

**Name:** Md.Abdullah

**ID** : IT-17015

**Lab report no:** 08

Name of the report: SDN Controllers and Mininet

**Objectives:**

- ✓ Understanding SDN Controllers
- ✓ Learn how mininet work

**Theory:**

**Explain how the traffic generators work?**

**Ans:**

**Traffic Generators**

A traffic generators is used to put traffic onto a network for other machines to consume. Logically, a traffic generator has a physical layer address (and usually higher-level address), because it is supposed to look like a machine on the network to the target machines receiving the traffic

**Question 5.3: Which is the main difference between configuring UDP and TCP traffic?**

**Ans:**

UDP traffic is fully customized in terms of packet length and bandwidth. Instead TCP only allows customizing the amount of traffic.

**Question 5.4: In your opinion, which are the main advantages and disadvantages of working with mininet? Provide at least 3.**

**Ans:**

Advantage:

1. free
2. easy to install
3. allow test scalability

Disadvantage:

1. It is not real devices
2. No real traffic can be injected
3. No INTERNET output.

**Question 5.5: Explain the architecture of Software defined Networks?**

**Ans:**

Controller role: Brain of the network

Switch role: Hands of the network

Host role: Clients

**Question 5.6: What is the advantage of having a programmable Controller?**

**Ans:**

Fully customize, program and control your network.

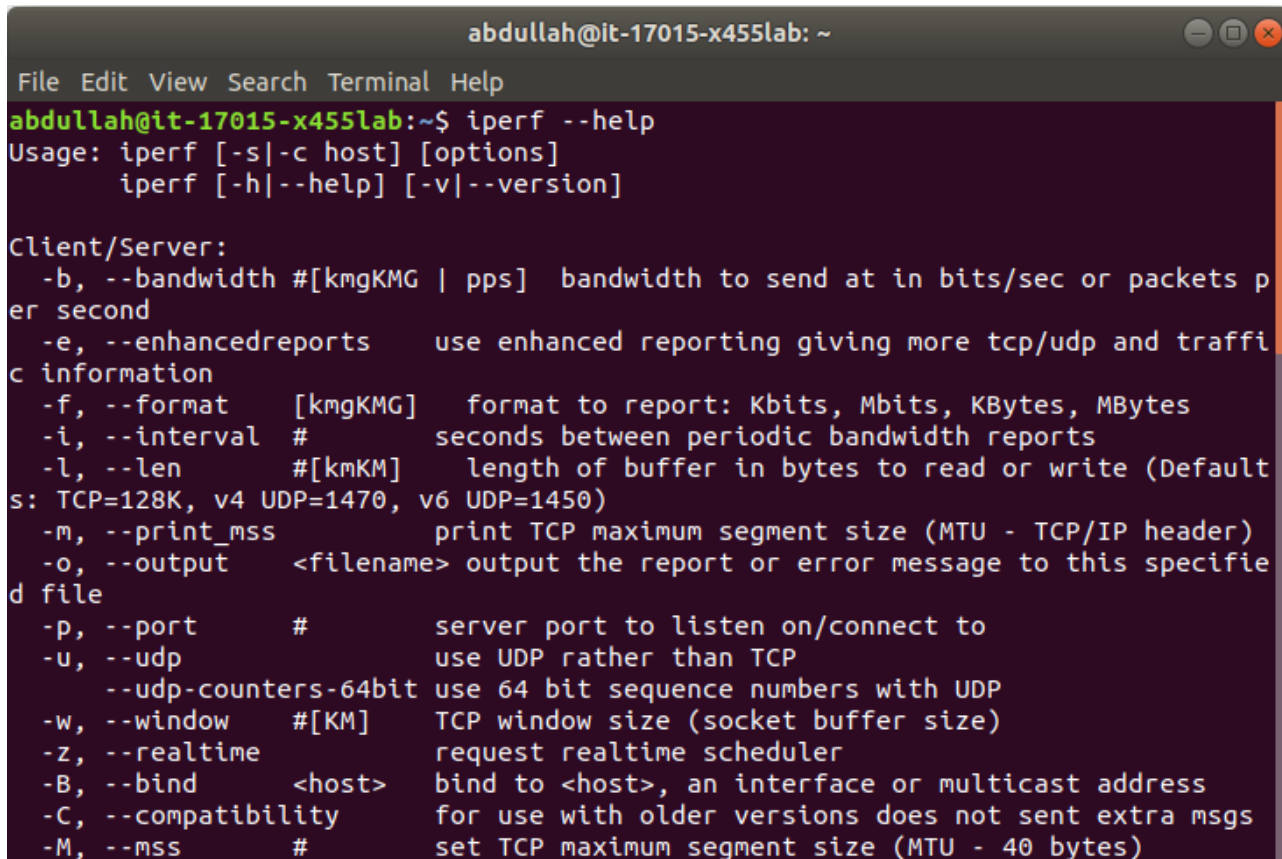
## Section 4.1: Using traffic Generators

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

**Ans:**

Client/Server:

- f, --format [kmKM] format to report: Kbits, Mbits, KBytes, MBytes
- i, --interval seconds between periodic bandwidth reports
- l, --len [KM] length of buffer to read or write (default 8 KB)
- m, --print\_mss print TCP maximum segment size (MTU - TCP/IP header)



```
abdullah@it-17015-x455lab: ~  
File Edit View Search Terminal Help  
abdullah@it-17015-x455lab:~$ 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)
```

**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). Which are the statistics provided at the end of transmission?**

**Ans:**

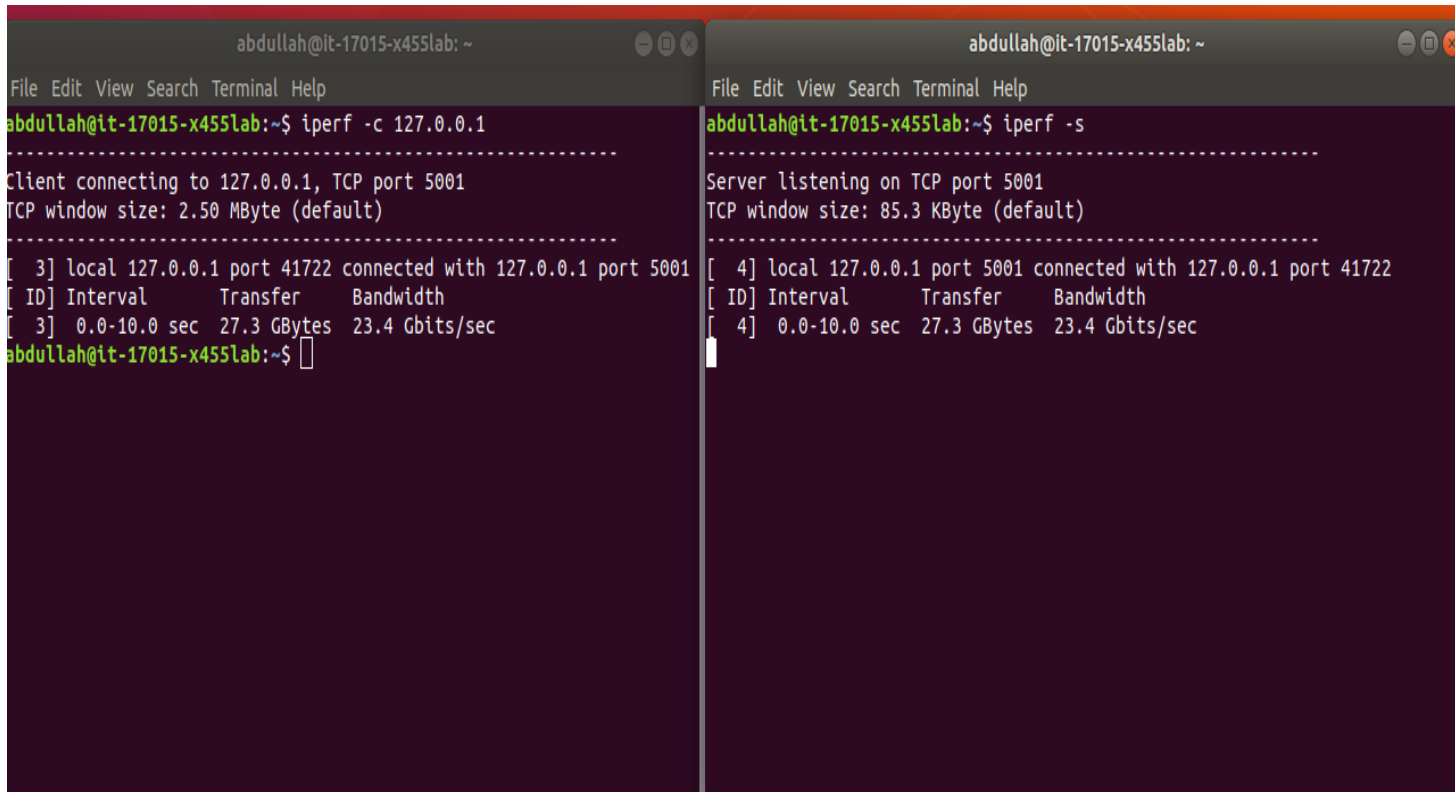
First we need to create a server by the following command:

`$ iperf -s`

Then we need to create a client by the following command:

`$ iperf -c 127.0.0.1`

Here is the output:



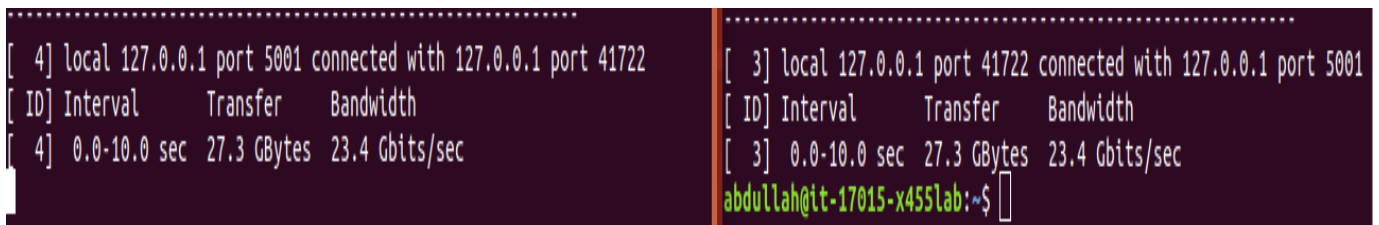
The image shows two terminal windows side-by-side. The left window is the client terminal, and the right window is the server terminal. Both show the execution of iperf commands and the resulting statistics.

```
abdullah@it-17015-x455lab: ~  
File Edit View Search Terminal Help  
abdullah@it-17015-x455lab:~$ iperf -c 127.0.0.1  
.....  
Client connecting to 127.0.0.1, TCP port 5001  
TCP window size: 2.50 MByte (default)  
.....  
[ 3] local 127.0.0.1 port 41722 connected with 127.0.0.1 port 5001  
[ ID] Interval      Transfer    Bandwidth  
[ 3]  0.0-10.0 sec  27.3 GBytes 23.4 Gbits/sec  
abdullah@it-17015-x455lab:~$
```

```
abdullah@it-17015-x455lab: ~  
File Edit View Search Terminal Help  
abdullah@it-17015-x455lab:~$ iperf -s  
.....  
Server listening on TCP port 5001  
TCP window size: 85.3 KByte (default)  
.....  
[ 4] local 127.0.0.1 port 5001 connected with 127.0.0.1 port 41722  
[ ID] Interval      Transfer    Bandwidth  
[ 4]  0.0-10.0 sec  27.3 GBytes 23.4 Gbits/sec
```

**The following statistics are provided at the end of the transmission:**

The time interval, the amount of data transfer and the bandwidth.



This block shows a close-up of the statistics output from the client and server terminals. The client output is on the left, and the server output is on the right.

```
[ 4] local 127.0.0.1 port 5001 connected with 127.0.0.1 port 41722  
[ ID] Interval      Transfer    Bandwidth  
[ 4]  0.0-10.0 sec  27.3 GBytes 23.4 Gbits/sec
```

```
[ 3] local 127.0.0.1 port 41722 connected with 127.0.0.1 port 5001  
[ ID] Interval      Transfer    Bandwidth  
[ 3]  0.0-10.0 sec  27.3 GBytes 23.4 Gbits/sec  
abdullah@it-17015-x455lab:~$
```

**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?**

**Ans:**

First we need to create a server by the following command:

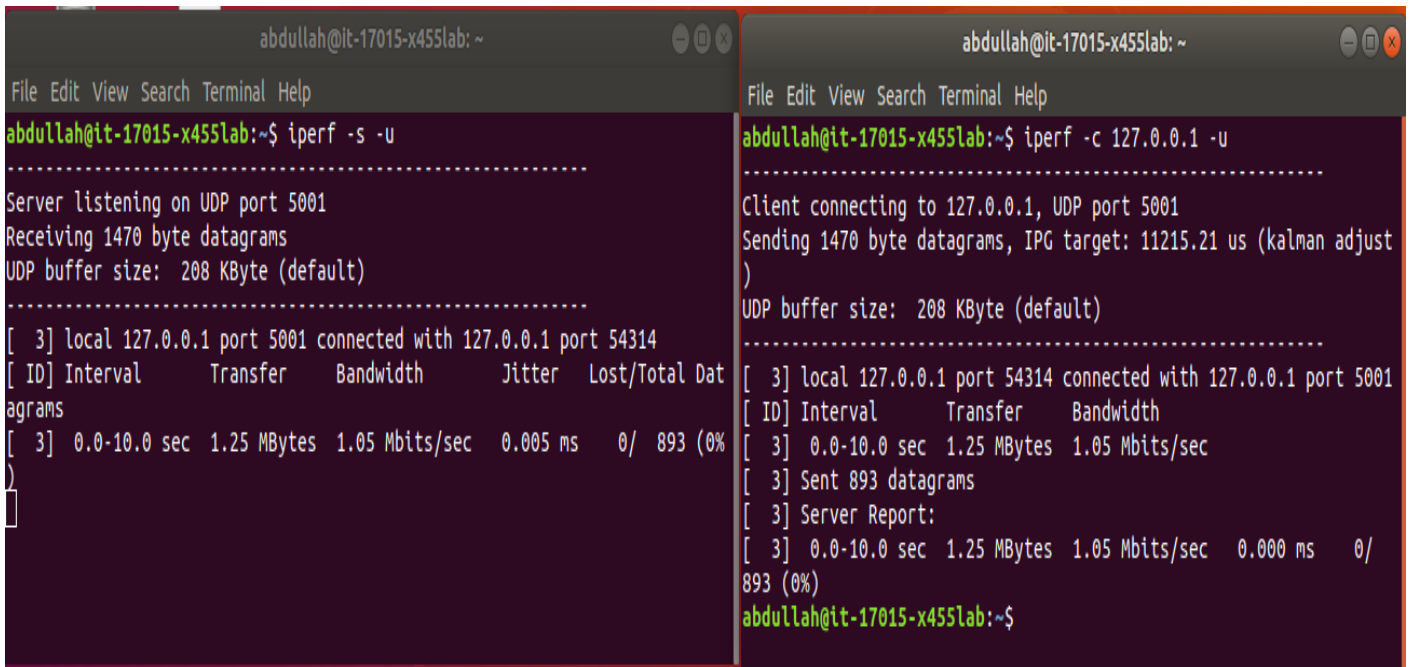
```
$ iperf -s -u
```

Here **u** for UDP traffic

Then we need to create a client by the following command:

```
$ iperf -c 127.0.0.1 -u
```

Here is the output of server and client:



The image shows two terminal windows side-by-side. The left window is the server terminal, and the right window is the client terminal. Both are running on a system named 'abdullah@it-17015-x455lab: ~'.

**Server Terminal Output:**

```
abdullah@it-17015-x455lab:~$ 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 54314
[ ID] Interval      Transfer    Bandwidth    Jitter  Lost/Total Datagrams
[ 3] 0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec  0.005 ms   0/ 893 (0%)
)
```

**Client Terminal Output:**

```
abdullah@it-17015-x455lab:~$ 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 54314 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 893 datagrams
[ 3] Server Report:
[ 3] 0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec  0.000 ms   0/ 893 (0%)
abdullah@it-17015-x455lab:~$
```

**Which are the statistics are provided at the end of transmission?**

**Ans:** The time interval, the amount of data transfer, the bandwidth, the jitter and the datagram loss percentage.

```
[ 3] local 127.0.0.1 port 5001 connected with 127.0.0.1 port 54314
[ ID] Interval      Transfer    Bandwidth    Jitter  Lost/Total Datagrams
[ 3] 0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec  0.005 ms   0/ 893 (0%)
```

**What is different from the statistics provided in exercise 4.1.1.?**

**Ans:**

The jitter and the datagram loss percentage, because TCP guaranties 0% packet loss.

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

- ✓ Packet length = 1000bytes
  - ✓ Time = 20 seconds
  - ✓ Bandwidth = 1Mbps
  - ✓ Port = 9900
- Which are the command lines?

**Ans:**

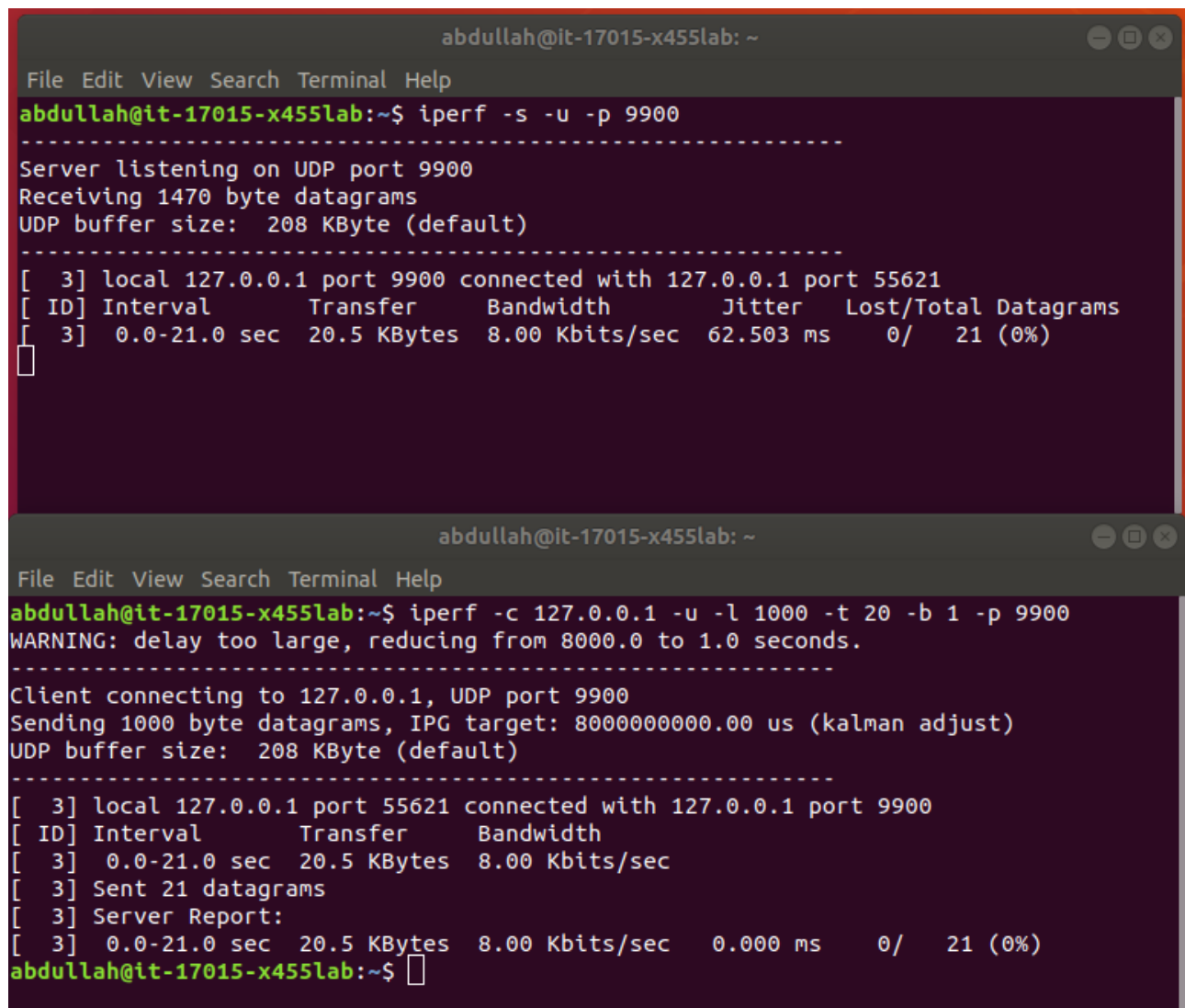
The client command is:

```
iperf -c 127.0.0.1 -u -l 1000 -t 20 -b 1 -p 9900
```

The server command is:

```
iperf -s -u -p 9900
```

**Output:**



The image shows two terminal windows from the user 'abdullah' on the host 'it-17015-x455lab'. The top window shows the server running 'iperf -s -u -p 9900'. It receives 1470 byte datagrams and reports a bandwidth of 8.00 Kbits/sec. The bottom window shows the client running 'iperf -c 127.0.0.1 -u -l 1000 -t 20 -b 1 -p 9900'. It connects to the server and reports a bandwidth of 8.00 Kbits/sec.

```
abdullah@it-17015-x455lab: ~  
File Edit View Search Terminal Help  
abdullah@it-17015-x455lab:~$ iperf -s -u -p 9900  
-----  
Server listening on UDP port 9900  
Receiving 1470 byte datagrams  
UDP buffer size: 208 KByte (default)  
-----  
[ 3] local 127.0.0.1 port 9900 connected with 127.0.0.1 port 55621  
[ ID] Interval      Transfer      Bandwidth      Jitter    Lost/Total Datagrams  
[ 3] 0.0-21.0 sec  20.5 KBytes   8.00 Kbits/sec 62.503 ms    0/ 21 (0%)  
  
abdullah@it-17015-x455lab: ~  
File Edit View Search Terminal Help  
abdullah@it-17015-x455lab:~$ 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 55621 connected with 127.0.0.1 port 9900  
[ ID] Interval      Transfer      Bandwidth  
[ 3] 0.0-21.0 sec  20.5 KBytes   8.00 Kbits/sec  
[ 3] Sent 21 datagrams  
[ 3] Server Report:  
[ 3] 0.0-21.0 sec  20.5 KBytes   8.00 Kbits/sec  0.000 ms    0/ 21 (0%)  
abdullah@it-17015-x455lab:~$
```

## Sections 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?

**Ans:**

```
File Edit View Search Terminal Help
abdullah@it-17015-x455lab:~$ ifconfig
enp2s0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
        ether 38:d5:47:90:e1:e2 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 1037247 bytes 29322277112 (29.3 GB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 1037247 bytes 29322277112 (29.3 GB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlp3s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 192.168.43.29 netmask 255.255.255.0 broadcast 192.168.43.255
        inet6 fe80::fdb9:febb:db0f:44bd prefixlen 64 scopeid 0x20<link>
        ether 74:c6:3b:d7:57:7d txqueuelen 1000 (Ethernet)
        RX packets 93818 bytes 116676394 (116.6 MB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 92333 bytes 11917582 (11.9 MB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

abdullah@it-17015-x455lab:~$
```

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

```
File Edit View Search Terminal Help
abdullah@it-17015-x455lab:~$ sudo mn
[sudo] password for abdullah:
*** 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>
```

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

Ans:

```
abdullah@it-17015-x455lab:~$ ifconfig
enp2s0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether 38:d5:47:90:e1:e2 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 1037369 bytes 29322287574 (29.3 GB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1037369 bytes 29322287574 (29.3 GB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

s1-eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet6 fe80::c6e:2dff:fe9d:4028 prefixlen 64 scopeid 0x20<link>
    ether 0e:6e:2d:9d:40:28 txqueuelen 1000 (Ethernet)
    RX packets 7 bytes 586 (586.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 28 bytes 3620 (3.6 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

s1-eth2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet6 fe80::dc37:aeff:fe0a:337b prefixlen 64 scopeid 0x20<link>
    ether de:37:ae:0a:33:7b txqueuelen 1000 (Ethernet)
    RX packets 8 bytes 656 (656.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 25 bytes 3390 (3.3 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlp3s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.43.29 netmask 255.255.255.0 broadcast 192.168.43.255
```



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

**mininet> help**

**Ans:**

Help for using mininet

```
mininet> help

Documented commands (type help <topic>):
=====
EOF      gterm  iperfudp  nodes      pingpair    py      switch
dpctl    help   link      noecho     pingpairfull  quit    time
dump     intfz  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**

**Ans:**

Description of the nodes

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

**mininet> net**

**Ans:**

Description of the network

```
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

Ans:

Details of the network nodes IP address.

```
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=13374>
<Host h2: h2-eth0:10.0.0.2 pid=13376>
<OVSBridge s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=13381>
mininet>
```

mininet> h1 ifconfig -a

Ans:

Configuration of host 1

```
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::305d:3fff:fe75:9f69 prefixlen 64 scopeid 0x20<link>
    ether 32:5d:3f:75:9f:69 txqueuelen 1000 (Ethernet)
    RX packets 44 bytes 5538 (5.5 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>
```

mininet> s1 ifconfig -a

```
mininet> s1 ifconfig -a
enp2s0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether 38:d5:47:90:e1:e2 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 1037373 bytes 29322287950 (29.3 GB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1037373 bytes 29322287950 (29.3 GB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ovs-system: flags=4098<BROADCAST,MULTICAST> mtu 1500
    ether 1e:e9:38:49:d8:77 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 32:97:00:da:37:41 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 45 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::c6e:2dff:fe9d:4028 prefixlen 64 scopeid 0x20<link>
    ether 0e:6e:2d:9d:40:28 txqueuelen 1000 (Ethernet)
    RX packets 13 bytes 1006 (1.0 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 44 bytes 5538 (5.5 KB)
```

**mininet> h1 ping -c 5 h2**

**Ans:**

ping between h1 and 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=0.532 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.076 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.075 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.071 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.074 ms

--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4098ms
rtt min/avg/max/mdev = 0.071/0.165/0.532/0.183 ms
mininet> █
```

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

**Ans:**

```
ovs-vsctl: missing command name (use --help for help)
abdullah@it-17015-x455lab:~$ 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:
-----
Caught exception. Cleaning up...

Exception: Error creating interface pair (h1-eth0,s1-eth1): RTNETLINK answers: File exists

-----
*** Removing excess controllers/ofprotocols/ofdatapaths/pings/noxes
killall controller ofprotocol ofdatapath ping nox_core lt-nox_core ovs-openflowd ovs-controller udpbwtest mnexec ivs 2> /dev/null
killall -9 controller ofprotocol ofdatapath ping nox_core lt-nox_core ovs-openflowd ovs-controller udpbwtest mnexec ivs 2> /dev/null
pkill -9 -f "sudo mnexec"
*** Removing junk from /tmp
rm -f /tmp/vconn* /tmp/vlogs* /tmp/*.out /tmp/*.log
*** Removing old X11 tunnels
*** Removing excess kernel datapaths
ps ax | egrep -o 'dp[0-9]+' | sed 's/dp/nl:/'
*** Removing OVS datapaths
ovs-vsctl --timeout=1 list-br
ovs-vsctl --if-exists del-br s1
ovs-vsctl --timeout=1 list-br
*** Removing all links of the pattern foo-ethX
ip link show | egrep -o '([_.:alnum:]+-eth[[:digit:]]+)'
( ip link del s1-eth1; ip link del s1-eth2 ) 2> /dev/null
ip link show
*** Killing stale mininet node processes
pkill -9 -f mininet:
*** Shutting down stale tunnels
pkill -9 -f Tunnel=Ethernet
```

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

**Ans:**

```
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=0.532 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.076 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.075 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.071 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.074 ms

--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4098ms
rtt min/avg/max/mdev = 0.071/0.165/0.532/0.183 ms
mininet> █
```

**Discussion:**

I can successfully understand the all the things and can be able to execute with my ubuntu terminal. This lab helps me to understand the basic mininet, how mininet works, advanced mininet and SDN controllers.