Mawlana Bhashani Science And Technology University

Lab-Report

Lab Report No: 04

Lab Report Name: Install and use traffic generators as powerful tools for testing network

Group member ID: IT-18013 and IT-18028

Date of Performance: 30-06-2021

Date of Submission: 30-06-2021

Submitted by

Name: Anjom Nour Anika & Fatema-tuz-zanat

ID: IT-18013 & IT-18028.

3rd Year 2nd Semester

Session: 2017-2018

Dept. of ICT, MBSTU

Submitted To

Nazrul Islam

Assistant Professor

Dept. of ICT

MBSTU.

Lab 4:SDN Controllers and Mininet

1.Objectives:

The objective of the lab 4 is to:

*Install and use traffic generators as powerful tools for testing network performance.

1. Install and configure SDN Controller

2. Install and understand how the mininet simulator works

3.Implement and run basic examples for understanding the role of the controller and how it interact with mininet

2.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.

Software Defined Networking: Software-defined networking was pioneered between 2008 and 2011 by work done at Stanford University and the Nicira Company (now part of VMware). The basic premise behind 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.

Controller: OVS-testcontroller is a simple OpenFlow controller that manages any number of switches over the OpenFlow protocol, causing them to function as L2

MAC-learning switches or hubs. It is suitable for initial testing of OpenFlow

networks.

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.

3.Methodology:

Install iperf:

```
anjom@anjom-VirtualBox:~$ sudo apt-get install iperf
[sudo] password for anjom:
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 246 not upgraded.
```

Install mininet:

```
anjom@anjom-VirtualBox:~$ 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 246 not upgraded.
```

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

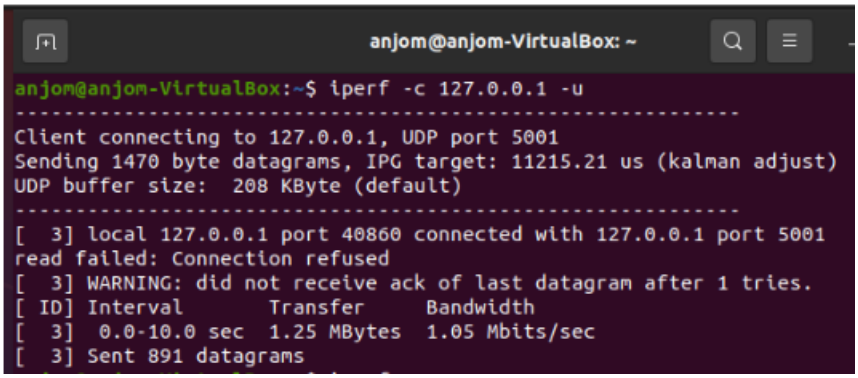
```
anjom@anjom-VirtualBox:~$ iperf--help
iperf--help: command not found
```

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).

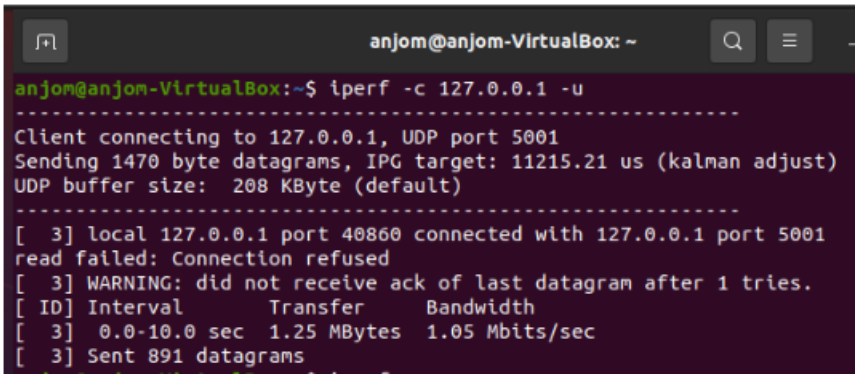
For terminal-1:

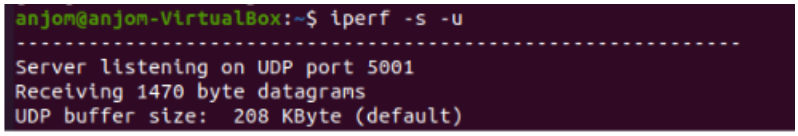
```
anjom@anjom-VirtualBox:~$ iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
```

For terminal-2:



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? What is different from the statistics provided in exercise 4.1.1





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?

For terminal 1:

```
anjom@anjom-VirtualBox: ~
anjom@anjom-VirtualBox:~$ 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 42993 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:

```
anjom@anjom-VirtualBox:~$ ipearf -s -u -p 9900

Command 'ipearf' not found, did you mean:

  command 'iperf' from deb iperf (2.0.13+dfsg1-1build1)

Try: sudo apt install <deb name>
```

4.2:Using mininet Exercise 4.2.1: Open two Linux terminals, and execute the command line ifconfig in terminal-1. 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?

```
anjom@anjom-VirtualBox:~$ 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::ff93:3d39:cbd5:1952  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:4b:e5:b0  txqueuelen 1000  (Ethernet)
        RX packets 12892  bytes 18581210 (18.5 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 6049  bytes 403178 (403.1 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 2235  bytes 1918179 (1.9 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2235  bytes 1918179 (1.9 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
anjom@anjom-VirtualBox:~$ sudo mn
[sudo] password for anjom:
*** 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:
```

```
anjom@anjom-VirtualBox:~$ 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::ff93:3d39:cbd5:1952  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:4b:e5:b0  txqueuelen 1000  (Ethernet)
        RX packets 12892  bytes 18581210 (18.5 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 6049  bytes 403178 (403.1 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 2235  bytes 1918179 (1.9 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2235  bytes 1918179 (1.9 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

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> nodes

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

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> dump

```
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=5840>
<Host h2: h2-eth0:10.0.0.2 pid=5842>
<OVSBridge s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=5847>
```

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::50e3:98ff:fe84:a384  prefixlen 64  scopeid 0x20<link>
        ether 52:e3:98:84:a3:84  txqueuelen 1000  (Ethernet)
        RX packets 40  bytes 4397 (4.3 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  collisio

mininet>
```


mininet> 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::b977:b09c:b4c1:bdcd  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:95:5c:6c  txqueuelen 1000  (Ethernet)
        RX packets 7411  bytes 10285787 (10.2 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3235  bytes 246459 (246.4 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 4078  bytes 3775763 (3.7 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4078  bytes 3775763 (3.7 MB)
```



```
        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 02:1c:b6:44:05:4c  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 22  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::f42d:afff:febb:c7a5  prefixlen 64  scopeid 0x20<link>
        ether f6:2d:af:bb:c7:a5  txqueuelen 1000  (Ethernet)
        RX packets 13  bytes 1006 (1.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 43  bytes 4662 (4.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

s1-eth2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::b030:b4ff:fe25:ec47  prefixlen 64  scopeid 0x20<link>
        ether b2:30:b4:25:ec:47  txqueuelen 1000  (Ethernet)
        RX packets 13  bytes 1006 (1.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 41  bytes 4486 (4.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisio

mininet>
```


mininet> h1 ping –c 5 h2

```
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=0.481 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.128 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.125 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.128 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.125 ms

--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4103ms
rtt min/avg/max/mdev = 0.125/0.197/0.481/0.141 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? Modify iperf for creating packet loss in the mininet network, which is the command line?

```
anjom@anjom-VirtualBox:~$ sudo mn--link tc,bw=0,delay=500ms
sudo: mn--link: command not found
```

5.Conclusion:

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's code is almost entirely Python, except for a short C utility.Mininet-based networks cannot (currently) exceed the CPU or bandwidth available on a single server.Mininet cannot (currently) run non-Linux-compatible OpenFlow switches or applications; this has not been a major issue in practice