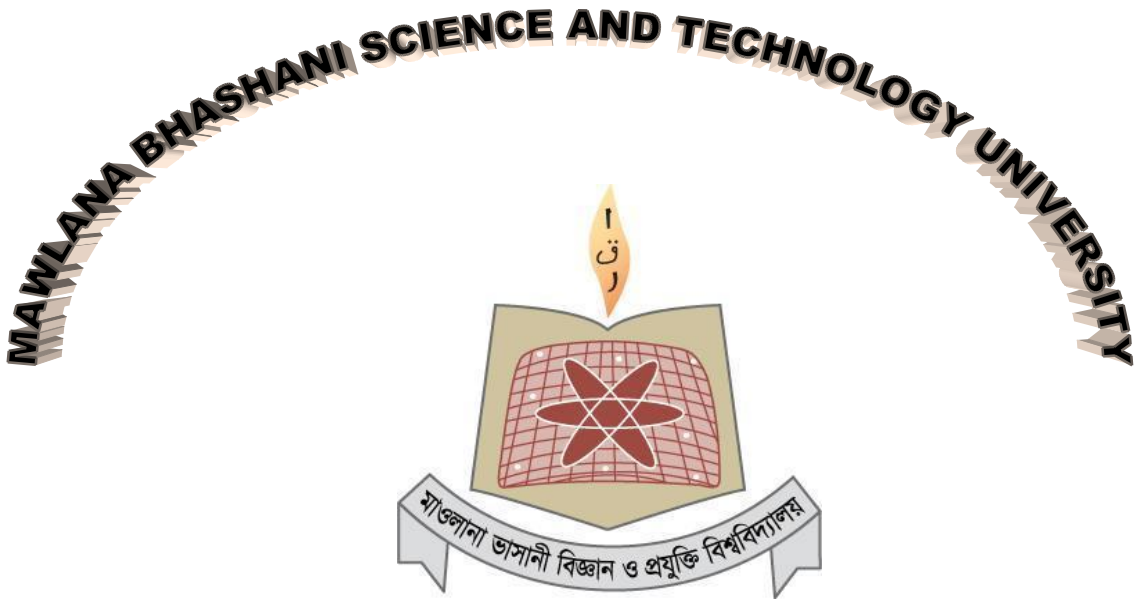


Lab Report: 04

Report Name: SDN Controllers and Mininet

Course code: ICT-3208

Course title: Computer Networks Lab

Date of Performance: 01/02/2021

Date of Submission: 01/02/2021

SUBMITTED BY

Name: Shakhera khanom & Masuk Mia

ID: IT-18033 & IT-18049

3rd year 2nd semester

Session: 2017-18

Department of ICT,

MBSTU.

SUBMITTED TO

Nazrul Islam

Assistant Professor

Department of ICT,

MBSTU.

Experiment No: 04

Experiment Name: SDN Controllers and Mininet

Objectives:

- Install and use traffic generators as powerful tools for testing network performance.
- Install and configure SDN Controller
- Install and understand how the mininet simulator works
- Implement and run basic examples for understanding the role of the controller and how it interact with mininet

Theory:

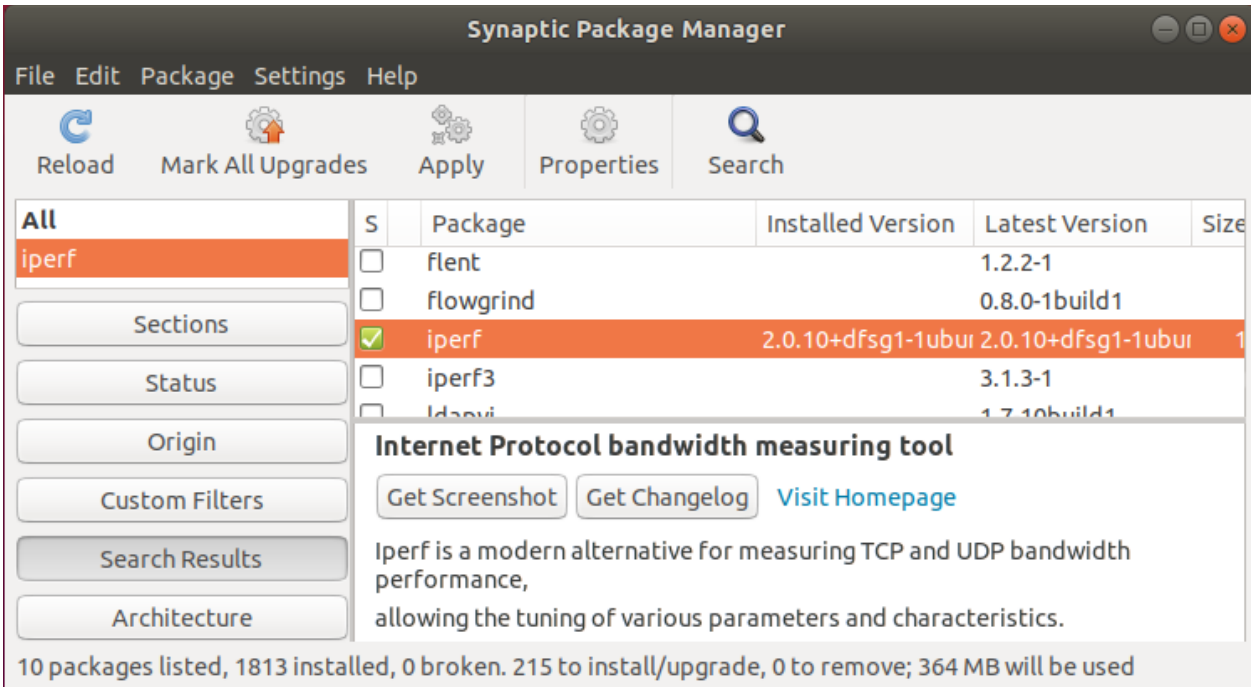
Traffic Generator:

What is iPerf?: iPerf is a tool for performing network throughput measurements. It can test either TCP or UDP throughput. To perform an iperf test the user must establish both a server (to discard traffic) and a client (to generate traffic). For each test it reports the bandwidth, loss, and other parameters

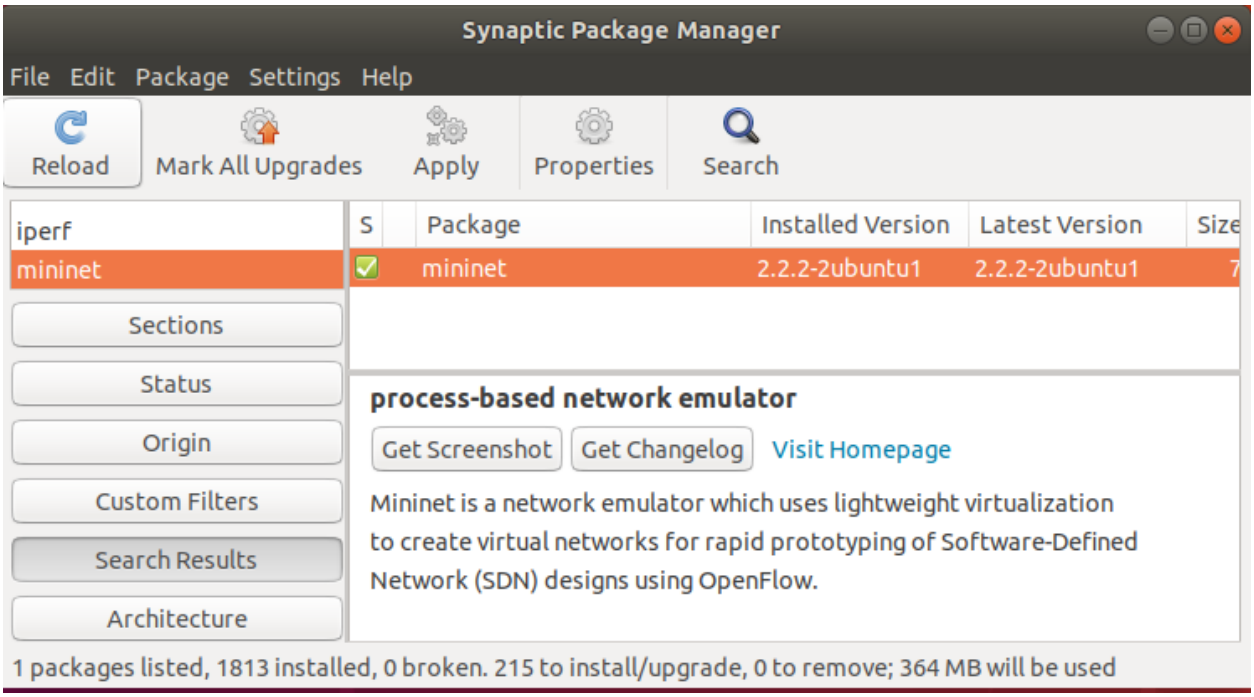
Mininet: Mininet creates a realistic virtual network, running real kernel, switch and application code, on a single machine (VM, cloud or native) Because you can easily interact with your network using the Mininet CLI (and API), customize it, share it with others, or deploy it on real hardware, Mininet 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

Methodology:

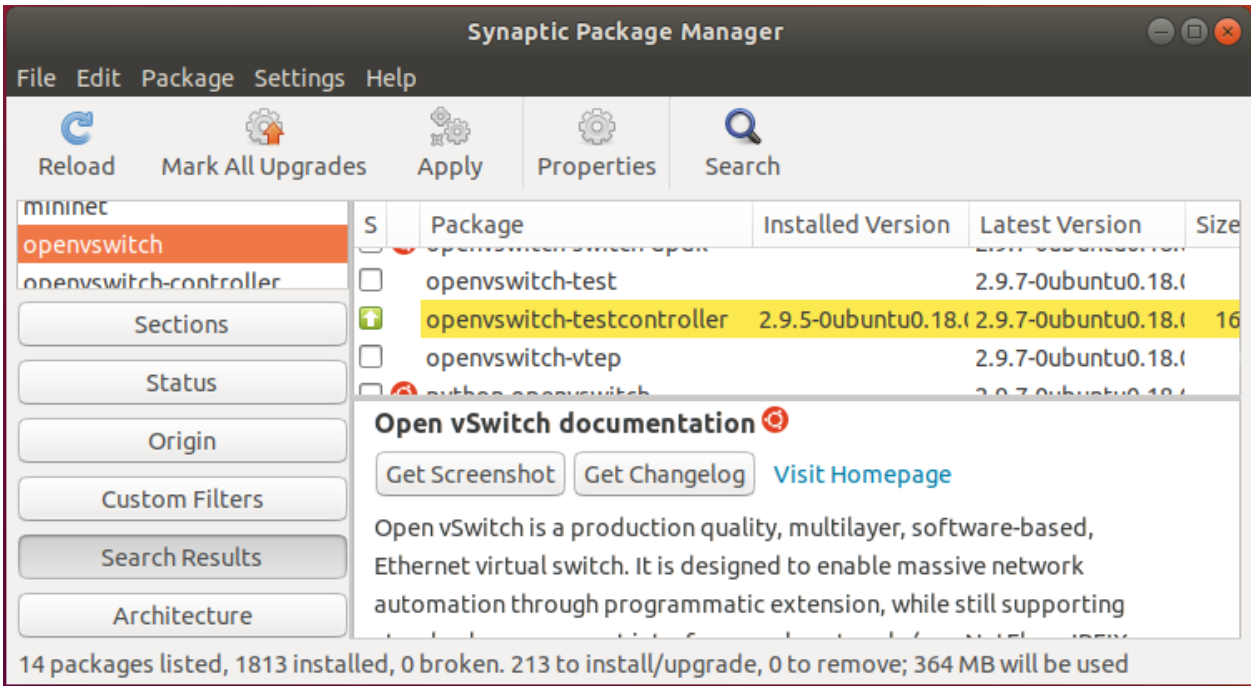
Install iperf:



Install mininet:



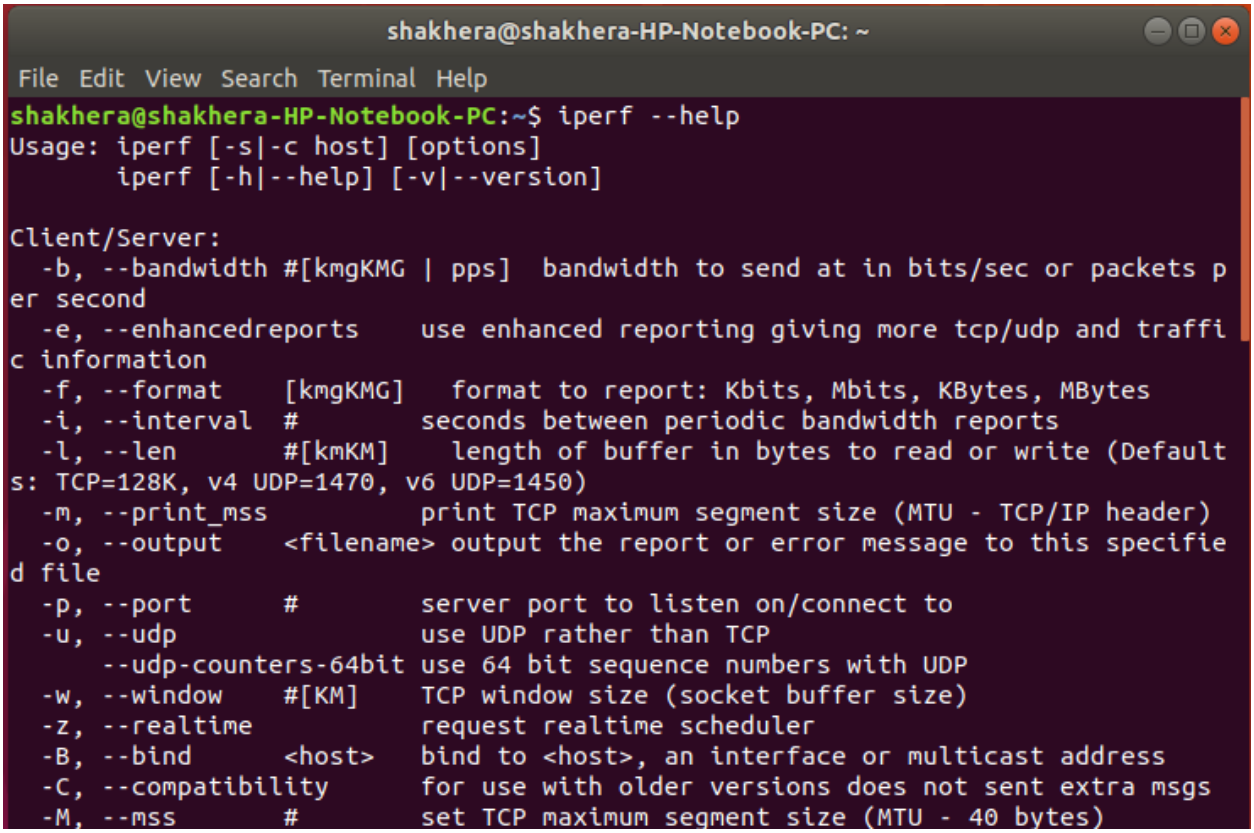
Install Controller:



Exercises:

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

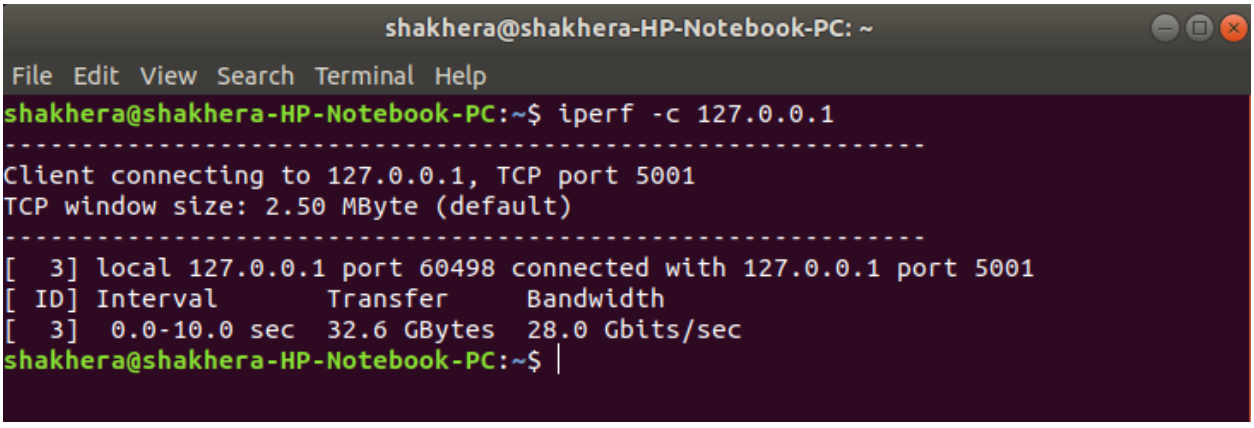
iperf –help



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). Note: use the loopback address. Which are the statistics provided at the end of transmission?

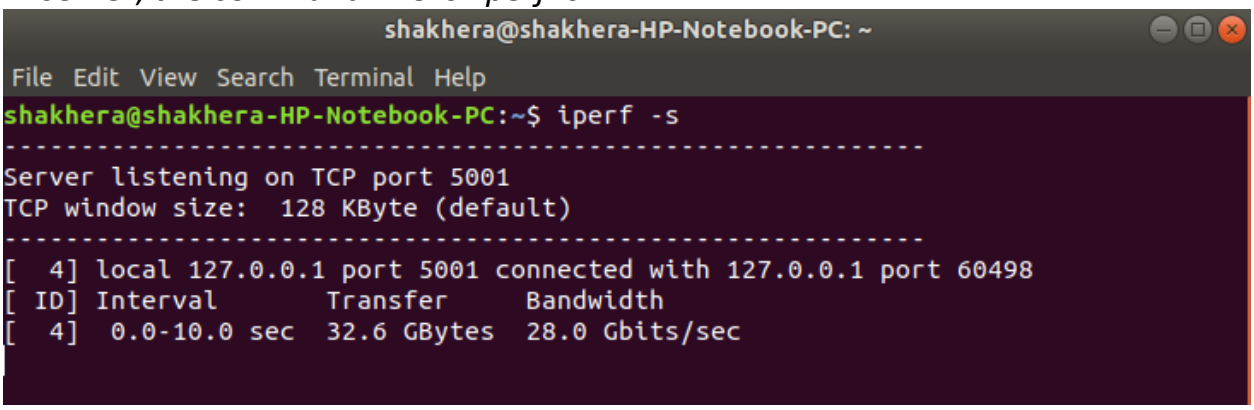
In terminal -1:

To run client in terminal, the command line is: *iperf -c 127.0.0.1*



In terminal -2:

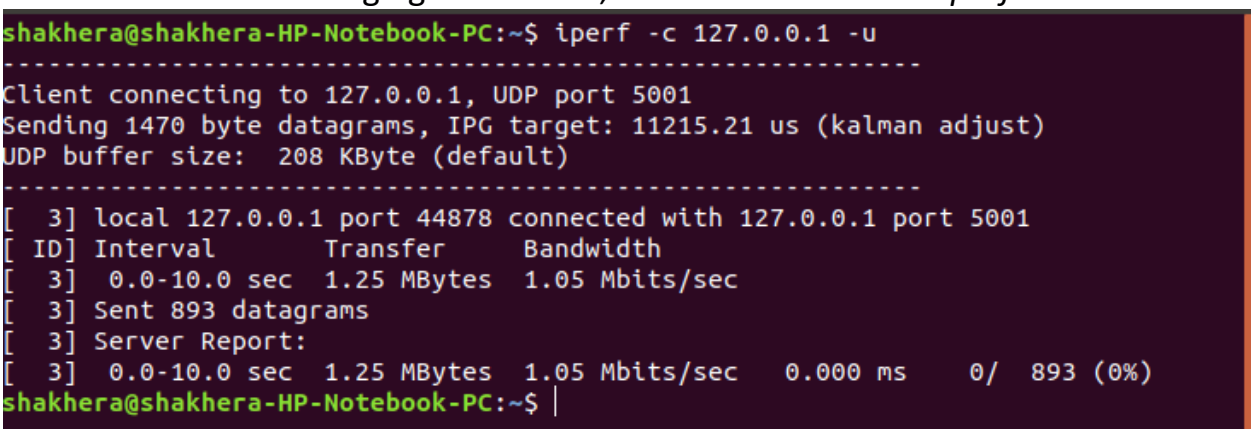
In server, the command line is: *iperf -s*



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.

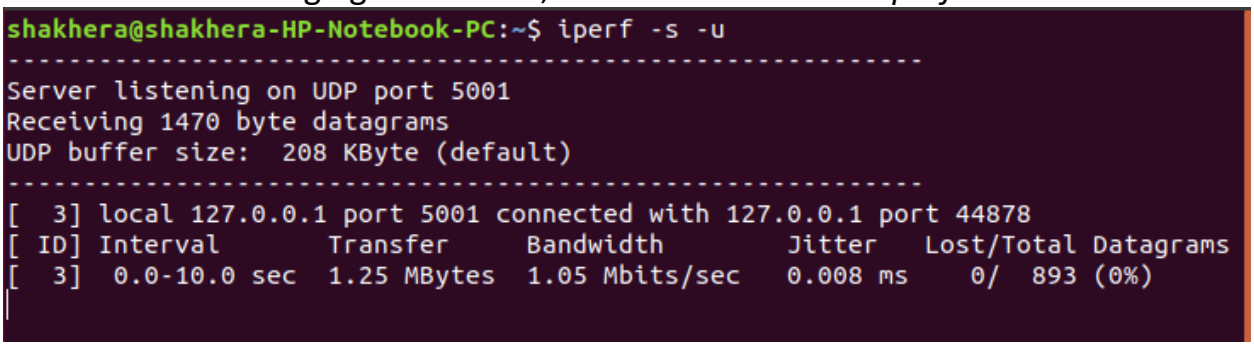
For terminal -1:

To run client for exchanging UDP traffic, the command line is: *iperf -c 127.0.0.1 -u*



For terminal -2:

In server for exchanging UDP traffic, the command line is: *iperf -s -u*



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?

For terminal -1:

To run client for exchanging UDP traffic, with pocket length, time, bandwidth and port, the command line is: *iperf -c 127.0.0.1 -u -l 1000 -t 20 -b 1M -p 9900*

```
shakhera@shakhera-HP-Notebook-PC:~$ iperf -c 127.0.0.1 -u -l 1000-t 20 -b 1 -p 9900

iperf: ignoring extra argument -- 20
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 39778 connected with 127.0.0.1 port 9900
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-12.0 sec  11.7 KBytes  8.00 Kbits/sec
[  3] Sent 12 datagrams
[  3] Server Report:
[  3]  0.0-12.0 sec  11.7 KBytes  8.00 Kbits/sec   0.000 ms    0/   12 (0%)
shakhera@shakhera-HP-Notebook-PC:~$
```

For terminal -2:

In server for exchanging UDP traffic, the command line is: *iperf -s -u -p 9900*

```
shakhera@shakhera-HP-Notebook-PC:~$ 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 39778
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  3]  0.0-12.0 sec  11.7 KBytes  8.00 Kbits/sec   0.000 ms    0/   12 (0%)
```

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?

In terminal-1:
The *ifconfig* command shows active network interfaces. This is generally the primary network interface – usually eth0 and the loopback address as shown in the example below.

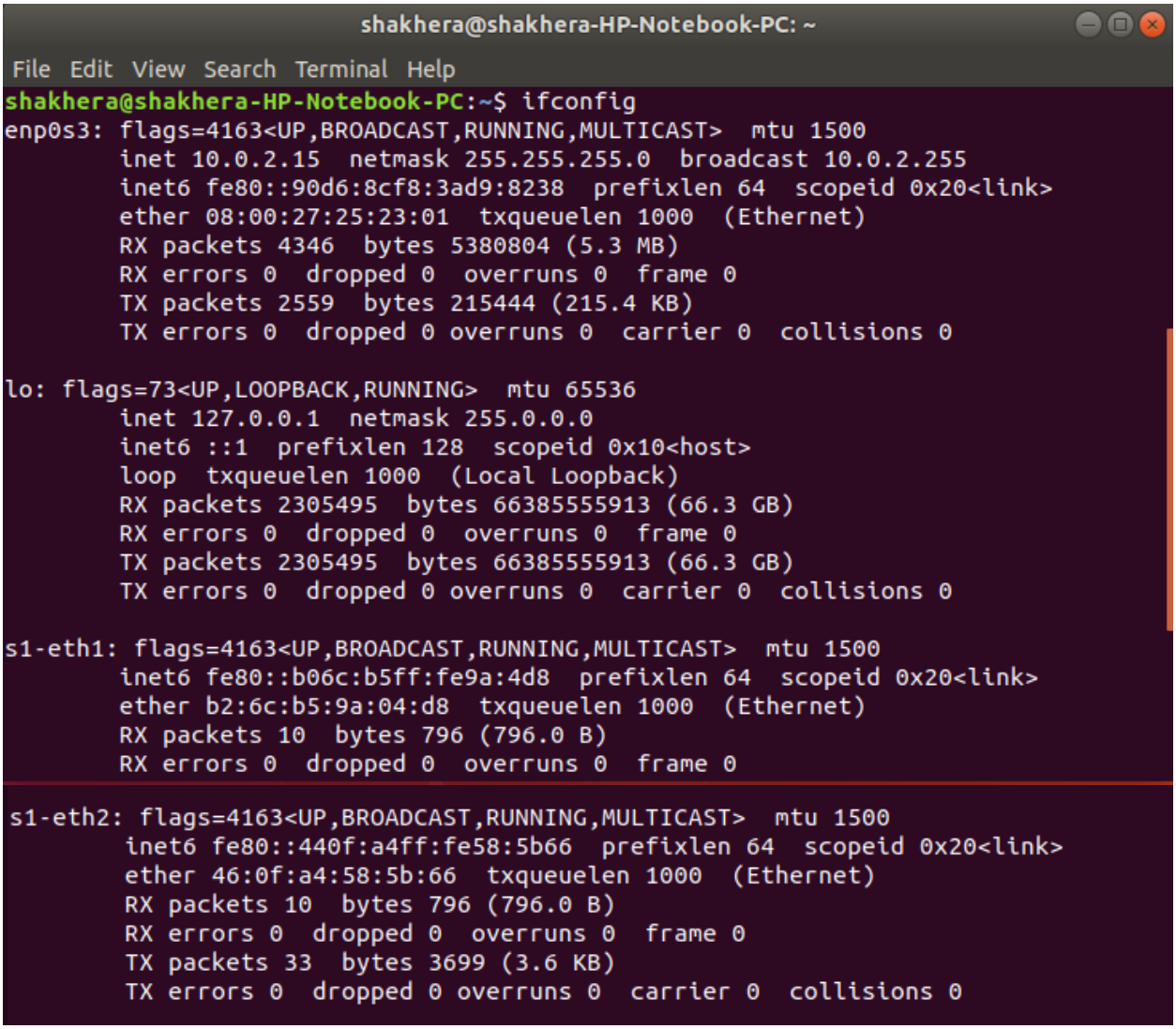
```
shakhera@shakhera-HP-Notebook-PC:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::90d6:8cf8:3ad9:8238  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:25:23:01  txqueuelen 1000  (Ethernet)
        RX packets 4343  bytes 5380624 (5.3 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2418  bytes 204388 (204.3 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 2305334  bytes 66385543417 (66.3 GB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2305334  bytes 66385543417 (66.3 GB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

shakhera@shakhera-HP-Notebook-PC:~$
```

In terminal-2: *Sudo mn*

```
shakhera@shakhera-HP-Notebook-PC:~$ sudo mn
[sudo] password for shakhera:
*** 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>
```

In terminal-1:

The *ifconfig* command shows active network interfaces. Usually the interface of eth0, loopback address, eth1 and eth2 are as shown in the example below.



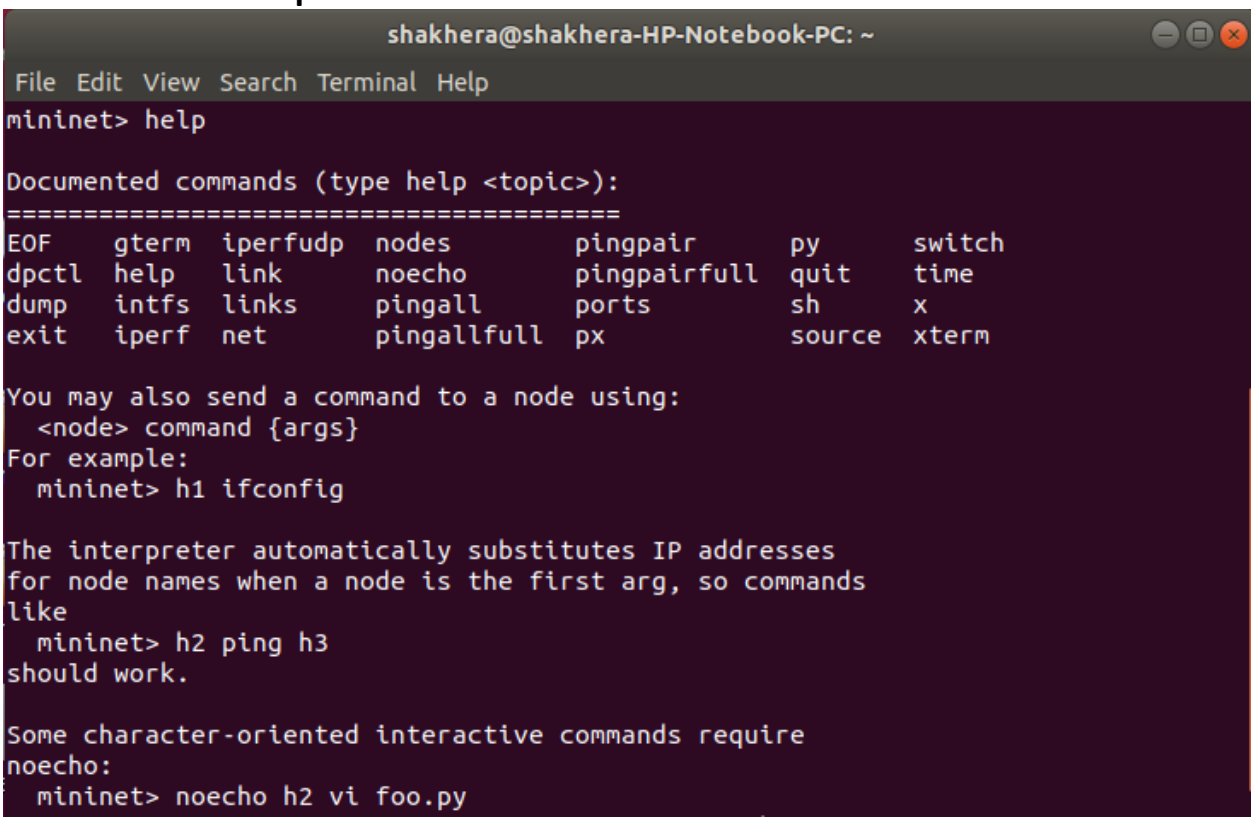
Exercise 4.2.2: Interacting with mininet; in terminal-2, display the following command lines and explain what it does:

- **mininet> help**
- **mininet> nodes**
- **mininet> net**
- **mininet> dump**
- **mininet> h1 ifconfig –a**

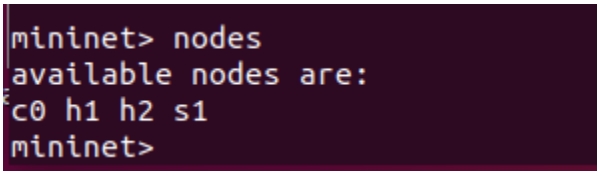
- **mininet> s1 ifconfig –a**
- **mininet> h1 ping -c 5 h2**
- **Draw the network topology that is created in mininet:**

In terminal-2:

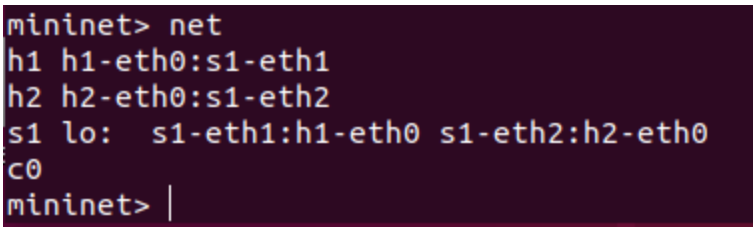
mininet> help



mininet> nodes



mininet> net



mininet> dump

```
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=6726>
<Host h2: h2-eth0:10.0.0.2 pid=6728>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=6733>
<Controller c0: 127.0.0.1:6653 pid=6719>
mininet>
```

mininet> h1 ifconfig –a: You should see the host's h1-eht0 and loopback l0 interfaces. Note that this interface h1-eth0 is not seen by primary Linux system when ifconfig is run, because it is specific to the network namespace of the host process.

```
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::c4aa:6dff:fe69:b978  prefixlen 64  scopeid 0x20<link>
        ether c6:aa:6d:69:b9:78  txqueuelen 1000  (Ethernet)
        RX packets 41  bytes 4370 (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  collisions 0

mininet>
```

mininet> s1 ifconfig –a: This will show the switch interfaces, plus the MV's connection out eth0.

For other examples highlighting that the hosts have isolated network state, run arp and route on both s1 and h1.

```
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::90d6:8cf8:3ad9:8238  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:25:23:01  txqueuelen 1000  (Ethernet)
        RX packets 4355  bytes 5381344 (5.3 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2712  bytes 227662 (227.6 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 2305803  bytes 66385574631 (66.3 GB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2305803  bytes 66385574631 (66.3 GB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ovs-system: flags=4098<BROADCAST,MULTICAST>  mtu 1500
        ether c6:f6:36:7e:dc:51  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 92:f5:c8:10:8f:41  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
```

- **mininet> h1 ping -c 5 h2:** if a string appears later in the command with a node name, that node name is replaced by it's IP address; that happened for 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=28.6 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.435 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.054 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.081 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.055 ms

--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4062ms
rtt min/avg/max/mdev = 0.054/5.857/28.663/11.403 ms
mininet>
```

- **Draw the network topology that is created in mininet:**

In terminal-1, command line: sudo ovs-vsctl show

```
shakhera@shakhera-HP-Notebook-PC:~$ sudo ovs-vsctl show
[sudo] password for shakhera:
c6bb5ced-c727-4a13-af98-17dc75201db1
    Bridge "s1"
        Controller "ptcp:6654"
        Controller "tcp:127.0.0.1:6653"
            is_connected: true
        fail_mode: secure
        Port "s1"
            Interface "s1"
                type: internal
        Port "s1-eth2"
            Interface "s1-eth2"
        Port "s1-eth1"
            Interface "s1-eth1"
    ovs_version: "2.9.5"
shakhera@shakhera-HP-Notebook-PC:~$
```

Finish the test using mininet>exit

```
mininet> exit
*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 854.767 seconds
shakhera@shakhera-HP-Notebook-PC:~$
```

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?

In terminal-2:

```
shakhera@shakhera-HP-Notebook-PC:~$ sudo mn --link tc,bw=10,delay=500ms
[sudo] password for shakhera:
*** 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>
```

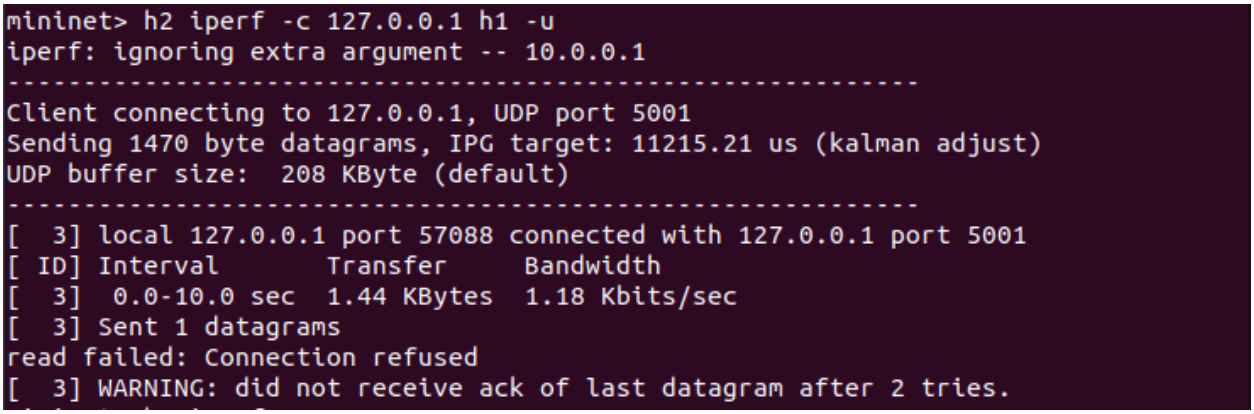
- **mininet> h1 ping -c 5 h2, What happen with the link?**

```
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=4060 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=2017 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=3062 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 4068ms
rtt min/avg/max/mdev = 2002.601/2629.097/4060.812/824.246 ms, pipe 4
mininet>
```

- **mininet> h1 iperf –s –u &**

```
mininet> h1 iperf -s -u
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
```

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



Question: Explain how the traffic generators work?

Answer:

Traffic generators are a way of injecting traffic into a network for utilization by other devices.

A traffic generator is meant to look like a device on a network, so it can target devices in receipt of traffic. This meant it will have a physical, typically higher-level address.

When implemented, a traffic generator attaches to the network via the same interfaces as other devices to establish brand-new packets. Certain traffic generators may also need to respond to traffic from other devices, for example, when they need to establish am Address Resolution Protocol (ARP) in an IP network or a TCP connection.

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

Answer:

TCP and UDP have many differences and similarities. They are the most commonly used protocols for sending packets over the internet. They both work on the transport layer of the TCP/IP protocol stack and both use the IP protocol.

TCP is a connection-oriented protocol. Connection-orientation means that the communicating devices should establish a connection before transmitting data and should close the connection after transmitting the data.

TCP is used by HTTP, HTTPs, FTP, SMTP and Telnet.

UDP is the Datagram oriented protocol. This is because there is no overhead for opening a connection, maintaining a connection, and terminating a connection. UDP is efficient for broadcast and multicast type of network transmission.
UDP is used by DNS, DHCP, TFTP, SNMP, RIP, and VoIP.

Question: In your opinions which are the main advantages and disadvantages of working with mininet? Provide at least 3.
Answer:
Advantages:

- Provides a simple and inexpensive **network testbed** for developing OpenFlow applications
- Includes a **CLI** that is topology-aware and OpenFlow-aware, for debugging or running network-wide tests
- also Provides a straightforward and extensible **Python API** for network creation and experimentation

disadvantage:

- 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.

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

There are some advantages of programmable Controller are given below,

- It has very faster scan time.
- It has capable to communication with computer in plant.
- It has great computational capabilities.
- It has shorter training time required.
- A wide range of control application.
- It Have interfacing for inputs and outputs already inside the controller.
- It is easily programmed and has an easily understood programming language.

- It has flexibility in programming and reprogramming.
- It has shorter project implementation time.
- It has reliability in operation

Discussion: From this lab, I have to know that how to Install and use traffic generators as powerful tools for testing network performance, and how to configure SDN Controller. To perform this lab I have to understand how the mininet simulator works, and how implement and run basic examples for understanding the role of the controller and how it interact with mininet.

In conclusion, it will suffice to say that TCP and UDP, two popular transport layer protocols, cannot be done without in any way. The selection between TCP vs UDP can be done via application developers and in line with user application connection requirements. From the above exercise we have learned the mechanism to sent some messages using UDP protocol.

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.