

1. See attached python file. Topology created with appropriate nodes.

```
mininet@mininet-vm:~$ sudo python ~/yshi81-lab3.py
mininet> nodes
available nodes are:
c0 h1 h2 h3 h4 s1 ser1 ser2 ser3
mininet> 
```

```
mininet@mininet-vm: ~
File Edit Tabs Help
mininet@mininet-vm:~$ date
Sun Nov  7 04:32:52 PST 2021
mininet@mininet-vm:~$ 
```

- 2.

```
mininet@mininet-vm: ~
File Edit Tabs Help
Serving HTTP on 0.0.0.0 port 80 ...
mininet> ser1 wget 20.1.1.1:80
--2021-11-08 18:45:21-- http://20.1.1.1/
Connecting to 20.1.1.1:80... connected.
20.1.1.1 - - [08/Nov/2021 18:45:21] "GET / HTTP/1.1" 200 -
HTTP request sent, awaiting response... 200 OK
Length: 1932 (1.9K) [text/html]
Saving to: 'index.html'

100%[=====>] 1,932      --.-K/s   in 0s

2021-11-08 18:45:21 (519 MB/s) - 'index.html' saved [1932/1932]

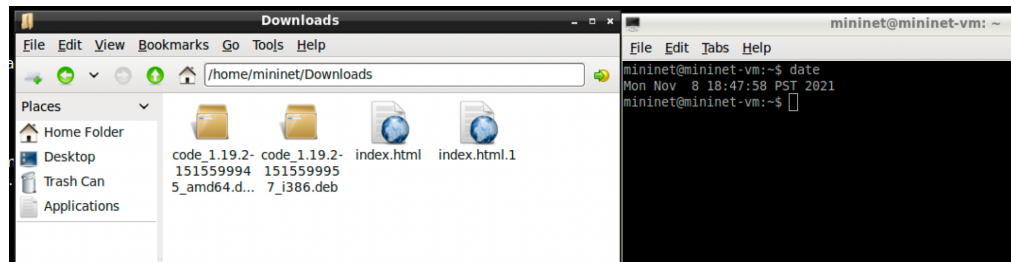
mininet> 
```

```
mininet@mininet-vm: ~
File Edit Tabs Help
--2021-11-08 18:46:00-- http://20.1.1.1/
Connecting to 20.1.1.1:80... ^C
mininet> h2 python -m SimpleHTTPServer 80 &
Serving HTTP on 0.0.0.0 port 80 ...
mininet> ser2 wget 20.1.1.2:80
--2021-11-08 18:46:45-- http://20.1.1.2/
Connecting to 20.1.1.2:80... failed: Connection refused.
mininet> ser1 wget 20.1.1.1:80
--2021-11-08 18:47:03-- http://20.1.1.1/
Connecting to 20.1.1.1:80... connected.
20.1.1.1 - - [08/Nov/2021 18:47:03] "GET / HTTP/1.1" 200 -
HTTP request sent, awaiting response... 200 OK
Length: 1932 (1.9K) [text/html]
Saving to: 'index.html.1'

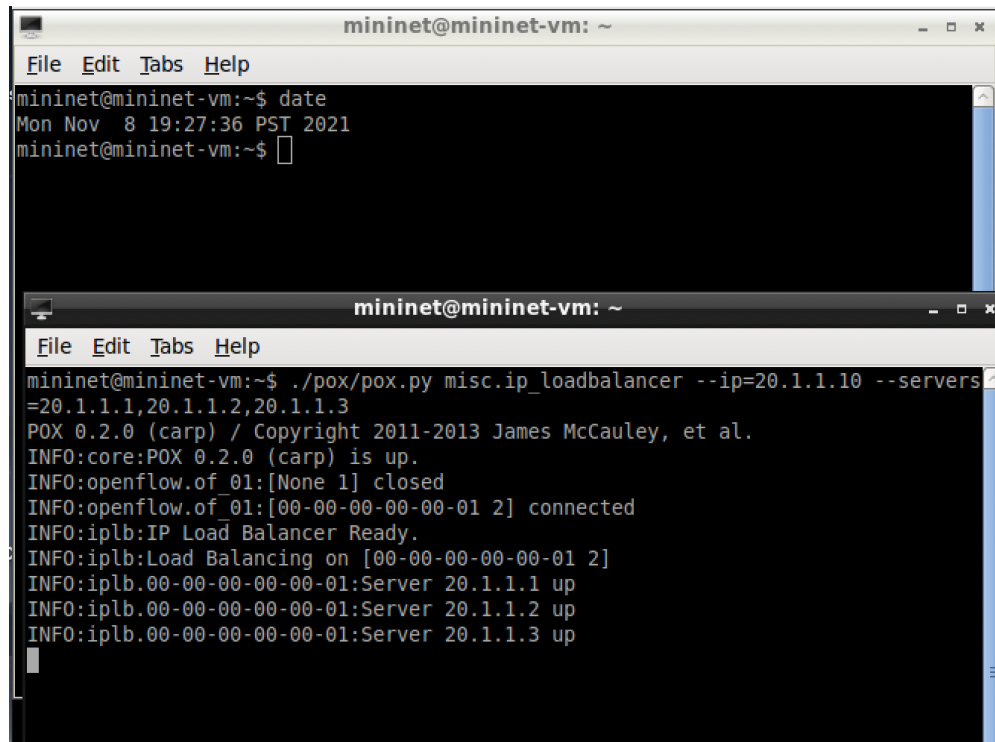
100%[=====>] 1,932      --.-K/s   in 0s

2021-11-08 18:47:03 (616 MB/s) - 'index.html.1' saved [1932/1932]

mininet> 
```

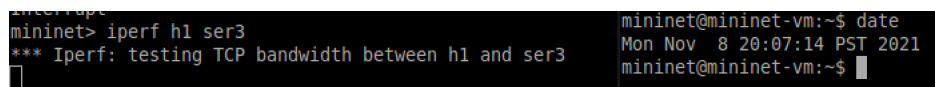


3. Remote controller terminal output.

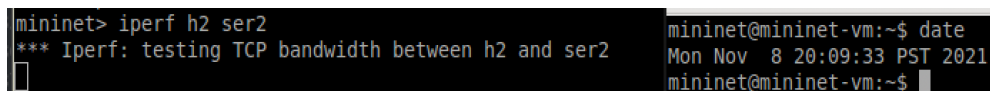


From this output after subsequently running “yshi81-lab3.py”, the load balance is up and ready. Load balancing is on the mac address “00-00-00-00-00-01 2” which distributes tasks onto the three servers. The last two lines indicate all three of the servers are up and running.

4. Iperf client 1 and server 3 command



Iperf client 2 server 2 command



Both iperf commands ran indefinitely or never ended, so I manually ended the testing after 20 seconds. Iperf command measures the TCP bandwidth between hosts. In this case, the VM kept trying to test the bandwidth between client 1 and server 3 indefinitely and also between

client 2 and server 2 indefinitely. The first command should be running indefinitely because the table does not specify source with client 1 and destination with server 3. The second command should be also running indefinitely because the table does not specify source with client 2 and destination with server 2 (only source with server 2 and destination with client 2). Therefore, there are no TCP packets to be measured for the bandwidth for these two commands.

5. Pingall command

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 ser1 ser2 ser3
h2 -> h1 h3 h4 ser1 ser2 ser3
h3 -> h1 h2 h4 ser1 ser2 ser3
h4 -> h1 h2 h3 ser1 ser2 ser3
ser1 -> h1 h2 h3 h4 ser2 ser3
ser2 -> h1 h2 h3 h4 ser1 ser3
ser3 -> h1 h2 h3 h4 ser1 ser2
*** Results: 0% dropped (42/42 received)
mininet>
```

```
mininet@mininet-vm:~$ date
Mon Nov  8 20:03:50 PST 2021
mininet@mininet-vm:~$
```

Pingall shows the connectivity between the clients and servers. They are all able to connect and reach each rest of the network hosts. To support this, no packets are dropped. If a host to host connection would be unreachable, there would be packets dropped.

6. The commands “msg.idle_timeout” and “msg.hard_timeout” help in installing the rules in a switch (l2_learning) and making them remembered. They can be modified by setting them to milliseconds for how long an entry stays in the switch. In my code for the firewall, I set them to 50ms.