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: Zahid Hasan Chowdhury

ID: IT-18017

Session: 2017-18

3rd year 2nd semester

Dept. of ICT

Submitted to,

Nazrul Islam
Assistant Professor
Dept. of ICT
MBSTU.

Theory:

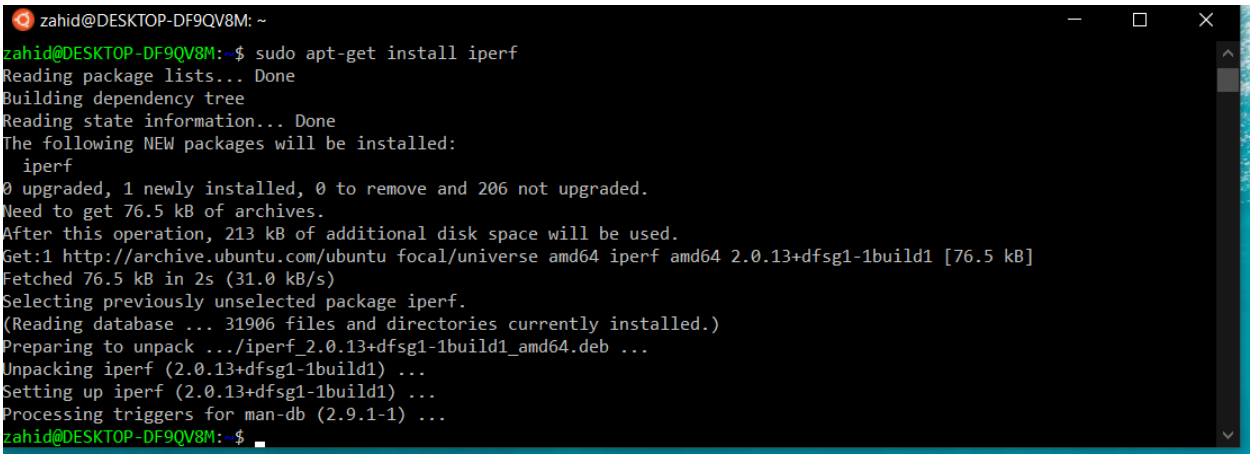
Traffic Generator:

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:



Install Mininet:

```
zahid@DESKTOP-DF9QV8M:~$ sudo apt-get install mininet
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  cgroup-tools libcgroup1 libpython2-stdlib libpython2.7-minimal libpython2.7-stdlib libunbound8 openvswitch-c
ommon
  openvswitch-switch python-pkg-resources python2 python2-minimal python2.7 python2.7-minimal python3-openvswi
tch
  python3-sortedcontainers socat
Suggested packages:
  openvswitch-doc python-setuptools python2-doc python-tk python2.7-doc binutils binfmt-support
  python-sortedcontainers-doc
The following NEW packages will be installed:
  cgroup-tools libcgroup1 libpython2-stdlib libpython2.7-minimal libpython2.7-stdlib libunbound8 mininet
  openvswitch-common openvswitch-switch python-pkg-resources python2 python2-minimal python2.7 python2.7-minim
al
  python3-openvswitch python3-sortedcontainers socat
0 upgraded, 17 newly installed, 0 to remove and 206 not upgraded.
Need to get 7662 kB of archives.
After this operation, 34.0 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu focal-updates/universe amd64 libpython2.7-minimal amd64 2.7.18-1~20.04.
1 [335 kB]Get:2 http://archive.ubuntu.com/ubuntu focal-updates/universe amd64 python2.7-minimal amd64 2.7.18-1
~20.04.1 [1285 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal/universe amd64 python2-minimal amd64 2.7.17-2ubuntu4 [27.5 kB]
```

4. Exercises Exercise

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

```
zahid@DESKTOP-DF9QV8M:~$ 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       #[kmKM]    length of buffer in bytes to read or write (Defaults: TCP=128K, v4 UDP=1470, v6 U
DP=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 a
nd device
  -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:
```

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:

```
ahid@DESKTOP-DF9QV8M:~$ iperf -s
------------------------------------------------------------
erver listening on TCP port 5001
CP window size: 1.00 MByte (default)
------------------------------------------------------------
```

For terminal -2:

```
zahid@DESKTOP-DF9QV8M:~$ 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:  160 KByte (default)
------------------------------------------------------------
[  3] local 127.0.0.1 port 58928 connected with 127.0.0.1 port 5001
[  3] WARNING: did not receive ack of last datagram after 10 tries.
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[  3] Sent 892 datagrams
zahid@DESKTOP-DF9QV8M:~$
```

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?

```
(trail) zahid@DESKTOP-DF9QV8M:~$ iperf -s -u 9900
iperf: ignoring extra argument -- 9900
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:  160 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:

```
trail) zahid@DESKTOP-DF9QV8M:~$ iperf -c 127.0.0.1 -u -l 1000 -t 20 -b 1 -p 9900
ARNING: delay too large, reducing from 8000.0 to 1.0 seconds.
------------------------------------------------------------
lient connecting to 127.0.0.1, UDP port 9900
ending 1000 byte datagrams, IPG target: 8000000000.00 us (kalman adjust)
DP buffer size:  160 KByte (default)
------------------------------------------------------------
  3] local 127.0.0.1 port 64766 connected with 127.0.0.1 port 9900
  3] WARNING: did not receive ack of last datagram after 10 tries.
 ID] Interval       Transfer     Bandwidth
  3]  0.0-20.0 sec  19.5 KBytes  8.00 Kbits/sec
  3] Sent 20 datagrams
trail) zahid@DESKTOP-DF9QV8M:~$
```

For term

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?

```
(trail) zahid@DESKTOP-DF9QV8M:~$ ifconfig
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 1500
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0xfe<compat,link,site,host>
        loop  (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

wifi2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.106  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::1186:ba95:e44b:5a42  prefixlen 64  scopeid 0xfd<compat,link,site,host>
        ether 44:03:2c:e9:6c:68  (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
```

```
trail) zahid@DESKTOP-DF9QV8M:~$ sudo mn
** No default OpenFlow controller found for default switch!
** Falling back to OVS Bridge
** Error setting resource limits. Mininet's performance may be affected.
** Creating network
** Adding controller
** Adding hosts:
1 h2
```

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

mininet> 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

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

mininet> net

```
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

```
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>
```

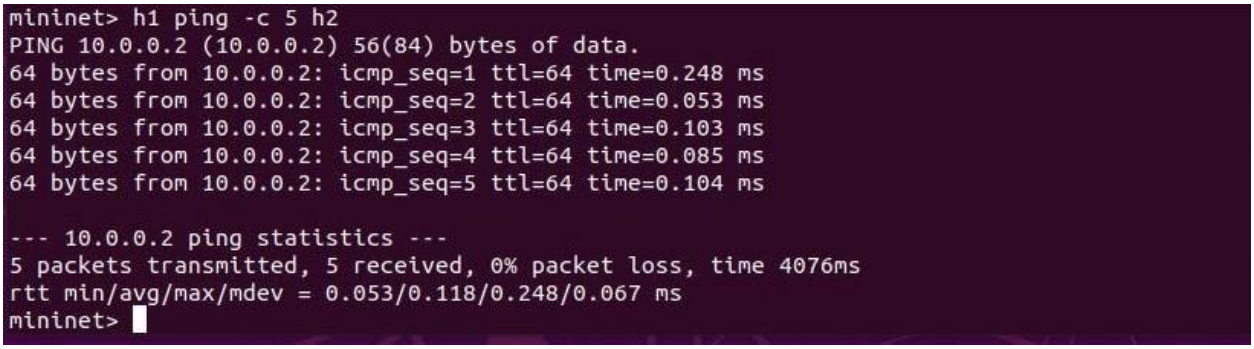
mininet> h1 ifconfig –a

```
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

```
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



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?**

```
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.

- 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.