Mawlana Bhashani Science and Technology University

Lab-Report

Report No: 04

Course code: ICT-3207

Course title: Computer Networking Lab

Date of Performance:

Date of Submission:

Submitted by

Name: Ashikur Rahman Miran & Rafiul Hasan

ID:IT-18014 & IT-18016

3rd year 2nd semester

Session: 2017-2018

Dept. of ICT

Submitted To

Nazrul Islam

Assistant Professor

Dept. of ICT

MBSTU.

Lab Report No.: 04

Lab Report Name: SDN controller and Mininet

Theory:

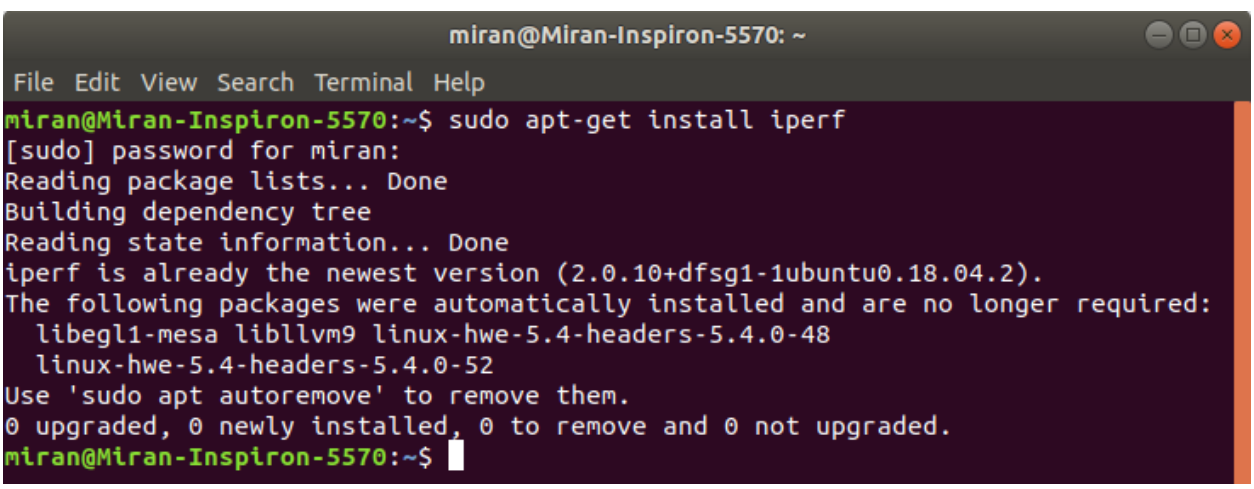
Traffic Generator:

What is 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:



Install Mininet:

```
miran@Miran-Inspiron-5570: ~
File  Edit  View  Search  Terminal  Help
miran@Miran-Inspiron-5570:~$ sudo apt-get install mininet
Reading package lists... Done
Building dependency tree
Reading state information... Done
mininet is already the newest version (2.2.2-2ubuntu1).
The following packages were automatically installed and are no longer required:
  libegl1-mesa libllvm9 linux-hwe-5.4-headers-5.4.0-48
  linux-hwe-5.4-headers-5.4.0-52
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
miran@Miran-Inspiron-5570:~$
```

4. Exercises Exercise

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

```
miran@Miran-Inspiron-5570: ~
File  Edit  View  Search  Terminal  Help
miran@Miran-Inspiron-5570:~$ 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 p
er second
  -e, --enhancedreports    use enhanced reporting giving more tcp/udp and traffi
c information
  -f, --format    [kmgKMG]   format to report: Kbits, Mbits, KBytes, MBytes
  -i, --interval  #        seconds between periodic bandwidth reports
  -l, --len       #[kmKM]    length of buffer in bytes to read or write (Default
s: 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 specifie
d 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>   bind to <host>, an interface or multicast address
  -C, --compatibility      for use with older versions does not sent extra msgs
  -M, --mss       #        set TCP maximum segment size (MTU - 40 bytes)
  -N, --nodelay            set TCP no delay, disabling Nagle's Algorithm
  -S, --tos       #        set the socket's IP_TOS (byte) field

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)
  -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 socke
t to AF_INET6 (Can receive on both IPv4 and IPv6)
```

```
miran@Miran-Inspiron-5570: ~
File  Edit  View  Search  Terminal  Help
Client specific:
  -c, --client    <host>   run in client mode, connecting to <host>
  -d, --dualtest           Do a bidirectional test simultaneously
  -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 src addr(s) from which to originate 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)
  -V, --ipv6_domain        Set the domain to IPv6 (send packets over IPv6)
  -X, --peer-detect        perform server version detection and version exchange
  -Z, --linux-congestion <algo>  set TCP congestion control algorithm (Linux onl
y)

Miscellaneous:
  -x, --reportexclude [CDMSV]   exclude C(connection) D(data) M(multicast) S(set
tings) V(server) reports
  -y, --reportstyle C      report as a Comma-Separated Values
  -h, --help               print this message and quit
  -v, --version            print version information and quit

[kmgKMG] Indicates options that support a k,m,g,K,M or G suffix
Lowercase format characters are 10^3 based and uppercase are 2^n based
(e.g. 1k = 1000, 1K = 1024, 1m = 1,000,000 and 1M = 1,048,576)

The TCP window size option can be set by the environment variable
TCP_WINDOW_SIZE. Most other options can be set by an environment variable
IPERF_<long option name>, such as IPERF_BANDWIDTH.

Source at <http://sourceforge.net/projects/iperf2/>
Report bugs to <iperf-users@lists.sourceforge.net>
miran@Miran-Inspiron-5570:~$
miran@Miran-Inspiron-5570:~$
```

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:

```
miran@Miran-Inspiron-5570: ~
File  Edit  View  Search  Terminal  Help
miran@Miran-Inspiron-5570:~$ iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size:  128 KByte (default)
------------------------------------------------------------
```

For terminal -2:

```
miran@Miran-Inspiron-5570: ~
File  Edit  View  Search  Terminal  Help
miran@Miran-Inspiron-5570:~$ 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 56744 connected with 127.0.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  1.44 KBytes  1.18 Kbits/sec
[  3] Sent 1 datagrams
read failed: Connection refused
[  3] WARNING: did not receive ack of last datagram after 2 tries.
miran@Miran-Inspiron-5570:~$
```

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?

```
miran@Miran-Inspiron-5570: ~
File  Edit  View  Search  Terminal  Help
miran@Miran-Inspiron-5570:~$ 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 56744 connected with 127.0.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  1.44 KBytes  1.18 Kbits/sec
[  3] Sent 1 datagrams
read failed: Connection refused
[  3] WARNING: did not receive ack of last datagram after 2 tries.
miran@Miran-Inspiron-5570:~$
```

```
miran@Miran-Inspiron-5570: ~
File  Edit  View  Search  Terminal  Help
miran@Miran-Inspiron-5570:~$ iperf -s -u
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
```

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:

Iperf –c 127.0.0.1 –u –l 1000 –t 20 –b 1 –p 9900

```
miran@Miran-Inspiron-5570: ~
File  Edit  View  Search  Terminal  Help
miran@Miran-Inspiron-5570:~$ 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 57483 connected with 127.0.0.1 port 9900
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-20.0 sec  1000 Bytes   400 bits/sec
[  3] Sent 1 datagrams
read failed: Connection refused
[  3] WARNING: did not receive ack of last datagram after 2 tries.
miran@Miran-Inspiron-5570:~$
```

For terminal 2:

Iperf –s –u –p 9900

```
miran@Miran-Inspiron-5570: ~
File  Edit  View  Search  Terminal  Help
miran@Miran-Inspiron-5570:~$ iperf -s -u -p 9900
------------------------------------------------------------
Server listening on UDP port 9900
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?

```
miran@Miran-Inspiron-5570: ~
File  Edit  View  Search  Terminal  Help
miran@Miran-Inspiron-5570:~$ ifconfig
enp1s0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 8c:ec:4b:07:61:ce  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 8711  bytes 2280813 (2.2 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8711  bytes 2280813 (2.2 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.104  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::6539:3c4c:a6a9:312b  prefixlen 64  scopeid 0x20<link>
        ether b0:52:16:12:4a:d1  txqueuelen 1000  (Ethernet)
        RX packets 15683  bytes 19176796 (19.1 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 10028  bytes 1349034 (1.3 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

miran@Miran-Inspiron-5570:~$
```

```
miran@Miran-Inspiron-5570: ~
File  Edit  View  Search  Terminal  Help
miran@Miran-Inspiron-5570:~$ sudo mn
[sudo] password for miran:
*** 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>
```

```
miran@Miran-Inspiron-5570: ~
File  Edit  View  Search  Terminal  Help
miran@Miran-Inspiron-5570:~$ ifconfig
enp1s0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 8c:ec:4b:07:61:ce  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 9166  bytes 2314473 (2.3 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 9166  bytes 2314473 (2.3 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

s1-eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::c08c:ebff:fed5:d122  prefixlen 64  scopeid 0x20<link>
        ether c2:8c:eb:d5:d1:22  txqueuelen 1000  (Ethernet)
        RX packets 10  bytes 796 (796.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 32  bytes 4261 (4.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

s1-eth2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::8c1d:29ff:fefa:9007  prefixlen 64  scopeid 0x20<link>
        ether 8e:1d:29:fa:90:07  txqueuelen 1000  (Ethernet)
        RX packets 10  bytes 796 (796.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 32  bytes 4261 (4.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.104  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::6539:3c4c:a6a9:312b  prefixlen 64  scopeid 0x20<link>
        ether b0:52:16:12:4a:d1  txqueuelen 1000  (Ethernet)
        RX packets 16020  bytes 19316874 (19.3 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 10428  bytes 1414835 (1.4 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

miran@Miran-Inspiron-5570:~$
```

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

mininet> help

```
miran@Miran-Inspiron-5570: ~
File  Edit  View  Search  Terminal  Help
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

```
miran@Miran-Inspiron-5570: ~
File  Edit  View  Search  Terminal  Help
mininet> nodes
available nodes are:
c0 h1 h2 s1
mininet>
```

mininet> net

```
miran@Miran-Inspiron-5570: ~
File  Edit  View  Search  Terminal  Help
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0
c0
mininet>
```

mininet> dump

```
miran@Miran-Inspiron-5570: ~
File  Edit  View  Search  Terminal  Help
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=5740>
<Host h2: h2-eth0:10.0.0.2 pid=5742>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=5747>
<Controller c0: 127.0.0.1:6653 pid=5733>
mininet>
```

mininet> h1 ifconfig –a

```
miran@Miran-Inspiron-5570: ~
File  Edit  View  Search  Terminal  Help
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::105a:7eff:fea0:caa2  prefixlen 64  scopeid 0x20<link>
        ether 12:5a:7e:a0:ca:a2  txqueuelen 1000  (Ethernet)
        RX packets 41  bytes 5290 (5.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 13  bytes 1006 (1.0 KB)
        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>
```

mininet> s1 ifconfig –a

miran@Miran-Inspiron-5570: ~

File Edit View Search Terminal Help

```
mininet> s1 ifconfig -a
enp1s0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 8c:ec:4b:07:61:ce  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 9495  bytes 2334398 (2.3 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 9495  bytes 2334398 (2.3 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ovs-system: flags=4098<BROADCAST,MULTICAST>  mtu 1500
        ether 6e:b0:28:20:88:43  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 82:00:e8:ef:ed:4c  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 30  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

miran@Miran-Inspiron-5570: ~

File Edit View Search Terminal Help

```
s1-eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::c08c:ebff:fed5:d122  prefixlen 64  scopeid 0x20<link>
        ether c2:8c:eb:d5:d1:22  txqueuelen 1000  (Ethernet)
        RX packets 13  bytes 1006 (1.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 41  bytes 5290 (5.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

s1-eth2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::8c1d:29ff:fefa:9007  prefixlen 64  scopeid 0x20<link>
        ether 8e:1d:29:fa:90:07  txqueuelen 1000  (Ethernet)
        RX packets 13  bytes 1006 (1.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 41  bytes 5290 (5.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.104  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::6539:3c4c:a6a9:312b  prefixlen 64  scopeid 0x20<link>
        ether b0:52:16:12:4a:d1  txqueuelen 1000  (Ethernet)
        RX packets 16516  bytes 19574130 (19.5 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 10850  bytes 1459929 (1.4 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

mininet>
```

mininet> h1 ping -c 5 h2

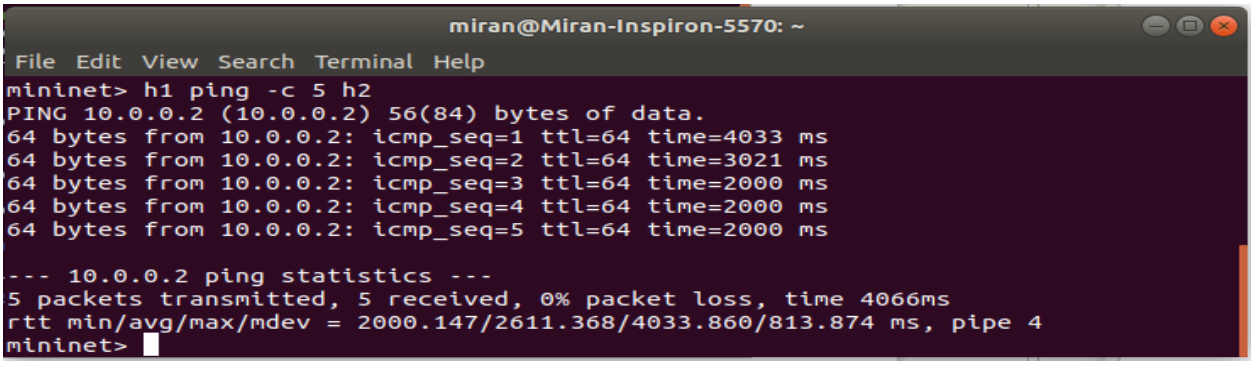
```
miran@Miran-Inspiron-5570: ~
File  Edit  View  Search  Terminal  Help
mininet> h1 ping -c 5 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=20.3 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.643 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.117 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.094 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.098 ms

--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4064ms
rtt min/avg/max/mdev = 0.094/4.268/20.389/8.063 ms
mininet>
```

Exercise 4.2.3:

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

- **mininet> h1 ping -c 5 h2, What happen with the link?**
- **mininet> h1 iperf –s –u &**
- **mininet> h2 iperf –c IPv4_h1 -u, Is there any packet loss?**

```
miran@Miran-Inspiron-5570: ~
File  Edit  View  Search  Terminal  Help
miran@Miran-Inspiron-5570:~$ sudo mn --link tc,bw=10,delay=500ms
[sudo] password for miran:
*** 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
c0
*** Starting 1 switches
s1 ...(10.00Mbit 500ms delay) (10.00Mbit 500ms delay)
*** Starting CLI:
mininet>
```



Conclusion:

Mininet is a useful tool for teaching, development and research. With it, a realistic virtual network, running a real kernel switch and application code, can be set up in a few seconds on a single machine, either virtual or native. It is actively developed and supported. Emulation refers to the running of unchanged code on virtual hardware on the top of the physical host, interactively. It is handy, practical and low cost. It comes with certain restrictions, though, like slower speeds compared to running the same code on a hardware test-bed which is fast and accurate, but expensive. While a simulator requires code modifications and is slow as well. Mininet is a network emulator that enables the creation of a network of virtual hosts, switches, controllers, and links. Mininet hosts standard Linux network software, and its switches support OpenFlow, a software defined network (SDN) for highly flexible custom routing. It constructs a virtual network that appears to be a real physical network. You can create a network topology, simulate it and implement the various network performance parameters such as bandwidth, latency, packet loss, etc, with Mininet, using simple code. You can create the virtual network on a single machine. Mininet permits the creation of multiple nodes enabling a big network to be simulated on a single PC. This is very useful in experimenting with various topologies and different controllers, for different network scenarios. The programs that you run can send packets through virtual switches that seem like real Ethernet interfaces, with a given link speed and delay. Packets get processed by what looks like a real Ethernet switch, router, or middle-box, with a given amount of queuing. The Mininet CLI and API facilitate easy interaction with our network. Virtual hosts, switches, links and controllers created through Mininet are the real thing. They are just created using the Mininet emulator rather than hardware and for the most part, their behaviour is similar to discrete hardware elements.