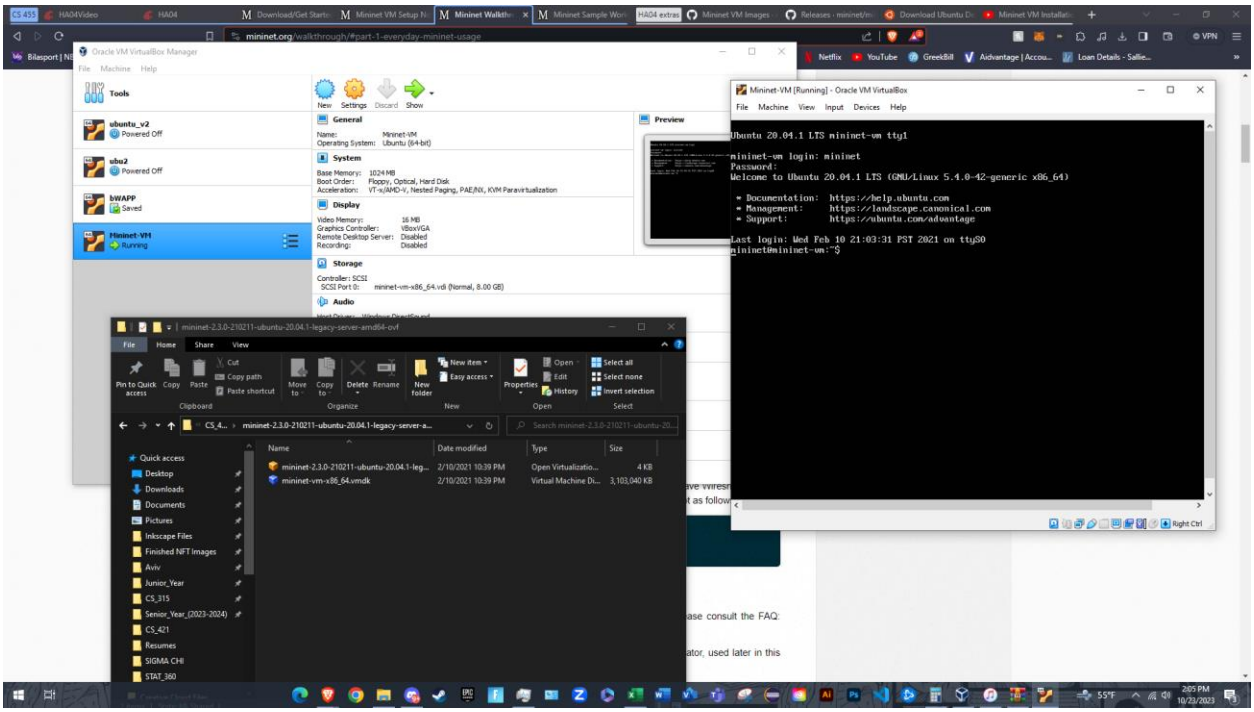
Aviv Y
Jillepalli
HA04
CS 455

NOTE: Explanation of simpleperf.py and test_simpleperf.py can be found on Page 23

Installation:

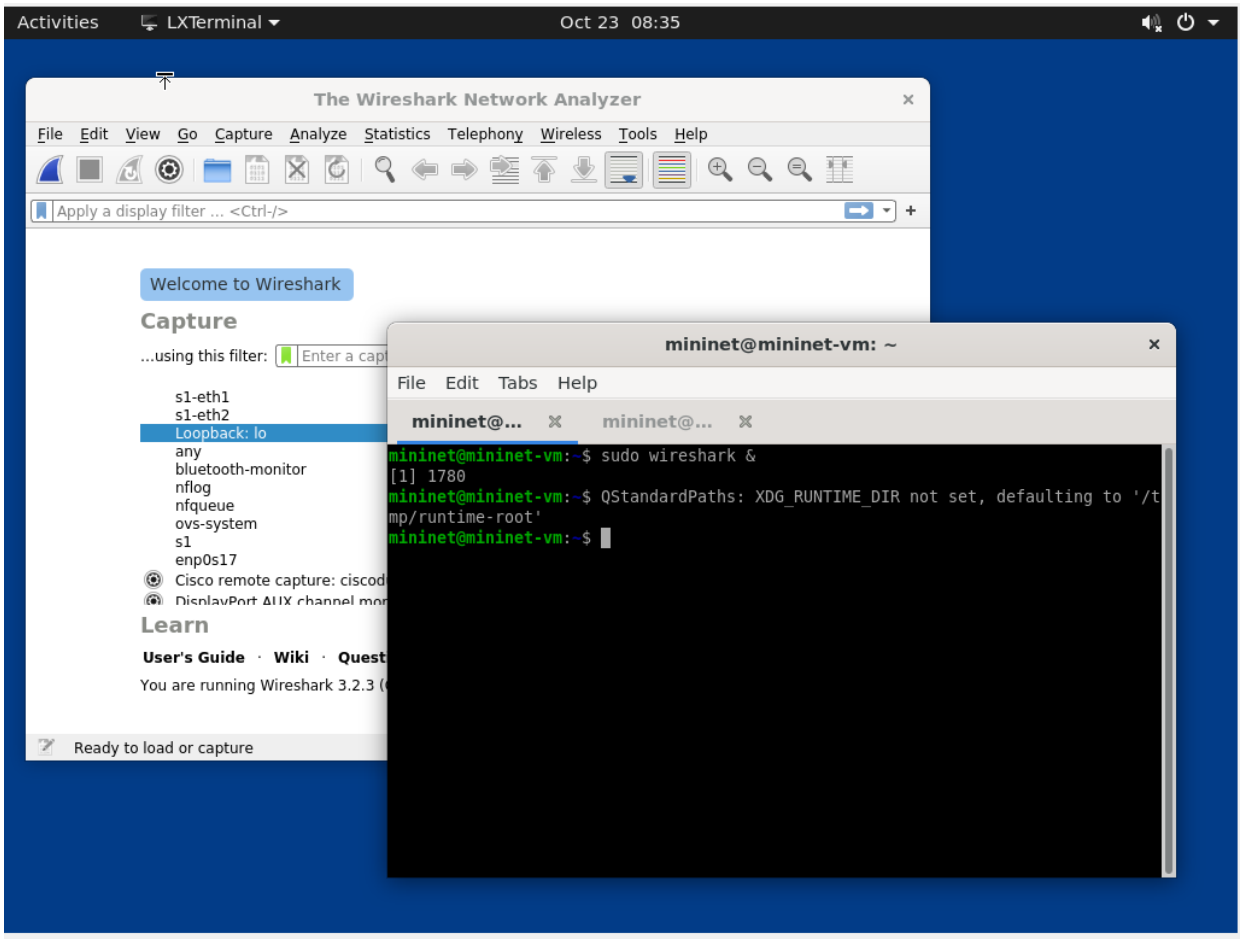
- Ran into issues while doing the default installation.
- Used a command to install the GUI which fixed the overall issue regarding Wireshark.
    - `Sudo apt-get update && apt-get install xinit lxde virtualbox-guest-dkms`
        - IF you get this error:
            - `Unable to acquire the dpkg frontend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



SC 1: Shows successful Mininet installation.

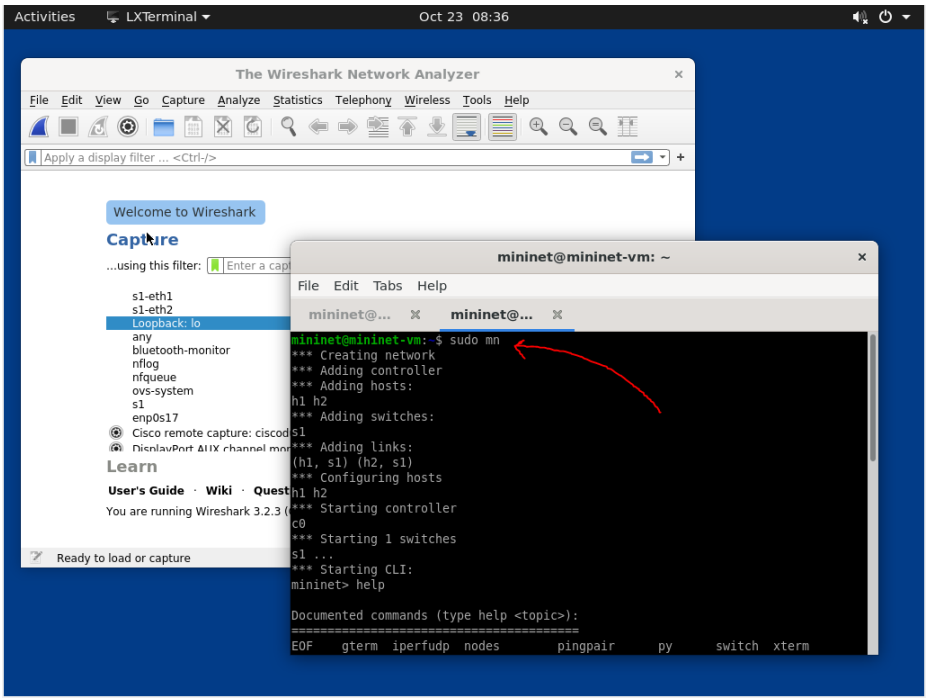
Aviv Y
Jillepalli
HA04
CS 455

Part 1: Everyday Mininet Usage

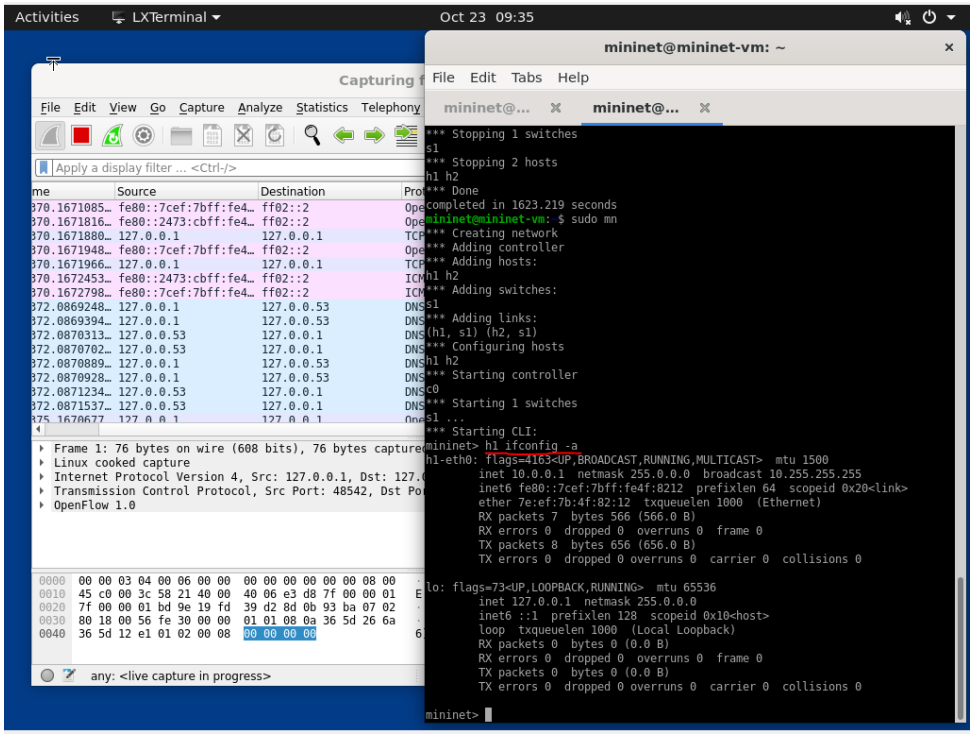


SC 2: Successfully Opening Wireshark.

Aviv Y
Jillepalli
HA04
CS 455

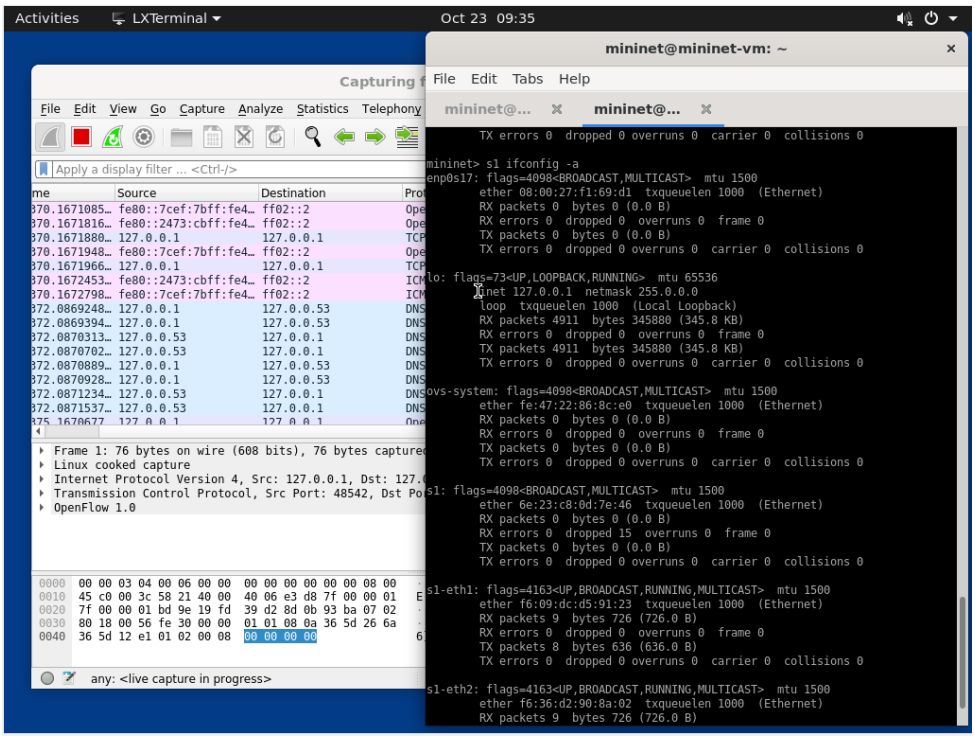


SC 3: Successfully creating minimal topology (1 OpenFlow switch + 2 hosts + OpenFlow ref. controller).



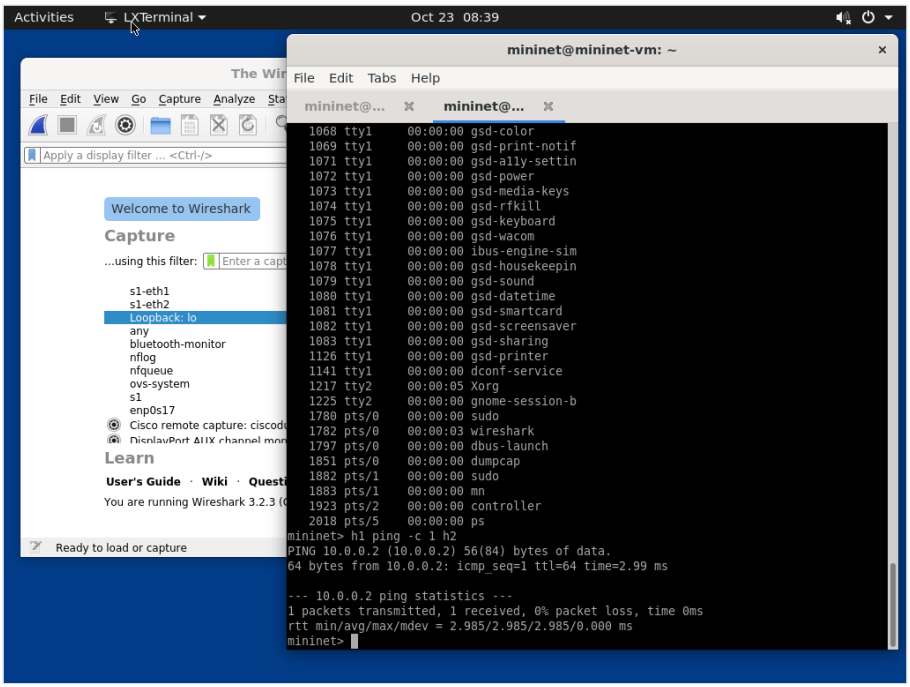
SC 4: Shows hosts interfaces listed.

Aviv Y
Jillepalli
HA04
CS 455



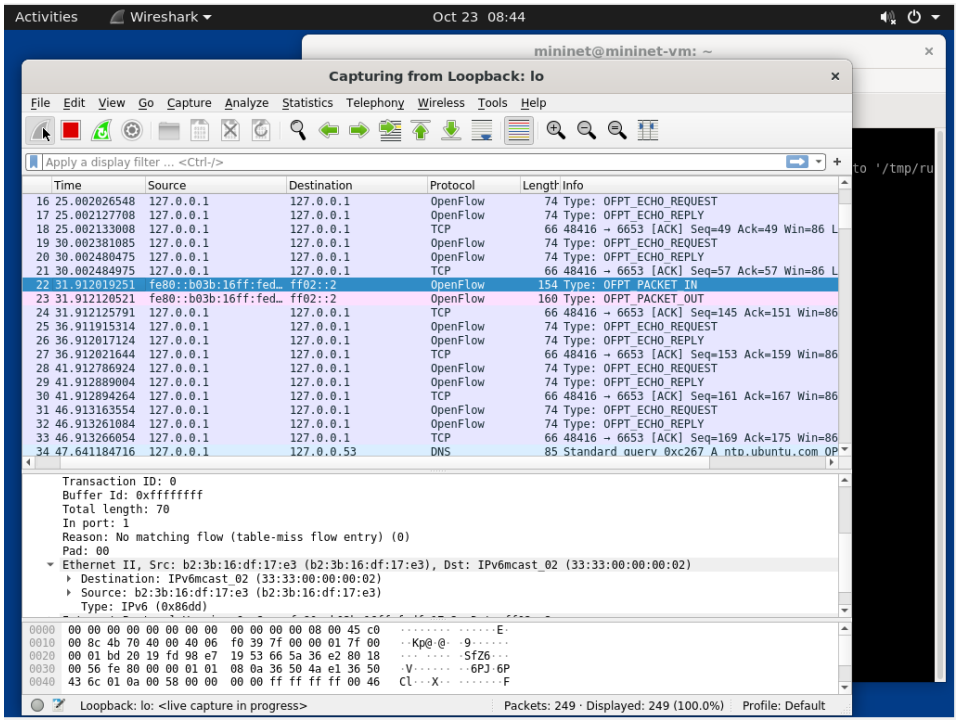
SC 5: Shows switch interfaces listed.

Testing Connectivity between hosts

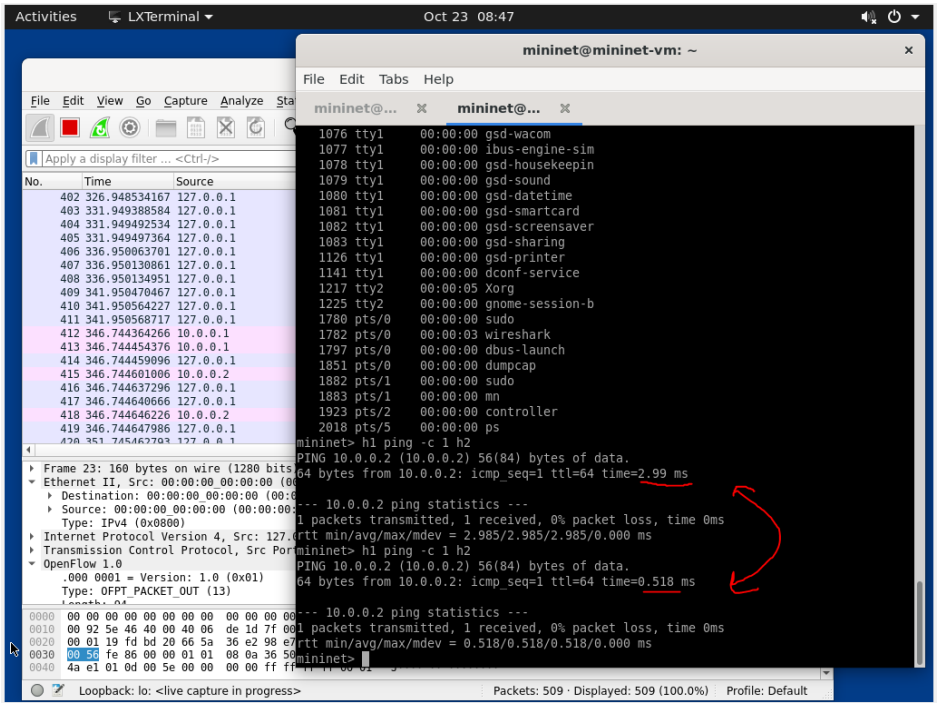


SC 6: running `h1 ping -c 1 h2` which pings host 0 to host 1.

Aviv Y
Jillepalli
HA04
CS 455

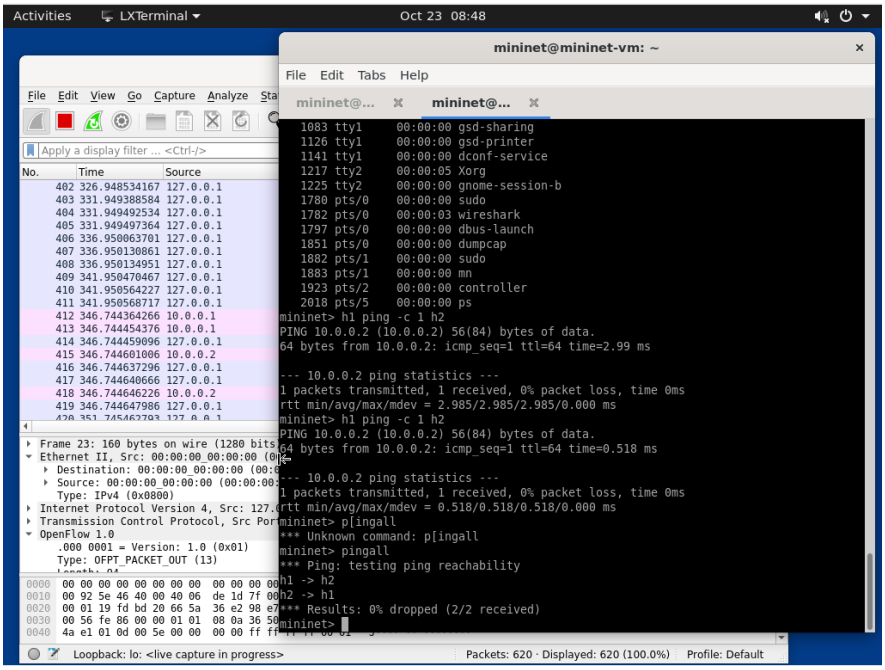


SC 7: Shows 1st Hosts ARPs for MAC address of 2nd, which shows PACKET IN and PACKET OUT.



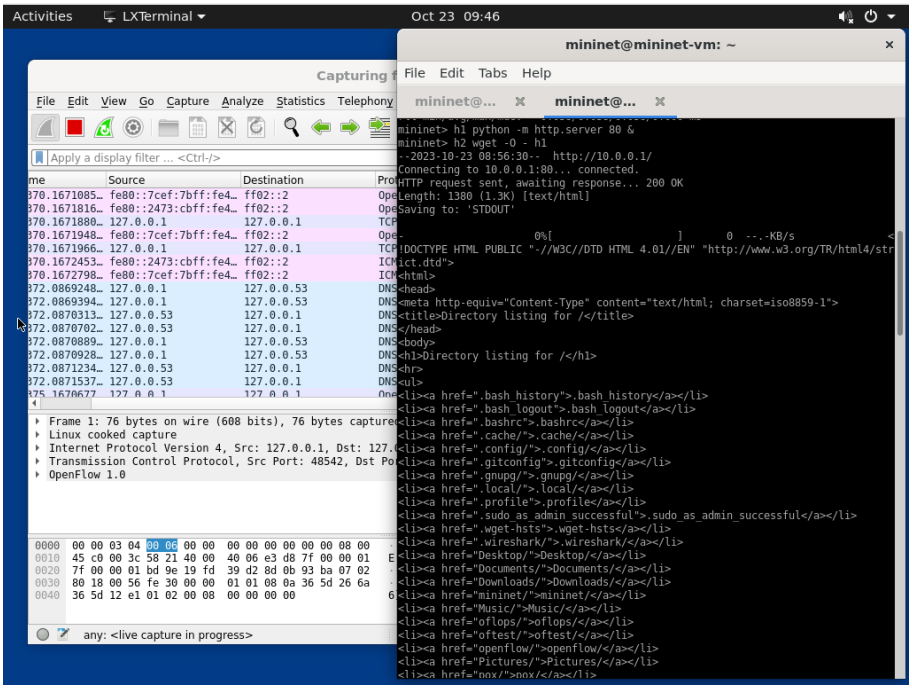
SC 8: running same command as SC 5 and shows much lower ping due to ICMP echo request, no control traffic was generated so packets immediately go to switch.

Aviv Y
Jillepalli
HA04
CS 455



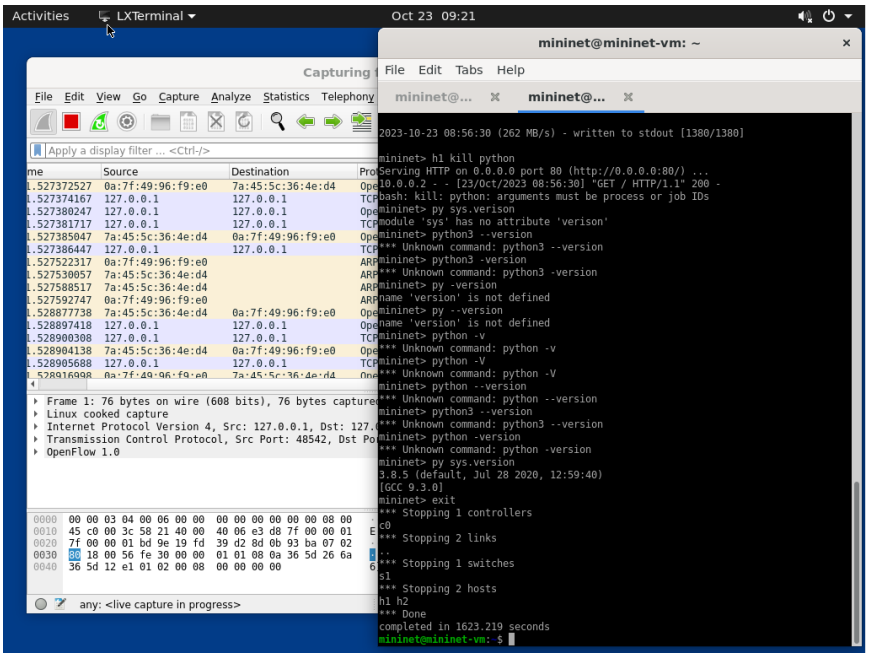
SC 9: Ran `pingall` which is an easier way to ping "all" of points in the topology.

Run a simple web server and client



SC 10: Shows running h1 `pyhton -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

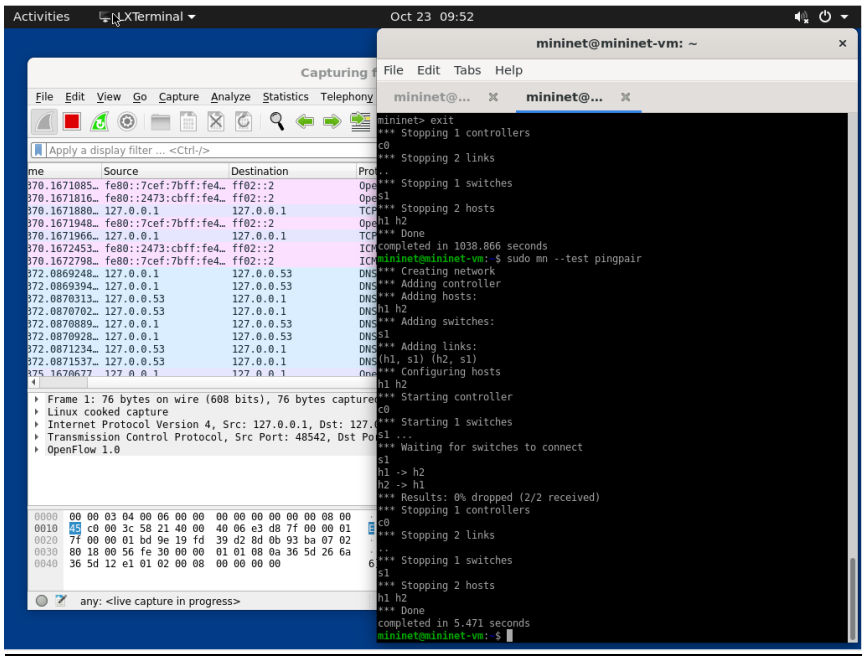


SC 11: Shows killing/shutting down the (web) server.

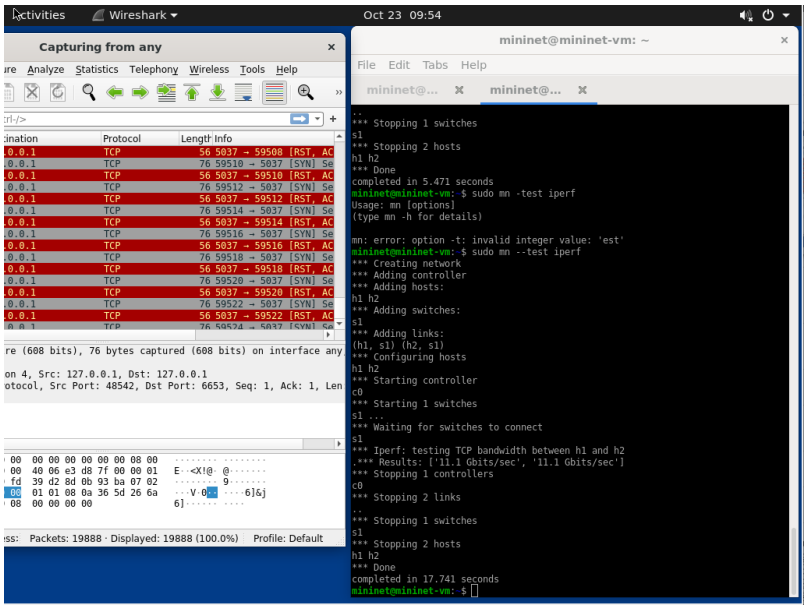
Aviv Y
Jillepalli
HA04
CS 455

Part 2: Advanced Starup Options

Running Regression Tests



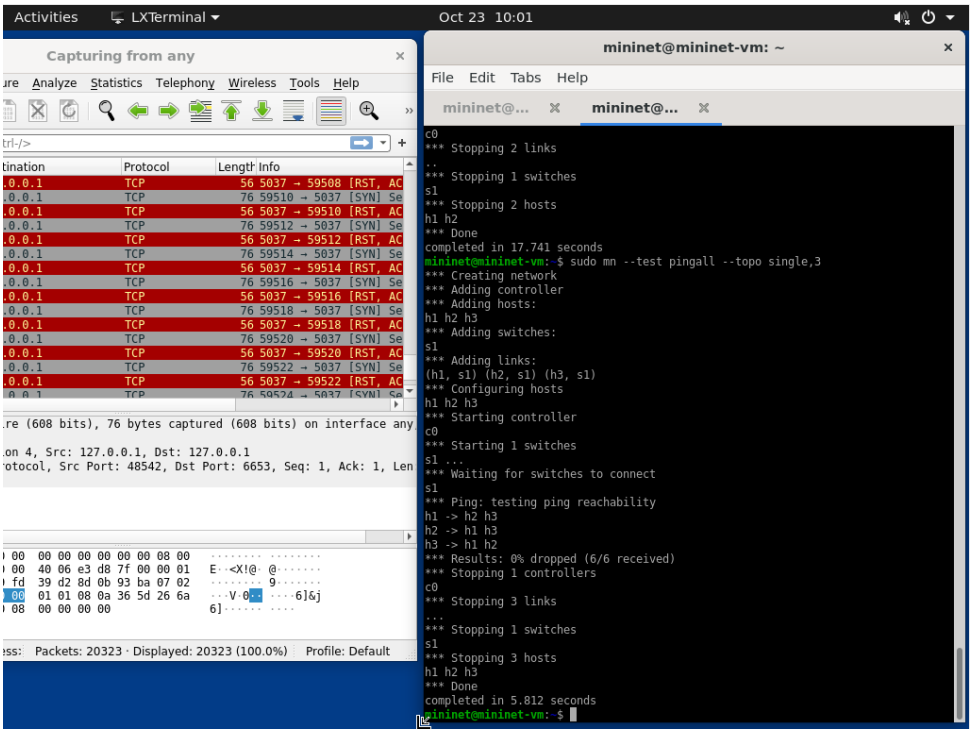
SC 12: Shows running command `sudo mn -test pingpair` which creates a minimal topology, ran an all-pairs-ping test, and broke down the topology and controller.



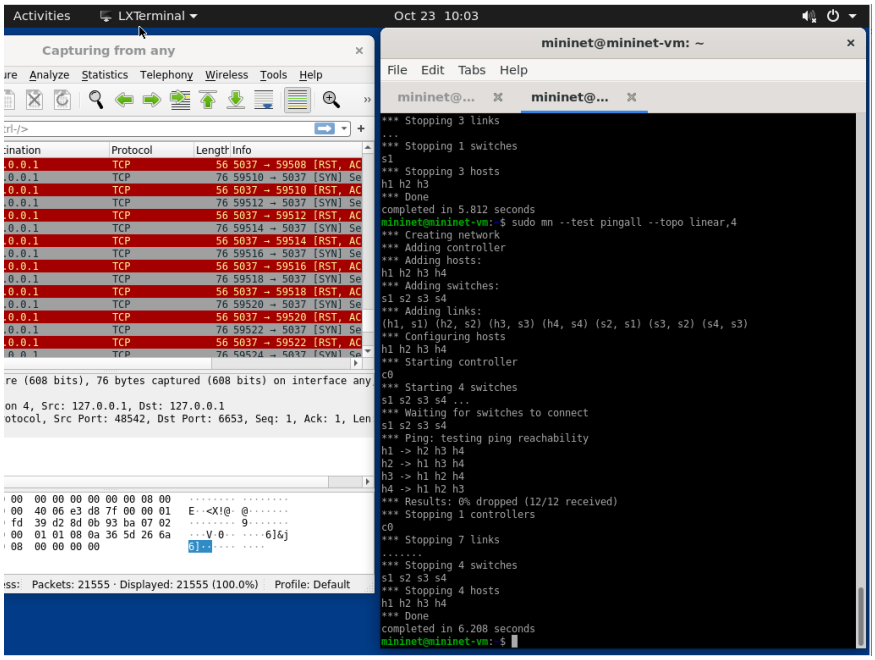
SC 13: The `sudo mn -test iperf` command creates the same Mininet, ran an iperf server on one host, ran an iperf client on the second host, and parsed the bandwidth achieved.

Aviv Y
Jillepalli
HA04
CS 455

Changing Topology Size and Type



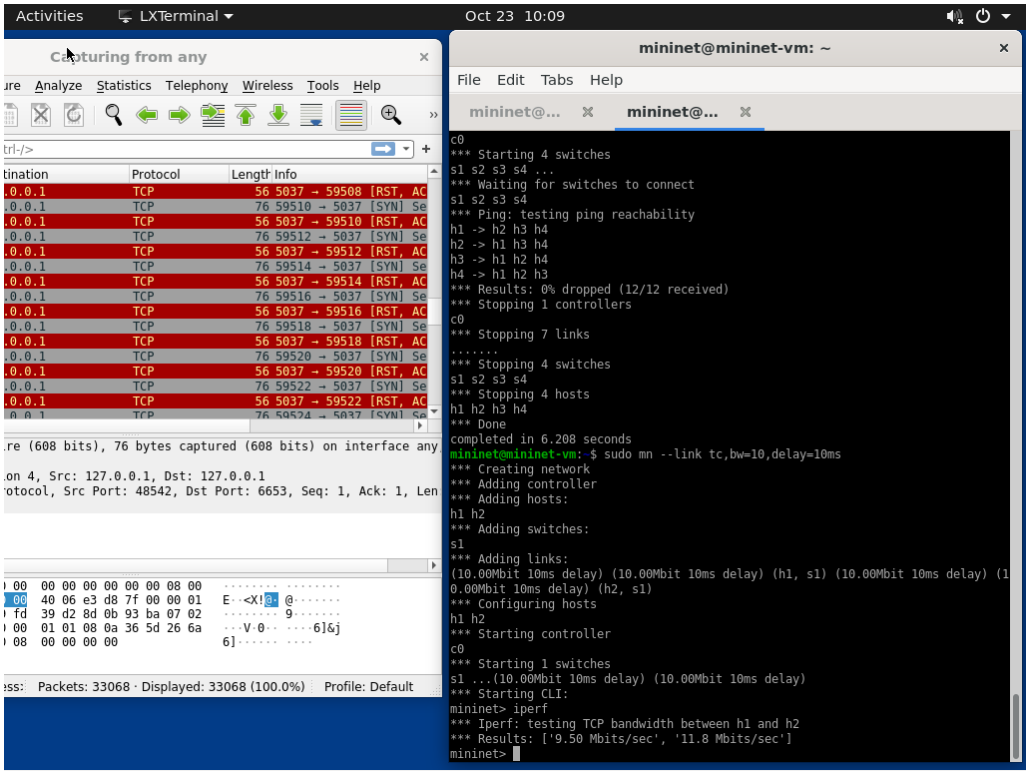
SC 14: Single topology test with 3 hosts.



SC 15: Linear topology test with 4 hosts.

Aviv Y
Jillepalli
HA04
CS 455

Link Variations



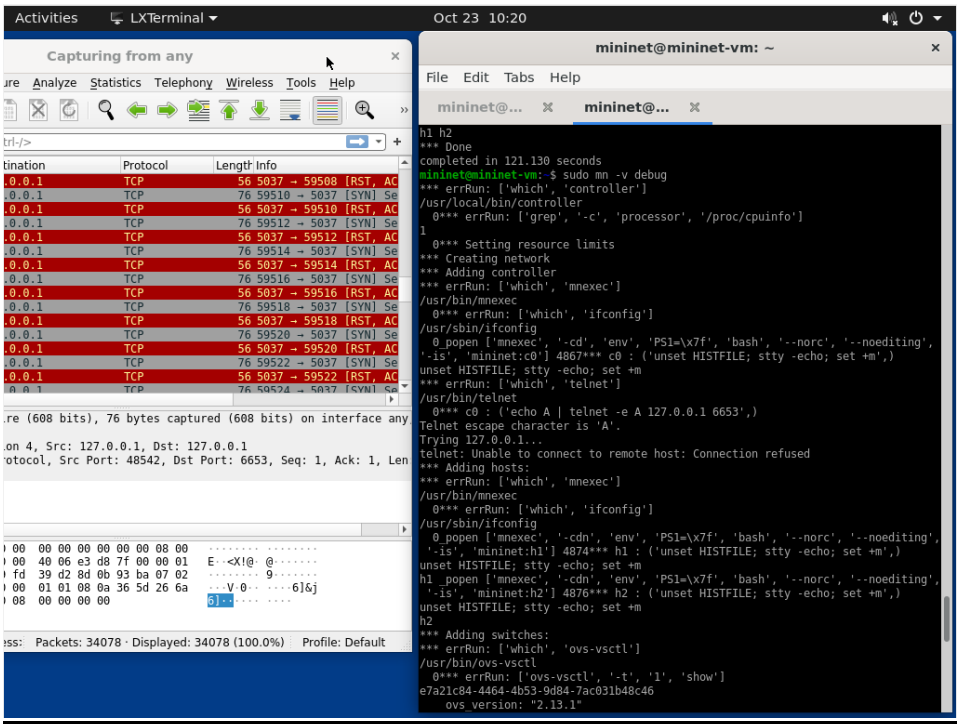
(The SC above runs `sudo mn –link tc,bw=10,delay=10ms`)

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

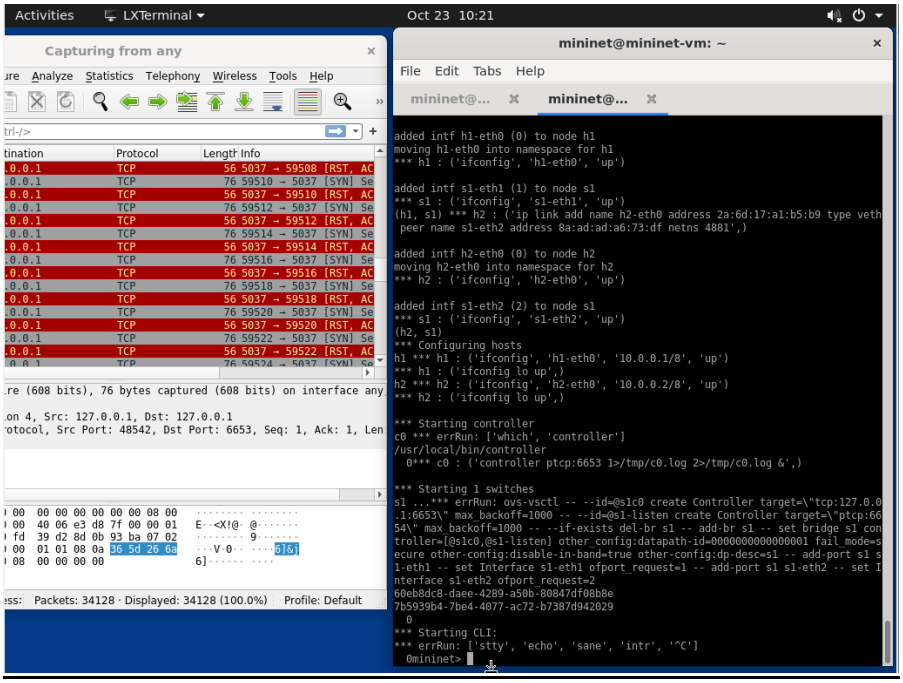
More info here.

Aviv Y
Jillepalli
HA04
CS 455

Adjustable Verbosity



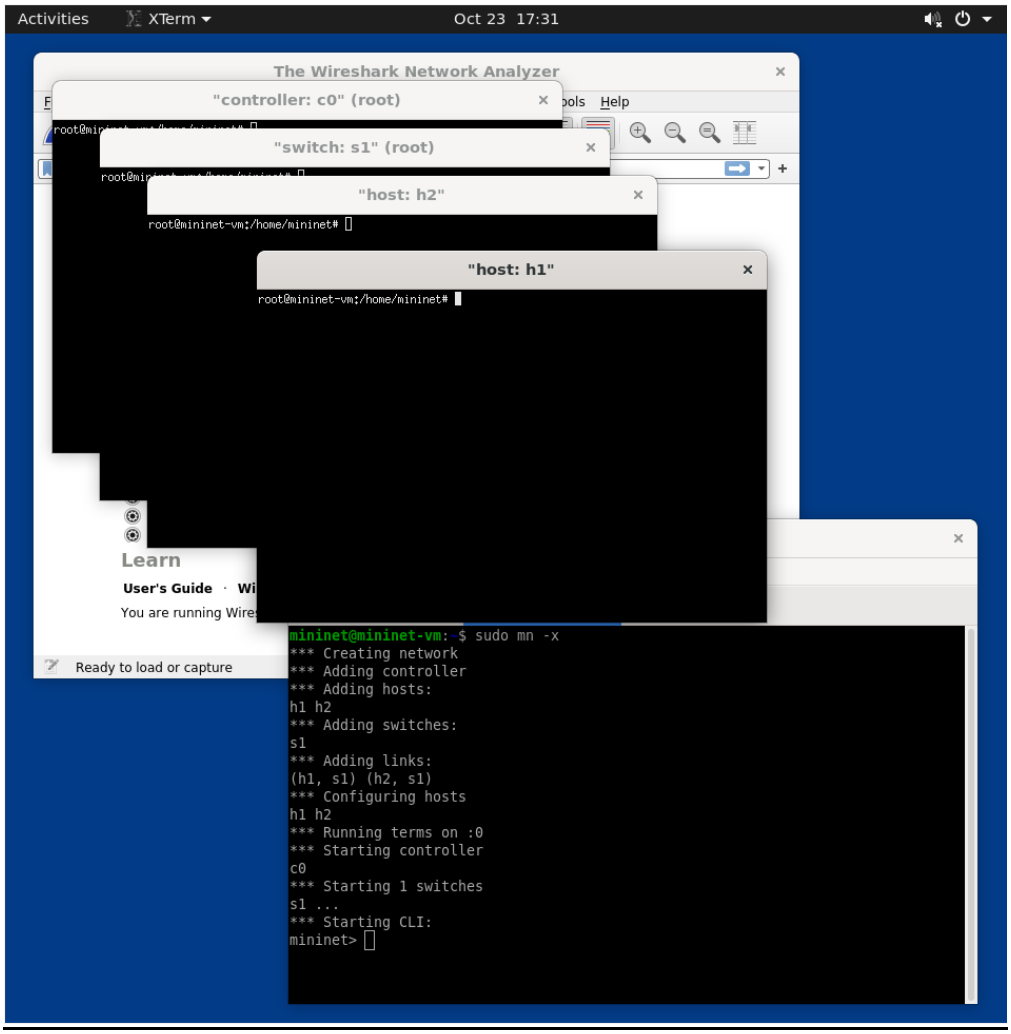
SC 17



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

Aviv Y
Jillepalli
HA04
CS 455

XTerm Display



Aviv Y
Jillepalli
HA04
CS 455

SC 19

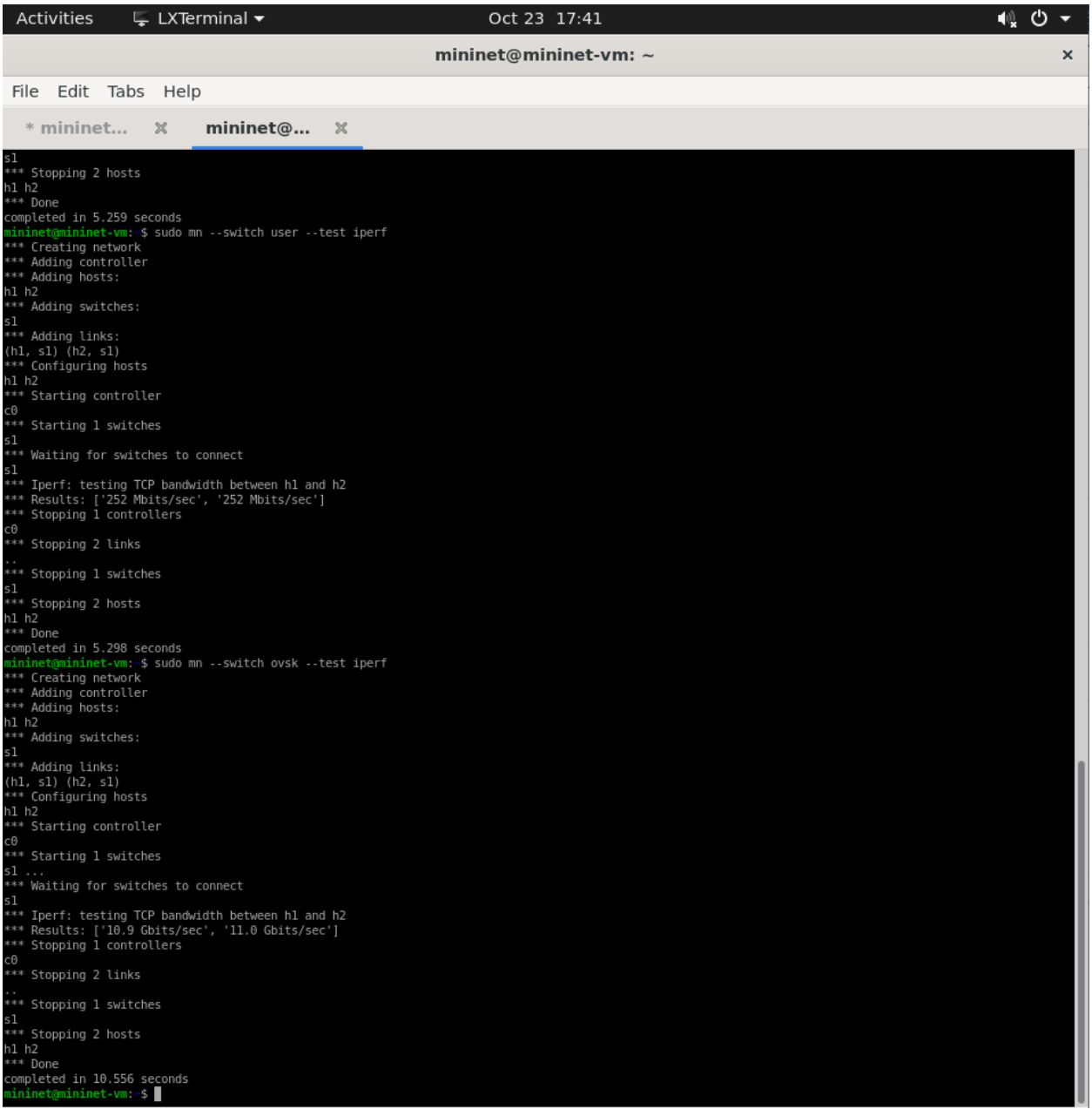
SC 18 & 19: runs command `sudo mn -x` and `ovs-ofctl dump-flows tcp:127.0.0.1:6654`.

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

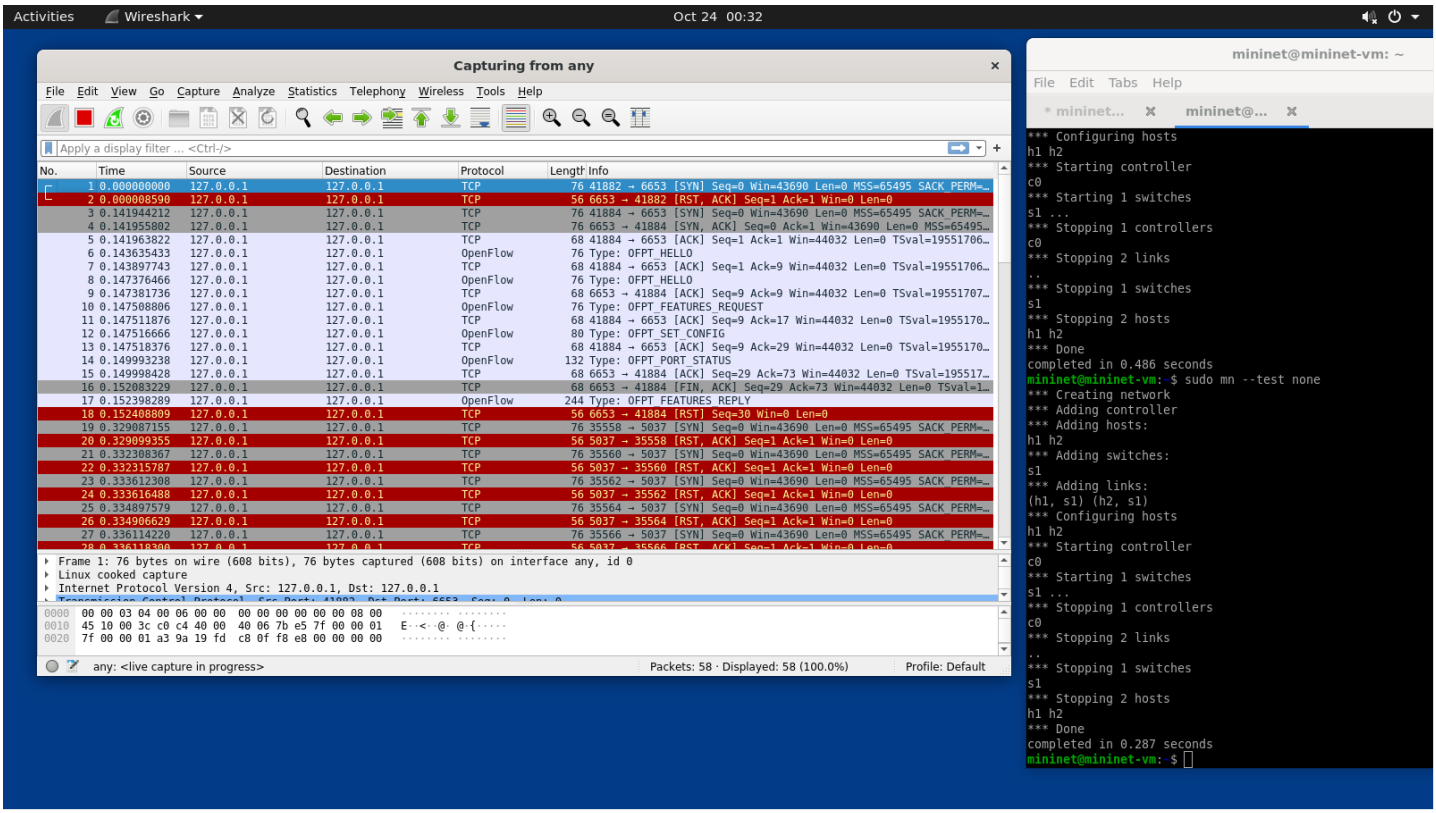
Other Switch Types



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

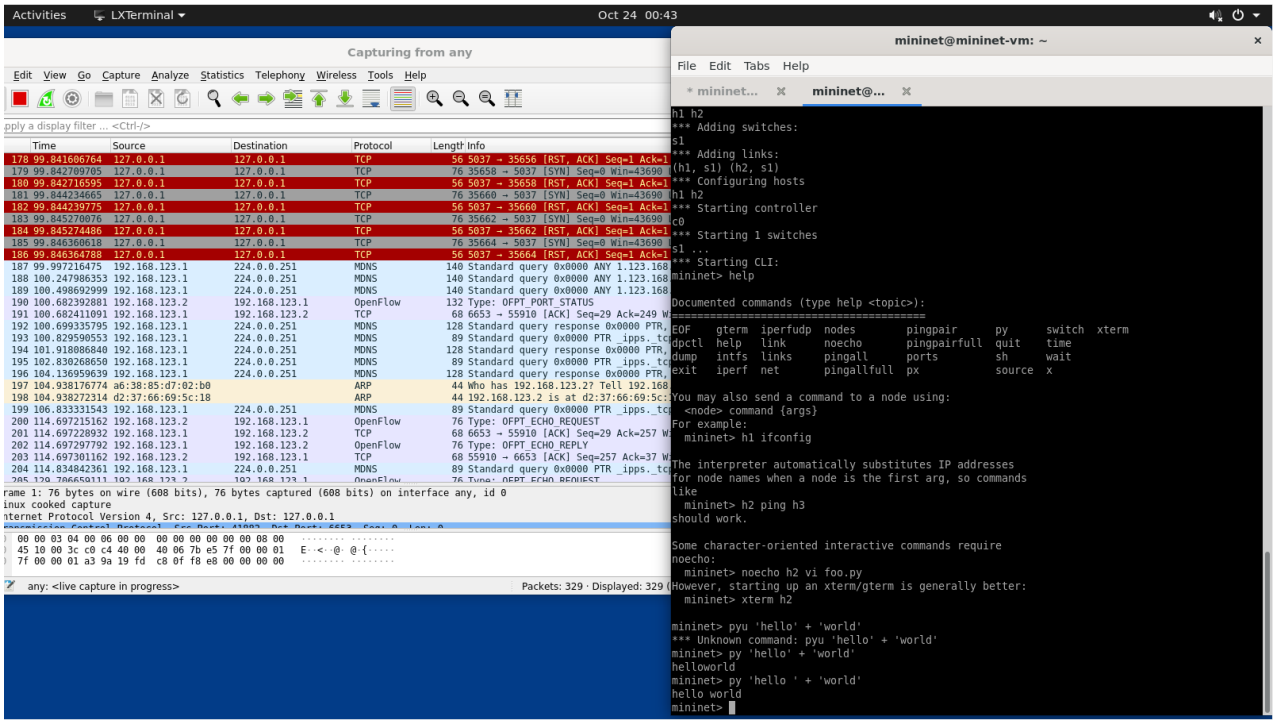
Everything in its own Namespace (user switch only)



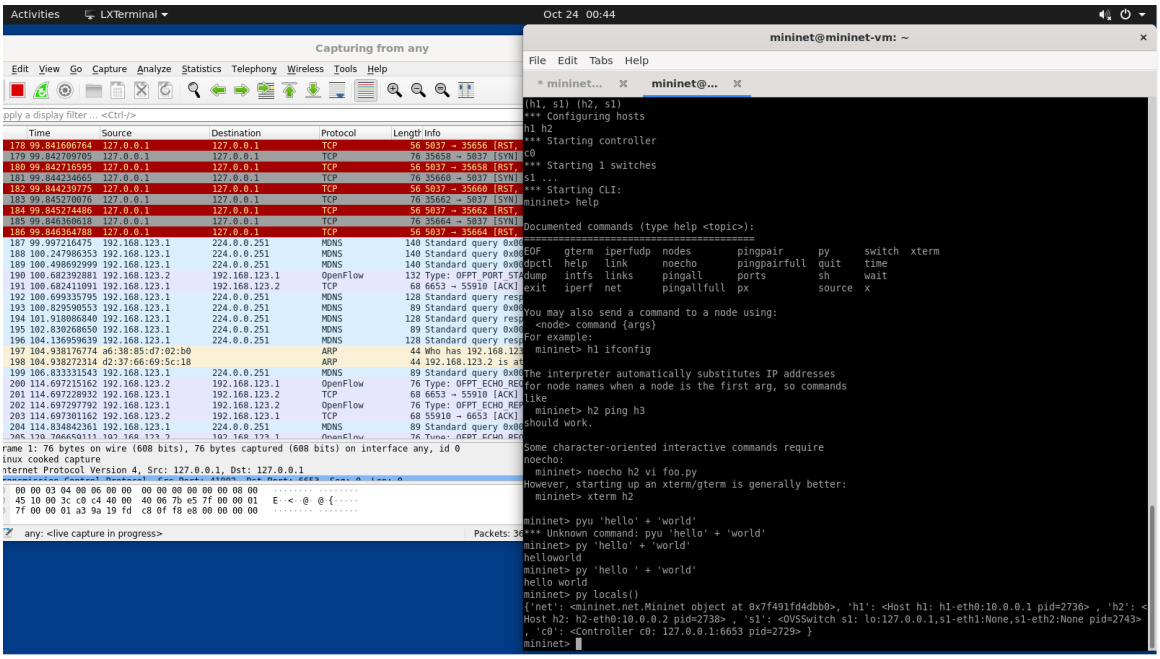
SC 21: The command `sudo mn --innamespace --switch user` puts switches in their own namespace as by default, hosts are put in their own space while switches and the controller are in the root namespace.

Aviv Y
Jillepalli
HA04
CS 455

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

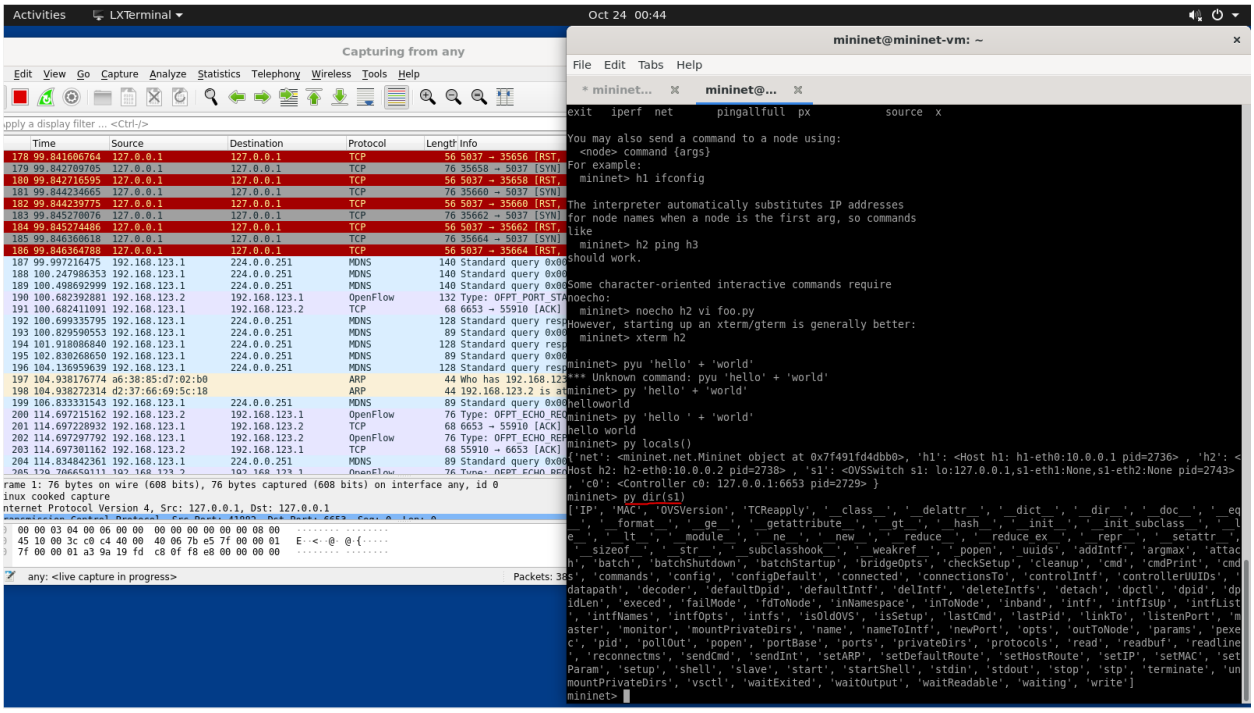


SC 22: `py 'hello ' + 'world'` shows that `py` extends Mininet, as well as probing its inner workings/.

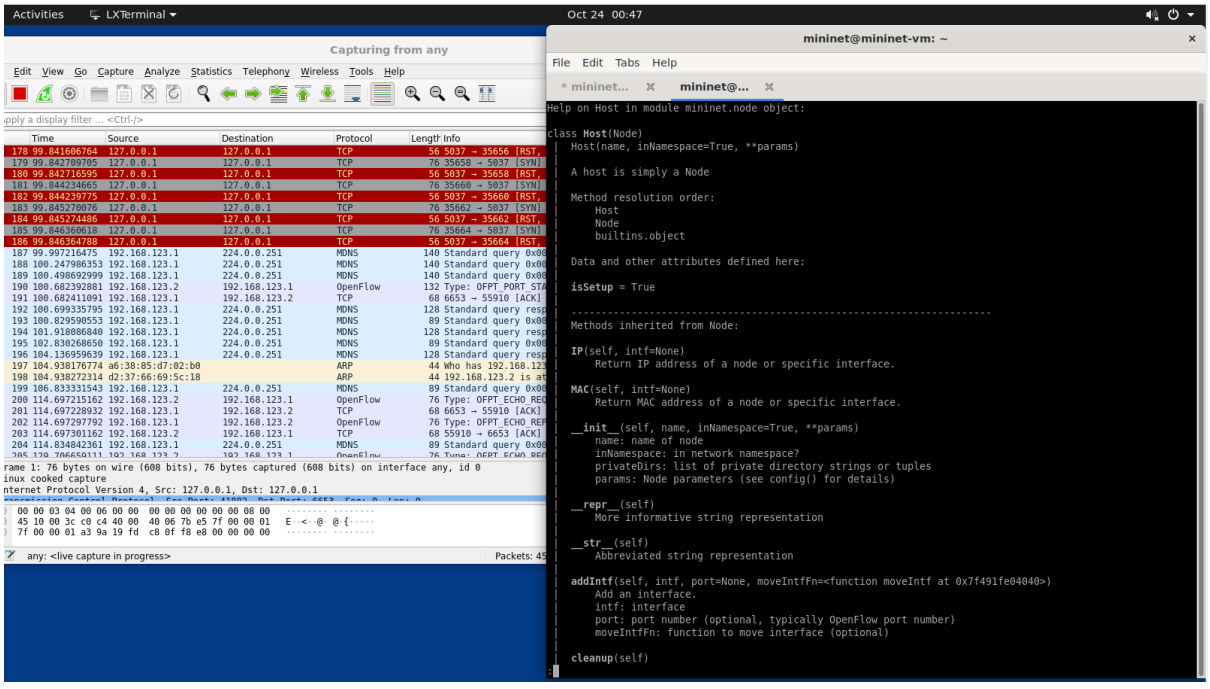


SC 23: `py locals()` prints the accessible local variables.

Aviv Y
Jillepalli
HA04
CS 455

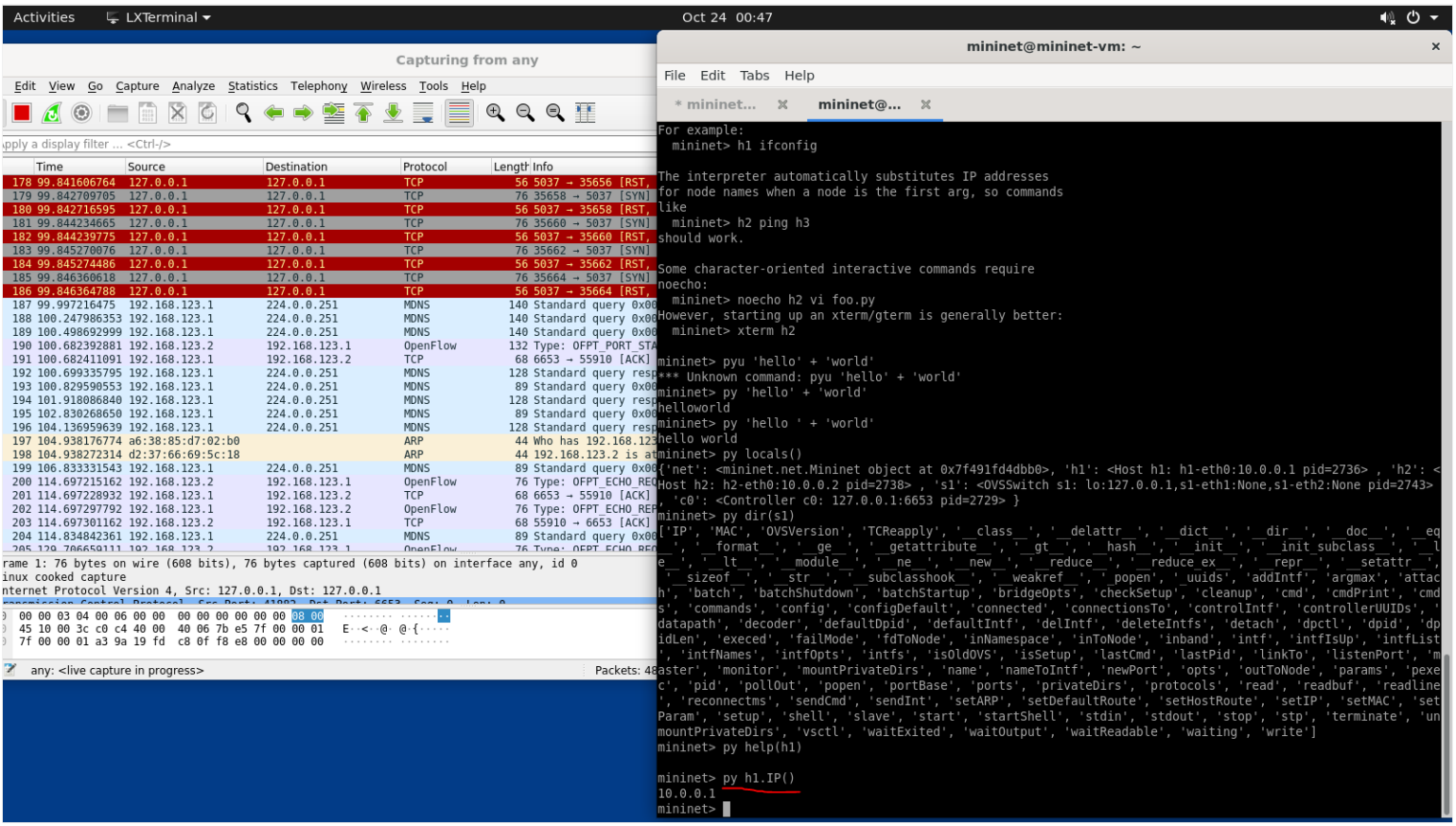


SC 24: `py dir(s1)` shows the methods and properties available for a node using the `dir()` function.



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

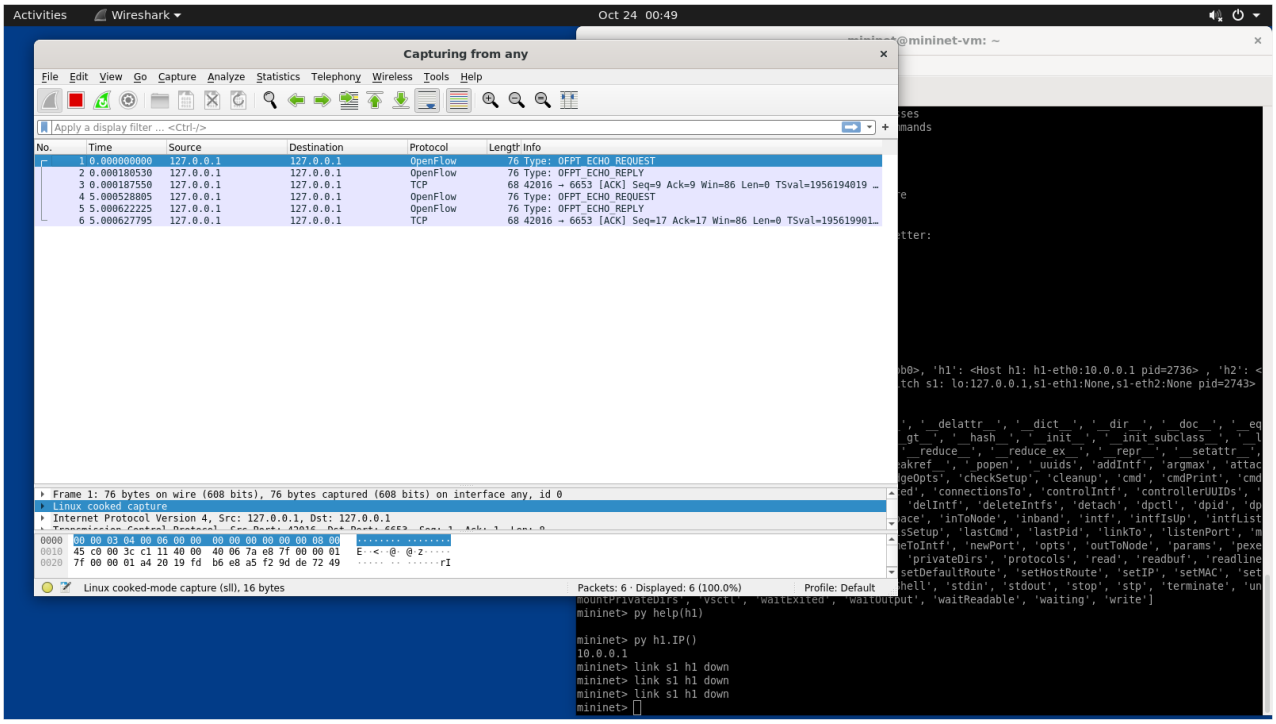
Aviv Y
Jillepalli
HA04
CS 455



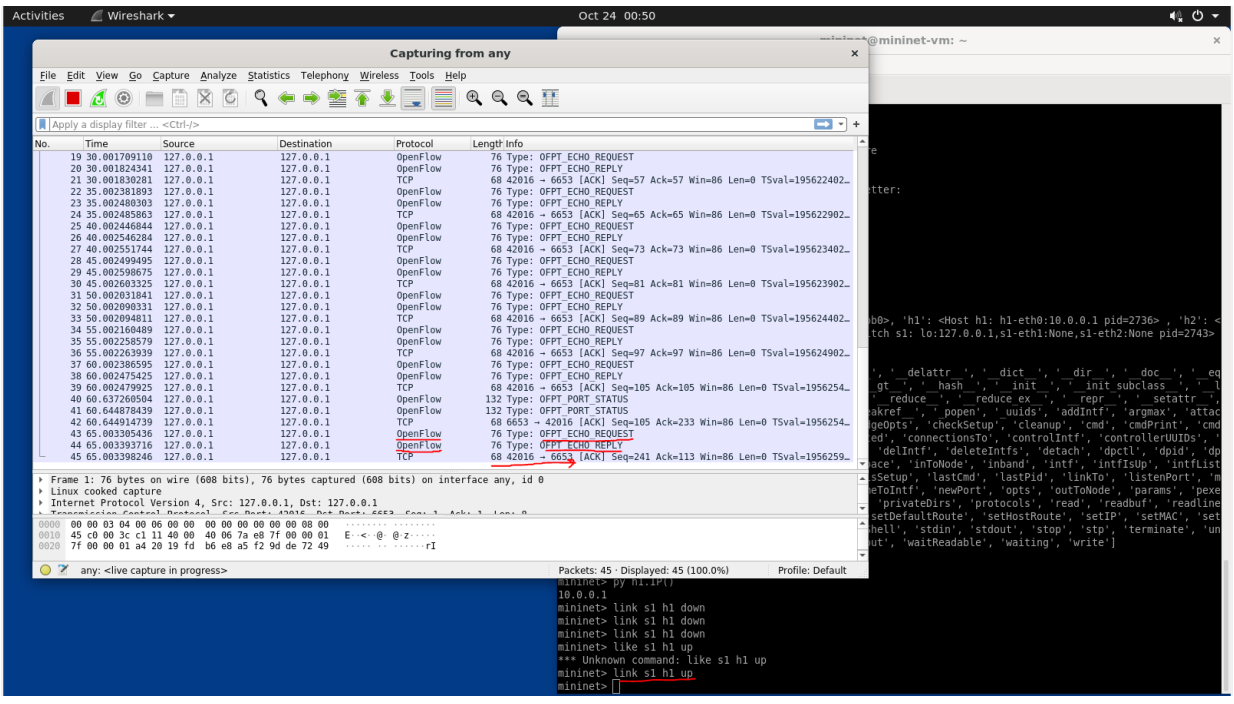
SC 26: Running the `py h1.IP()` command, this lets us evaluate methods of variables, in this command, it prints the hosts IP Address.

Aviv Y
Jillepalli
HA04
CS 455

Link Up/Down



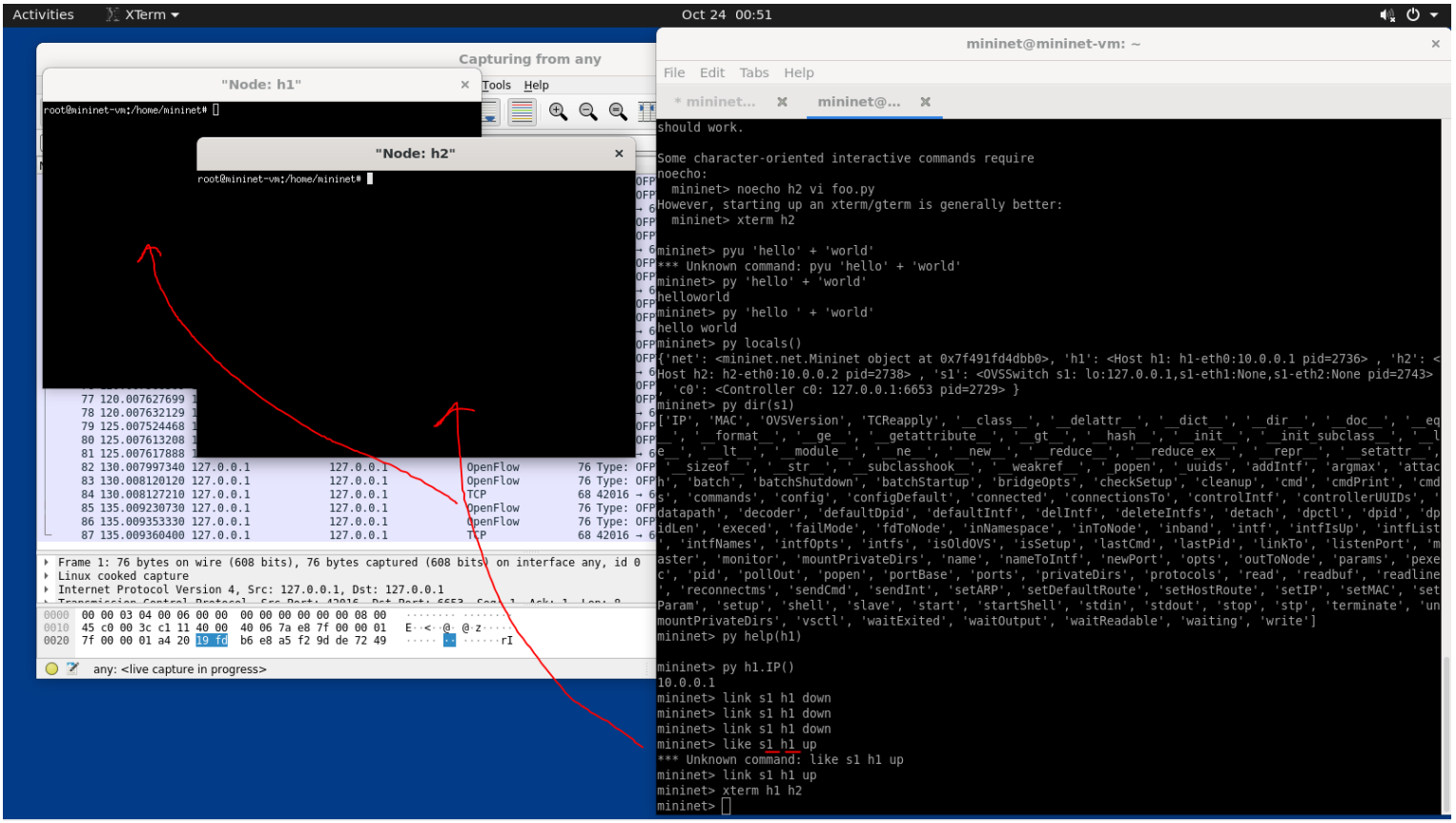
SC 27: The `link s1 h1 down` command disables both halves of the virtual ethernet pair. This is for fault tolerance testing.



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

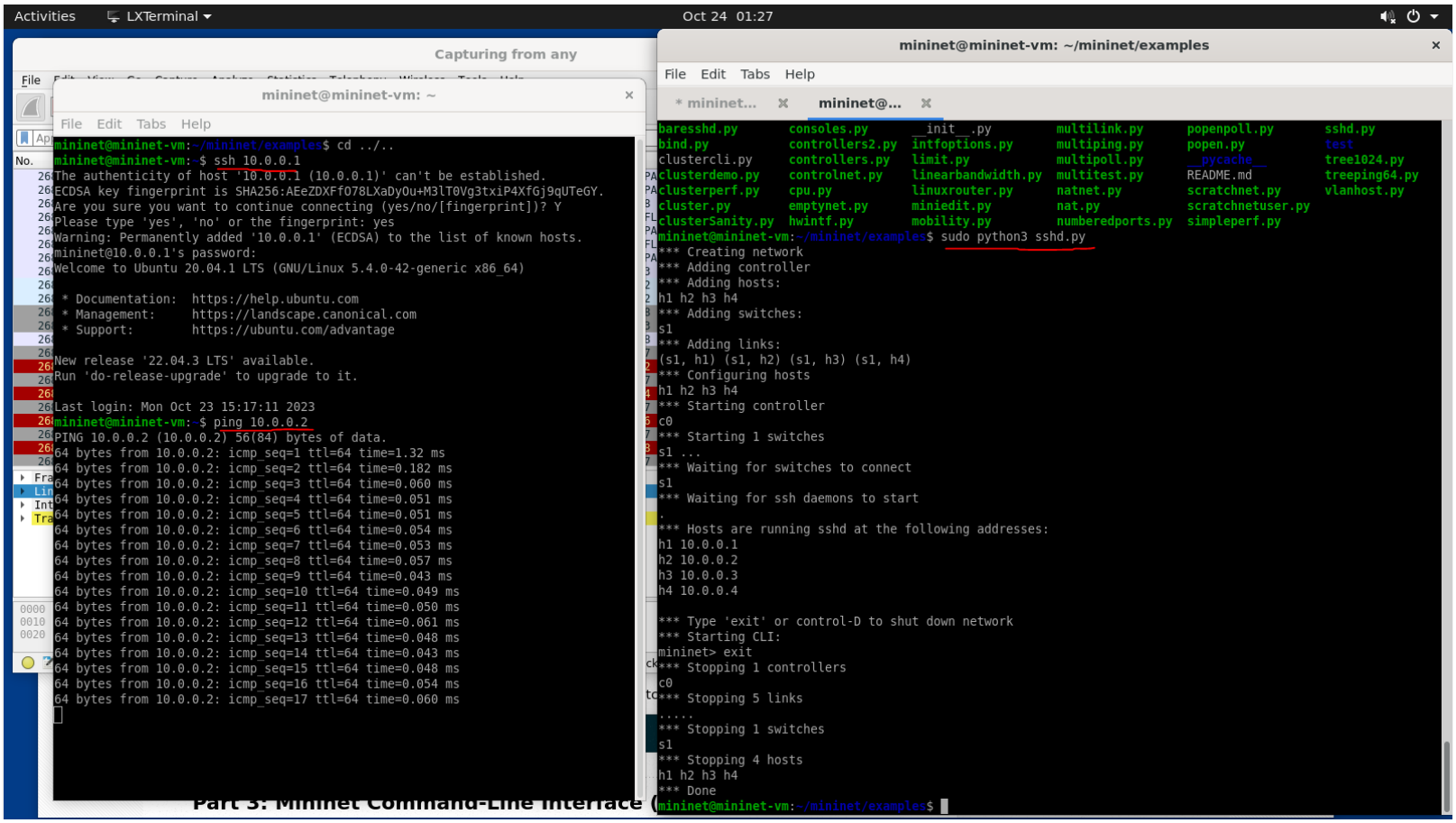


SC 29: `xterm h1 h2` displays xterm for host 1 (`h1`) and host 2 (`h2`).

Aviv Y
Jillepalli
HA04
CS 455

Part 4: Python API Examples

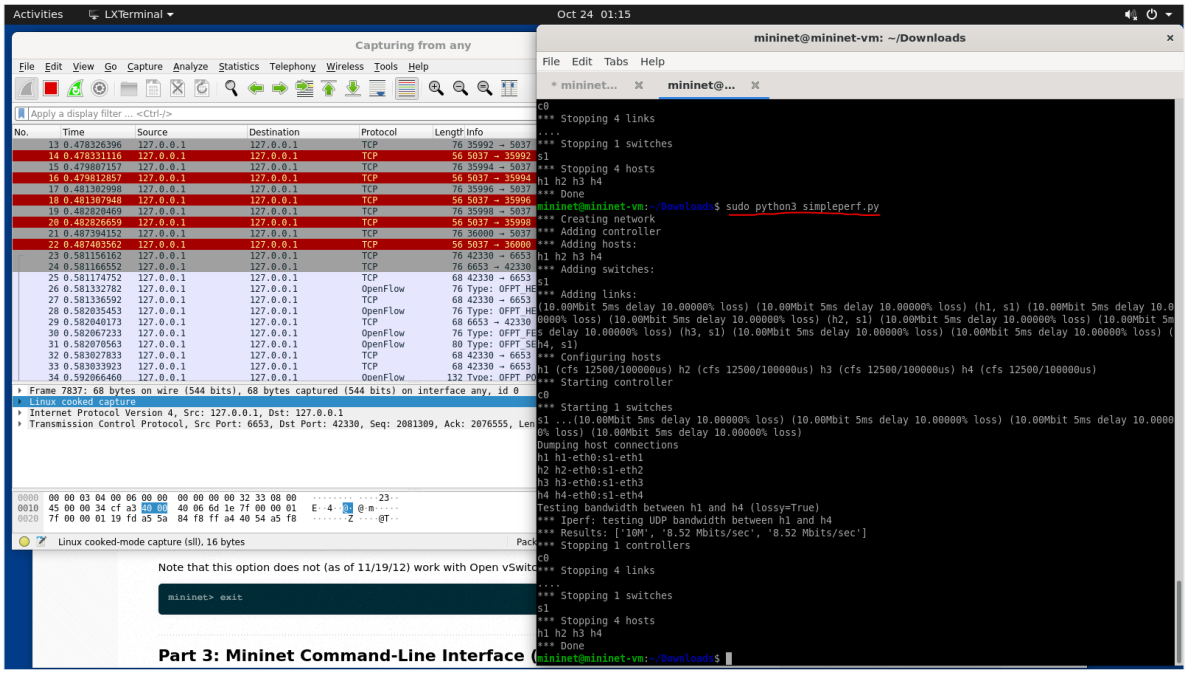
SSH Daemon per host



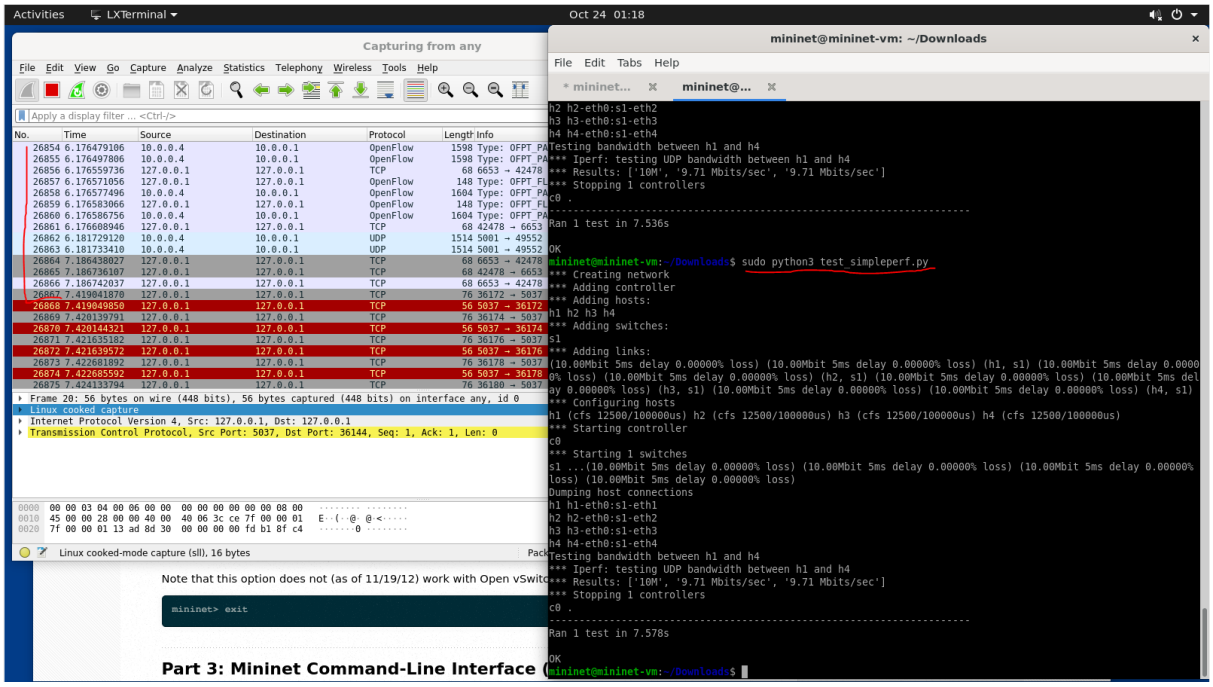
SC 30: Shows the SSH Daemon on every host. It's done by going into the Mininet/examples directory; `~/mininet/examples/sshd.py`. This is done by running `sudo python3 sshd.py` in the designated directory.

Aviv Y
Jillepalli
HA04
CS 455

simpleperf.py & test_simpleperf.py



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



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 but 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.

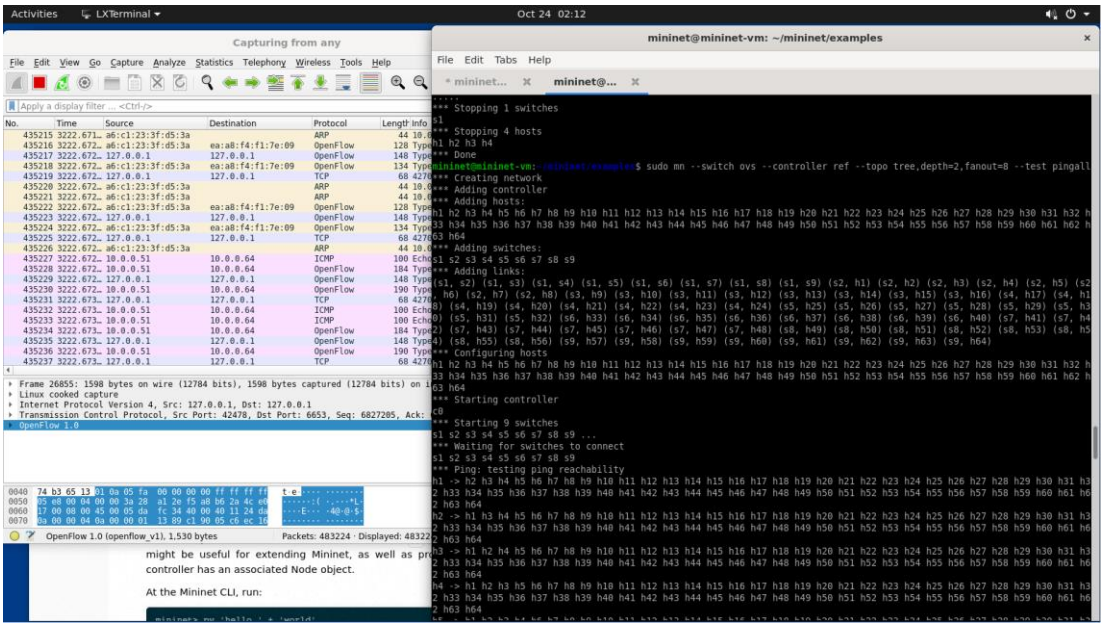
test_simpleperf.py

- 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.

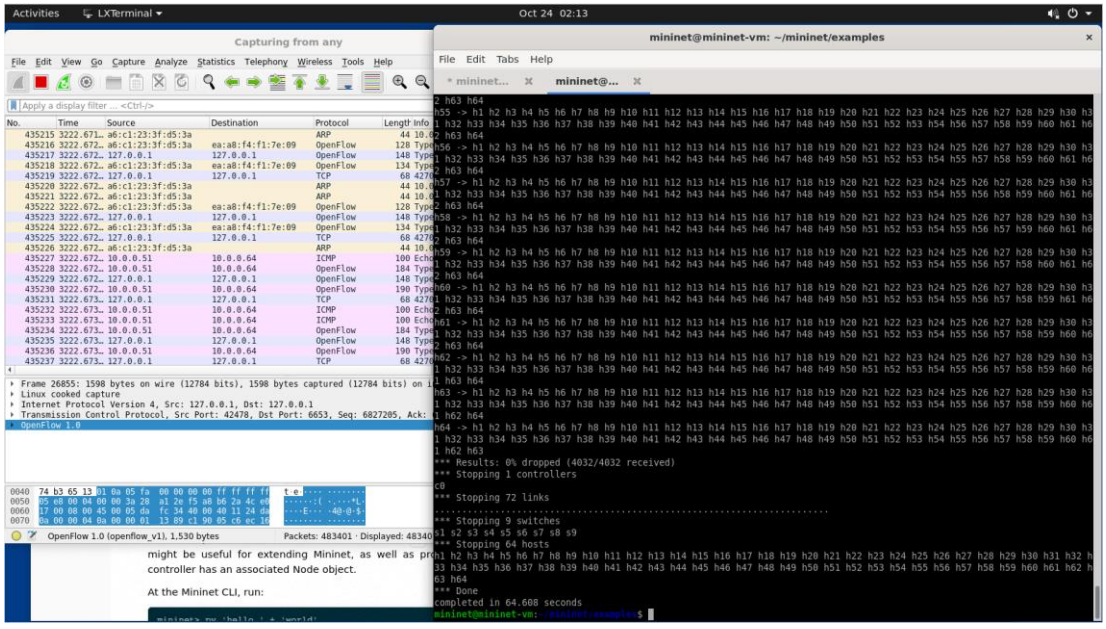
Aviv Y
Jillepalli
HA04
CS 455

Workflow

Creating a Network



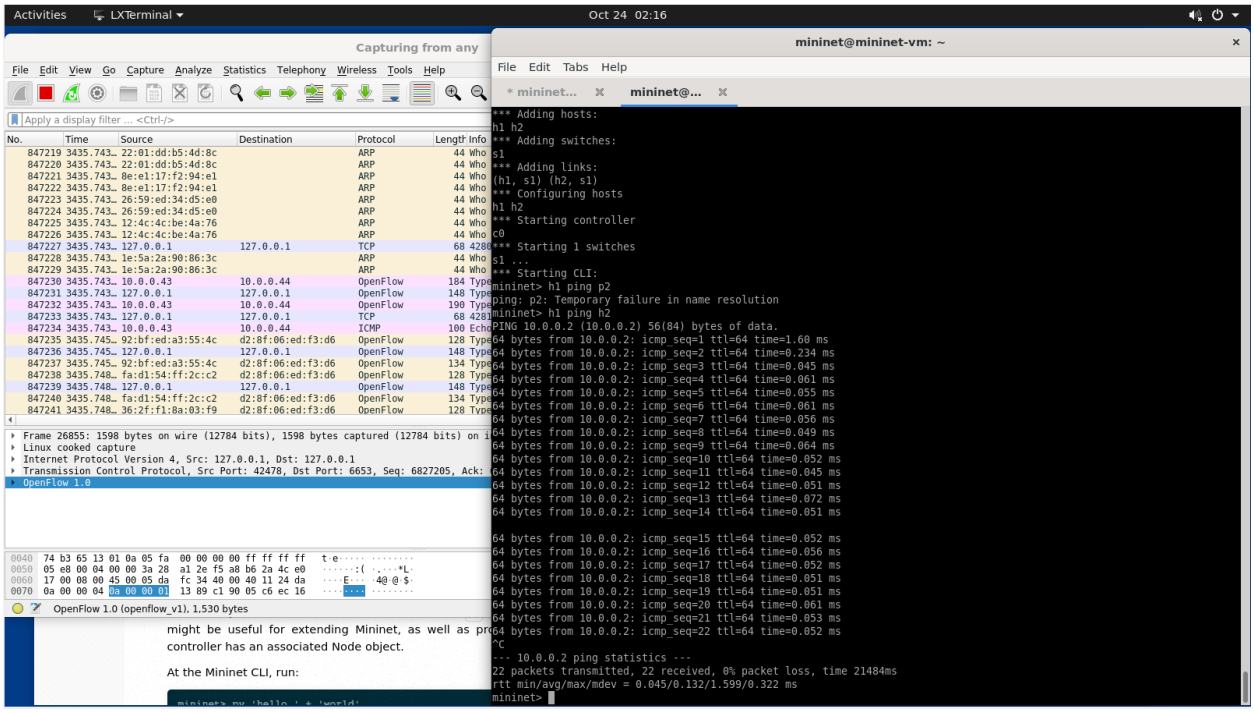
SC 33: The command `sudo mn --switch ovs --controller ref --topo tree,depth=2,fanout=8 --test pingall`, this creates a network with 9 switches and 64 hosts (b/c of depth of 2 and fanout of 8). Within this command, we are testing all points in the network with `pingall`.



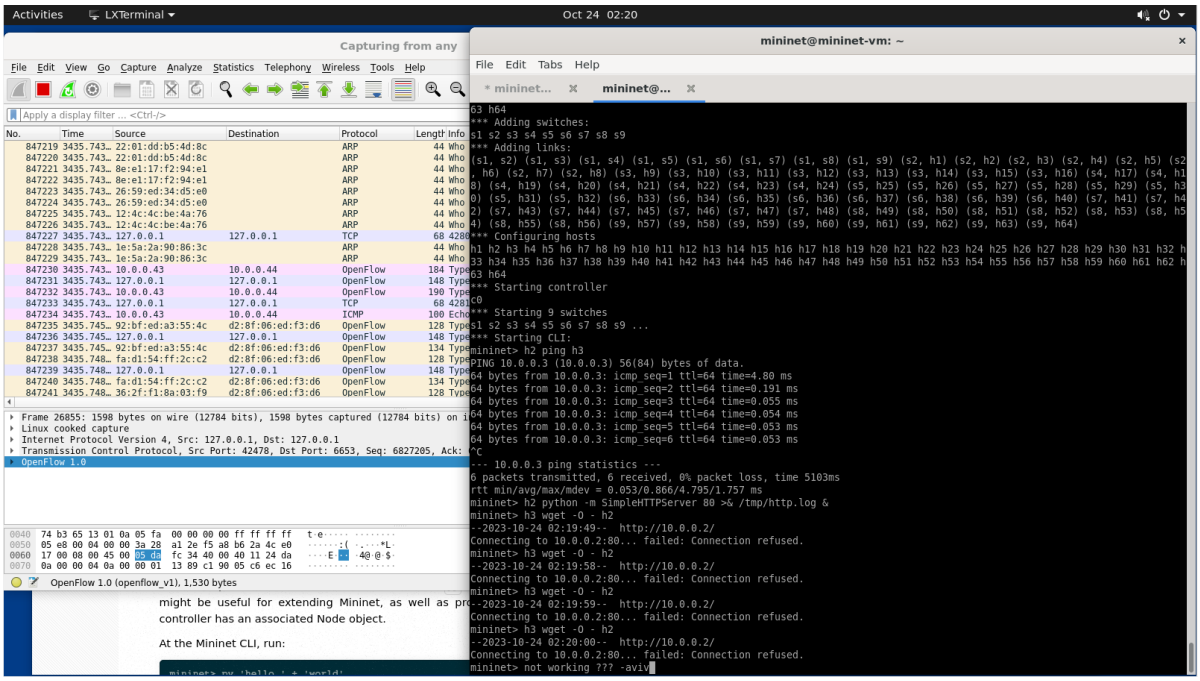
SC 34: end of result of command above.

Aviv Y
Jillepalli
HA04
CS 455

Interacting with a Network



SC 35: Runs `h1 ping h2`



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.