MAWLANA BHASHANI SCIENCE AND TECHNOLOGY UNIVERSITY

DEPARTMENT OF ICT

Lab Report No : 04

Course Code	: ICT-3208
Course Title	: Network Planning and Designing Lab
Lab Report name	: SDN Controllers and Mininet

Submitted by	***Submitted to***
Md. Faruk Hosen	Nazrul Islam
ID : IT-17035	Assistant Professor,
Session : 2016-2017	Department of ICT,MBSTU
Year : 3rd Semester : 2nd	Santosh,Tangail-1902

Date of Submission : 5 September 2020

Objective :

- Installl and use traffic generators as powerful tools for testing network performance.
- Install and configure SDN controller.
- Install and understand how the mininet simulator works.
- Implement and run basic examples for understanding the role of the controller and how it interact with mininet.

1. **What is iperf? Describe the features of iperf.**

Ans : **iperf:** iperf is a tool for active measurements of the maximum achievable bandwidth on IP networks. It supports tuning of various parameters related to timing, buffers and protocols(TCP, UDP, SCTP with IPv4 and IPv6). For each test it reports the bandwidth, loss, and other parameters.

Features of iperf :

- TCP and SCTP
  - ➢ Measure bandwidth
  - ➢ Report MSS/MTU size and observed read sizes.
- UDP
  - ➢ Client can create UDP streams of specified bandwidth.
  - ➢ Measure packet loss.
- Client and server can have multiple simultaneous connections (-P option).
- Server handles multiple connections, rather than quitting after a single test.
- Can run for specified time (-t option), rather than a set amount of data to transfer (-n or -k option).
- Run the server as a daemon (-D option).
- Use representative streams to test out how link layer compression affects your achievable bandwidth (-F option).

2. What is software defines networking(SDN) ? What is the difference between software defined networking and traditional networking?

Ans: **SDN :** SDN is that by separating control of network functions from hardware devices, administrators acquire more power to route and direct traffic in response to changing requirements.

The key difference between traditional and software-defined networking is how SDNs handle data packets. In a traditional network, the way a switch handles an incoming data packet is written into its firmware.SDN provides admins with granular control over the way switches handle data, giving them the ability to automatically prioritize or block certain types of packets.

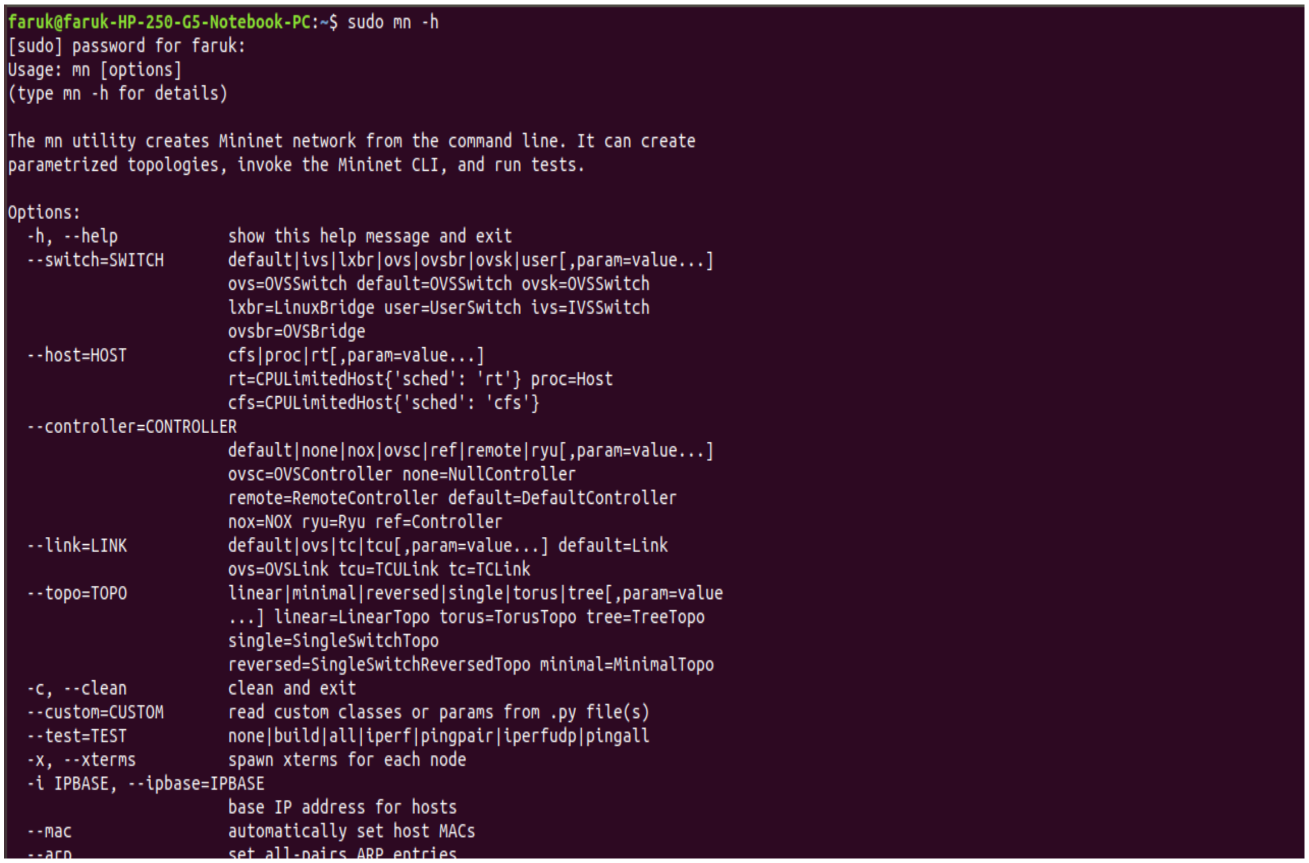
3. What is mininet and Why we use mininet?

Ans: **Mininet :** Mininet is a network emulation software that allows one to launch a virtual network with switches, host and SDN controller all with a single command.

Mininet creates a realistic virtual network, running real kernel, switch and application code, on a single machine (VM, cloud or native) Because you can easily interactwith your network using the Mininet CLI (and API), customize it, share it with others, or deploy it on real hardware, Mininet is useful for the development, teaching, and research. Mininet is also a great way to develop, share, and experiment with OpenFlow and Software-Defined Networking systems.

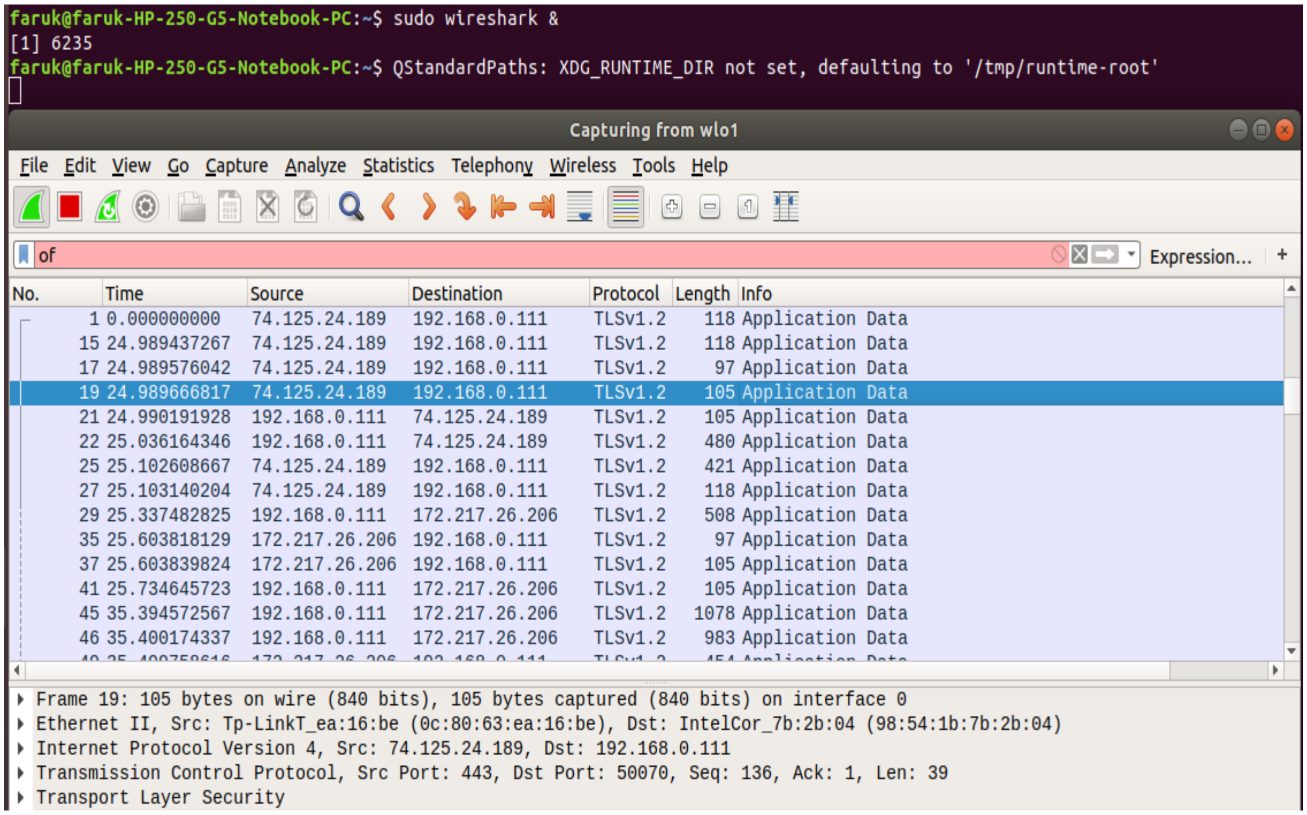
4. Some important Mininet Commands :

<u>Display startup options :</u> Type the following command to display a help message describing Mininet’s startup options:



```
faruk@faruk-HP-250-G5-Notebook-PC:~$ sudo mn -h
[sudo] password for faruk:
Usage: mn [options]
(type mn -h for details)

The mn utility creates Mininet network from the command line. It can create
parametrized topologies, invoke the Mininet CLI, and run tests.

Options:
  -h, --help            show this help message and exit
  --switch=SWITCH       default|ivs|lxbr|ovs|ovsbr|ovsk|user[,param=value...]
                        ovs=OVSSwitch default=OVSSwitch ovsk=OVSSwitch
                        lxbr=LinuxBridge user=UserSwitch ivs=IVSSwitch
                        ovsbr=OVSBridge
  --host=HOST           cfs|proc|rt[,param=value...]
                        rt=CPULimitedHost{'sched': 'rt'} proc=Host
                        cfs=CPULimitedHost{'sched': 'cfs'}
  --controller=CONTROLLER
                        default|none|nox|ovsc|ref|remote|ryu[,param=value...]
                        ovsc=OVSController none=NullController
                        remote=RemoteController default=DefaultController
                        nox=NOX ryu=Ryu ref=Controller
  --link=LINK           default|ovs|tc|tcu[,param=value...] default=Link
                        ovs=OVSLink tcu=TCULink tc=TCLink
  --topo=TOPO           linear|minimal|reversed|single|torus|tree[,param=value
                        ...] linear=LinearTopo torus=TorusTopo tree=TreeTopo
                        single=SingleSwitchTopo
                        reversed=SingleSwitchReversedTopo minimal=MinimalTopo
  -c, --clean           clean and exit
  --custom=CUSTOM       read custom classes or params from .py file(s)
  --test=TEST           none|build|all|iperf|pingpair|iperfudp|pingall
  -x, --xterms          spawn xterms for each node
  -i IPBASE, --ipbase=IPBASE
                        base IP address for hosts
  --mac                 automatically set host MACs
  --arp                 set all-pairs ARP entries
```

Start Wireshark: To view control traffic using the OpenFlow Wireshark dissector, first open wireshark in the background:

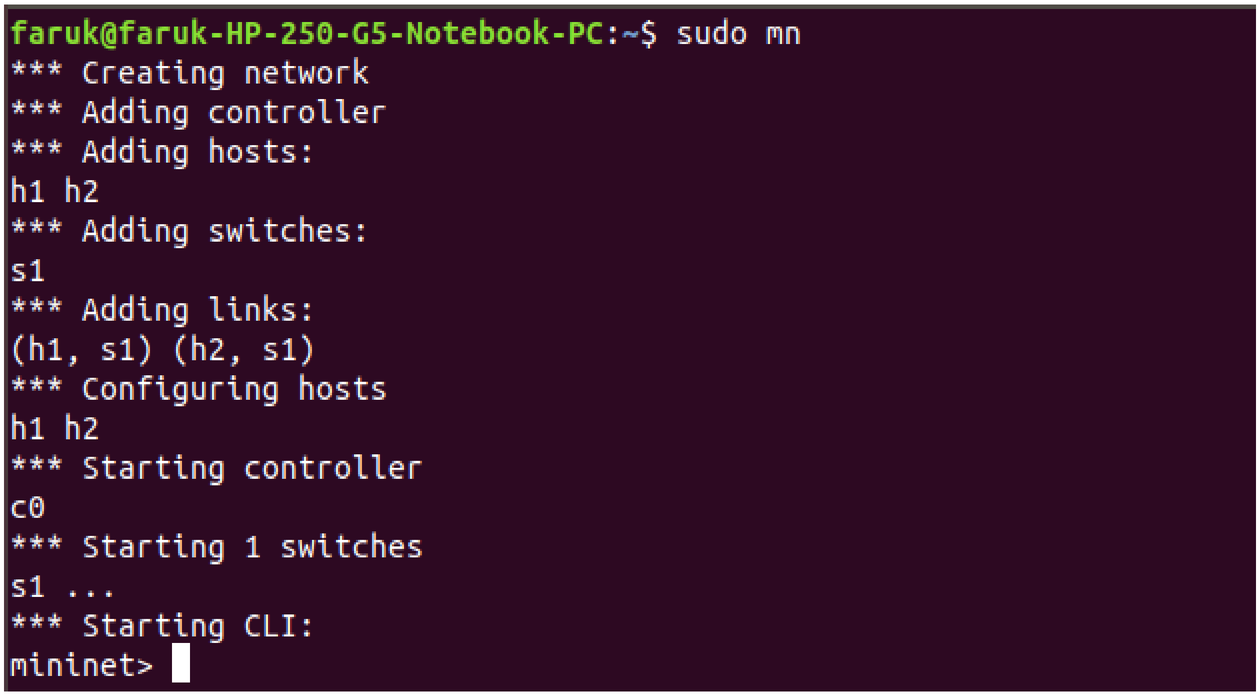


```
faruk@faruk-HP-250-G5-Notebook-PC:~$ sudo wireshark &
[1] 6235
faruk@faruk-HP-250-G5-Notebook-PC:~$ QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
```

Capturing from wlo1
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
of
Expression…

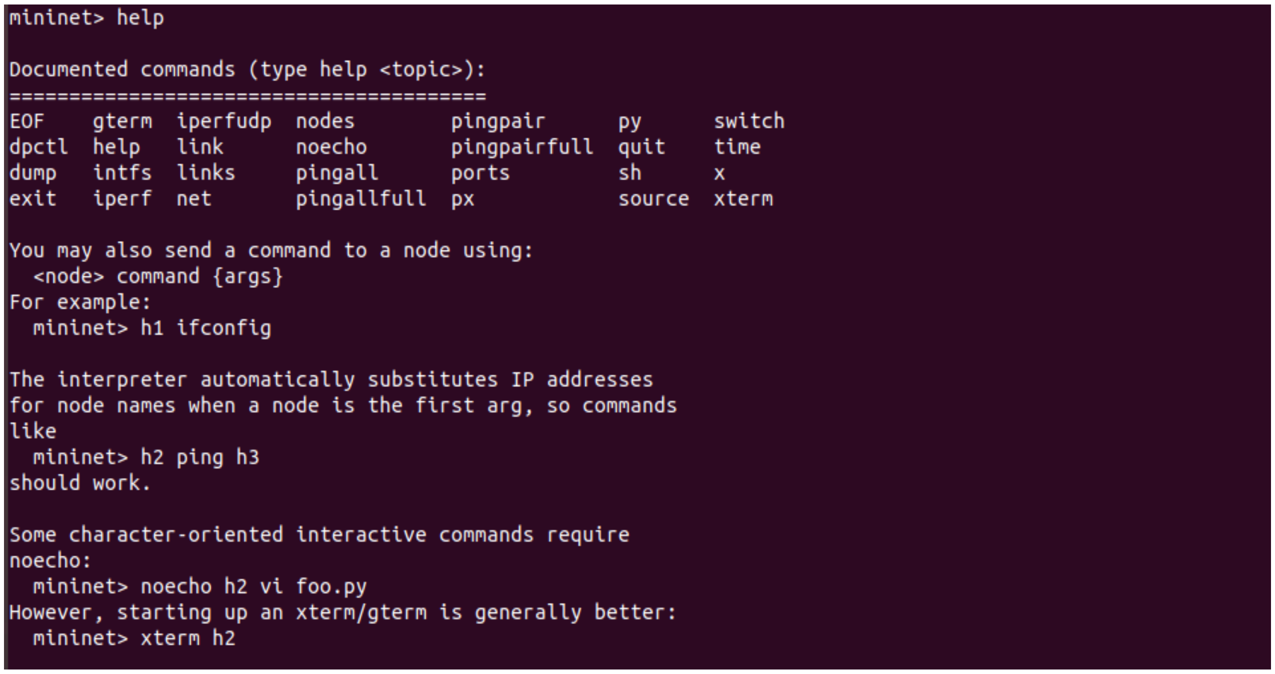
```
No.  Time          Source          Destination     Protocol Length Info
   1 0.000000000   74.125.24.189   192.168.0.111   TLSv1.2   118 Application Data
  15 24.989437267  74.125.24.189   192.168.0.111   TLSv1.2   118 Application Data
  17 24.989576042  74.125.24.189   192.168.0.111   TLSv1.2    97 Application Data
  19 24.989666817  74.125.24.189   192.168.0.111   TLSv1.2   105 Application Data
  21 24.990191928  192.168.0.111   74.125.24.189   TLSv1.2   105 Application Data
  22 25.036164346  192.168.0.111   74.125.24.189   TLSv1.2   480 Application Data
  25 25.102608667  74.125.24.189   192.168.0.111   TLSv1.2   421 Application Data
  27 25.103140204  74.125.24.189   192.168.0.111   TLSv1.2   118 Application Data
  29 25.337482825  192.168.0.111   172.217.26.206  TLSv1.2   508 Application Data
  35 25.603818129  172.217.26.206  192.168.0.111   TLSv1.2    97 Application Data
  37 25.603839824  172.217.26.206  192.168.0.111   TLSv1.2   105 Application Data
  41 25.734645723  192.168.0.111   172.217.26.206  TLSv1.2   105 Application Data
  45 35.394572567  192.168.0.111   172.217.26.206  TLSv1.2  1078 Application Data
  46 35.400174337  192.168.0.111   172.217.26.206  TLSv1.2   983 Application Data

Frame 19: 105 bytes on wire (840 bits), 105 bytes captured (840 bits) on interface 0
Ethernet II, Src: Tp-LinkT_ea:16:be (0c:80:63:ea:16:be), Dst: IntelCor_7b:2b:04 (98:54:1b:7b:2b:04)
Internet Protocol Version 4, Src: 74.125.24.189, Dst: 192.168.0.111
Transmission Control Protocol, Src Port: 443, Dst Port: 50070, Seq: 136, Ack: 1, Len: 39
Transport Layer Security
```

<u>Interact with Hosts and switches :</u> Start a minimal topology and enter the CLI:



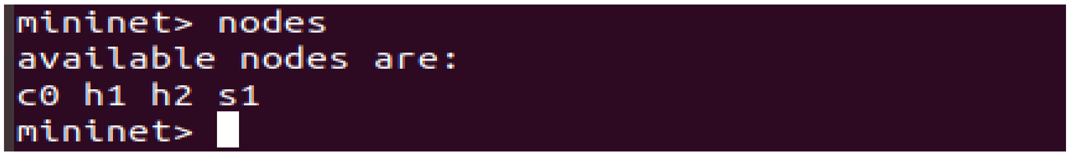
```
faruk@faruk-HP-250-G5-Notebook-PC:~$ sudo mn
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

Display Mininet CLI commands:



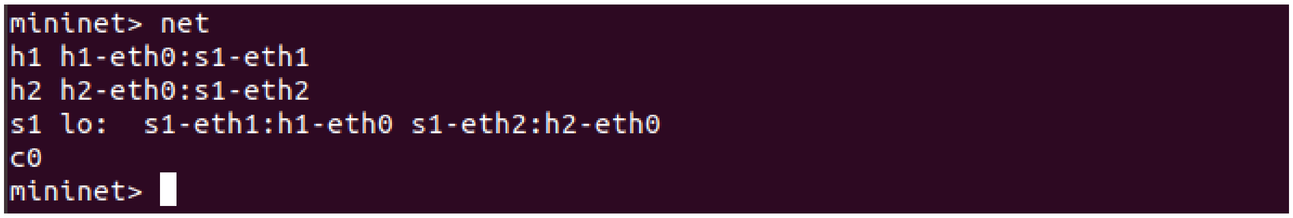
```
mininet> help

Documented commands (type help <topic>):
========================================
EOF    gterm  iperfudp  nodes        pingpair      py      switch
dpctl  help   link      noecho       pingpairfull  quit    time
dump   intfs  links     pingall      ports         sh      x
exit   iperf  net       pingallfull  px            source  xterm

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
```

Display nodes:



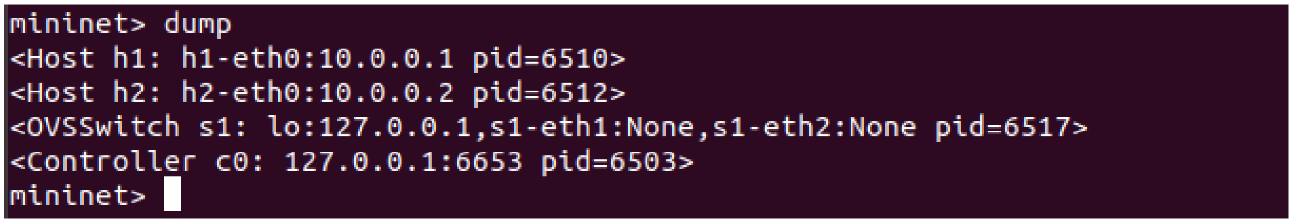
```
mininet> nodes
available nodes are:
c0 h1 h2 s1
mininet>
```

Display links:



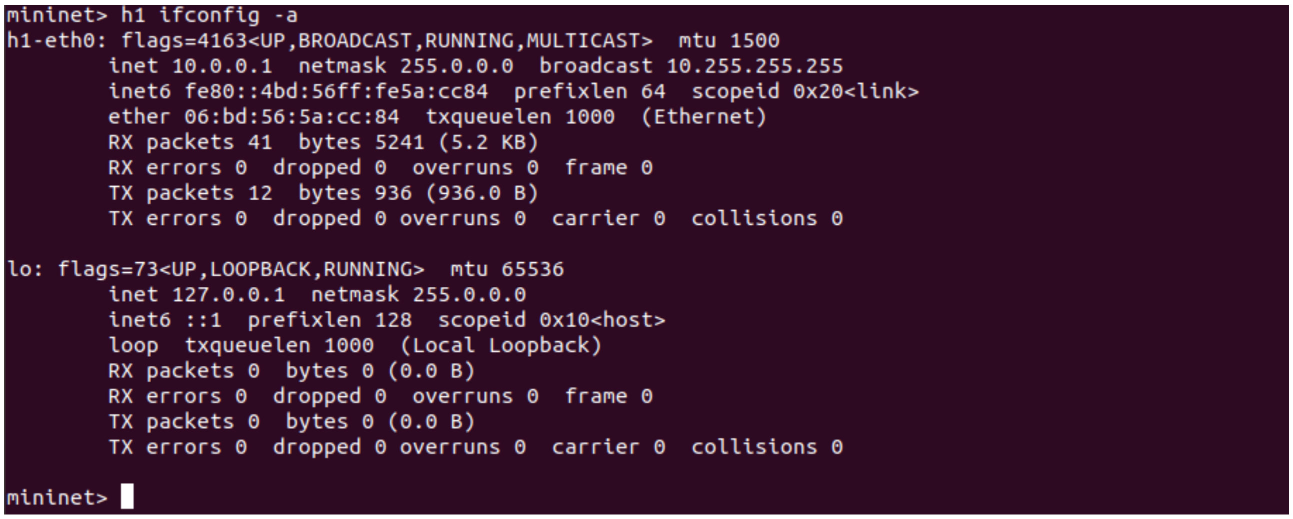
```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0
c0
mininet>
```

Dump information about all nodes:



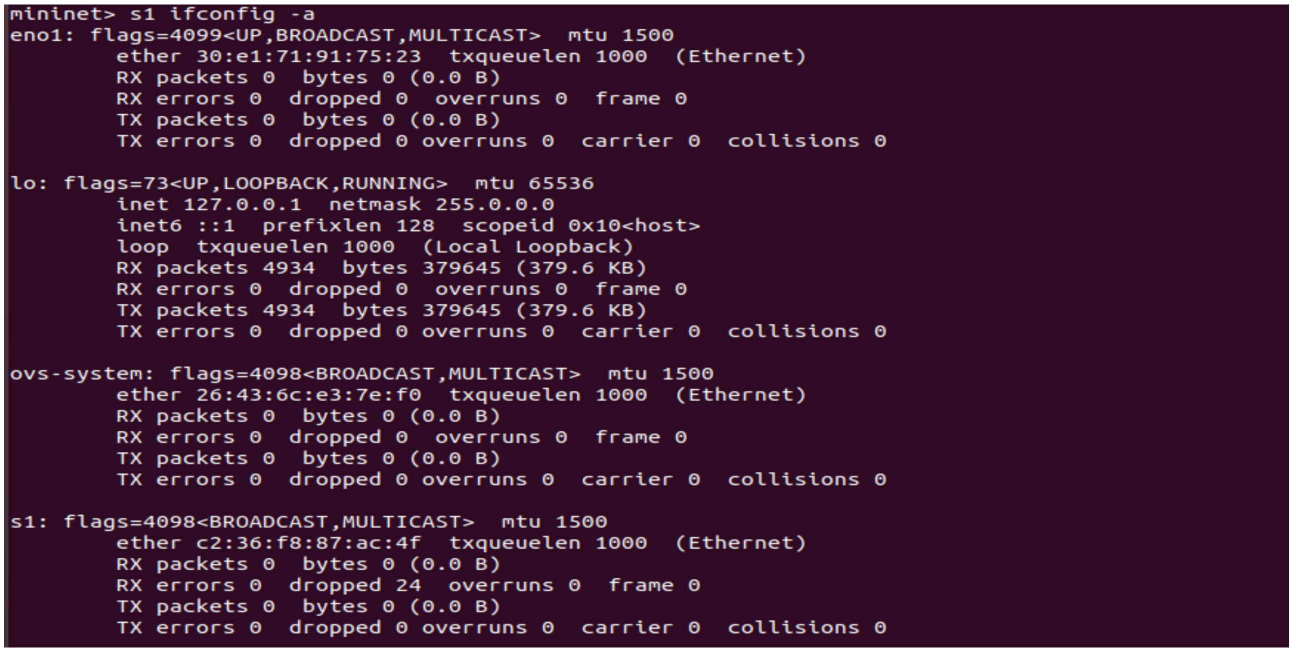
```
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=6510>
<Host h2: h2-eth0:10.0.0.2 pid=6512>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=6517>
<Controller c0: 127.0.0.1:6653 pid=6503>
mininet>
```

Run a command on a host process:



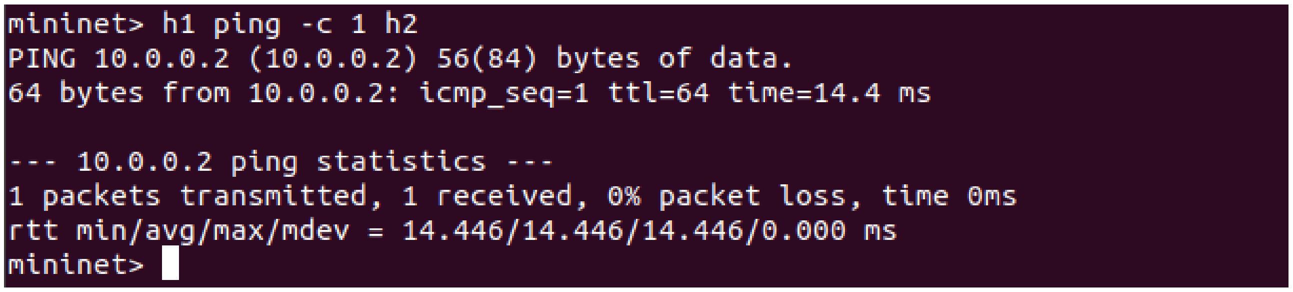
```
mininet> h1 ifconfig -a
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.1  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::4bd:56ff:fe5a:cc84  prefixlen 64  scopeid 0x20<link>
        ether 06:bd:56:5a:cc:84  txqueuelen 1000  (Ethernet)
        RX packets 41  bytes 5241 (5.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 12  bytes 936 (936.0 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>
```

This command will show the switch interfaces, plus the VM’s connection out (eth0):



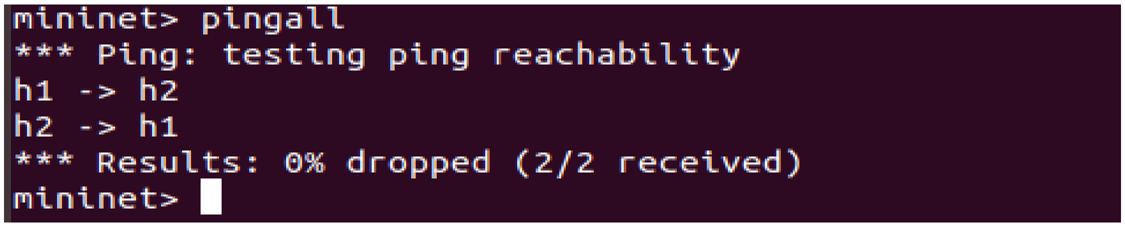
```
mininet> s1 ifconfig -a
eno1: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 30:e1:71:91:75:23  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

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 4934  bytes 379645 (379.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4934  bytes 379645 (379.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ovs-system: flags=4098<BROADCAST,MULTICAST>  mtu 1500
        ether 26:43:6c:e3:7e:f0  txqueuelen 1000  (Ethernet)
        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

s1: flags=4098<BROADCAST,MULTICAST>  mtu 1500
        ether c2:36:f8:87:ac:4f  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 24  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

<u>Test connectivity between hosts:</u> Now, verify that one can ping from host 0 to host 1:



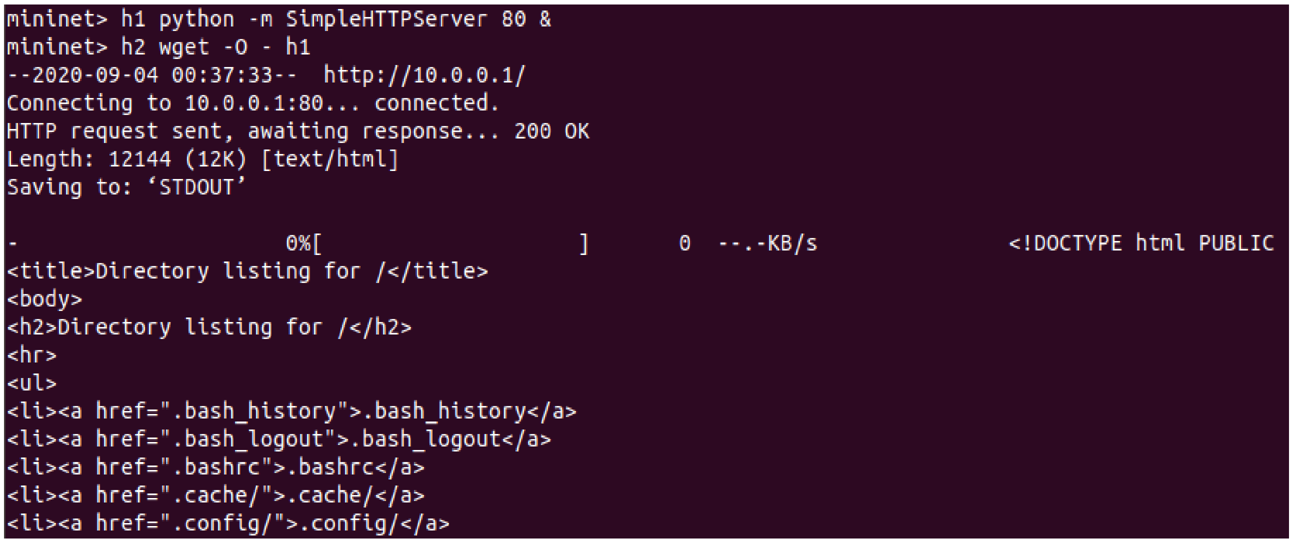
```
mininet> h1 ping -c 1 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=14.4 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 14.446/14.446/14.446/0.000 ms
mininet> 
```

<u>Pingall command:</u> An easier way to run this test is to use the Mininet CLI built-in pingall command, which does an all-pairs ping:



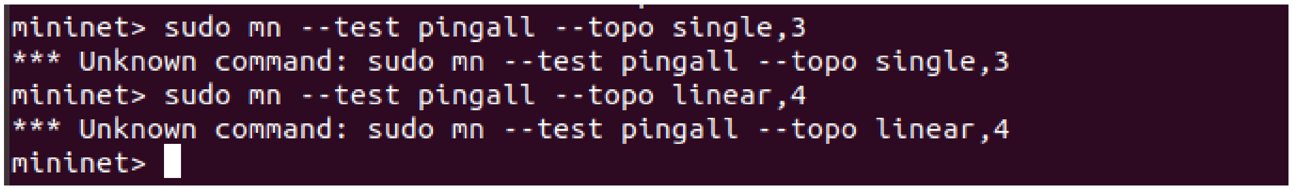
```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet> 
```

Run a simple web server and client :



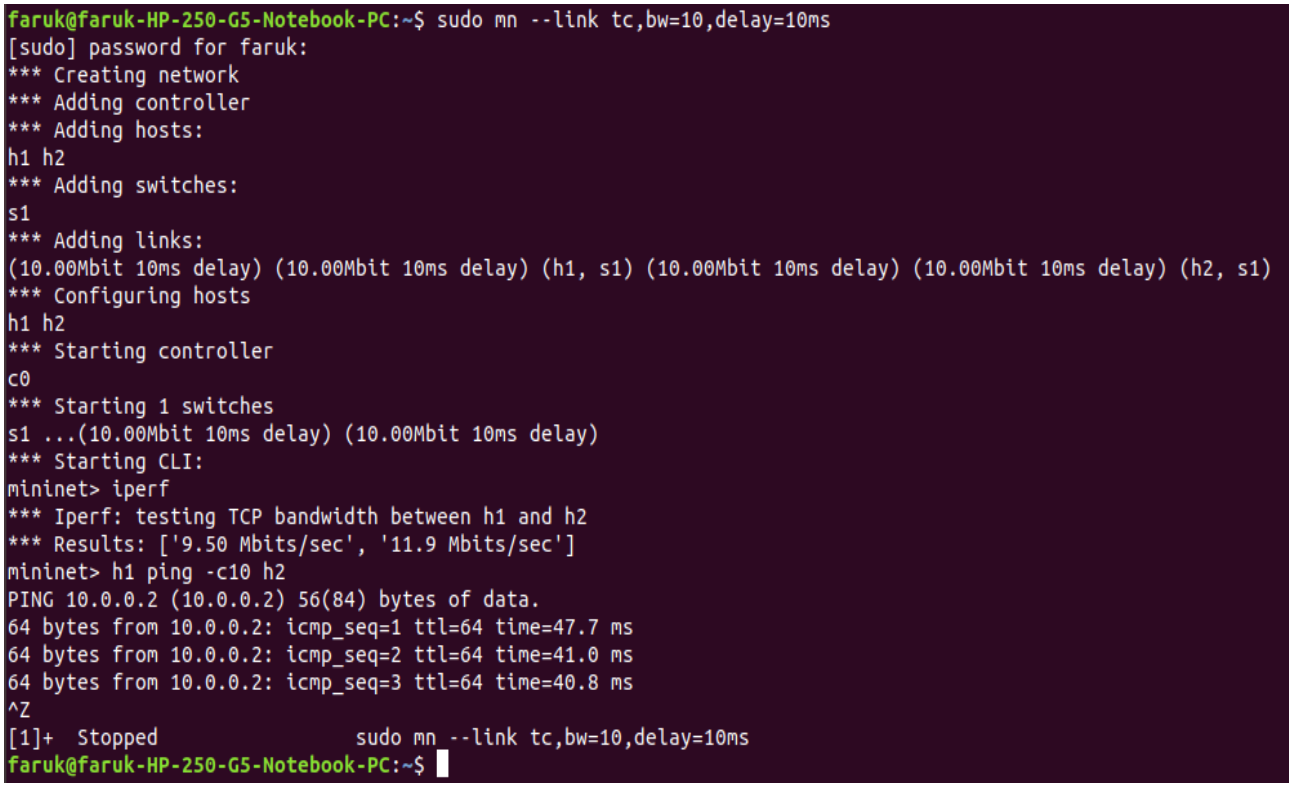
```
mininet> h1 python -m SimpleHTTPServer 80 &
mininet> h2 wget -O - h1
--2020-09-04 00:37:33--  http://10.0.0.1/
Connecting to 10.0.0.1:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 12144 (12K) [text/html]
Saving to: ‘STDOUT’

-                    0%[                   ]       0  --.-KB/s               <!DOCTYPE html PUBLIC
<title>Directory listing for /</title>
<body>
<h2>Directory listing for /</h2>
<hr>
<ul>
<li><a href=".bash_history">.bash_history</a>
<li><a href=".bash_logout">.bash_logout</a>
<li><a href=".bashrc">.bashrc</a>
<li><a href=".cache/">.cache/</a>
<li><a href=".config/">.config/</a>
```

Changing Topology Size and Type: The default topology is a single switch connected to two hosts. You could change this to a different topo with --topo, and pass parameters for that topology’s creation. For example, to verify all-pairs ping connectivity with one switch and three hosts:



```
mininet> sudo mn --test pingall --topo single,3
*** Unknown command: sudo mn --test pingall --topo single,3
mininet> sudo mn --test pingall --topo linear,4
*** Unknown command: sudo mn --test pingall --topo linear,4
mininet> 
```

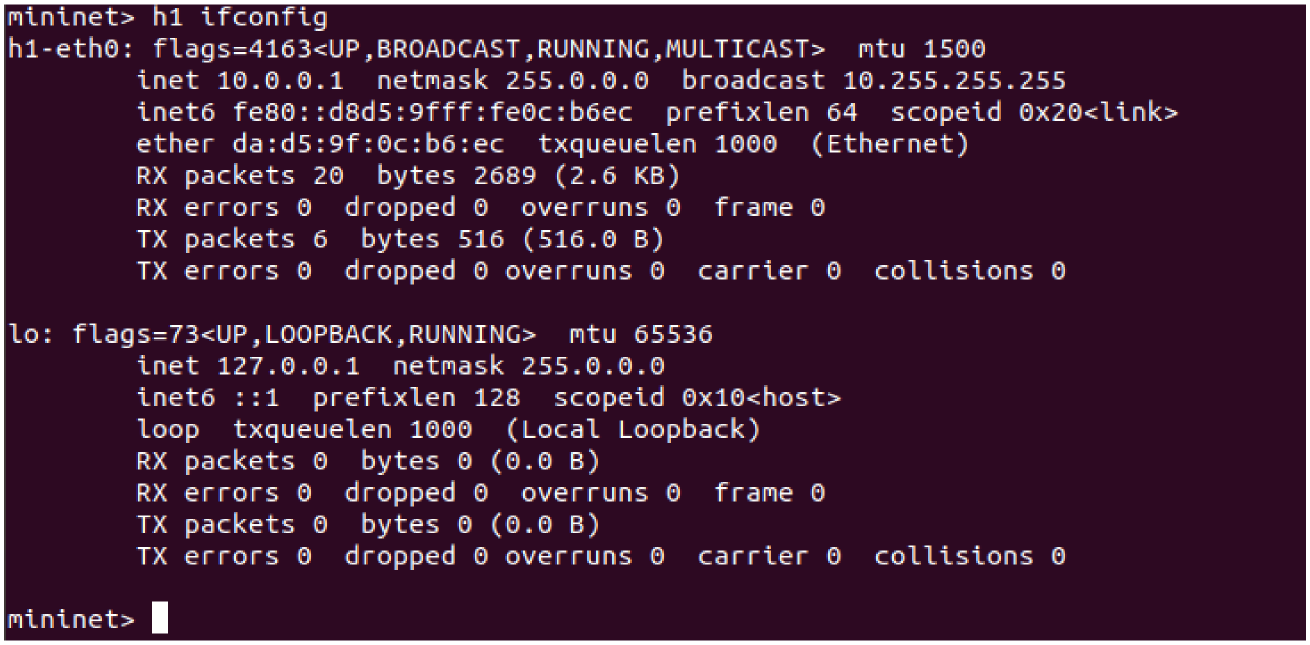
Link variations :



```
faruk@faruk-HP-250-G5-Notebook-PC:~$ sudo mn --link tc,bw=10,delay=10ms
[sudo] password for faruk:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(10.00Mbit 10ms delay) (10.00Mbit 10ms delay) (h1, s1) (10.00Mbit 10ms delay) (10.00Mbit 10ms delay) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...(10.00Mbit 10ms delay) (10.00Mbit 10ms delay)
*** Starting CLI:
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['9.50 Mbits/sec', '11.9 Mbits/sec']
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=47.7 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=41.0 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=40.8 ms
^Z
[1]+  Stopped                 sudo mn --link tc,bw=10,delay=10ms
faruk@faruk-HP-250-G5-Notebook-PC:~$
```

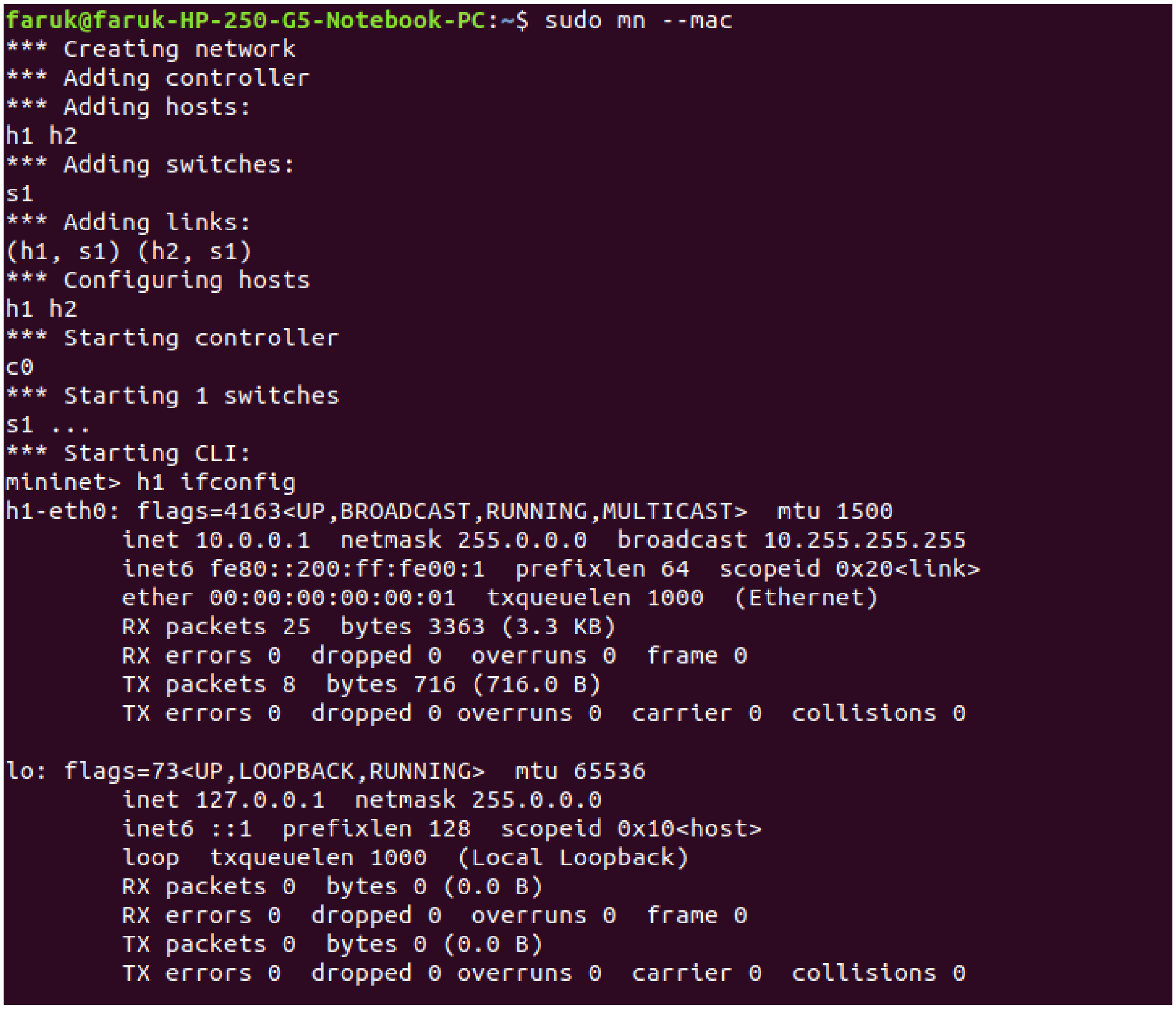
ID = MAC:

Before :



```
mininet> h1 ifconfig
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.1  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::d8d5:9fff:fe0c:b6ec  prefixlen 64  scopeid 0x20<link>
        ether da:d5:9f:0c:b6:ec  txqueuelen 1000  (Ethernet)
        RX packets 20  bytes 2689 (2.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 6  bytes 516 (516.0 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>
```

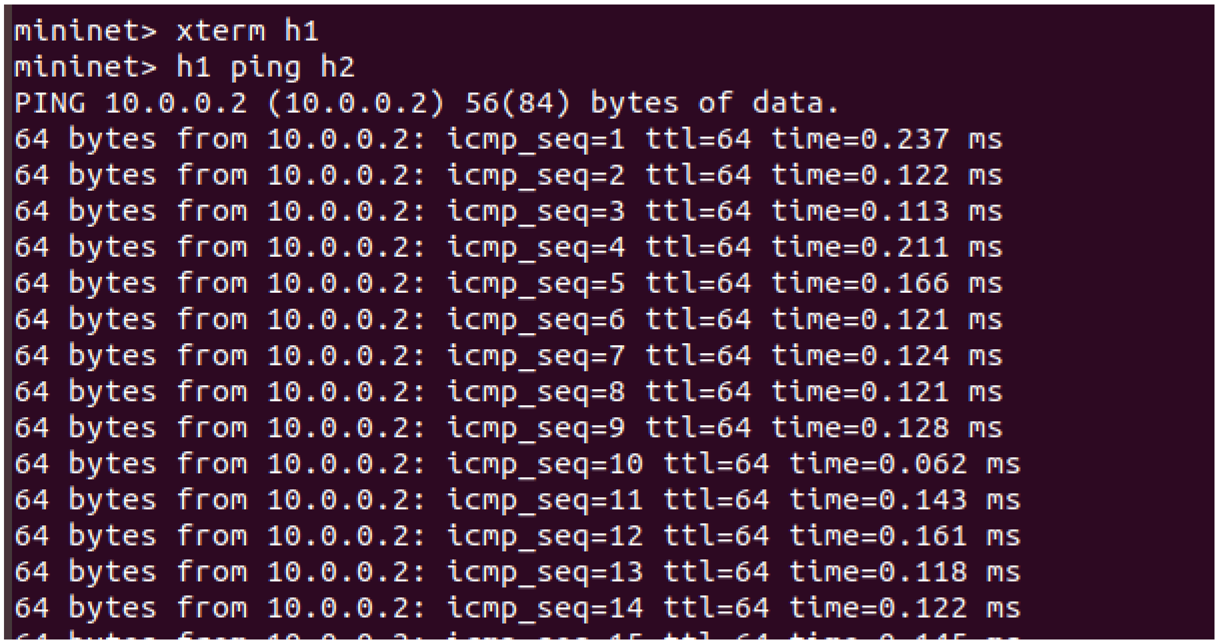
After:



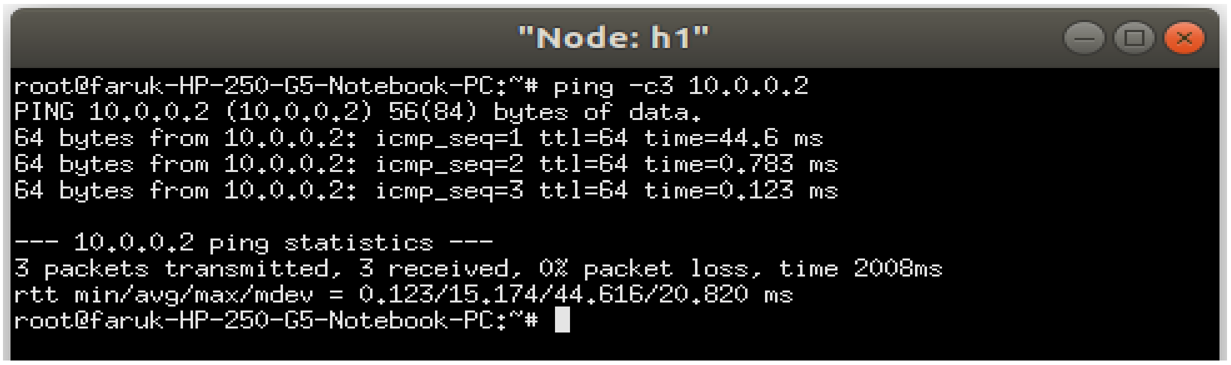
```
faruk@faruk-HP-250-G5-Notebook-PC:~$ sudo mn --mac
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> h1 ifconfig
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.1  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::200:ff:fe00:1  prefixlen 64  scopeid 0x20<link>
        ether 00:00:00:00:00:01  txqueuelen 1000  (Ethernet)
        RX packets 25  bytes 3363 (3.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 716 (716.0 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
```

<u>XTerm Display:</u>

For more complex debugging, you can start Mininet so that it spawns one or more xterms.



```
mininet> xterm h1
mininet> h1 ping 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=0.237 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.122 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.113 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.211 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.166 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.121 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.124 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.121 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.128 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.062 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=0.143 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=0.161 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=0.118 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=0.122 ms
```

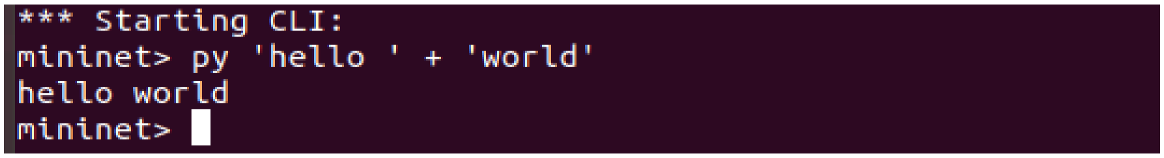


```
root@faruk-HP-250-G5-Notebook-PC:~# ping -c3 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=44.6 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.783 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.123 ms

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2008ms
rtt min/avg/max/mdev = 0.123/15.174/44.616/20.820 ms
root@faruk-HP-250-G5-Notebook-PC:~# 
```

<u>Python Interpreter :</u>

If the first phrase on the Mininiet command line is py, then that command is executed with Python. This might be useful for extending Mininet, as well as probing its inner workings. Each host, switch, and controller has an associated Node object.

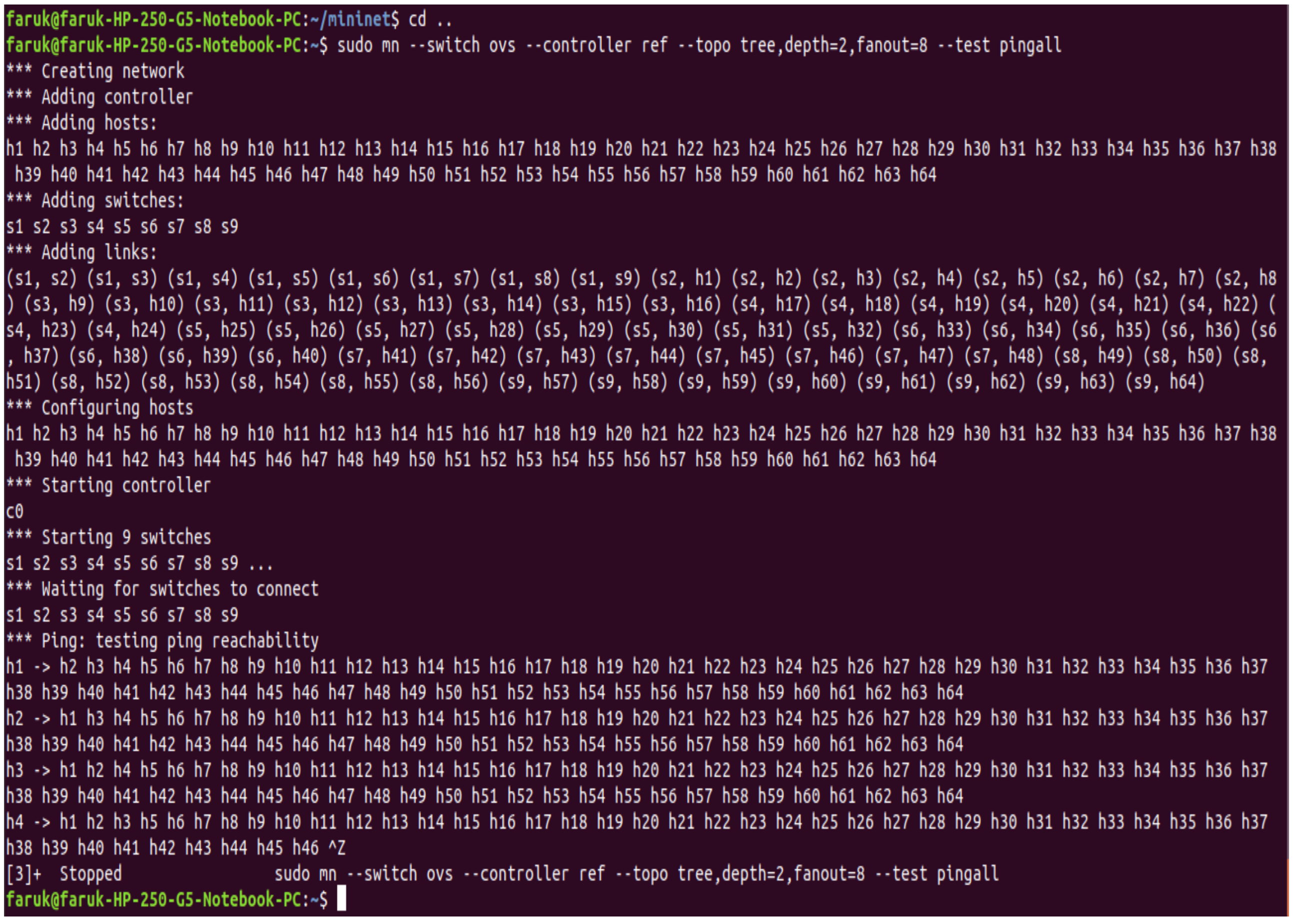
At the Mininet CLI, run:



```
*** Starting CLI:
mininet> py 'hello ' + 'world'
hello world
mininet> 
```

Creating a Network:

You can create a network with a single command. For example :



```
faruk@faruk-HP-250-G5-Notebook-PC:~/mininet$ cd ..
faruk@faruk-HP-250-G5-Notebook-PC:~$ sudo mn --switch ovs --controller ref --topo tree,depth=2,fanout=8 --test pingall
*** 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 h23 h24 h25 h26 h27 h28 h29 h30 h31 h32 h33 h34 h35 h36 h37 h38
 h39 h40 h41 h42 h43 h44 h45 h46 h47 h48 h49 h50 h51 h52 h53 h54 h55 h56 h57 h58 h59 h60 h61 h62 h63 h64
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7 s8 s9
*** Adding links:
(s1, s2) (s1, s3) (s1, s4) (s1, s5) (s1, s6) (s1, s7) (s1, s8) (s1, s9) (s2, h1) (s2, h2) (s2, h3) (s2, h4) (s2, h5) (s2, h6) (s2, h7) (s2, h8
) (s3, h9) (s3, h10) (s3, h11) (s3, h12) (s3, h13) (s3, h14) (s3, h15) (s3, h16) (s4, h17) (s4, h18) (s4, h19) (s4, h20) (s4, h21) (s4, h22) (
s4, h23) (s4, h24) (s5, h25) (s5, h26) (s5, h27) (s5, h28) (s5, h29) (s5, h30) (s5, h31) (s5, h32) (s6, h33) (s6, h34) (s6, h35) (s6, h36) (s6
, h37) (s6, h38) (s6, h39) (s6, h40) (s7, h41) (s7, h42) (s7, h43) (s7, h44) (s7, h45) (s7, h46) (s7, h47) (s7, h48) (s8, h49) (s8, h50) (s8,
h51) (s8, h52) (s8, h53) (s8, h54) (s8, h55) (s8, h56) (s9, h57) (s9, h58) (s9, h59) (s9, h60) (s9, h61) (s9, h62) (s9, h63) (s9, h64)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26 h27 h28 h29 h30 h31 h32 h33 h34 h35 h36 h37 h38
 h39 h40 h41 h42 h43 h44 h45 h46 h47 h48 h49 h50 h51 h52 h53 h54 h55 h56 h57 h58 h59 h60 h61 h62 h63 h64
*** Starting controller
c0
*** Starting 9 switches
s1 s2 s3 s4 s5 s6 s7 s8 s9 ...
*** Waiting for switches to connect
s1 s2 s3 s4 s5 s6 s7 s8 s9
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26 h27 h28 h29 h30 h31 h32 h33 h34 h35 h36 h37
h38 h39 h40 h41 h42 h43 h44 h45 h46 h47 h48 h49 h50 h51 h52 h53 h54 h55 h56 h57 h58 h59 h60 h61 h62 h63 h64
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26 h27 h28 h29 h30 h31 h32 h33 h34 h35 h36 h37
h38 h39 h40 h41 h42 h43 h44 h45 h46 h47 h48 h49 h50 h51 h52 h53 h54 h55 h56 h57 h58 h59 h60 h61 h62 h63 h64
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26 h27 h28 h29 h30 h31 h32 h33 h34 h35 h36 h37
h38 h39 h40 h41 h42 h43 h44 h45 h46 h47 h48 h49 h50 h51 h52 h53 h54 h55 h56 h57 h58 h59 h60 h61 h62 h63 h64
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26 h27 h28 h29 h30 h31 h32 h33 h34 h35 h36 h37
h38 h39 h40 h41 h42 h43 h44 h45 h46 ^Z
[3]+  Stopped                 sudo mn --switch ovs --controller ref --topo tree,depth=2,fanout=8 --test pingall
faruk@faruk-HP-250-G5-Notebook-PC:~$ 
```

Conclusion : Mininet is a network emulation software that allows one to launch a virtual network with switches, Host and SDN controller. Doing this lab, I learn a new technology mininet. One can easily interact with the network using Mininet.