



MAWLANA BHASHANI SCIENCE AND TECHNOLOGY UNIVERSITY

Santosh, Tangail-1902

LAB REPORT

Lab Report No : 04
Lab Report name : SDN controllers and mininet
Course Title : Computer Networks
Course Code : ICT- 3207
Date of Performance : 05/02/2021
Date of Submission :

Submitted by,

Name: Tanvir Ahmed and Farhana
Afrin Shikha

ID: IT-18043 and IT-18038

Session: 2017-18

3rd year 2nd semester

Dept. of ICT

Submitted to,

Nazrul Islam
Assistant Professor
Dept. of ICT
MBSTU.

Theory:

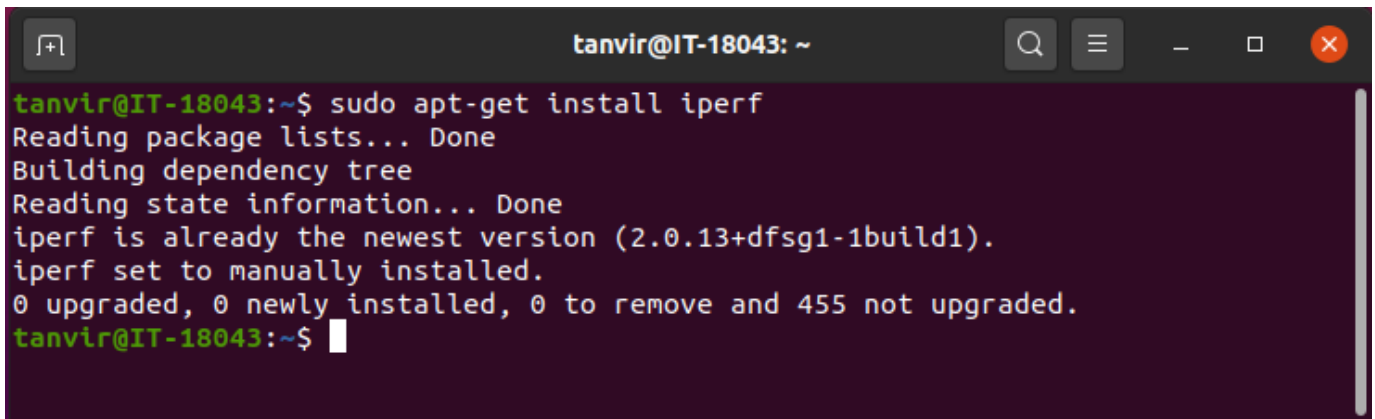
Traffic Generator:

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.

Mininet:

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 interact with your network using the Mininet CLI (and API), customize it, share it with others, or deploy it on real hardware, Mininet is useful for development, teaching, and research. Mininet is also a great way to develop, share, and experiment with OpenFlow and Software-Defined Networking systems.

Install iperf:

A terminal window titled 'tanvir@IT-18043: ~' with standard window controls. The terminal output shows the command 'sudo apt-get install iperf' being executed. The output indicates that iperf is already installed at the latest version (2.0.13+dfsg1-1build1) and is set to manually installed. It also shows that 0 packages were upgraded, 0 newly installed, 0 to be removed, and 455 not upgraded.

```
tanvir@IT-18043:~$ sudo apt-get install iperf
Reading package lists... Done
Building dependency tree
Reading state information... Done
iperf is already the newest version (2.0.13+dfsg1-1build1).
iperf set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 455 not upgraded.
tanvir@IT-18043:~$
```

Install Mininet:

```
tanvir@IT-18043: ~  
tanvir@IT-18043:~$ sudo apt-get install mininet  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
mininet is already the newest version (2.2.2-5ubuntu1).  
0 upgraded, 0 newly installed, 0 to remove and 455 not upgraded.  
tanvir@IT-18043:~$
```

4. Exercises Exercise

4.1.1: Open a Linux terminal, and execute the command line `iperf --help`. Provide four configuration options of `iperf`.

```
tanvir@IT-18043: ~  
tanvir@IT-18043:~$ iperf --help  
Usage: iperf [-s|-c host] [options]  
iperf [-h|--help] [-v|--version]  
  
Client/Server:  
-b, --bandwidth #[kmgKMG | pps] bandwidth to send at in bits/sec or packets per second  
-e, --enhancedreports use enhanced reporting giving more tcp/udp and traffic information  
-f, --format [kmgKMG] format to report: Kbits, Mbits, KBytes, MBytes  
-i, --interval # seconds between periodic bandwidth reports  
-l, --len #[kmmKM] length of buffer in bytes to read or write (Defaults: TCP=128K, v4 UDP=1470, v6 UDP=1450)  
-m, --print_mss print TCP maximum segment size (MTU - TCP/IP header)  
-o, --output <filename> output the report or error message to this specified file  
-p, --port # server port to listen on/connect to  
-u, --udp use UDP rather than TCP  
--udp-counters-64bit use 64 bit sequence numbers with UDP  
-w, --window #[KM] TCP window size (socket buffer size)  
-Z, --realtime request realtime scheduler  
-B, --bind <host>[:<port>][%<dev>] bind to <host>, ip addr (including multicast address) and optional port and device  
for use with older versions does not sent extra msgs  
-C, --compatibility # set TCP maximum segment size (MTU - 40 bytes)  
-M, --mss # set TCP no delay, disabling Nagle's Algorithm  
-N, --nodelay # set the socket's IP_TOS (byte) field  
-S, --tos #  
  
Server specific:  
-s, --server run in server mode  
-t, --time # time in seconds to listen for new connections as well as to receive traffic (default not set)  
--udp-histogram #, # enable UDP latency histogram(s) with bin width and count, e.g. 1,1000=1(ms),1000(bins)  
-B, --bind <ip>[%<dev>] bind to multicast address and optional device  
-H, --ssm-host <ip> set the SSM source, use with -B for (S,G)  
-U, --single_udp run in single threaded UDP mode  
-D, --daemon run the server as a daemon  
-V, --ipv6_domain Enable IPv6 reception by setting the domain and socket to AF_INET6 (Can receive on both IPv4 and IPv6)  
  
Client specific:  
-c, --client <host> run in client mode, connecting to <host>  
-d, --dualtest Do a bidirectional test simultaneously  
--ipg set the the interpacket gap (milliseconds) for packets within an isochronous frame  
--isochronous <frames-per-second>:<mean>,<stddev> send traffic in bursts (frames - emulate video traffic)  
-n, --num #[kmgKMG] number of bytes to transmit (instead of -t)  
-r, --tradeoff Do a bidirectional test individually  
-t, --time # time in seconds to transmit for (default 10 secs)  
-B, --bind [<ip> | <ip:port>] bind ip (and optional port) from which to source traffic  
-F, --fileinput <name> input the data to be transmitted from a file  
-I, --stdin input the data to be transmitted from stdin  
-L, --listenport # port to receive bidirectional tests back on  
-P, --parallel # number of parallel client threads to run  
-R, --reverse reverse the test (client receives, server sends)  
-T, --ttl # time-to-live, for multicast (default 1)
```

Exercise 4.1.2: Open two Linux terminals, and configure terminal-1 as client (`iperf -c IPv4_server_address`) and terminal-2 as server (`iperf -s`).

For terminal -1:

```
tanvir@IT-18043: ~  
tanvir@IT-18043:~$ iperf -s  
-----  
Server listening on TCP port 5001  
TCP window size: 85.3 KByte (default)  
-----  
█
```

For terminal -2:

```
tanvir@IT-18043: ~  
tanvir@IT-18043:~$ iperf -c 127.0.0.1 -u  
-----  
Client connecting to 127.0.0.1, UDP port 5001  
Sending 1470 byte datagrams, IPG target: 11215.21 us (kalman adjust)  
UDP buffer size: 208 KByte (default)  
-----  
[ 3] local 127.0.0.1 port 53547 connected with 127.0.0.1 port 5001  
read failed: Connection refused  
[ 3] WARNING: did not receive ack of last datagram after 1 tries.  
[ ID] Interval      Transfer      Bandwidth  
[ 3] 0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec  
[ 3] Sent 891 datagrams  
tanvir@IT-18043:~$ █
```

Exercise 4.1.3: Open two Linux terminals, and configure terminal-1 as client and terminal-2 as server for exchanging UDP traffic, which are the command lines? Which are the statistics are provided at the end of transmission?

```
tanvir@IT-18043: ~  
-----  
^Ctanvir@IT-18043:~$ iperf -s -u  
-----  
Server listening on UDP port 5001  
Receiving 1470 byte datagrams  
UDP buffer size: 208 KByte (default)  
-----  
[ 3] local 127.0.0.1 port 5001 connected with 127.0.0.1 port 47061  
[ ID] Interval      Transfer    Bandwidth    Jitter    Lost/Total Datagrams  
[ 3] 0.0-10.0 sec  1.25 MBytes 1.05 Mbits/sec 0.002 ms   0/ 892 (0%)
```

```
tanvir@IT-18043: ~  
-----  
tanvir@IT-18043:~$ iperf -c 127.0.0.1 -u  
-----  
Client connecting to 127.0.0.1, UDP port 5001  
Sending 1470 byte datagrams, IPG target: 11215.21 us (kalman adjust)  
UDP buffer size: 208 KByte (default)  
-----  
[ 3] local 127.0.0.1 port 47061 connected with 127.0.0.1 port 5001  
[ ID] Interval      Transfer    Bandwidth  
[ 3] 0.0-10.0 sec  1.25 MBytes 1.05 Mbits/sec  
[ 3] Sent 892 datagrams  
[ 3] Server Report:  
[ 3] 0.0-10.0 sec  1.25 MBytes 1.05 Mbits/sec 0.002 ms   0/ 892 (0%)  
tanvir@IT-18043:~$
```

Exercise 4.1.4: Open two Linux terminals, and configure terminal-1 as client and terminal-2 as server for exchanging UDP traffic, with:

- o Packet length = 1000bytes

- o Time = 20 seconds

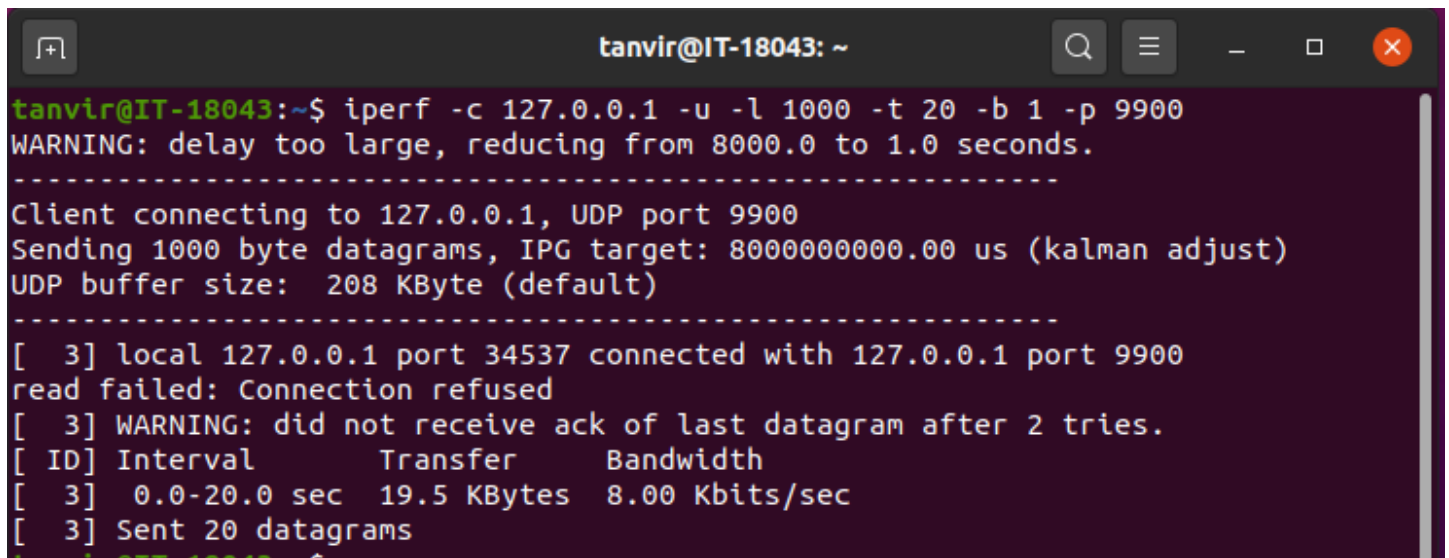
- o Bandwidth = 1Mbps

- o Port = 9900

Which are the command lines?

The command lines are:

For terminal 1:



```
tanvir@IT-18043: ~  
tanvir@IT-18043:~$ iperf -c 127.0.0.1 -u -l 1000 -t 20 -b 1 -p 9900  
WARNING: delay too large, reducing from 8000.0 to 1.0 seconds.  
-----  
Client connecting to 127.0.0.1, UDP port 9900  
Sending 1000 byte datagrams, IPG target: 8000000000.00 us (kalman adjust)  
UDP buffer size: 208 KByte (default)  
-----  
[ 3] local 127.0.0.1 port 34537 connected with 127.0.0.1 port 9900  
read failed: Connection refused  
[ 3] WARNING: did not receive ack of last datagram after 2 tries.  
[ ID] Interval      Transfer    Bandwidth  
[ 3]  0.0-20.0 sec  19.5 KBytes  8.00 Kbits/sec  
[ 3] Sent 20 datagrams
```

For terminal 2:

```
tanvir@IT-18043: ~  
tanvir@IT-18043:~$ iperf -s -u 9900  
iperf: ignoring extra argument -- 9900  
-----  
Server listening on UDP port 5001  
Receiving 1470 byte datagrams  
UDP buffer size: 208 KByte (default)  
-----  
□
```

Using Mininet

Exercise 4.2.1: Open two Linux terminals, and execute the command line `ifconfig` in terminal1. How many interfaces are present?

In terminal-2, execute the command line `sudo mn`, which is the output?

In terminal-1 execute the command line `ifconfig`. How many real and virtual interfaces are present now?

```
tanvir@IT-18043: ~  
tanvir@IT-18043:~$ ifconfig  
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255  
    inet6 fe80::5526:8c52:746f:a669 prefixlen 64 scopeid 0x20<link>  
    ether 08:00:27:54:1d:b2 txqueuelen 1000 (Ethernet)  
    RX packets 291406 bytes 298856425 (298.8 MB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 85193 bytes 5211254 (5.2 MB)  
    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 3027 bytes 3250743 (3.2 MB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 3027 bytes 3250743 (3.2 MB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
tanvir@IT-18043:~$
```

```
tanvir@IT-18043: ~  
tanvir@IT-18043:~$ sudo mn  
[sudo] password for tanvir:  
*** No default OpenFlow controller found for default switch!  
*** Falling back to OVS Bridge  
*** Creating network  
*** Adding controller  
*** Adding hosts:  
h1 h2  
*** Adding switches:  
s1  
*** Adding links:  
(h1, s1) (h2, s1)  
*** Configuring hosts  
h1 h2  
*** Starting controller  
  
*** Starting 1 switches  
s1 ...  
*** Starting CLI:  
mininet>
```

Exercise 4.2.2: Interacting with mininet; in terminal-2, display the following command lines and explain what it does:

mininet> help

```
tanvir@IT-18043: ~  
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  
  
mininet> █
```


mininet> nodes

```
tanvir@IT-18043: ~  
mininet> nodes  
available nodes are:  
h1 h2 s1  
mininet>
```

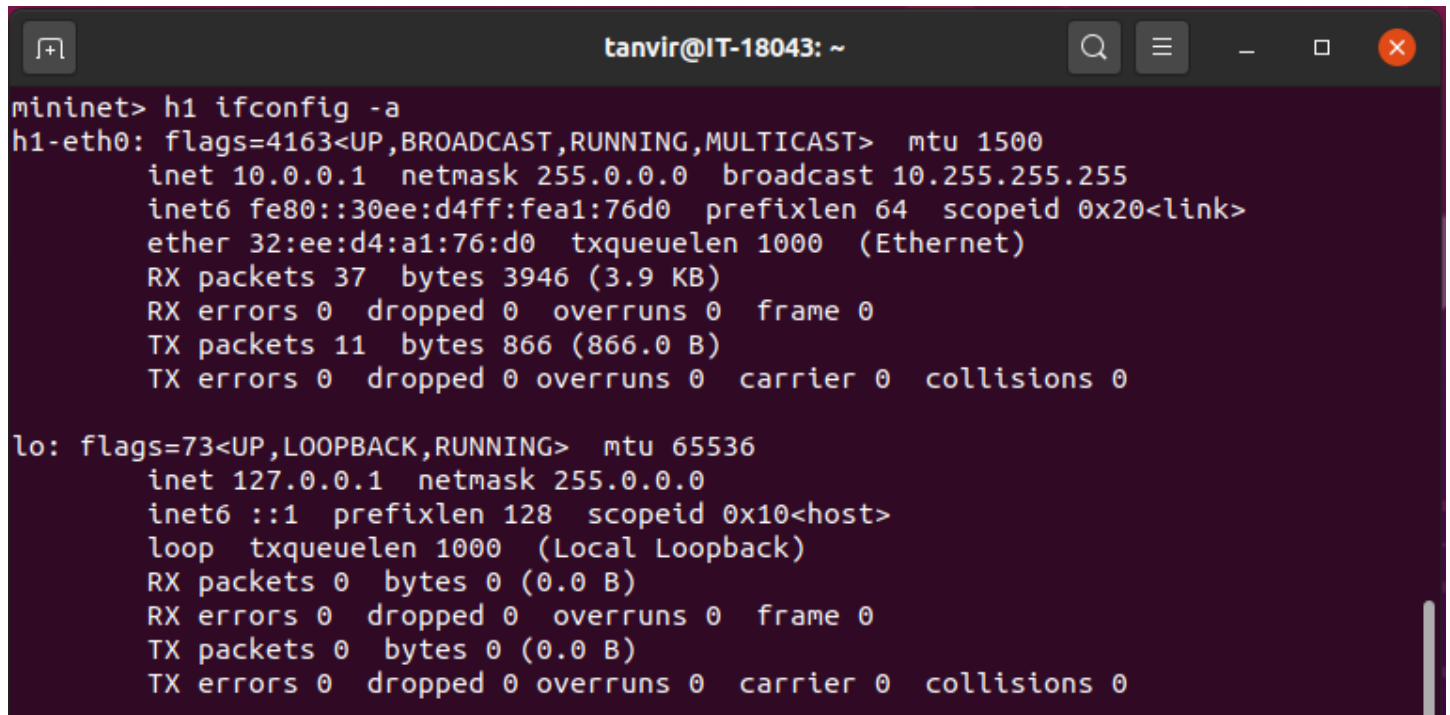
mininet> net

```
tanvir@IT-18043: ~  
mininet> net  
h1 h1-eth0:s1-eth1  
h2 h2-eth0:s1-eth2  
s1 lo: s1-eth1:h1-eth0 s1-eth2:h2-eth0  
mininet>
```

mininet> dump

```
tanvir@IT-18043: ~  
mininet> dump  
<Host h1: h1-eth0:10.0.0.1 pid=6629>  
<Host h2: h2-eth0:10.0.0.2 pid=6631>  
<OVSBridge s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=6636>  
mininet>
```

mininet> h1 ifconfig -a



A terminal window titled 'tanvir@IT-18043: ~' with standard window controls. The terminal displays the output of the command 'h1 ifconfig -a'. It shows details for the 'h1-eth0' interface, including its flags (UP, BROADCAST, RUNNING, MULTICAST), MTU (1500), IP address (10.0.0.1), netmask (255.0.0.0), broadcast address (10.255.255.255), IPv6 address (fe80::30ee:d4ff:fea1:76d0), prefix length (64), scope ID (0x20), and MAC address (32:ee:d4:a1:76:d0). It also shows statistics for RX and TX packets, bytes, errors, dropped, overruns, carrier, and collisions. Below this, it shows details for the 'lo' (loopback) interface, including its flags (UP, LOOPBACK, RUNNING), MTU (65536), IP address (127.0.0.1), netmask (255.0.0.0), IPv6 address (::1), prefix length (128), scope ID (0x10), and statistics.

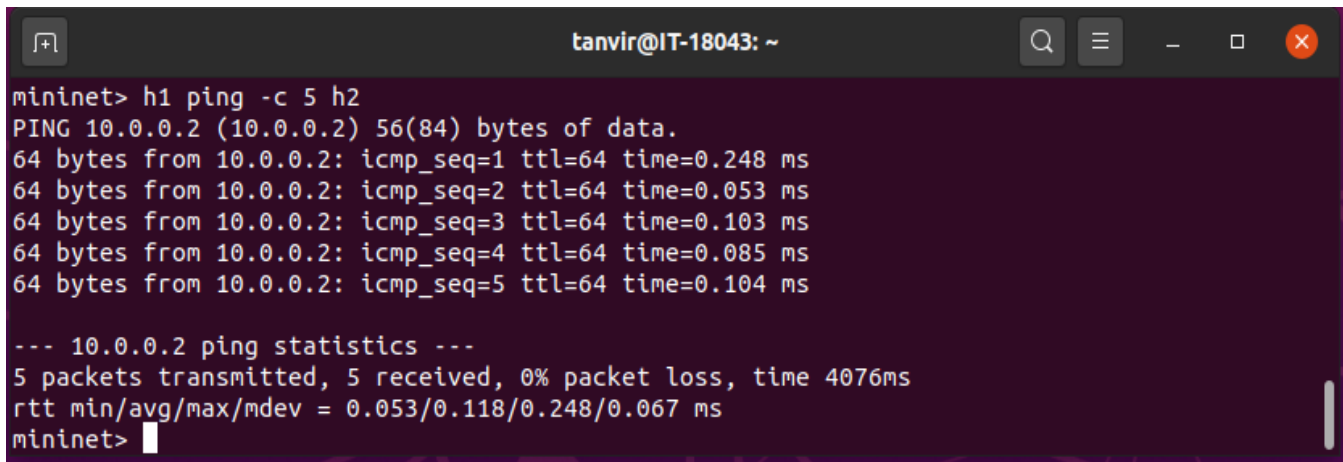
```
mininet> h1 ifconfig -a
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.0.1 netmask 255.0.0.0 broadcast 10.255.255.255
    inet6 fe80::30ee:d4ff:fea1:76d0 prefixlen 64 scopeid 0x20<link>
    ether 32:ee:d4:a1:76:d0 txqueuelen 1000 (Ethernet)
    RX packets 37 bytes 3946 (3.9 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 11 bytes 866 (866.0 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
```

s1 ifconfig -a

```
tanvir@IT-18043: ~  
mininet> s1 ifconfig -a  
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255  
    inet6 fe80::5526:8c52:746f:a669 prefixlen 64 scopeid 0x20<link>  
    ether 08:00:27:54:1d:b2 txqueuelen 1000 (Ethernet)  
    RX packets 393529 bytes 399893184 (399.8 MB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 118008 bytes 7190304 (7.1 MB)  
    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 3069 bytes 3254155 (3.2 MB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 3069 bytes 3254155 (3.2 MB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
ovs-system: flags=4098<BROADCAST,MULTICAST> mtu 1500  
    ether 9a:9d:56:d6:e5:90 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 d2:93:ea:74:a5:44 txqueuelen 1000 (Ethernet)  
    RX packets 0 bytes 0 (0.0 B)  
    RX errors 0 dropped 20 overruns 0 frame 0  
    TX packets 0 bytes 0 (0.0 B)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
s1-eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet6 fe80::8808:2eff:fe5f:86f0 prefixlen 64 scopeid 0x20<link>  
    ether 8a:08:2e:5f:86:f0 txqueuelen 1000 (Ethernet)  
    RX packets 11 bytes 866 (866.0 B)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 37 bytes 3946 (3.9 KB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
s1-eth2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet6 fe80::a44b:baff:fedf:b77 prefixlen 64 scopeid 0x20<link>  
    ether a6:4b:ba:df:0b:77 txqueuelen 1000 (Ethernet)  
    RX packets 11 bytes 866 (866.0 B)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 37 bytes 3946 (3.9 KB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

mininet> h1 ping -c 5 h2



A terminal window titled 'tanvir@IT-18043: ~' with standard window controls. The terminal output shows a successful ping from h1 to h2. The command 'mininet> h1 ping -c 5 h2' is entered. The output shows 'PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.' followed by five lines of ping results, each showing '64 bytes from 10.0.0.2: icmp_seq=X ttl=64 time=Y ms' for X from 1 to 5. The times are 0.248, 0.053, 0.103, 0.085, and 0.104 ms respectively. Below this, it shows '--- 10.0.0.2 ping statistics ---', '5 packets transmitted, 5 received, 0% packet loss, time 4076ms', and 'rtt min/avg/max/mdev = 0.053/0.118/0.248/0.067 ms'. The prompt 'mininet>' is followed by a cursor.

```
mininet> h1 ping -c 5 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=0.248 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.053 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.103 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.085 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.104 ms

--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4076ms
rtt min/avg/max/mdev = 0.053/0.118/0.248/0.067 ms
mininet> 
```

Exercise 4.2.3: In terminal-2, display the following command line: `sudo mn --link tc,bw=10,delay=500ms`

o mininet> h1 ping -c 5 h2, What happen with the link?

o mininet> h1 iperf -s -u &

o mininet> h2 iperf -c IPv4_h1 -u, Is there any packet loss?

```
tanvir@IT-18043: ~  
tanvir@IT-18043:~$ sudo mn --link tc,bw=10,delay=500ms  
*** No default OpenFlow controller found for default switch!  
*** Falling back to OVS Bridge  
*** Creating network  
*** Adding controller  
*** Adding hosts:  
h1 h2  
*** Adding switches:  
s1  
*** Adding links:  
(10.00Mbit 500ms delay) (10.00Mbit 500ms delay) (h1, s1) (10.00Mbit 500ms delay) (10.00Mbit 500ms delay) (h2, s1)  
*** Configuring hosts  
h1 h2  
*** Starting controller  
  
*** Starting 1 switches  
s1 ... (10.00Mbit 500ms delay) (10.00Mbit 500ms delay)  
*** Starting CLI:  
mininet> 
```

```
mininet> h1 ping -c 5 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=4002 ms  
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=2980 ms  
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=2001 ms  
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=2002 ms  
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=2002 ms  
  
--- 10.0.0.2 ping statistics ---  
5 packets transmitted, 5 received, 0% packet loss, time 4070ms  
rtt min/avg/max/mdev = 2001.407/2597.549/4002.440/798.132 ms, pipe 4  
mininet> 
```

Discussion :

Mininet is a network emulator which creates a network of virtual hosts, switches, controllers, and links. Mininet hosts run standard Linux network software, and its switches support OpenFlow for highly flexible custom routing and Software-Defined Networking.

Mininet supports research, development, learning, prototyping, testing, debugging, and any other tasks that could benefit from having a complete experimental network on a laptop or other PC.

Mininet:

- Provides a simple and inexpensive network testbed for developing OpenFlow applications
- Enables multiple concurrent developers to work independently on the same topology

- Supports system-level regression tests, which are repeatable and easily packaged
- Enables complex topology testing, without the need to wire up a physical network
- Includes a CLI that is topology-aware and OpenFlow-aware, for debugging or running network-wide tests
- Supports arbitrary custom topologies, and includes a basic set of parametrized topologies
- is usable out of the box without programming, but
- also Provides a straightforward and extensible Python API for network creation and experimentation

Mininet provides an easy way to get correct system *behavior* (and, to the extent supported by your hardware, performance) and to experiment with topologies.