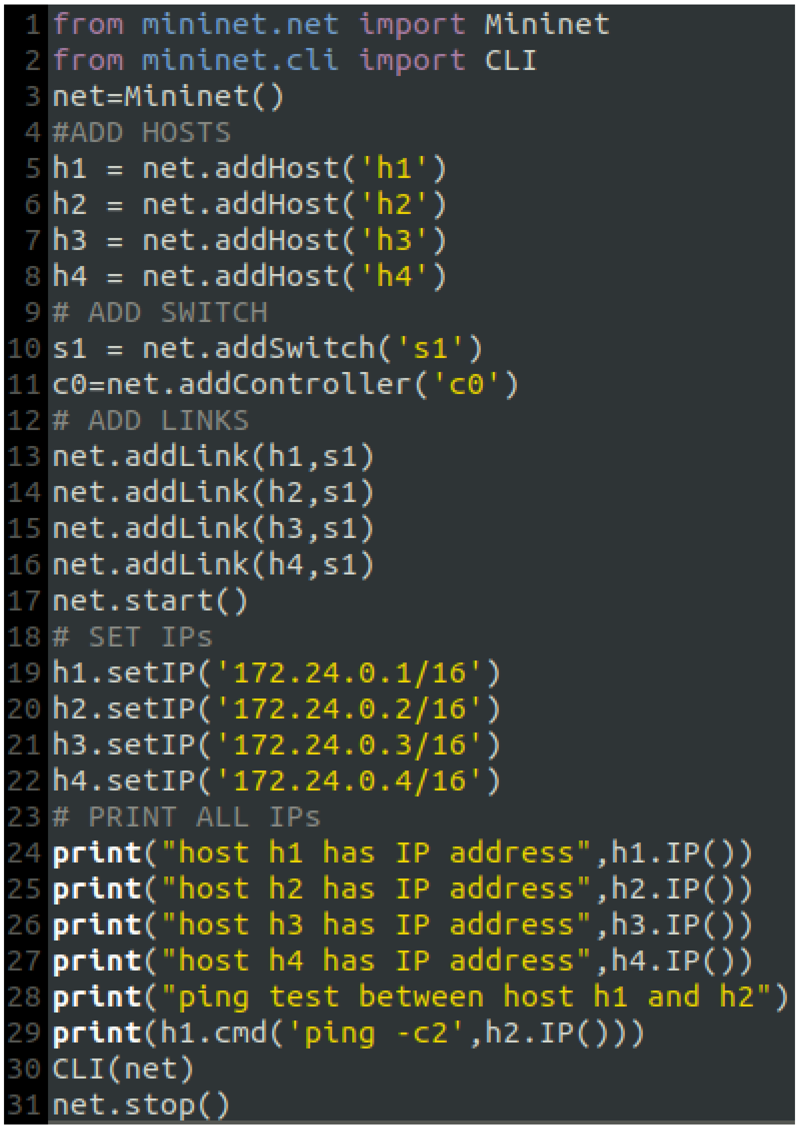


```
nikolas@ubuntu:~$ sudo mn --topo=linear,7
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (h5, s5) (h6, s6) (h7, s7) (s2, s1) (s3, s2)
 (s4, s3) (s5, s4) (s6, s5) (s7, s6)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7
*** Starting controller
c0
*** Starting 7 switches
s1 s2 s3 s4 s5 s6 s7 ...
*** Starting CLI:
mininet>
```

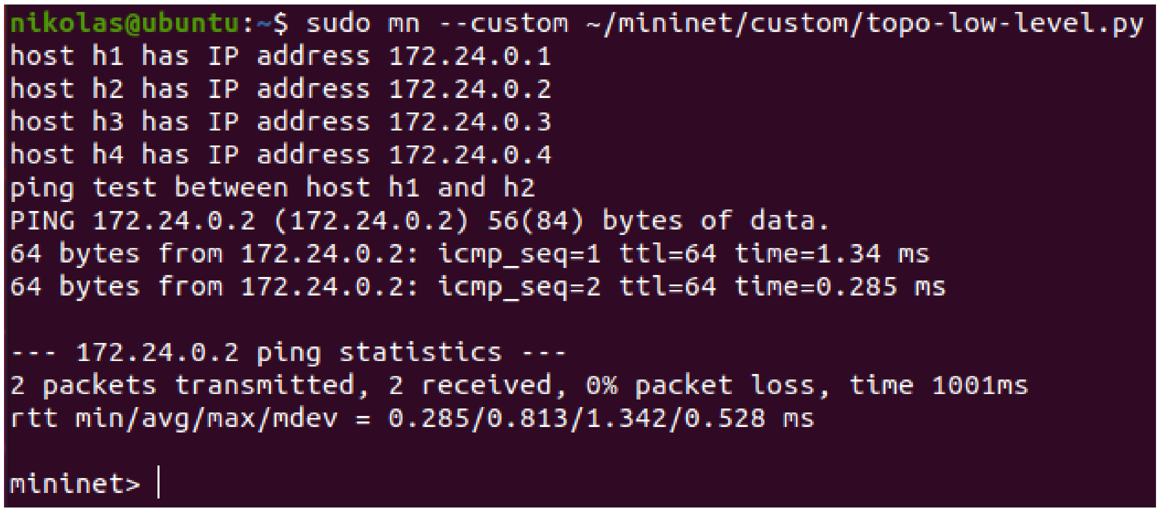


```
nikolas@ubuntu:~$ sudo mn --topo=tree,depth=3,fanout=3
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h2
3 h24 h25 h26 h27
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 s13
*** Adding links:
(s1, s2) (s1, s6) (s1, s10) (s2, s3) (s2, s4) (s2, s5) (s3, h1) (s3, h2) (s3, h3)
 (s4, h4) (s4, h5) (s4, h6) (s5, h7) (s5, h8) (s5, h9) (s6, s7) (s6, s8) (s6, s9)
 (s7, h10) (s7, h11) (s7, h12) (s8, h13) (s8, h14) (s8, h15) (s9, h16) (s9, h17)
(s9, h18) (s10, s11) (s10, s12) (s10, s13) (s11, h19) (s11, h20) (s11, h21) (s12,
 h22) (s12, h23) (s12, h24) (s13, h25) (s13, h26) (s13, h27)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h2
3 h24 h25 h26 h27
*** Starting controller
c0
*** Starting 13 switches
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 s13 ...
*** Starting CLI:
mininet>
```

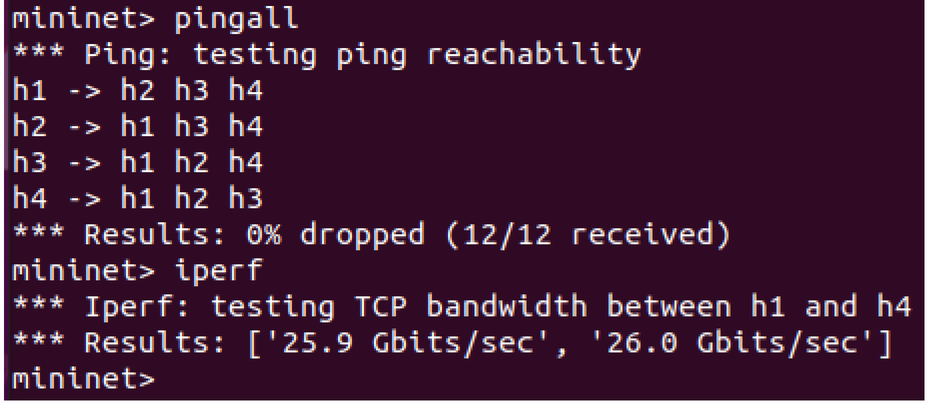
Q9. Add any missing lines of Python code into the file “custom-topo.py”, in order to create the following custom topology. Use the Mininet Python API reference manual to find the necessary functions: http://mininet.org/api/classmininet_1_1net_1_1Mininet.html
Then use the following command to run it and show the output:
sudo mn --custom ~/mininet/custom/topo-low-level.py
Also perform pingall and iperf tests.



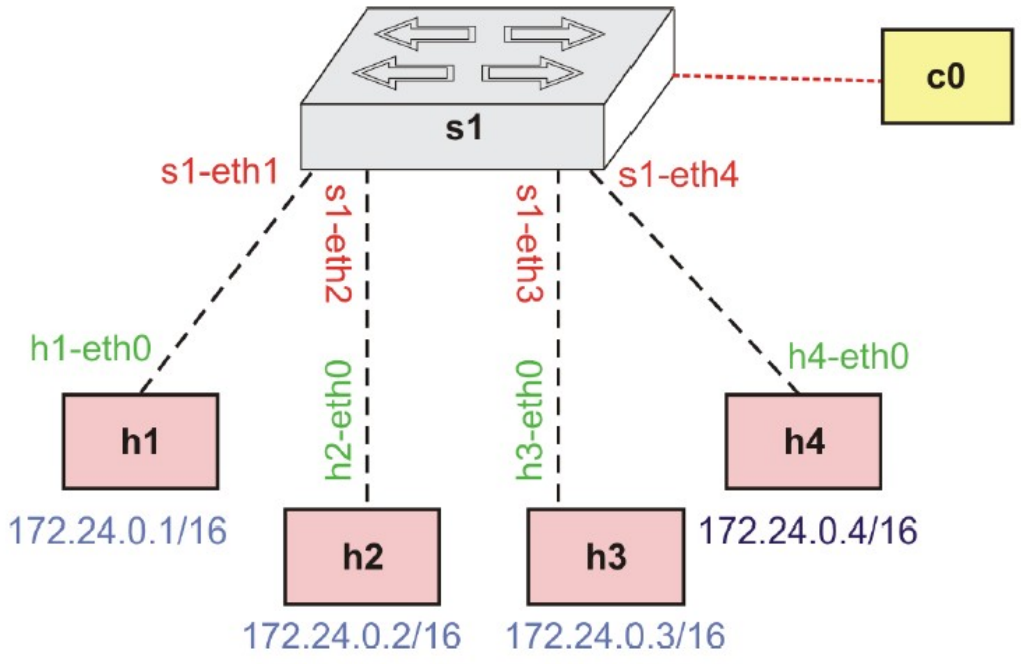
```python
from mininet.net import Mininet
from mininet.cli import CLI
net=Mininet()
#ADD HOSTS
h1 = net.addHost('h1')
h2 = net.addHost('h2')
h3 = net.addHost('h3')
h4 = net.addHost('h4')
# ADD SWITCH
s1 = net.addSwitch('s1')
c0=net.addController('c0')
# ADD LINKS
net.addLink(h1,s1)
net.addLink(h2,s1)
net.addLink(h3,s1)
net.addLink(h4,s1)
net.start()
# SET IPs
h1.setIP('172.24.0.1/16')
h2.setIP('172.24.0.2/16')
h3.setIP('172.24.0.3/16')
h4.setIP('172.24.0.4/16')
# PRINT ALL IPs
print("host h1 has IP address",h1.IP())
print("host h2 has IP address",h2.IP())
print("host h3 has IP address",h3.IP())
print("host h4 has IP address",h4.IP())
print("ping test between host h1 and h2")
print(h1.cmd('ping -c2',h2.IP()))
CLI(net)
net.stop()
```



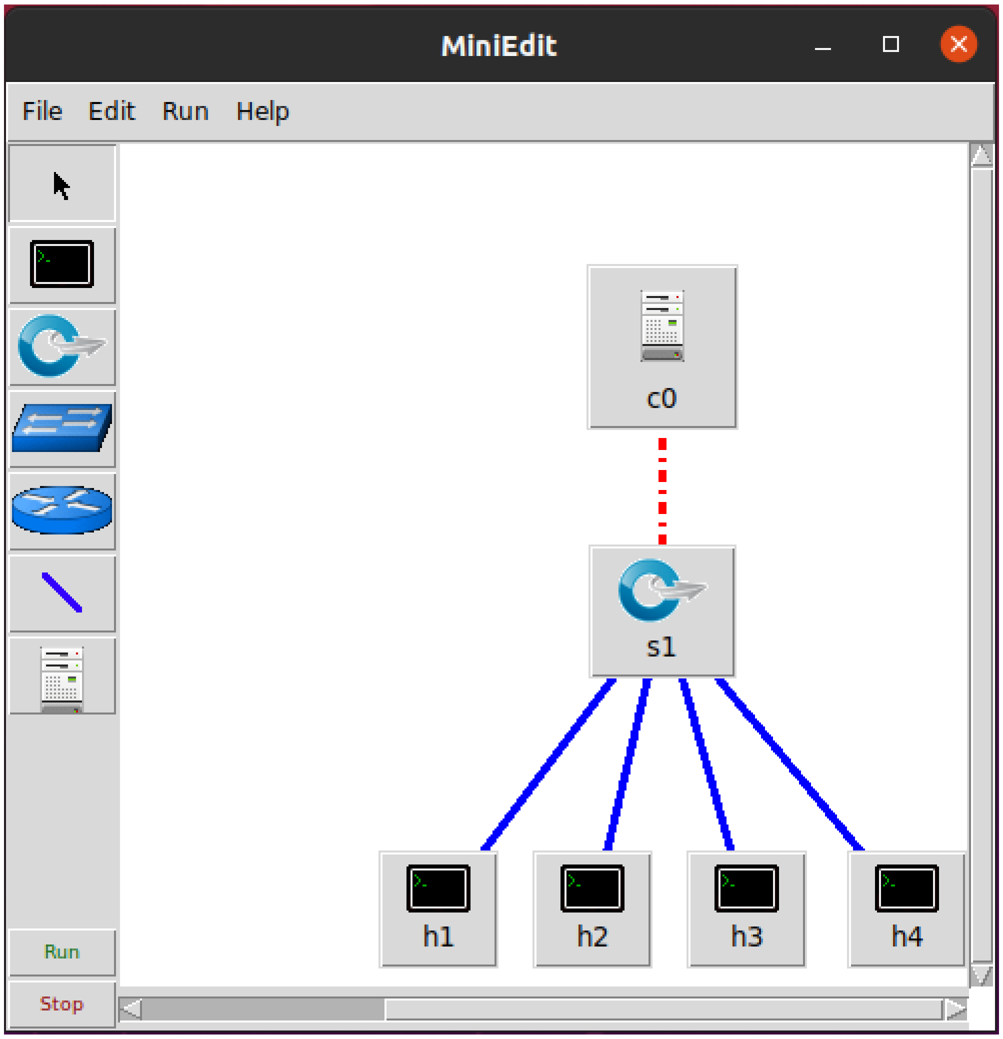
```
nikolas@ubuntu:~$ sudo mn --custom ~/mininet/custom/topo-low-level.py
host h1 has IP address 172.24.0.1
host h2 has IP address 172.24.0.2
host h3 has IP address 172.24.0.3
host h4 has IP address 172.24.0.4
ping test between host h1 and h2
PING 172.24.0.2 (172.24.0.2) 56(84) bytes of data.
64 bytes from 172.24.0.2: icmp_seq=1 ttl=64 time=1.34 ms
64 bytes from 172.24.0.2: icmp_seq=2 ttl=64 time=0.285 ms

--- 172.24.0.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.285/0.813/1.342/0.528 ms

mininet> 
```

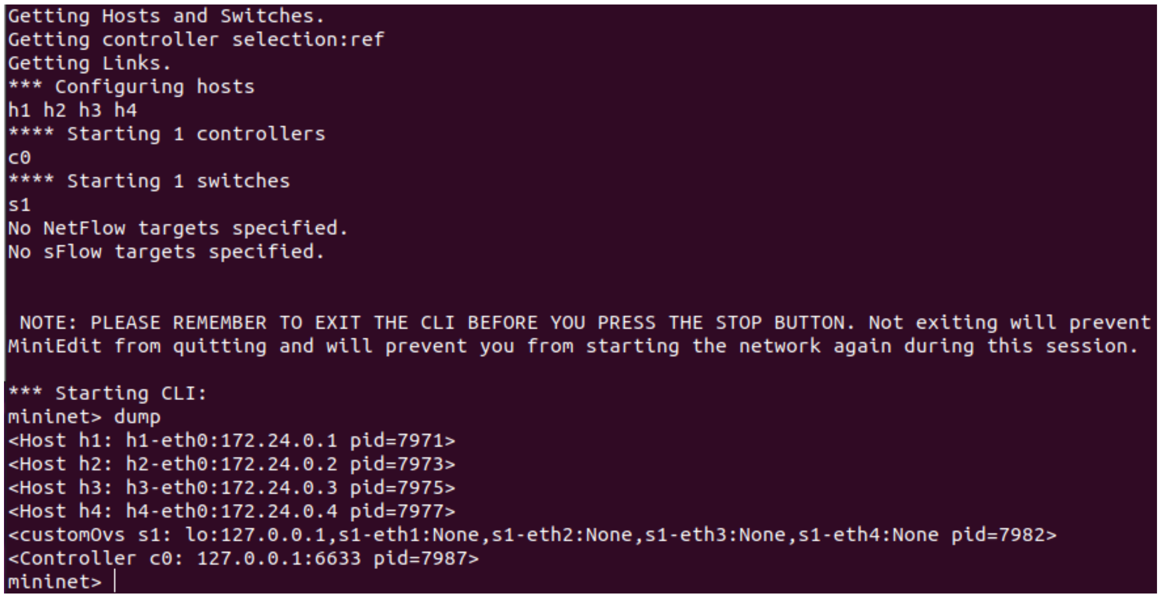


```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h4
*** Results: ['25.9 Gbits/sec', '26.0 Gbits/sec']
mininet>
```



Q10. Now we will work with MiniEdit. Refer to the manual: https://www.brianlinkletter.com/2015/04/how-to-use-miniedit-mininets-graphical-user-interface/
Run:
sudo ~/mininet/examples/miniedit.py
Using MiniEdit, create the same topology as before (including IP addresses at hosts). Don't forget to add the Controller! Note that you may use the Edit -> Properties -> IP Base option. Then "Run" this topology and present here both the MiniEdit window and the Mininet output. Also run any command that shows the configured Ips.





```
Getting Hosts and Switches.
Getting controller selection:ref
Getting Links.
*** Configuring hosts
h1 h2 h3 h4
**** Starting 1 controllers
c0
**** Starting 1 switches
s1
No NetFlow targets specified.
No sFlow targets specified.


 NOTE: PLEASE REMEMBER TO EXIT THE CLI BEFORE YOU PRESS THE STOP BUTTON. Not exiting will prevent
MiniEdit from quitting and will prevent you from starting the network again during this session.

*** Starting CLI:
mininet> dump
<Host h1: h1-eth0:172.24.0.1 pid=7971>
<Host h2: h2-eth0:172.24.0.2 pid=7973>
<Host h3: h3-eth0:172.24.0.3 pid=7975>
<Host h4: h4-eth0:172.24.0.4 pid=7977>
<customOvs s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None,s1-eth4:None pid=7982>
<Controller c0: 127.0.0.1:6633 pid=7987>
mininet> |
```