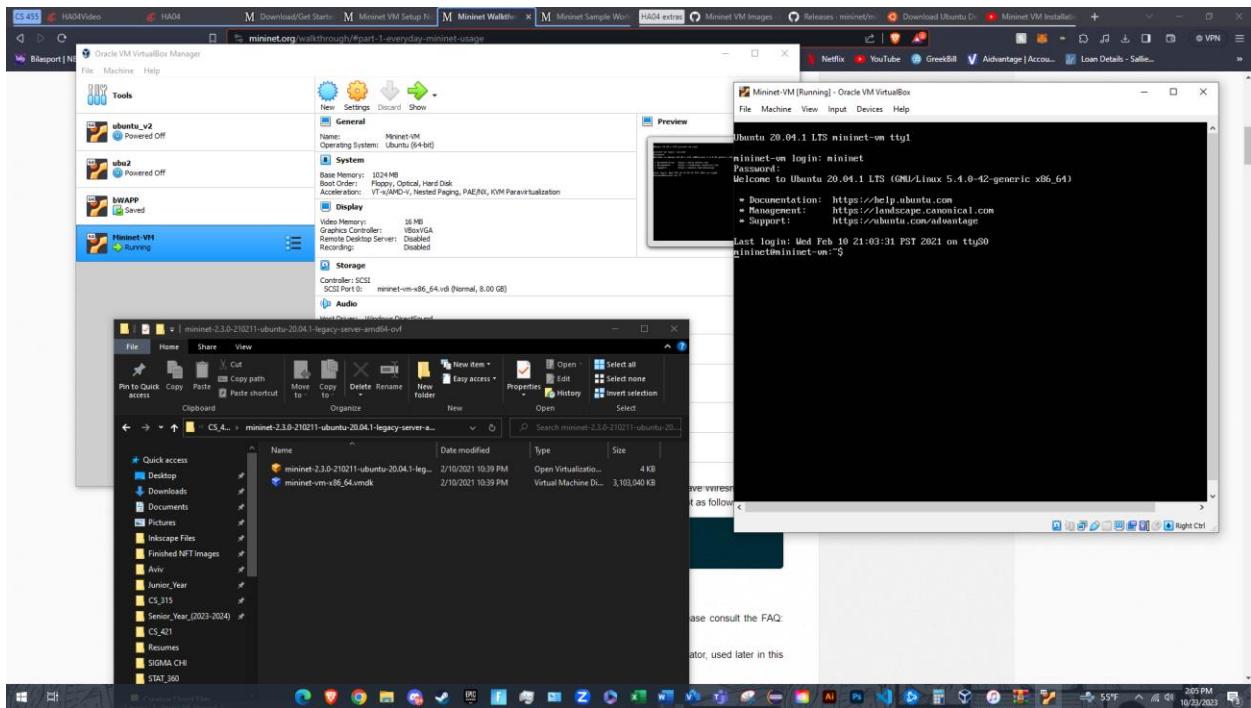


Aviv Y
Jillepalli
HA04
CS 455

NOTE: Explanation of simpleperf.py and test_simpleperf.py can be found on [Page 23](#)

Installation:

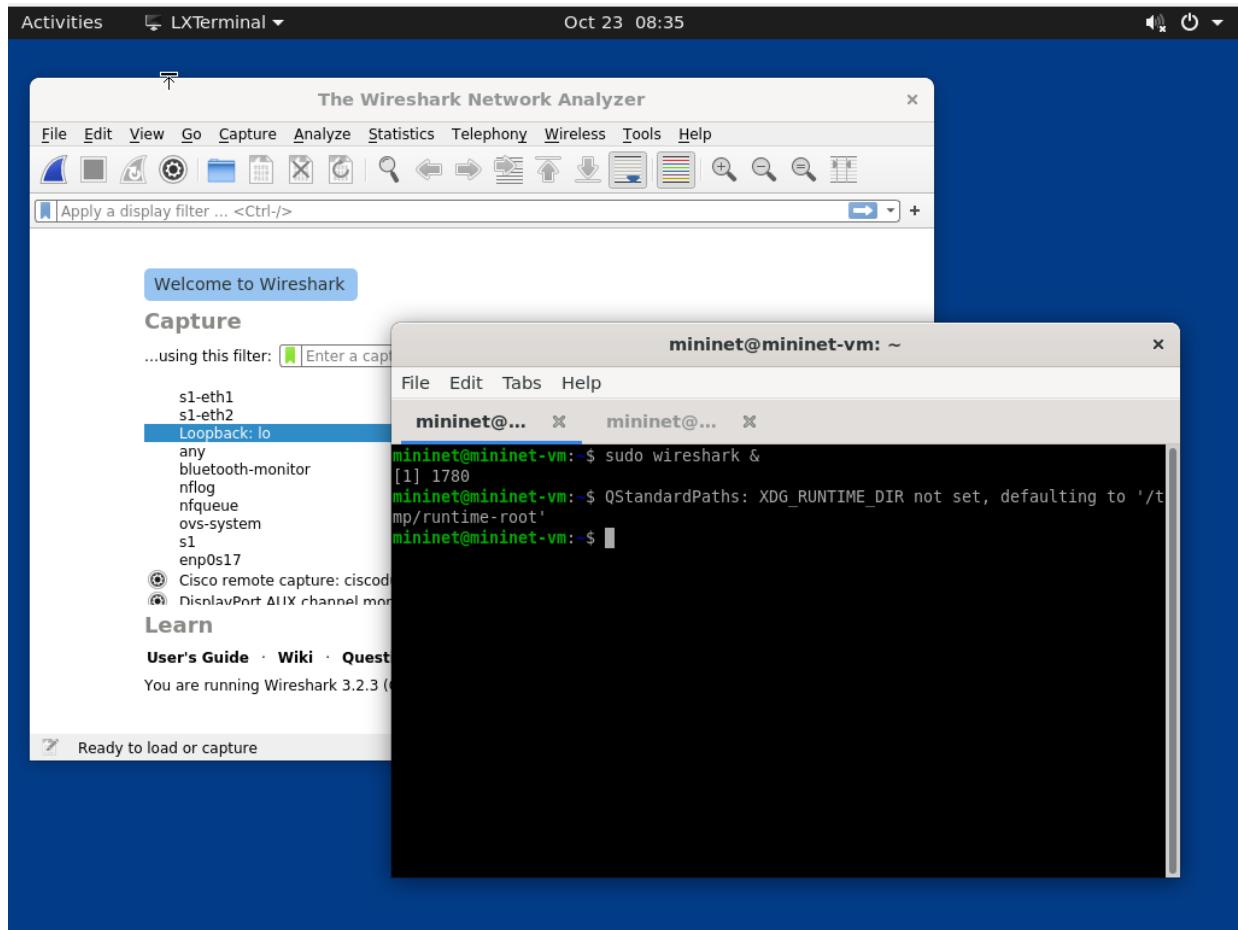
- Ran into issues while doing the default installation.
- Used a command to install the GUI which fixed the overall issue regarding Wireshark.
 - o **Sudo apt-get update & apt-get install xinit lxde virtualbox-guest-dkms**
 - IF you get this error:
 - **Unable to acquire the dpkg frontend lock**
 - o Run these commands:
 - **Sudo apt get update**
 - **Sudo apt-get install xinit lxde virtualbox-guest-dkms**



SC 1: Shows successful Mininet installation.

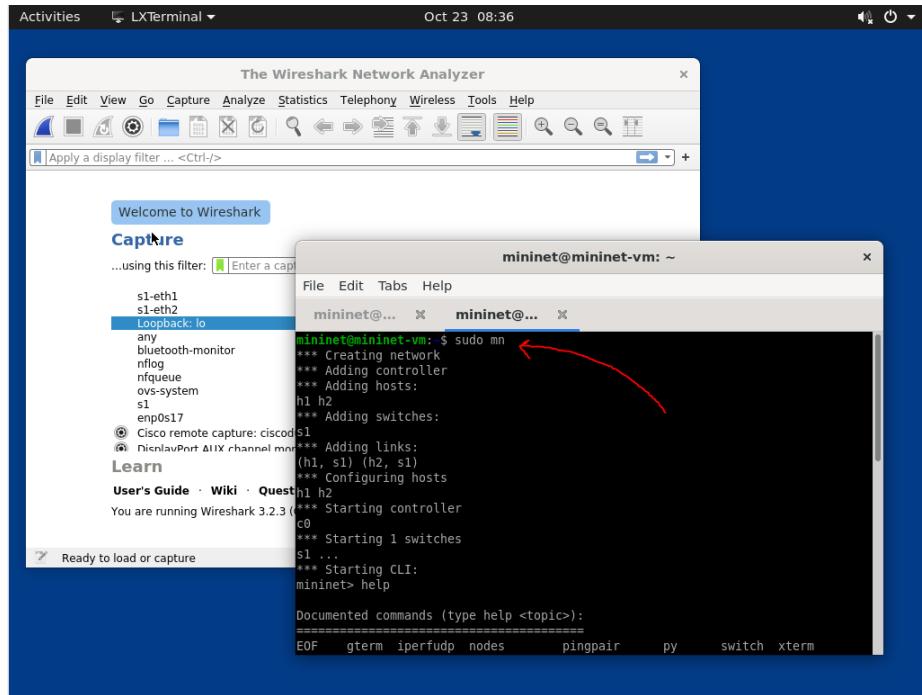
Aviv Y
Jillepalli
HA04
CS 455

Part 1: Everyday Mininet Usage

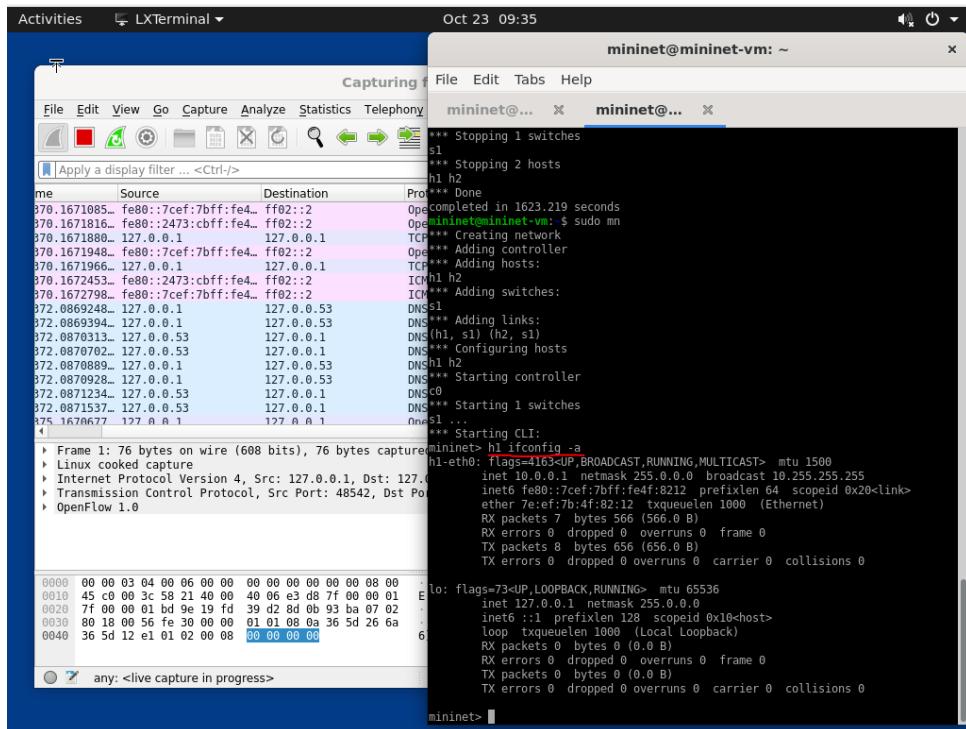


SC 2: Successfully Opening Wireshark.

Aviv Y
Jillepalli
HA04
CS 455

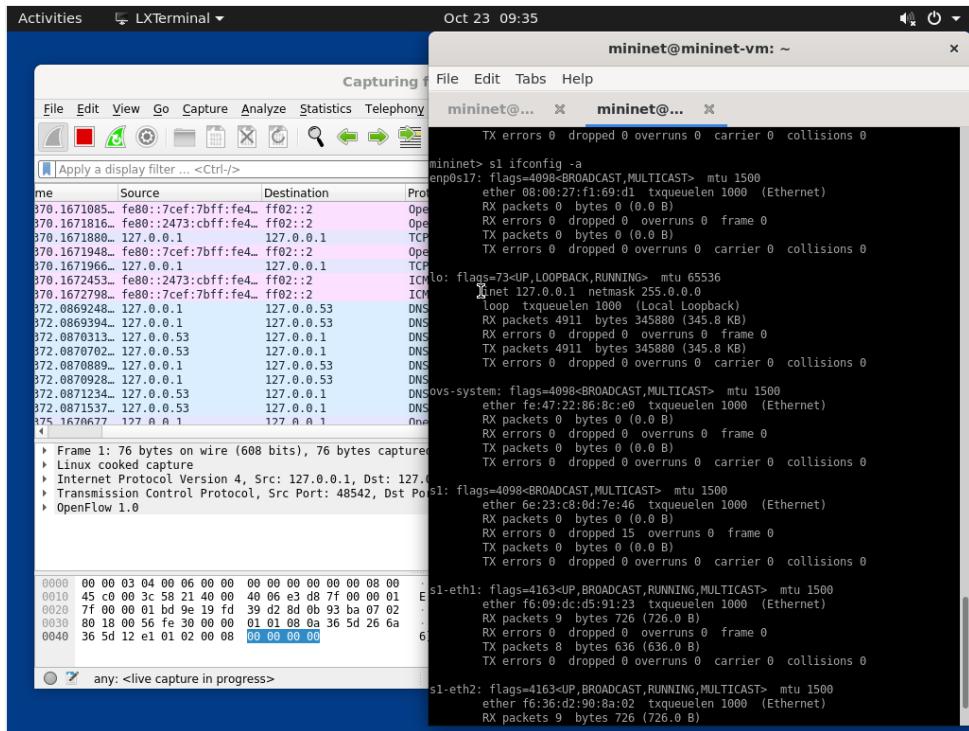


SC 3: Successfully creating minimal topology (1 OpenFlow switch + 2 hosts + OpenFlow ref. controller).



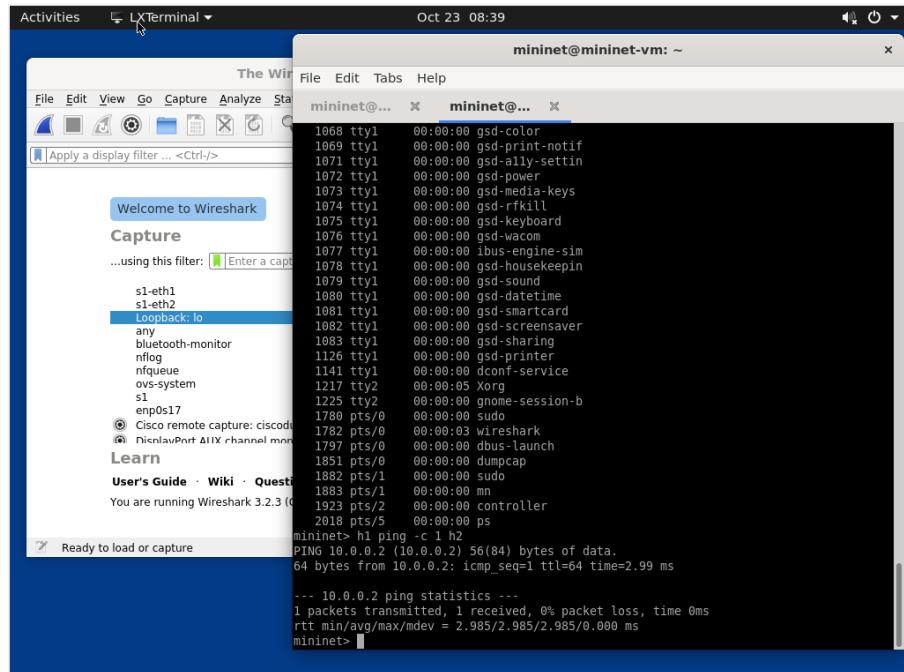
SC 4: Shows hosts interfaces listed.

Aviv Y
Jillepalli
HA04
CS 455



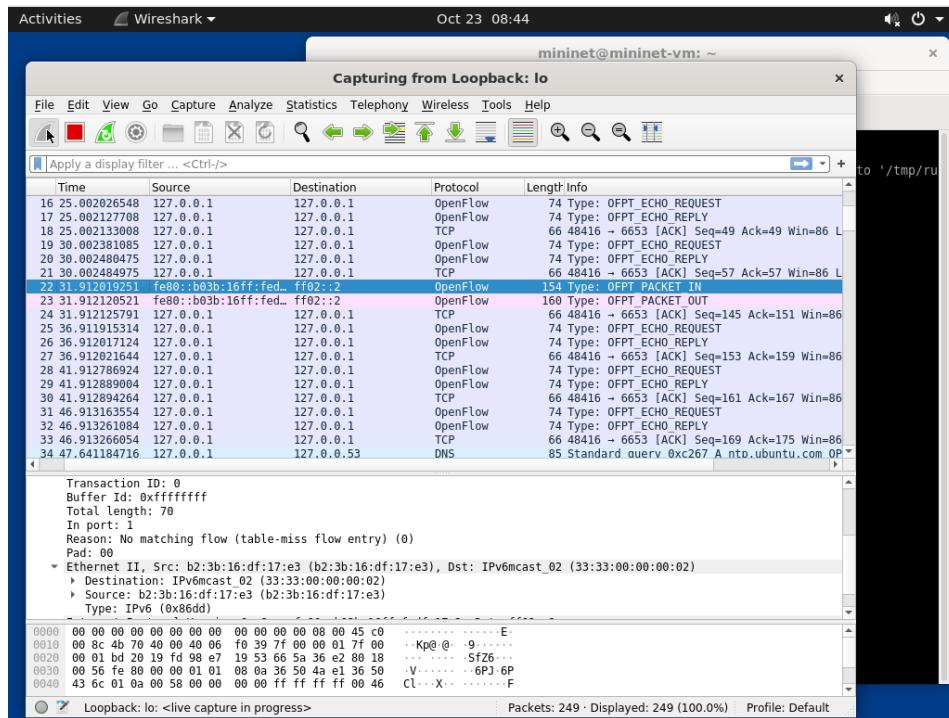
SC 5: Shows switch interfaces listed.

Testing Connectivity between hosts

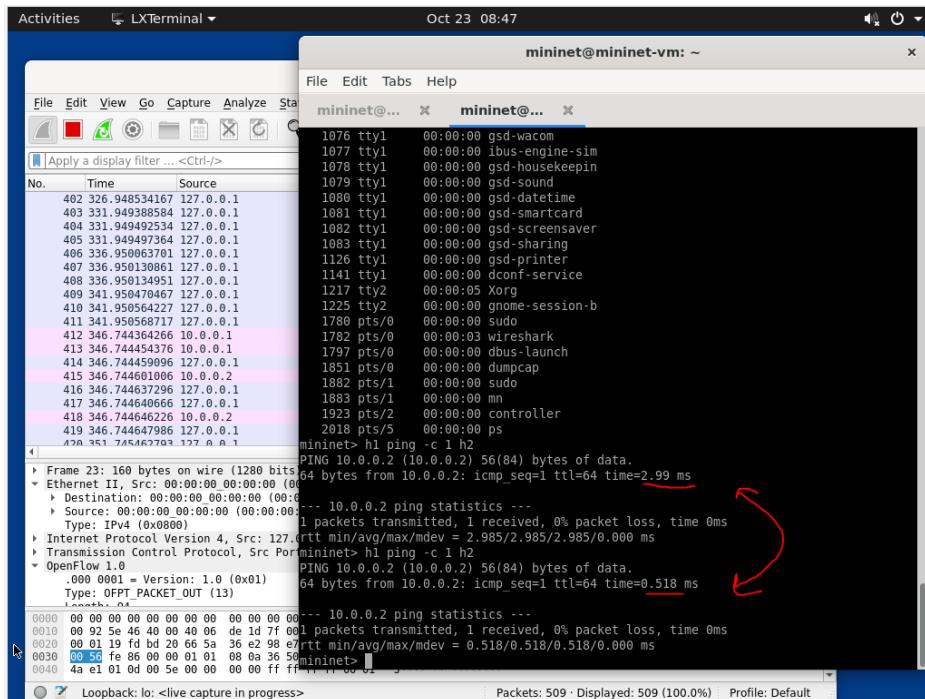


SC 6: running `h1 ping -c 1 h2` which pings host 0 to host 1.

Aviv Y
Jillepalli
HA04
CS 455



SC 7: Shows 1st Hosts ARPs for MAC address of 2nd, which shows **PACKET IN** and **PACKET OUT**.



SC 8: running same command as SC 5 and shows much lower ping due to ICMP echo request, no control traffic was generated so packets immediately go to switch.

Aviv Y
Jillepalli
HA04
CS 455

SC 9: Ran **pingall** which is an easier way to ping “all” of points in the topology.

Run a simple web server and client

The figure shows two windows side-by-side. The left window is titled 'Capturing' and displays a list of network traffic captured on interface 'mon0'. The right window is titled 'mininet@mininet-vm: ~' and shows a web browser displaying a directory listing for 'http://10.0.0.1/'.

```

Activities LXTerminal Oct 23 09:46
File Edit Tabs Help
mininet@mininet-vm: ~
File Edit View Go Capture Analyze Statistics Telephone
Apply a display filter... <Ctrl-/>
me Source Destination Proto Length/TIME
170.1671085... fe80::7cef:7bff:fea4:ff02::2 HTTP 1380 [1.3K] [text/html]
170.1671816... fe80::2473:cbff:fea4:ff02::2
170.1671880... 127.0.0.1 TCP 0 [0B] 0 ...-KB/s
170.1671948... fe80::7cef:7bff:fea4:ff02::2
170.1671966... 127.0.0.1 TCP [DCTCP] HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd"
170.1672043... fe80::2473:cbff:fea4:ff02::2
170.1672798... fe80::7cef:7bff:fea4:ff02::2
172.0869248... 127.0.0.1 TCP [HEAD] DNS</head>
172.0869394... 127.0.0.1 TCP [HEAD] DNS<meta http-equiv="Content-Type" content="text/html; charset=iso8859-1">
172.0870702... 127.0.0.53 TCP [HEAD] DNS<title>Directory listing for /</title>
172.0870702... 127.0.0.53 TCP [HEAD] DNS</head>
172.0870899... 127.0.0.1 TCP [HEAD] DNS<body>
172.0870928... 127.0.0.1 TCP [HEAD] DNS<h1>Directory listing for /</h1>
172.0871234... 127.0.0.53 TCP [HEAD] DNS<ul>
172.0871537... 127.0.0.53 TCP [HEAD] DNS<ul>
175.1676677... 127.0.0.1 TCP [HEAD] DNS<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=".gitconfig">.gitconfig</a></li>
<li><a href=".gnupg">.gnupg</a></li>
<li><a href=".local">.local</a></li>
<li><a href=".profile">.profile</a></li>
<li><a href=".sudo_as_admin_successful">.sudo_as_admin_successful</a></li>
<li><a href=".wget-hsts">.wget-hsts</a></li>
<li><a href=".wireshard">.wireshard</a></li>
<li><a href="Desktop">Desktop</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="OFL">OFL</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="nox">nox</a></li>

```

Frame 1: 76 bytes on wire (608 bits), 76 bytes captured
 Linux cooked capture
 Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
 Transmission Control Protocol, Src Port: 40842, Dst Port: 80
 OpenFlow 1.0

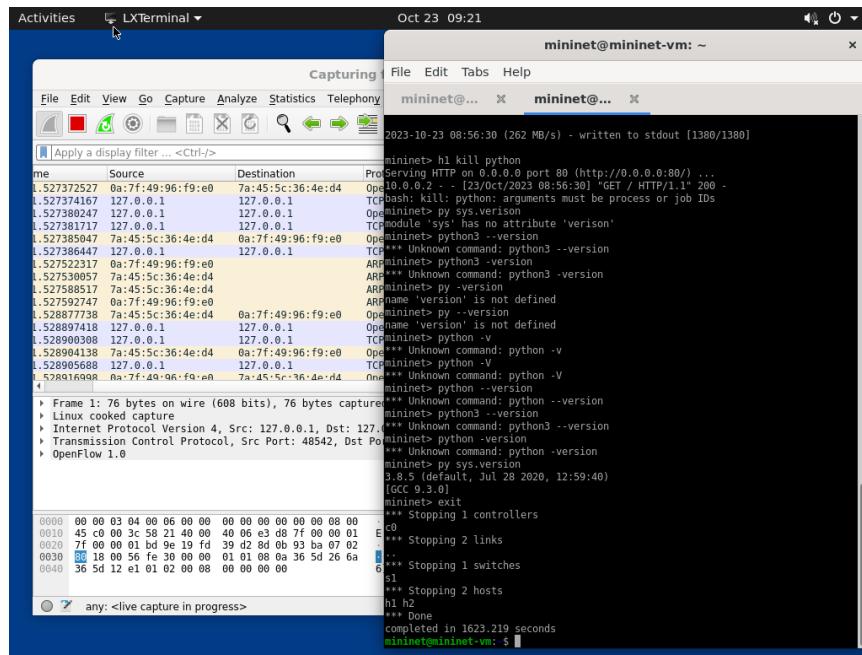
Index	Time	Source	Destination	Length
0000	00:00:03.04	00:00:06	00:00:00	00:00:00
0001	45 c9 00 3c 58 21	40 09	40 06 e3 d8	0f 00 00 01
0002	7f 00 00 01 bd 9e	19 fd	39 0d 88 0b	93 b4 07 02
0003	00 18 00 56 fe 30	00 00	01 01 68 0e	36 5d 26 6a
0004	36 5d 12 e1 01 02	00 00	00 00 00 00	00 00 00 00

any: <live capture in progress>

SC 10: Shows running h1 `python -m http.server 80` &, as well as h2 `wget -O - h1`

the first command starts the server on h1 and the second command makes a request from h2.

Aviv Y
Jillepalli
HA04
CS 455

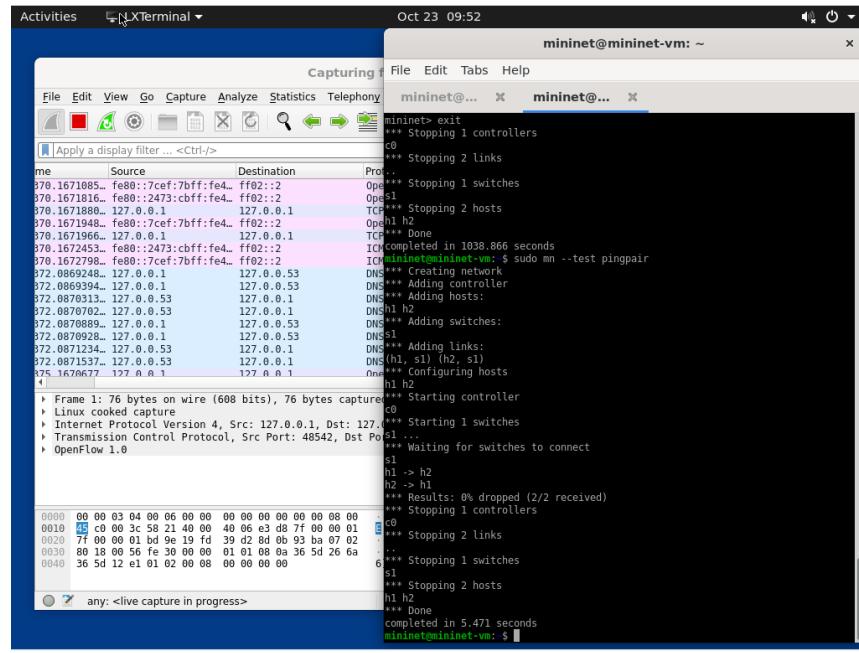


SC 11: Shows killing/shutting down the (web) server.

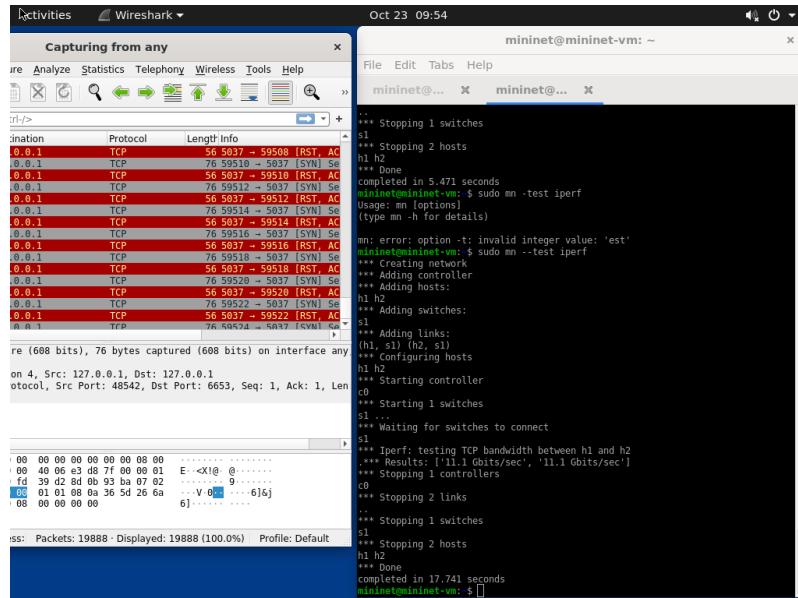
Aviv Y
Jillepalli
HA04
CS 455

Part 2: Advanced Startup Options

Running Regression Tests



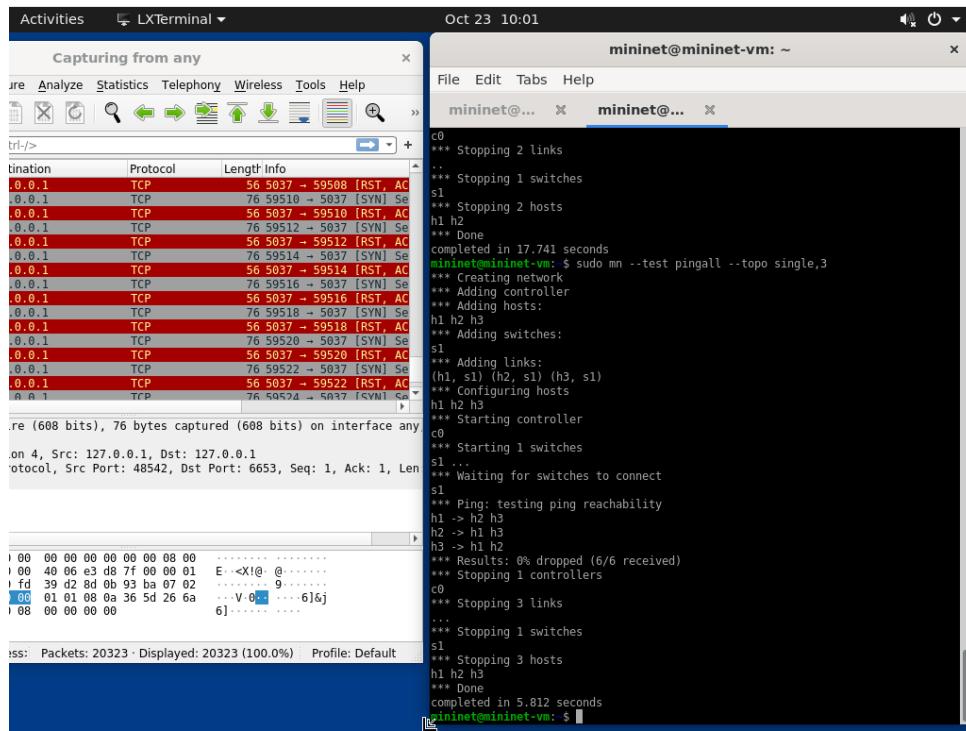
SC 12: Shows running command **sudo mn --test pingpair** which creates a minimal topology, ran an all-pairs-ping test, and broke down the topology and controller.



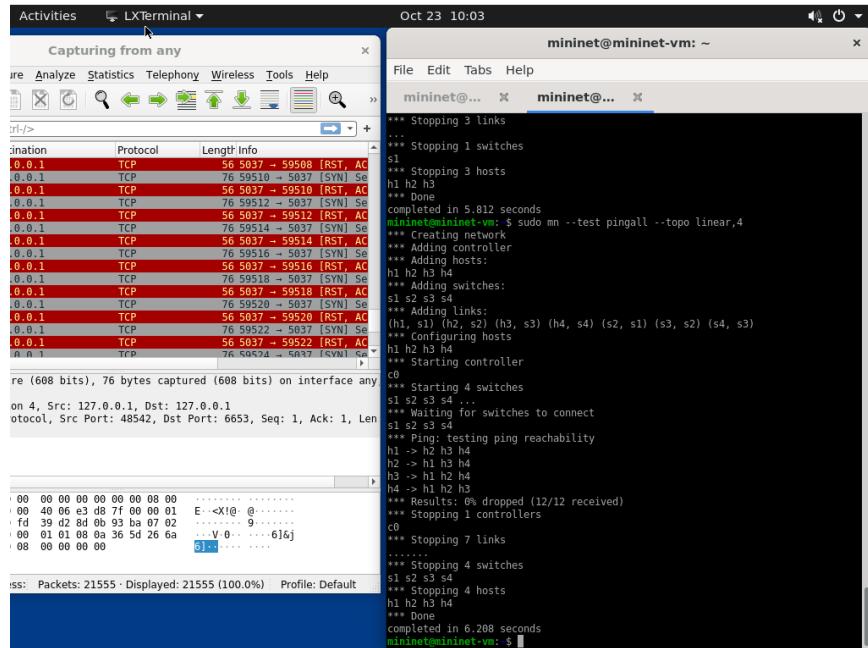
SC 13: The **sudo mn --test iperf** command creates the same Mininet, ran an iperf server on one host, ran an iperf client on the second host, and parsed the bandwidth achieved.

Aviv Y
Jillepalli
HA04
CS 455

Changing Topology Size and Type



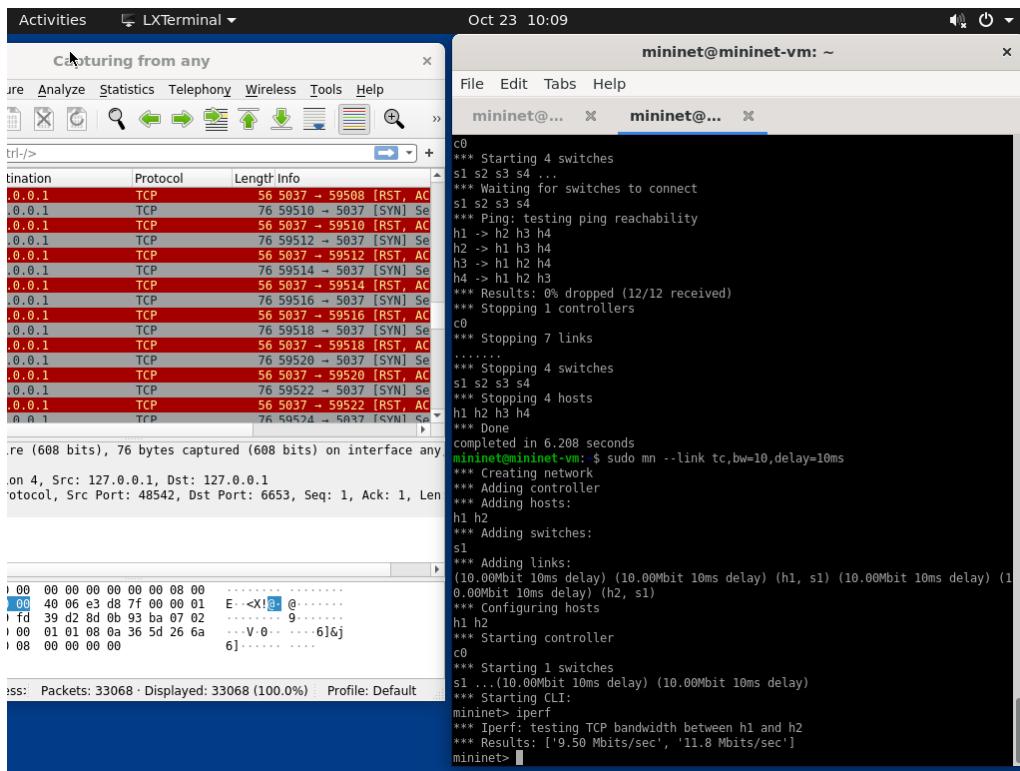
SC 14: Single topology test with 3 hosts.



SC 15: Linear topology test with 4 hosts.

Aviv Y
Jillepalli
HA04
CS 455

Link Variations



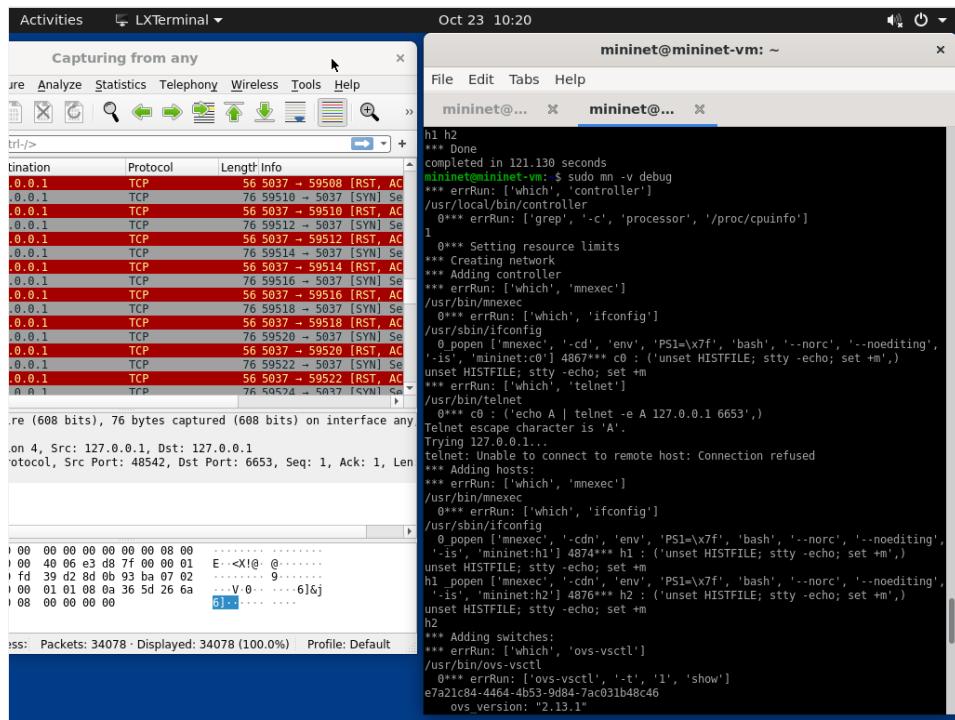
(The SC above runs `sudo mn --link tc,bw=10,delay=10ms`)

SC 16: Mininet command that allows users to set link parameters.

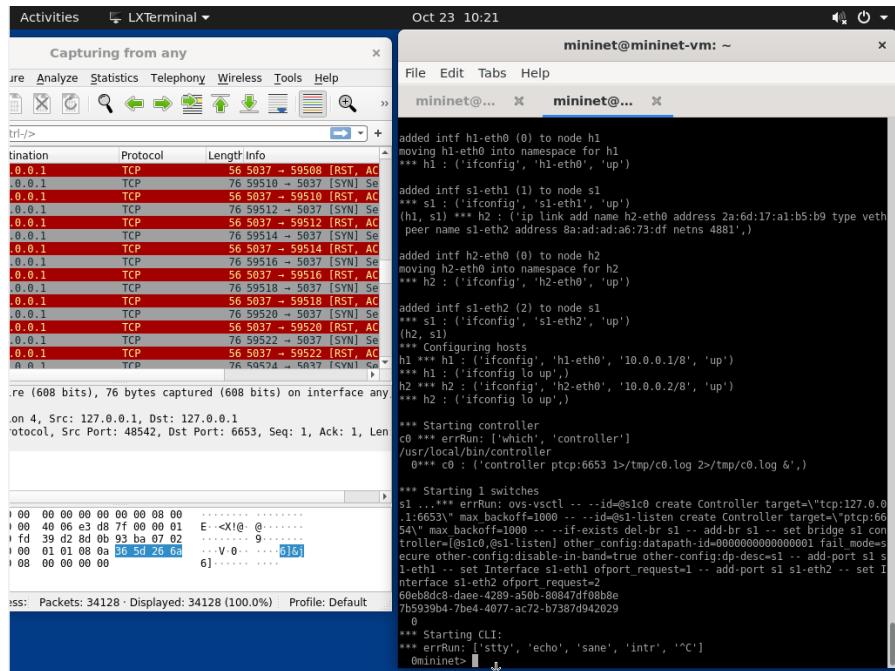
More info [here](#).

Aviv Y
Jillepalli
HA04
CS 455

Adjustable Verbosity



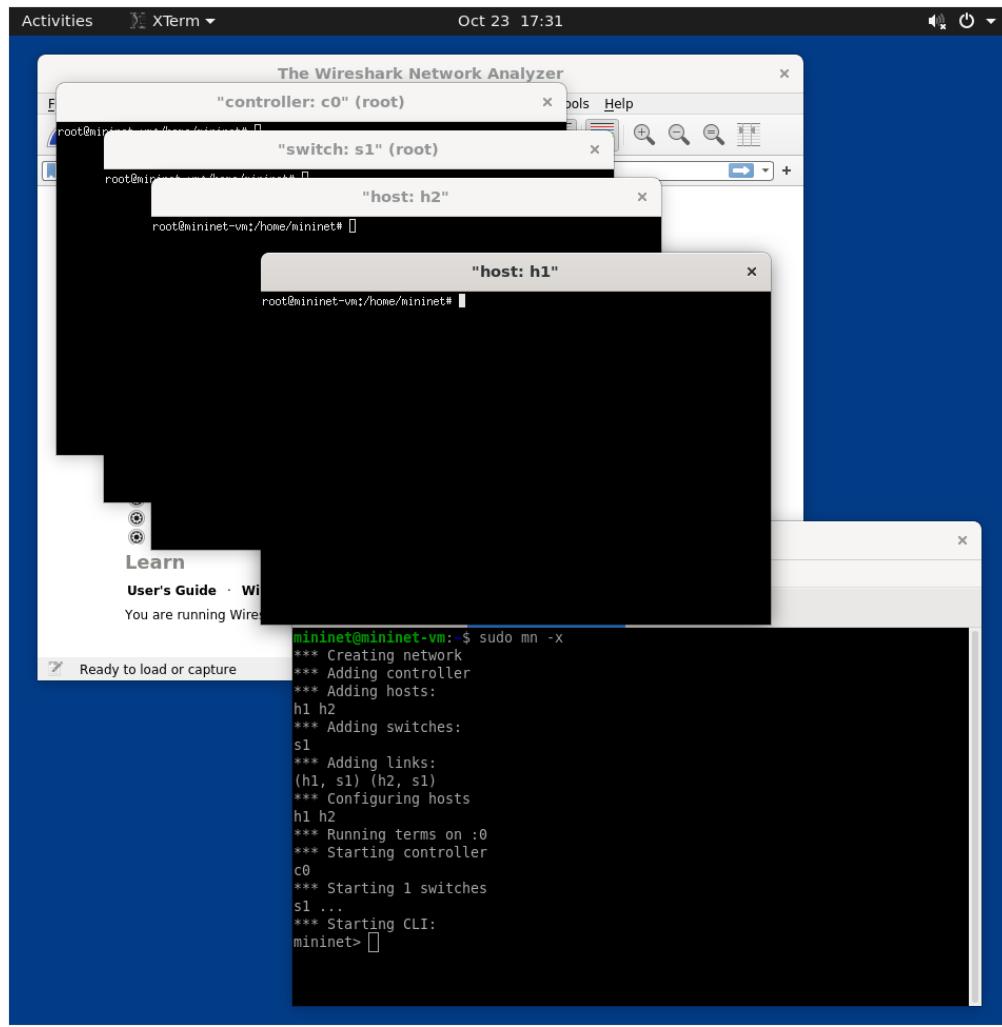
SC 17



SC 17: SC 16 & 17 shows the adjustable verbosity of the Mininet CLI

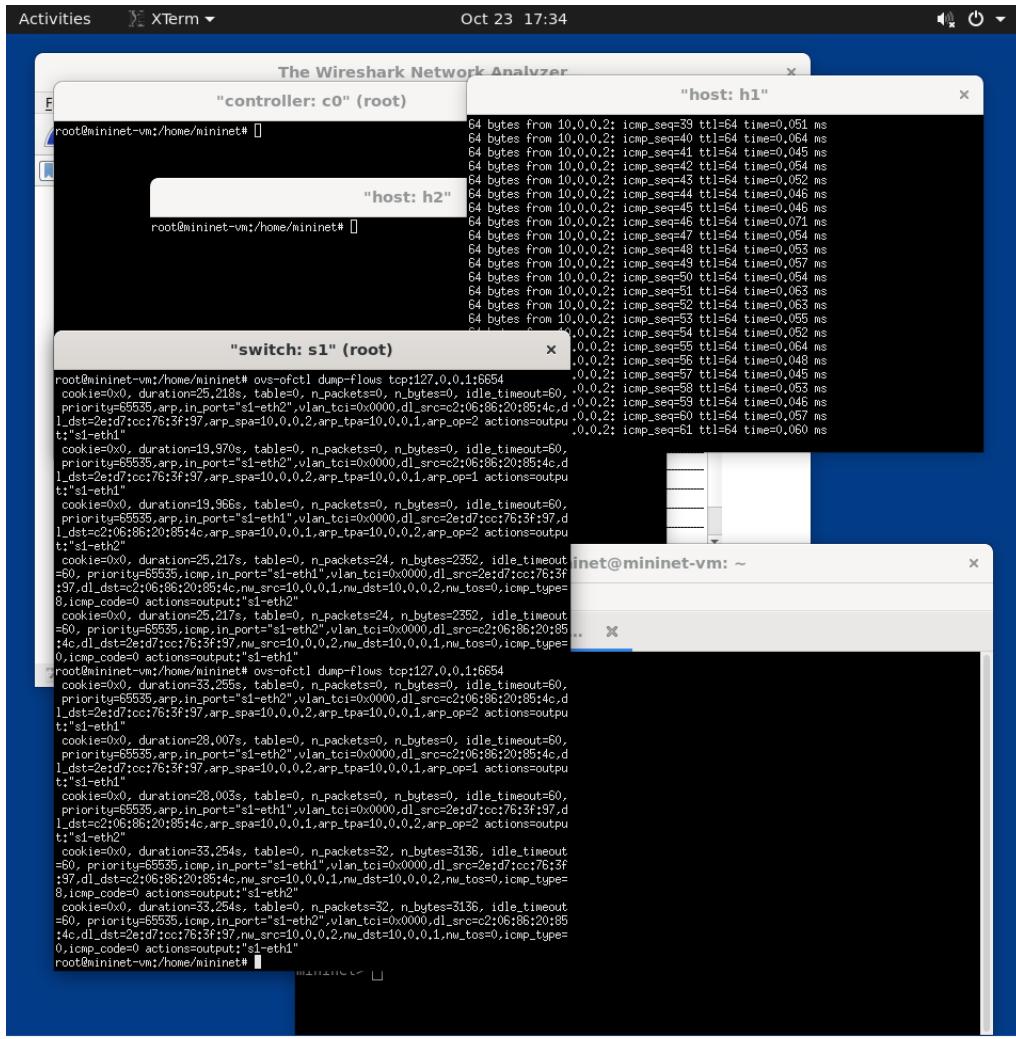
Aviv Y
Jillepalli
HA04
CS 455

XTerm Display



SC 18

Aviv Y
Jillepalli
HA04
CS 455



SC 19

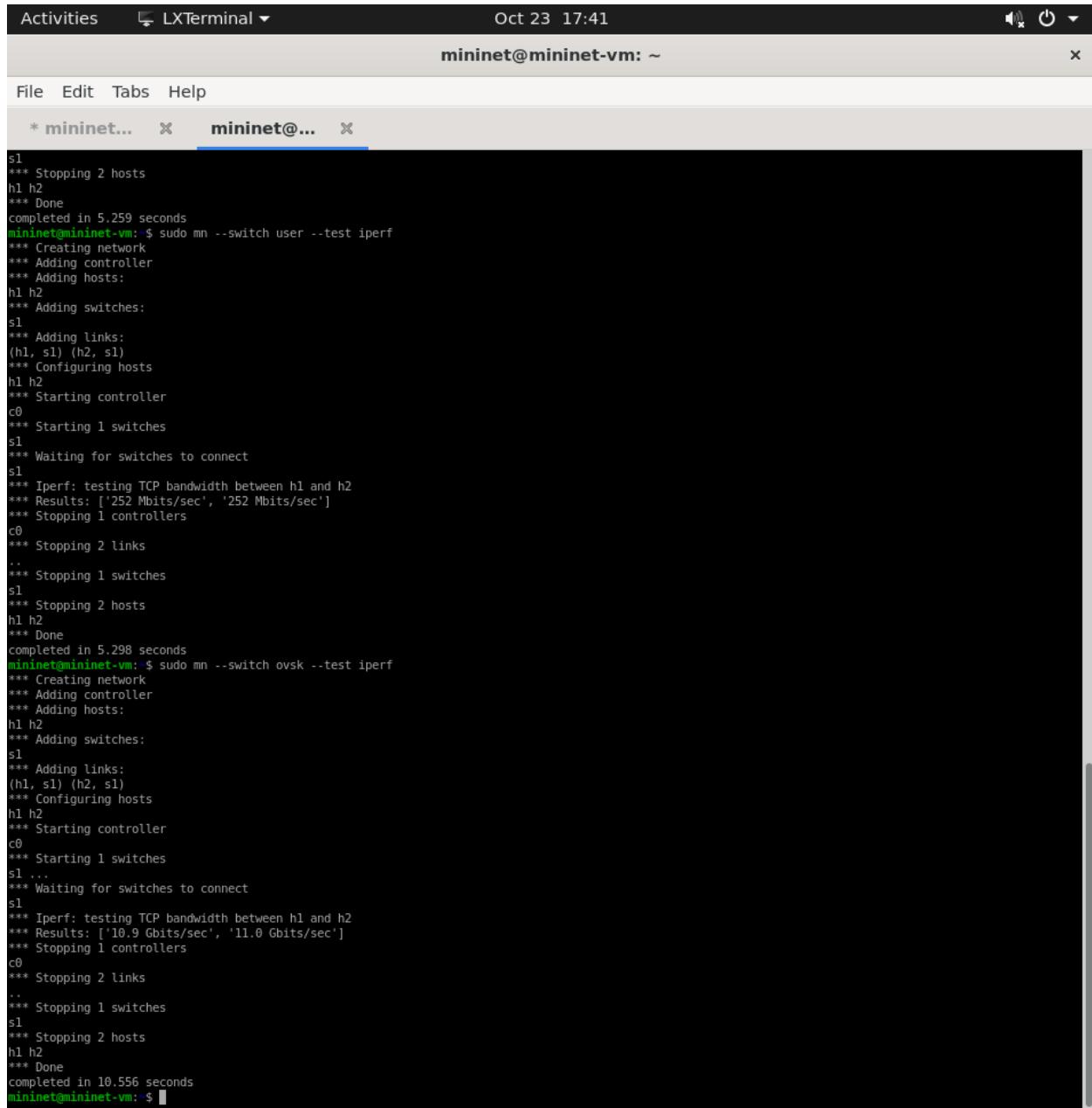
SC 18 & 19: runs command `sudo mn -x` and `ovs-ofctl dump-flows tcp:127.0.0.1:6654`.

The 1st command starts an `xterm` for every host and switch using the `-x` option.

The 2nd command enables Open Vswitch so the switch can have multiple flow entries coming in.

Aviv Y
Jillepalli
HA04
CS 455

Other Switch Types



```

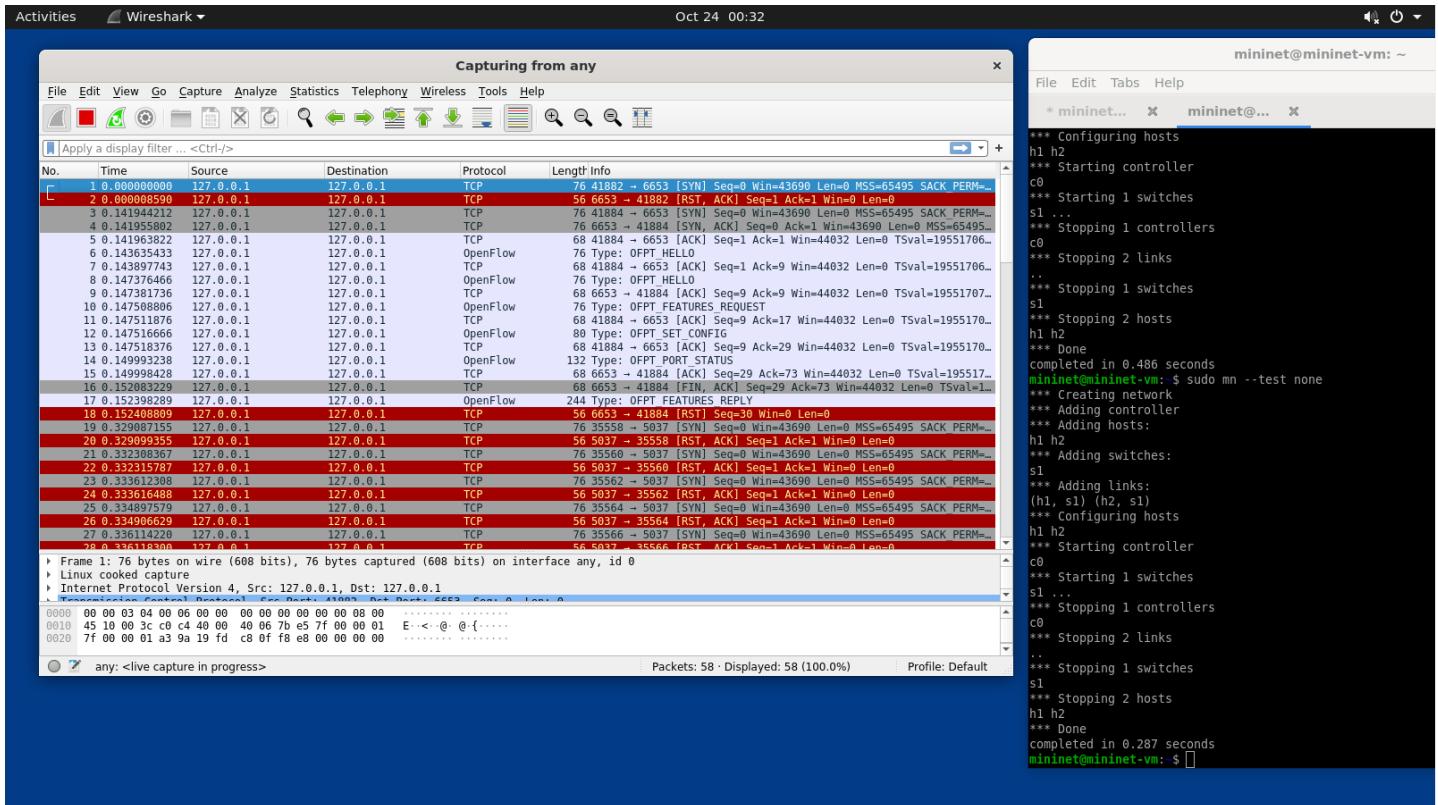
Activities LXTerminal ▾ Oct 23 17:41
mininet@mininet-vm: ~
File Edit Tabs Help
* mininet... × mininet@... ×

s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 5.259 seconds
mininet@mininet-vm: $ sudo mn --switch user --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: ['252 Mbits/sec', '252 Mbits/sec']
*** Stopping 1 controllers
c0
*** Stopping 2 links
...
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 5.298 seconds
mininet@mininet-vm: $ sudo mn --switch ovsk --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: ['10.9 Gbits/sec', '11.0 Gbits/sec']
*** Stopping 1 controllers
c0
*** Stopping 2 links
...
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 10.556 seconds
mininet@mininet-vm: $ 
```

SC 20: The two commands: `sudo mn --switch user --test iperf` and `sudo mn --switch ovsk --test iperf`, the first command runs the user-space switch (`user`) while the second command activates the Open vSwitch (`ovsk`) Switch Type. It was said the ovsk would run faster so I ran several tests on it, and it was not.

Aviv Y
Jillepalli
HA04
CS 455

Everything in its own Namespace (user switch only)



SC 21: The command **sudo mn --innamespace --switch user** puts switches in their own namespace as by default, hosts are put in their own space while switches and the controller are in the root namespace.

Aviv Y
Jillepalli
HA04
CS 455

Part 3: Mini Command-Line Interface (CLI) Commands

```

Oct 24 00:43
File Edit Tabs Help
* mininet... * mininet@...
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
(h1, s1) (h2, s1)
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
=====
EOF gterm iperfudp nodes pingpair py switch xterm
dpctl help link noecho pingairfull quit time
dpdump infits links pingall ports sh wait
exit iperf net pingallfull px source x
You may also send a command to a node using:
<node> command {args}
For example:
mininet> h1 ifconfig
The interpreter automatically substitutes IP addresses
for node names when a node is the first arg, so commands
like
mininet> h2 ping h3
should work.

Some character-oriented interactive commands require
noecho:
mininet> noecho h2 vi foo.py
However, starting up an xterm/gterm is generally better:
mininet> xterm h2

mininet> pyu 'hello' + 'world'
*** Unknown command: pyu 'hello' + 'world'
mininet> py 'hello' + 'world'
helloworld
mininet> py 'hello' + 'world'
hello world
mininet>

```

SC 22: `py 'hello' + 'world'` shows that `py` extends Mininet, as well as probing its inner workings.

```

Oct 24 00:44
File Edit Tabs Help
* mininet... * mininet@...
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
=====
EOF gterm iperfudp nodes pingpair py switch xterm
dpctl help link noecho pingairfull quit time
dpdump infits links pingall ports sh wait
exit iperf net pingallfull px source x
You may also send a command to a node using:
<node> command {args}
For example:
mininet> h1 ifconfig
The interpreter automatically substitutes IP addresses
for node names when a node is the first arg, so commands
like
mininet> h2 ping h3
should work.

mininet> pyu 'hello' + 'world'
*** Unknown command: pyu 'hello' + 'world'
mininet> py 'hello' + 'world'
helloworld
mininet> py 'hello' + 'world'
hello world
mininet> locals()
{'net': <mininet.net.Mininet object at 0x7f491ff4db0>, 'h1': <Host h1: h1-eth0:10.0.0.1 pid=2736>, 'h2': <Host h2: h2-eth0:10.0.0.2 pid=2738>, 's1': <OVSSwitch s1: lo:127.0.0.1, s1-eth1:None, s1-eth2:None pid=2743>, 'c0': <Controller c0: 127.0.0.1:6653 pid=2729>}
mininet>

```

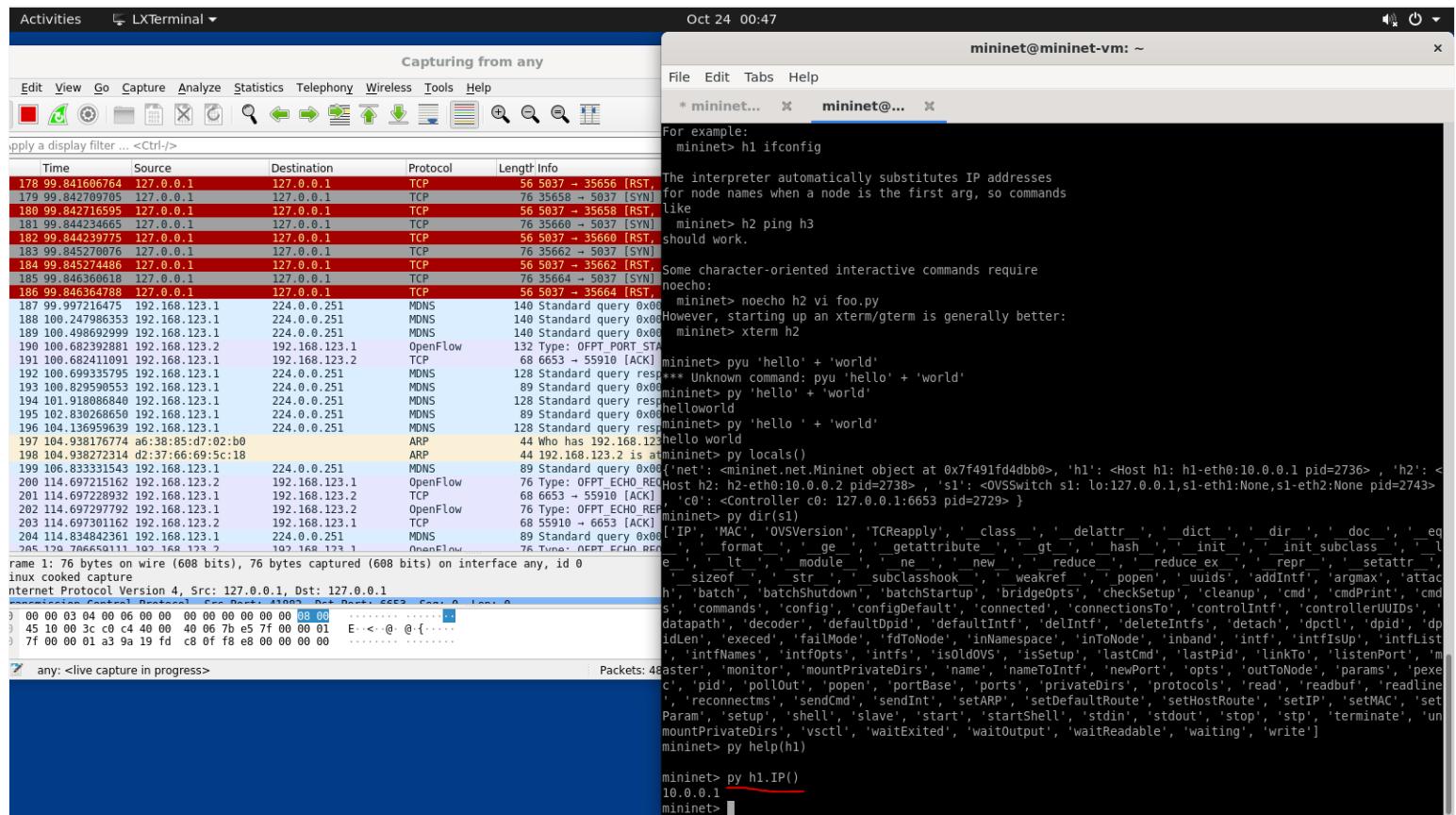
SC 23: `py locals()` prints the accessible local variables.

Aviv Y
Jillepalli
HA04
CS 455

SC 24: `py dir(s1)` shows the methods and properties available for a node using the `dir()` function.

SC 25: `py help(h1)` shows the on-line documentation for methods available on a node by using the `help()` function.

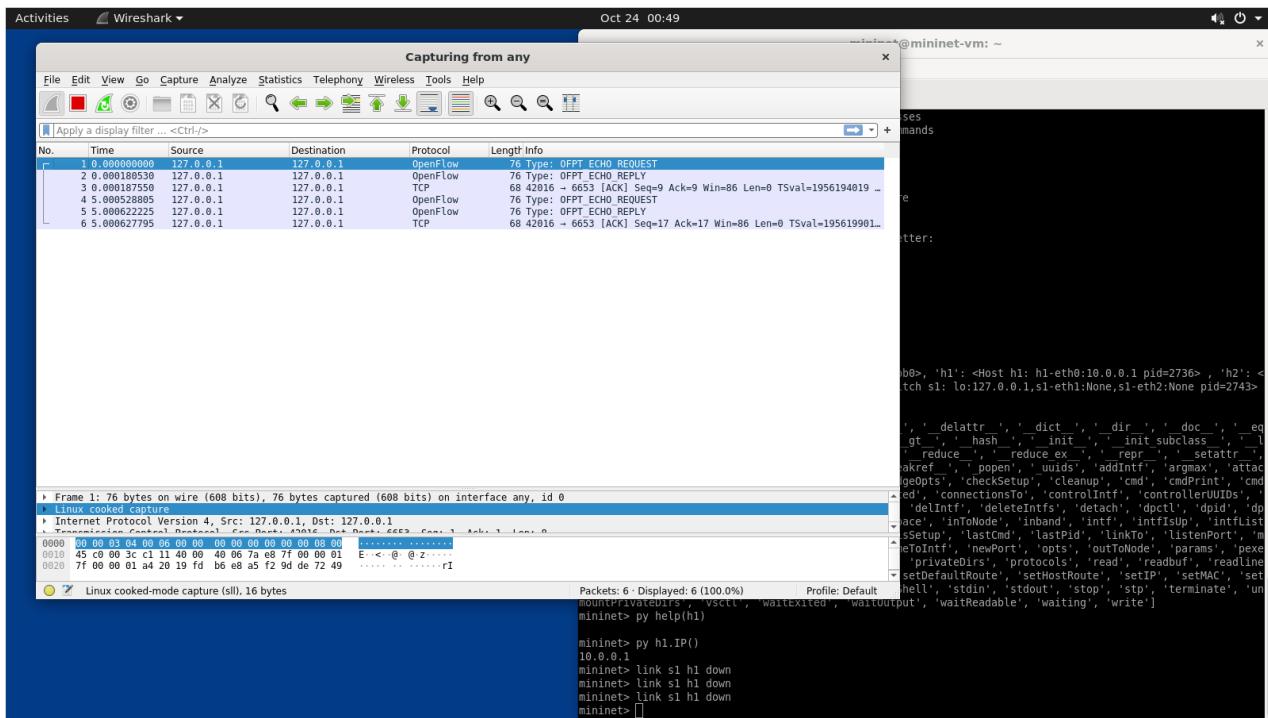
Aviv Y
Jillepalli
HA04
CS 455



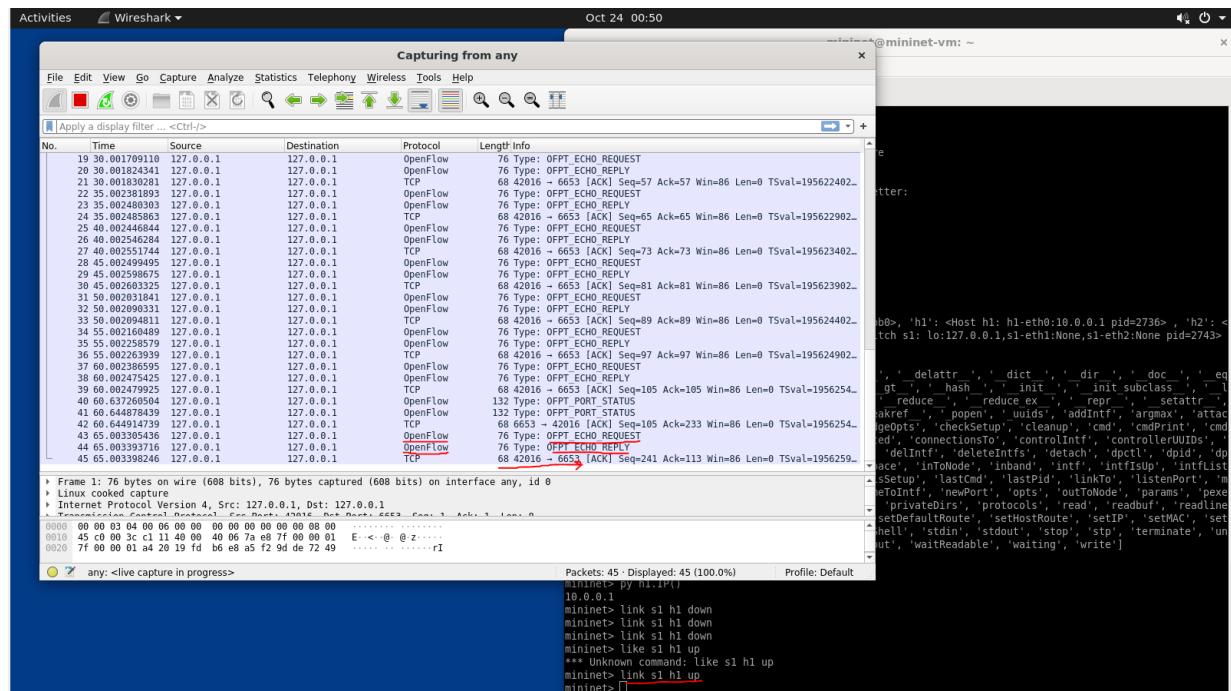
SC 26: Running the `py h1.IP()` command, this lets us evaluate methods of variables, in this command, it prints the hosts IP Address.

Aviv Y
Jillepalli
HA04
CS 455

Link Up/Down



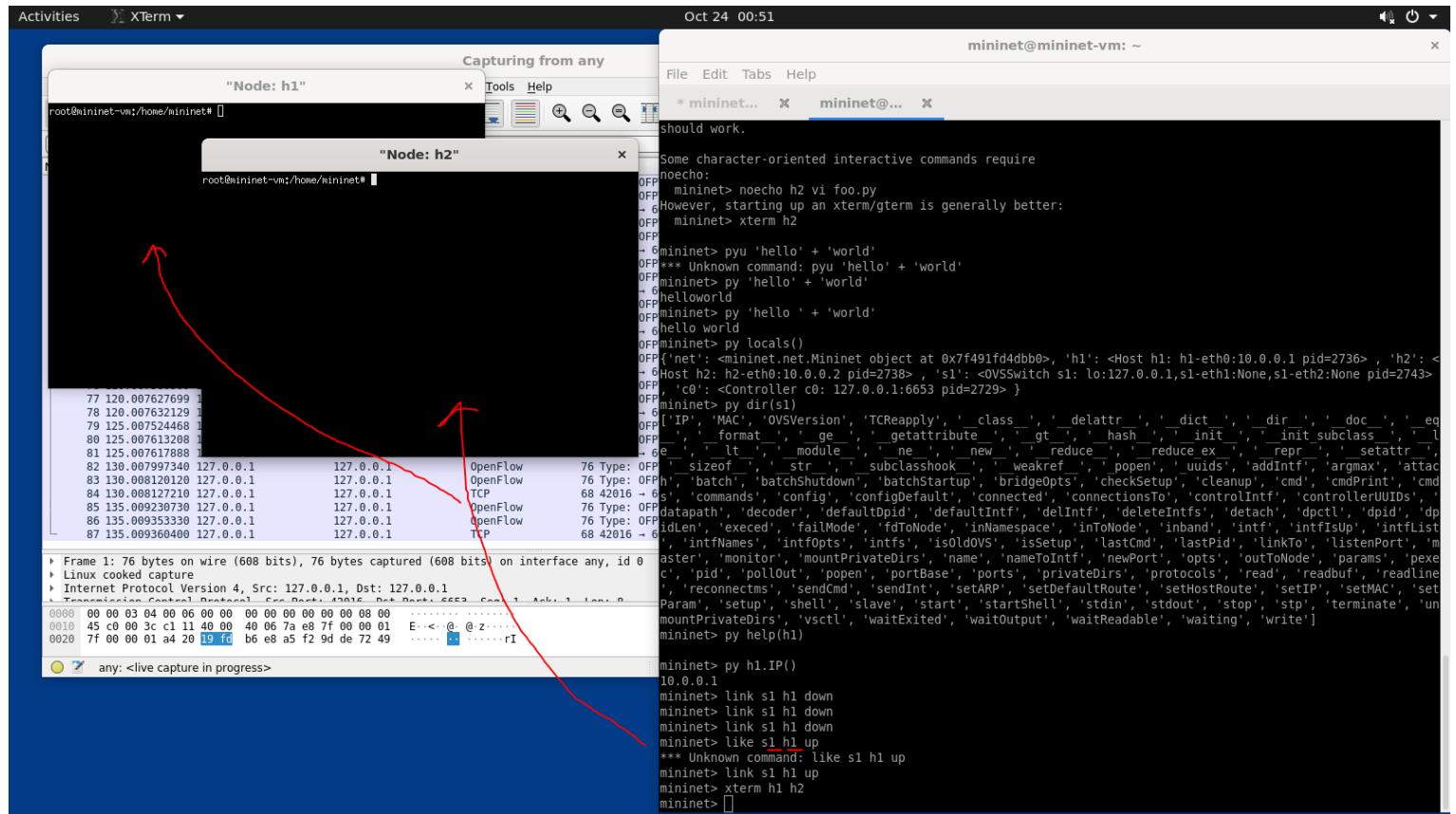
SC 27: The **link s1 h1 down** command disables both halves of the virtual ethernet pair. This is for fault tolerance testing.



SC 28: The **link s1 h1 up** command links back both halves of the virtual ethernet pair.

Aviv Y
Jillepalli
HA04
CS 455

XTerm Display



SC 29: **xterm h1 h2** displays xterm for host 1 (h1) and host 2 (h2).

Aviv Y
Jillepalli
HA04
CS 455

Part 4: Python API Examples

SSH Daemon per host

Activities LXTerminal ▾ Oct 24 01:27

Capturing from any mininet@mininet-vm: ~

```
File Edit Tabs Help
File Edit Tabs Help
mininet@mininet-vm: ~
mininet@mininet-vm: $ cd ../..
26The authenticity of host '10.0.0.1' (10.0.0.1) can't be established.
26ECDSA key fingerprint is SHA256:AEeZDXFf078LxDy0uM3lT0Vg3txiP4XfGj9qUTeGY.
26Are you sure you want to continue connecting (yes/no/[fingerprint])? Y
26Please type 'yes', 'no' or the fingerprint: yes
26Warning: Permanently added '10.0.0.1' (ECDSA) to the list of known hosts.
26mininet@10.0.0.1's password:
26Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-42-generic x86_64)
26
26 * Documentation: https://help.ubuntu.com
26 * Management: https://landscape.canonical.com
26 * Support: https://ubuntu.com/advantage
26
26New release '22.04.3 LTS' available.
26Run 'do-release-upgrade' to upgrade to it.
26
26Last login: Mon Oct 23 15:17:11 2023
26mininet@mininet-vm: $ ping 10.0.0.2
26PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
2664 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=1.32 ms
2664 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.182 ms
> Frag64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.060 ms
> Lin64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.051 ms
> Int64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.051 ms
> Tra64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.054 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.053 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.057 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.043 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.049 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=0.050 ms
000864 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=0.061 ms
001864 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=0.048 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=0.043 ms
64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=0.048 ms
64 bytes from 10.0.0.2: icmp_seq=16 ttl=64 time=0.054 ms
64 bytes from 10.0.0.2: icmp_seq=17 ttl=64 time=0.060 ms
[...]
mininet@mininet-vm: ~
```

mininet@mininet-vm: ~/mininet/examples

```
File Edit Tabs Help
* mininet... x mininet@...
baressh.py consoles.py init_.py multilink.py popenpoll.py sshd.py
bind.py controllers2.py intoptions.py multiping.py popen.py test
clustercli.py controllers.py limit.py multipoll.py _pycache_
clusterdemo.py controlnet.py linearbandwidth.py natnet.py README.md tree1024.py
clusterperf.py cpu.py linuxrouter.py nat.py scratchnet.py treeping64.py
cluster.py emptynet.py miniedit.py numberedports.py scratchnetuser.py vlanhost.py
clustersanity.py hwinif.py mobility.py simpleperf.py
mininet@mininet-vm: ~/mininet/examples$ sudo python3 sshd.py
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(s1, h1) (s1, h2) (s1, h3) (s1, h4)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Waiting for switches to connect
s1
*** Waiting for ssh daemons to start
.
*** Hosts are running sshd at the following addresses:
h1 10.0.0.1
h2 10.0.0.2
h3 10.0.0.3
h4 10.0.0.4
*** Type 'exit' or control-D to shut down network
*** Starting CLI:
mininet> exit
*** Stopping 1 controllers
c0
*** Stopping 5 links
.....
*** Stopping 1 switches
s1
*** Stopping 4 hosts
h1 h2 h3 h4
*** Done
mininet@mininet-vm: ~/mininet/examples$
```

SC 30: Shows the SSH Daemon on every host. It's done by going into the Mininet/examples directory; `~/mininet/examples/sshd.py`. This is done by running `sudo python3 sshd.py` in the designated directory.

Aviv Y
Jillepalli
HA04
CS 455

simpleperf.py & test_simpleperf.py

```

Oct 24 01:15 mininet@mininet-vm: ~/Downloads
* mininet... * mininet@...
c0
*** Stopping 4 links
*** Stopping 1 switches
s1
*** Stopping 4 hosts
h1 h2 h3 h4
*** Done
mininet@mininet-vm: ~/Downloads$ sudo python3 simpleperf.py
Creating network
* Adding hosts:
h1 h2 h3 h4
* Adding controller:
h1
* Adding switches:
h2 h3 h4
* Adding links:
h1-h2 h2-h3 h3-h4
* Configuring hosts
h1 h2 h3 h4
* Starting controller
c0
*** Starting 1 switches
s1
*** Starting 4 links
*** Stopping 1 switches
s1
*** Stopping 4 hosts
h1 h2 h3 h4
*** Done
mininet@mininet-vm: ~/Downloads$ 
Part 3: Mininet Command-Line Interface [mininet@mininet-vm: ~/Downloads$ ]

```

SC 31: Runs `sudo python3 simpleperf.py`. Explanation of code and process below.

```

Oct 24 01:18 mininet@mininet-vm: ~/Downloads
* mininet... * mininet@...
h2 h2-eth0:s1-eth2
h3 h3-eth0:s1-eth3
h4 h4-eth0:s1-eth4
h1 ...
*** Starting 1 switches
s1
*** Starting 4 links
*** Stopping 1 switches
s1
*** Stopping 4 hosts
h1 h2 h3 h4
*** Done
mininet@mininet-vm: ~/Downloads$ sudo python3 test_simpleperf.py
Creating network
* Adding controller
h1
* Adding switches:
h2 h3 h4
* Adding links:
h1-h2 h2-h3 h3-h4
* Configuring hosts
h1 h2 h3 h4
* Starting controller
c0
*** Starting 1 switches
s1
*** Starting 4 links
*** Stopping 1 switches
s1
*** Stopping 4 hosts
h1 h2 h3 h4
*** Done
Ran 1 test in 7.578s
mininet@mininet-vm: ~/Downloads$ 
Part 3: Mininet Command-Line Interface [mininet@mininet-vm: ~/Downloads$ ]

```

SC 32: Runs `sudo python3 test_simpleperf.py`. Explanation of code and process below.

Aviv Y
Jillepalli
HA04
CS 455

Part 4: simpleperf.py and test_simpleperf.py Explanations:

simpleperf.py

- This script tests the performance between `h1` and `h4` in a Mininet setup where all hosts are connected to a single switch. The script has functionality to test transmissions with or without packet loss '`lossy`'. The script first starts by initializing the custom topology of the network by building a class (`SingleSwitchTopo()`) which builds the actual network. Then begins testing the network performance using the `perfTest()` function.

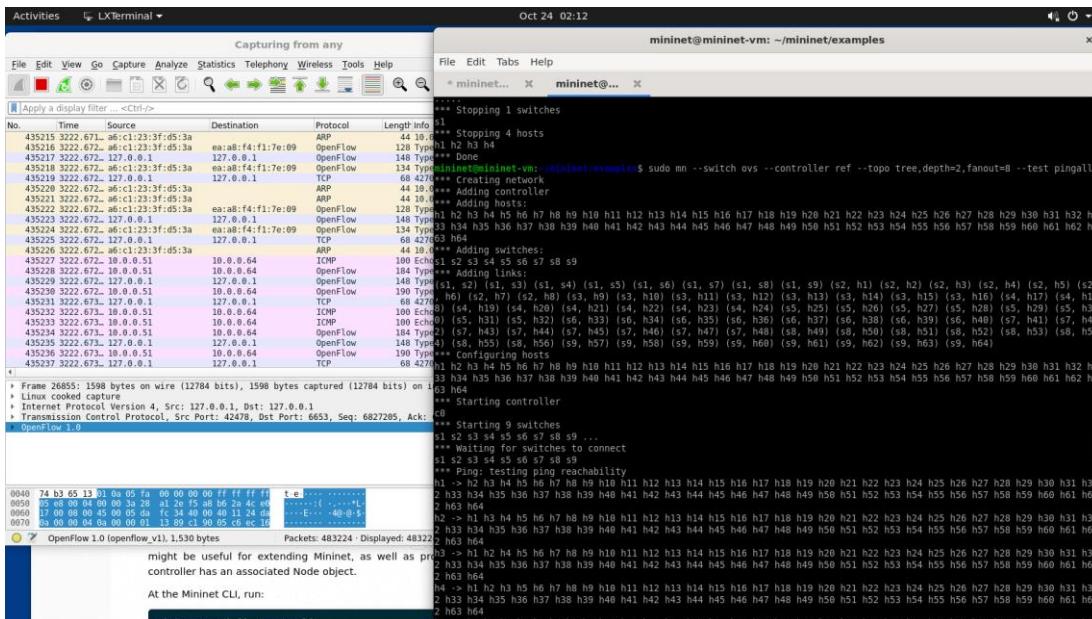
test_simpleperf.py

- This script tests and verifies the performance of a Mininet setup based on `simpleperf.py` script. It makes sure that the measured bandwidth (from `iperf`) is within an acceptable range of the expected bandwidth. In the case that the bandwidth deviates beyond the acceptable tolerance, the test fails.
- The script inherits from `unittest.TestCase(testSimplePerf(unittest.TestCase))` which contains the unit tests. `testE2E(self)` tests end to end performance where the bandwidth and tolerance is defined according to the test's requirements.

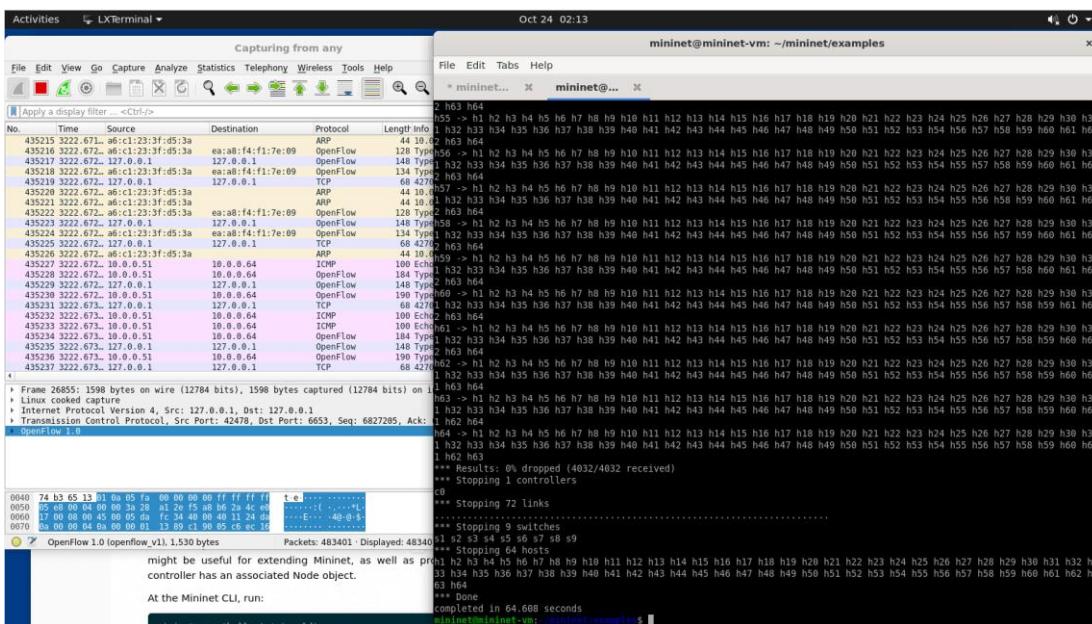
Aviv Y
Jillepalli
HA04
CS 455

Workflow

Creating a Network



SC 33: The command `sudo mn --switch ovs --controller ref --topo tree,depth=2,fanout=8 --test pingall`, this creates a network with 9 switches and 64 hosts (b/c of depth of 2 and fanout of 8). Within this command, we are testing all points in the network with `pingall`.



SC 34; end of result of command above.

Aviv Y
Jillepalli
HA04
CS 455

Interacting with a Network

The figure shows a terminal window titled "Capturing from any" and another titled "mininet@mininet-vm: ~".

Capturing from any:

- Shows a list of network captures (e.g., 4345, 743, 127.0.0.1).
- Logs of captured frames (e.g., Frame 26855: 1598 bytes on wire (12784 bits), 1598 bytes captured (12784 bits) on interface eth0).
- Logs of transmitted packets (e.g., 00:40 7b b3 65 13 01 00 05 fa 00:00:00:ff:ff:ff t.e., 00:50 05 e8 00 00 00 00 3a 2a a2:2e:b6:04:e0E. 48 @ \$:, 00:60 17 00 08 00 05 05 05 fc 34 40 00 40 11 24 d4 0a:00:00:00:00:01 13:89:11:90:05:c6 ec 16).
- Logs of OpenFlow 1.0 traffic.

mininet@mininet-vm: ~

- Logs of host addition: *** Adding hosts: h1 h2, *** Adding switches: s1 s2, *** Configuring hosts h1 s1 (h2, s1).
- Logs of controller setup: *** Starting controller, *** Starting 1 switches s1 ...
- Logs of CLI startup: *** Starting CLI: mininet-h1 ping p2, mininet-p2: Temporary failure in name resolution.
- Logs of ICMP echo requests and responses between hosts.
- Logs of OpenFlow 1.0 messages (e.g., 128 Type04 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=1.60 ms, 128 Type04 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.234 ms, 128 Type04 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.045 ms, 128 Type04 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.055 ms, 128 Type04 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.061 ms, 128 Type04 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.056 ms, 128 Type04 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.049 ms, 128 Type04 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.064 ms, 128 Type04 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.052 ms, 128 Type04 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.052 ms, 128 Type04 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=0.045 ms, 128 Type04 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=0.051 ms, 128 Type04 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=0.072 ms, 128 Type04 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=0.051 ms, 128 Type04 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=0.052 ms, 128 Type04 bytes from 10.0.0.2: icmp_seq=16 ttl=64 time=0.056 ms, 128 Type04 bytes from 10.0.0.2: icmp_seq=17 ttl=64 time=0.052 ms, 128 Type04 bytes from 10.0.0.2: icmp_seq=18 ttl=64 time=0.051 ms, 128 Type04 bytes from 10.0.0.2: icmp_seq=19 ttl=64 time=0.051 ms, 128 Type04 bytes from 10.0.0.2: icmp_seq=20 ttl=64 time=0.061 ms, 128 Type04 bytes from 10.0.0.2: icmp_seq=21 ttl=64 time=0.053 ms, 128 Type04 bytes from 10.0.0.2: icmp_seq=22 ttl=64 time=0.052 ms).
- Logs of ping statistics and packet loss.

SC 35: Runs h1 ping h2

The figure shows a terminal window titled "mininet@mininet-vm: ~" with the command "mininet>". Below it is a Wireshark capture window titled "Capturing from any" showing network traffic. The traffic list shows several ARP requests and responses between hosts 127.0.0.1 and 127.0.0.2. The details and bytes panes show the structure of the ARP frames. At the bottom, a status bar indicates "Frame 26855: 1598 bytes on wire (12784 bits), 1598 bytes captured (12784 bits) on interface eth0".

SC 36: Shows trying to start a webserver but it doesn't want to connect. I ran the command above with creates a network with 64 hosts and 9 switches so it should work but doesn't.