# Networking Module
# Assignment 1 Report

K K Gayatri Priyadarsini 21310015

14th November 2021

## 1 Introduction

The assignment consists the following exercises:

- Experimenting with IP, MAC addresses

- Concurrent Servers: Fork vs threads

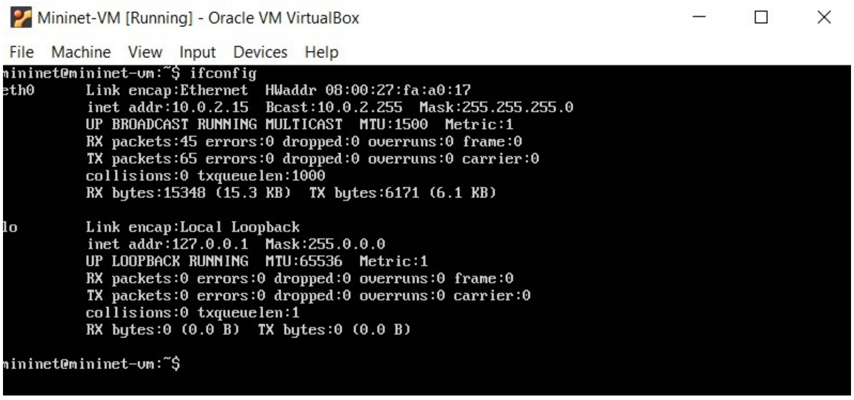
- Experimenting with Mininet

## 2 Tool Setup

In this lab, we use Mininet , a network simulator to compare a SDN i.e., Software Defined Network with a Hardware Defined Network. A virtual machine image for the same was uploaded in VirtualBox environment. Another alternative for using mininet is to install it on the host machine. For the 5th part of the assignment, Linux machines (Kali OS) was used.

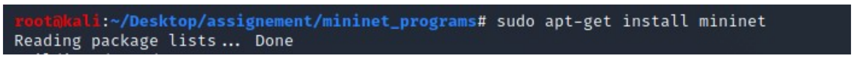
Once the Mininet-VM was up and running, the following commands were executed:

- 'ifconfig' to identify the interface address, IP address and subnet details



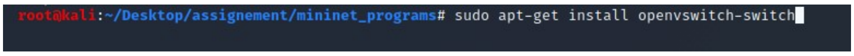
```
ininet@mininet-vm:~$ ifconfig
th0       Link encap:Ethernet  HWaddr 08:00:27:fa:a0:17
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:45 errors:0 dropped:0 overruns:0 frame:0
          TX packets:65 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:15348 (15.3 KB)  TX bytes:6171 (6.1 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

ininet@mininet-vm:~$
```

Figure 1: Mininet-VM : ifconfig



```
root@kali:~/Desktop/assignement/mininet_programs# sudo apt-get install mininet
Reading package lists ... Done
```

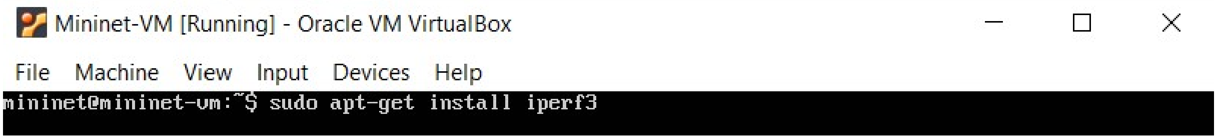
Figure 2: To install mininet on Linux machine: ’apt-get install mininet’



```
root@kali:~/Desktop/assignement/mininet_programs# sudo apt-get install openvswitch-switch
```

Figure 3: To install Open VSwitch

- ’iperf3’ , a cross-platform network performance benchmark tool, was downloaded to measure the download time and bandwidth between hosts in mininet

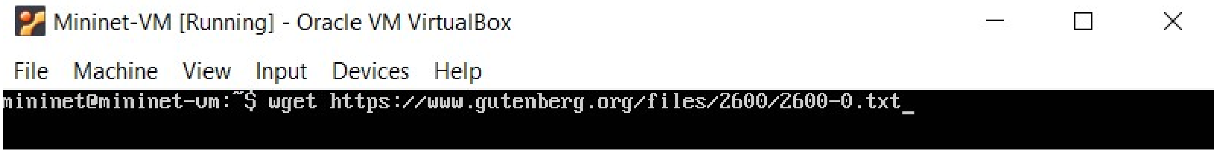


Mininet-VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

```
mininet@mininet-vm:~$ sudo apt-get install iperf3
```

Figure 4: iperf3 in Mininet

- Download a huge file to be exchanged between hosts to while measuring the performance.



Mininet-VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

```
mininet@mininet-vm:~$ wget https://www.gutenberg.org/files/2600/2600-0.txt_
```

Figure 5: 2600-0.txt - File for transmission

Another method to create a large file is to use the following commands:

- ”dd if=/dev/zero of=1GBFile.txt bs=1M count=1024”

- ”fallocate -l 1G 1GBFile.txt”

- ”truncate -s 1G 1GBFile.txt”

# 3 Experimenting with IP, MAC addresses

For this part of the assignment, Parrot OS was used (IP address: 10.0.2.15)

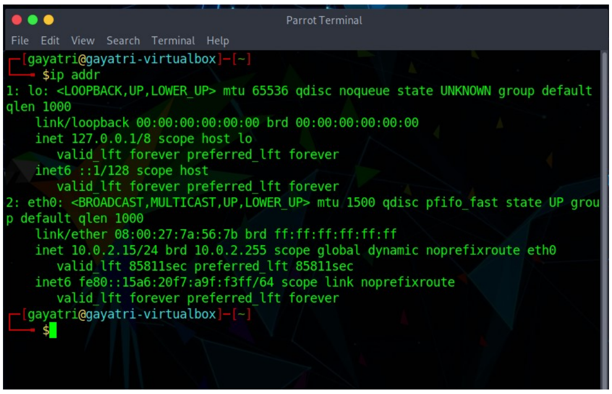
## 3.1 Secondary IP

The objective of this question was to add a secondary IP address to a chosen interface. Figure 6 shows the current ip addr output.

A secondary IP can be added to the interface temporarily using the command ip add (Figure 7)

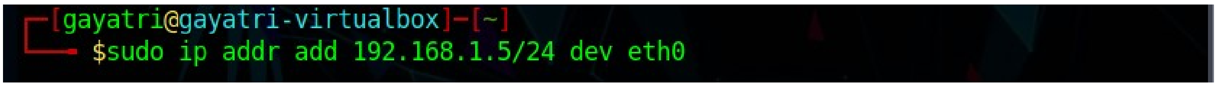
After adding the ip address, it can be seen in the output of ip addr. (Figure 8)

One can add an IP address from the GUI option as well. In Parrot OS, Advanced Network Settings - IPv4 has the option to add IP address to the selected interface permanently.



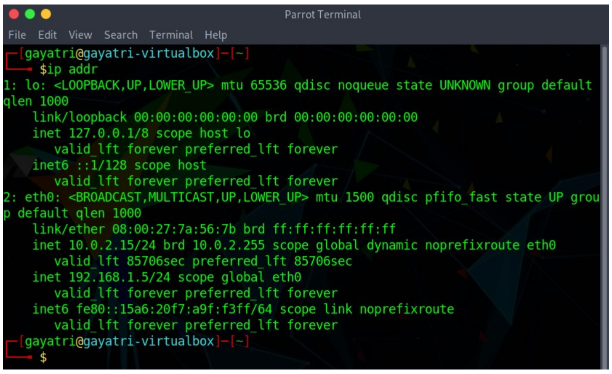
```
┌─[gayatri@gayatri-virtualbox]─[~]
└──╼ $ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP grou
p default qlen 1000
    link/ether 08:00:27:7a:56:7b brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute eth0
       valid_lft 85811sec preferred_lft 85811sec
    inet6 fe80::15a6:20f7:a9f:f3ff/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
┌─[gayatri@gayatri-virtualbox]─[~]
└──╼ $
```

Figure 6: ip addr



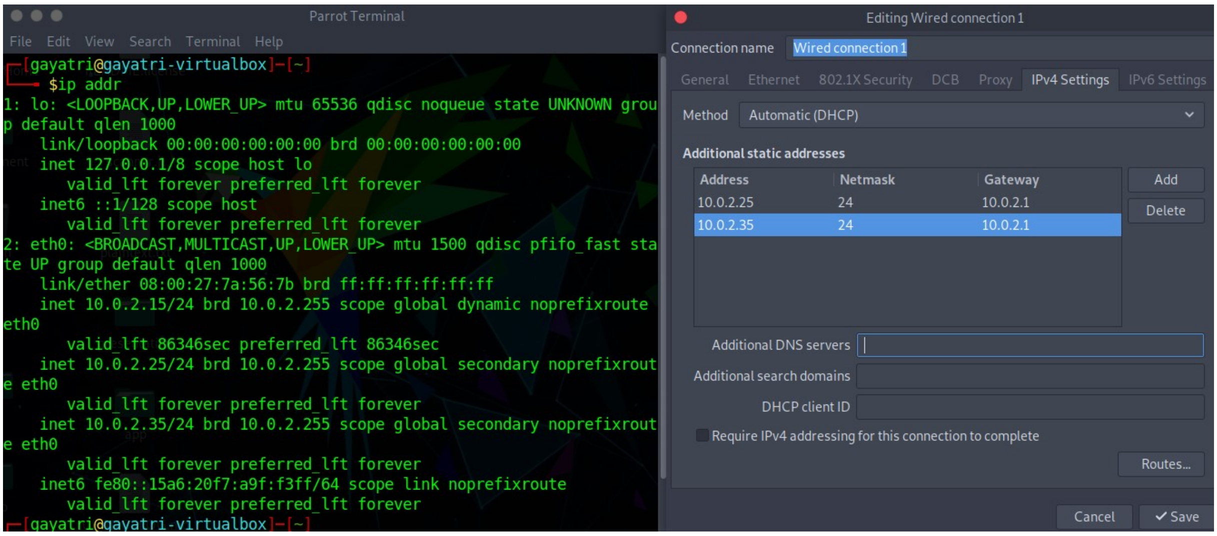
```
┌─[gayatri@gayatri-virtualbox]─[~]
└──╼ $sudo ip addr add 192.168.1.5/24 dev eth0
```

Figure 7: Adding a secondary IP address



```
┌─[gayatri@gayatri-virtualbox]─[~]
└──╼ $ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP grou
p default qlen 1000
    link/ether 08:00:27:7a:56:7b brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute eth0
       valid_lft 85706sec preferred_lft 85706sec
    inet 192.168.1.5/24 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::15a6:20f7:a9f:f3ff/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
┌─[gayatri@gayatri-virtualbox]─[~]
└──╼ $
```

Figure 8: Temporary secondary IP added



```
┌─[gayatri@gayatri-virtualbox]─[~]
└──╼ $ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN grou
p default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast sta
te UP group default qlen 1000
    link/ether 08:00:27:7a:56:7b brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute
eth0
       valid_lft 86346sec preferred_lft 86346sec
    inet 10.0.2.25/24 brd 10.0.2.255 scope global secondary noprefixrout
e eth0
       valid_lft forever preferred_lft forever
    inet 10.0.2.35/24 brd 10.0.2.255 scope global secondary noprefixrout
e eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::15a6:20f7:a9f:f3ff/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
┌─[gayatri@gayatri-virtualbox]─[~]
```

Editing Wired connection 1

Connection name: Wired connection 1

Method: Automatic (DHCP)

Additional static addresses

| Address | Netmask | Gateway |
|---|---|---|
| 10.0.2.25 | 24 | 10.0.2.1 |
| 10.0.2.35 | 24 | 10.0.2.1 |

Additional DNS servers:

Additional search domains:

DHCP client ID:

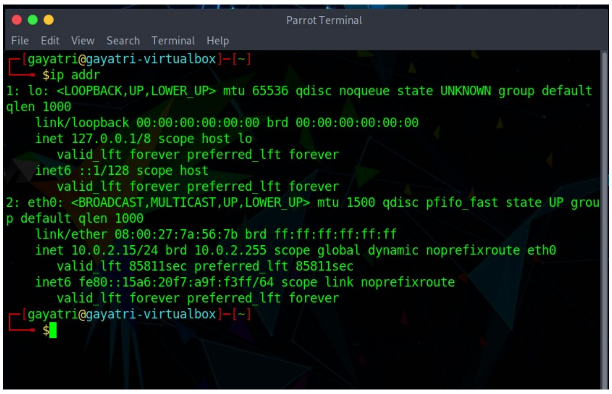
□ Require IPv4 addressing for this connection to complete

Figure 9: Advanced Network settings to add IPs

## 3.2 IP Aliasing

IP aliasing can be used to provide multiple network addresses on a single physical interface. This demonstrates using IP version 4 addresses only. One reason for using this could be to make a computer look as though it is multiple computers.

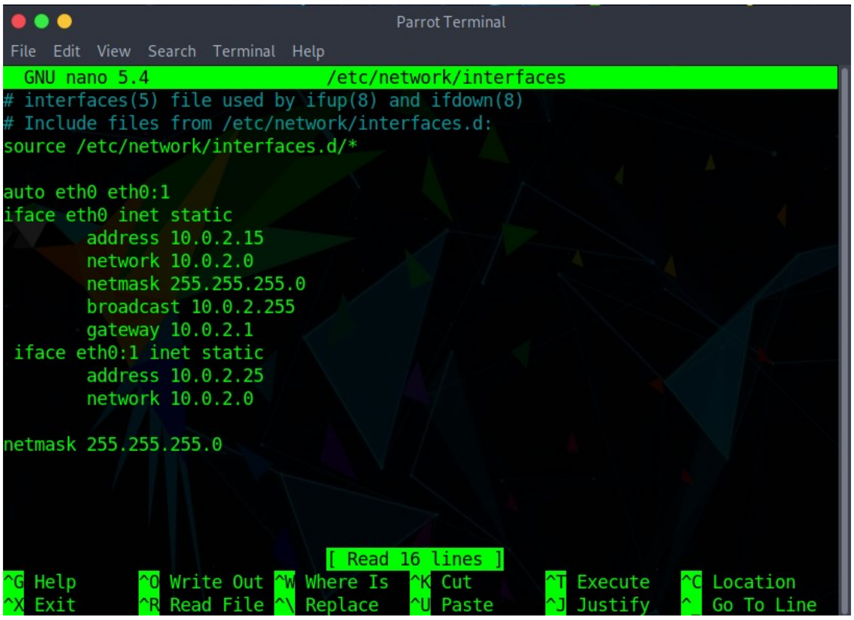
Initially there is only one default interface Ethernet (eth0). These device network files are located in /etc/network/interfaces.



```
┌─[gayatri@gayatri-virtualbox]─[~]
└──╼ $ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP grou
p default qlen 1000
    link/ether 08:00:27:7a:56:7b brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute eth0
       valid_lft 85811sec preferred_lft 85811sec
    inet6 fe80::15a6:20f7:a9f:f3ff/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
┌─[gayatri@gayatri-virtualbox]─[~]
└──╼ $
```

Figure 10: ip addr

Objective of this question was to add an interface to the existing Ethernet device. If we want to create an additional virtual interface to bind a new IP address to the NIC (one mac address), we need to create an additional alias file. This can be done by changing the /etc/network/interfaces file (Parrot OS).

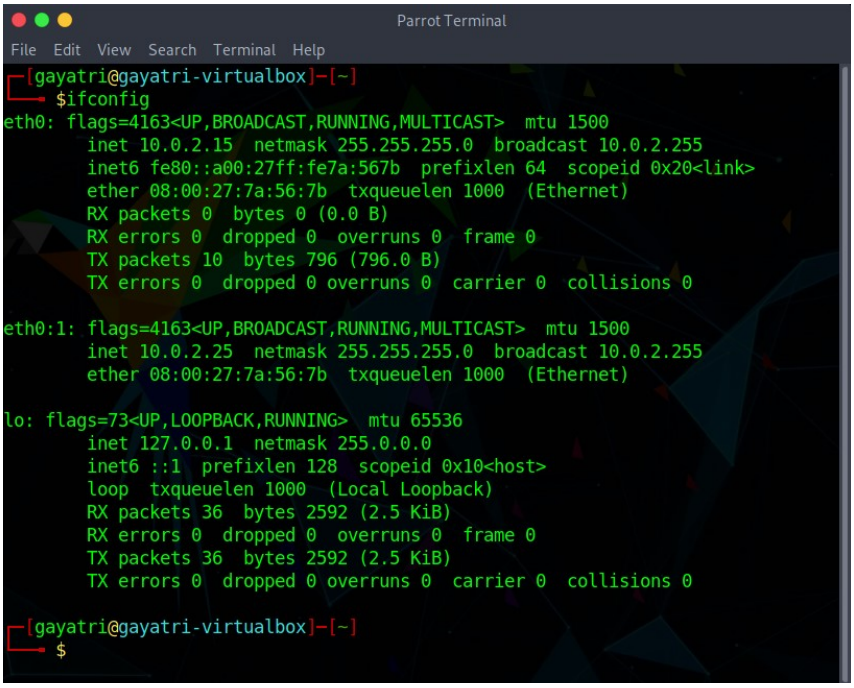


```
  GNU nano 5.4                  /etc/network/interfaces
# interfaces(5) file used by ifup(8) and ifdown(8)
# Include files from /etc/network/interfaces.d:
source /etc/network/interfaces.d/*

auto eth0 eth0:1
iface eth0 inet static
        address 10.0.2.15
        network 10.0.2.0
        netmask 255.255.255.0
        broadcast 10.0.2.255
        gateway 10.0.2.1
 iface eth0:1 inet static
        address 10.0.2.25
        network 10.0.2.0

netmask 255.255.255.0




                              [ Read 16 lines ]
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

Figure 11: Changes made to /etc/network/interfaces

Here, the ":X" is the device (interface) number to create the alias for interface eth0.

The following changes appear after adding the interface



```
┌─[gayatri@gayatri-virtualbox]─[~]
└──╼ $ifconfig
eth0: 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::a00:27ff:fe7a:567b  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:7a:56:7b  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 10  bytes 796 (796.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth0:1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.25  netmask 255.255.255.0  broadcast 10.0.2.255
        ether 08:00:27:7a:56:7b  txqueuelen 1000  (Ethernet)

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 36  bytes 2592 (2.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 36  bytes 2592 (2.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

┌─[gayatri@gayatri-virtualbox]─[~]
└──╼ $
```

Figure 12: Adding interface - ethO:1

# 4 Concurrent Servers: Fork vs Threads(TCP)

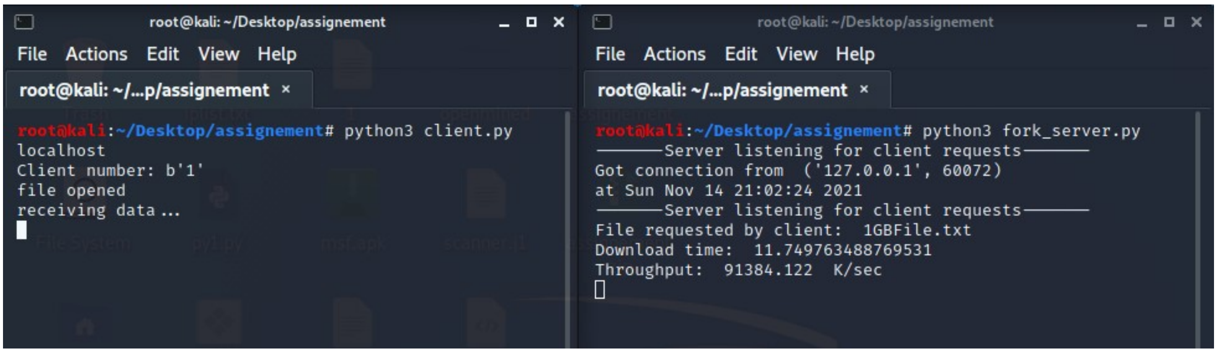
For this part of the assignment, Kali OS was used (IP address: 10.0.2.15)

## 4.1 Server Models

Objective of this exercise is to write TCP server and client programs.

### 4.1.1 Fork Model

To handle a client request, this server code uses a fork() system call, which returns 0 if it was successful in creating this child process. Once a child process is created with a new PID, the parent process return to listen for new client requests.


root@kali: ~/Desktop/assignement
File Actions Edit View Help
root@kali: ~/...p/assignement
root@kali: ~/Desktop/assignement
File Actions Edit View Help
root@kali: ~/...p/assignement

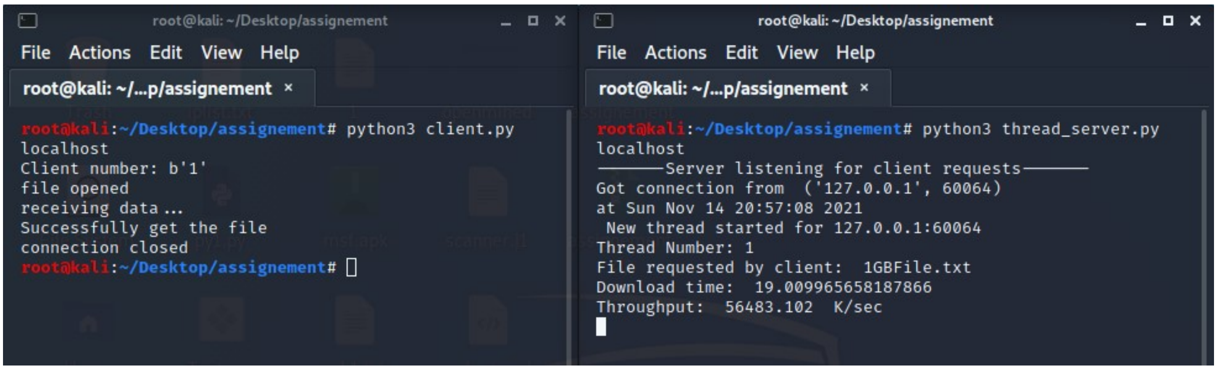
```
root@kali:~/Desktop/assignement# python3 client.py
localhost
Client number: b'1'
file opened
receiving data ...
```

```
root@kali:~/Desktop/assignement# python3 fork_server.py
———————Server listening for client requests———————
Got connection from  ('127.0.0.1', 60072)
at Sun Nov 14 21:02:24 2021
———————Server listening for client requests———————
File requested by client:  1GBFile.txt
Download time:  11.749763488769531
Throughput:  91384.122  K/sec
```

Figure 13: Fork Server and 1 client request

### 4.1.2 Threads Model

In this server code, the server assigns a thread to each client request. All the threads are stored in a list and then concurrency is handled by join().


root@kali: ~/Desktop/assignement
File Actions Edit View Help
root@kali: ~/...p/assignement

```
root@kali:~/Desktop/assignement# python3 client.py
localhost
Client number: b'1'
file opened
receiving data ...
Successfully get the file
connection closed
root@kali:~/Desktop/assignement# 
```

```
root@kali:~/Desktop/assignement# python3 thread_server.py
localhost
———————Server listening for client requests———————
Got connection from  ('127.0.0.1', 60064)
at Sun Nov 14 20:57:08 2021
 New thread started for 127.0.0.1:60064
Thread Number: 1
File requested by client:  1GBFile.txt
Download time:  19.009965658187866
Throughput:  56483.102  K/sec
```

Figure 14: Threads Server and 1 clients request

Server and Client code is accessible at:

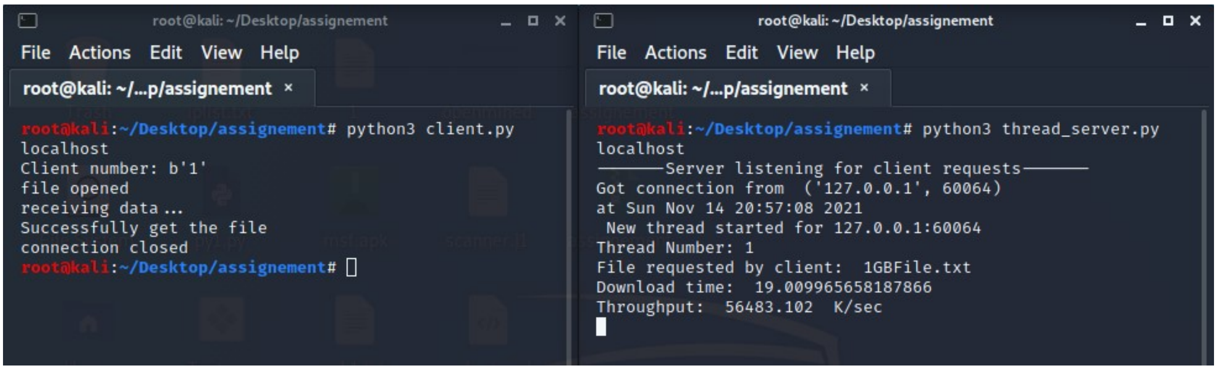
https://github.com/Gayatri-Priyadarsini/CS612-Networking_Asisgnment1/blob/main/thread_server.py

https://github.com/Gayatri-Priyadarsini/CS612-Networking_Asisgnment1/blob/main/fork_server.py

https://github.com/Gayatri-Priyadarsini/CS612-Networking_Asisgnment1/blob/main/client.py

## 4.2 Use-Case : File Download

### 4.2.1 Single Connection


root@kali: ~/Desktop/assignement
File Actions Edit View Help
root@kali: ~/...p/assignement

```
root@kali:~/Desktop/assignement# python3 client.py
localhost
Client number: b'1'
file opened
receiving data ...
Successfully get the file
connection closed
root@kali:~/Desktop/assignement# 
```

```
root@kali:~/Desktop/assignement# python3 thread_server.py
localhost
———————Server listening for client requests———————
Got connection from  ('127.0.0.1', 60064)
at Sun Nov 14 20:57:08 2021
 New thread started for 127.0.0.1:60064
Thread Number: 1
File requested by client:  1GBFile.txt
Download time:  19.009965658187866
Throughput:  56483.102  K/sec
```

Figure 15: Threads Server and 1 clients request

In this case, only a single client request is managed by the server. The client requests for a 1GB file (Created in section 2). The download time is calculated by the server using the time() function, before starting the request and after the request is completed.

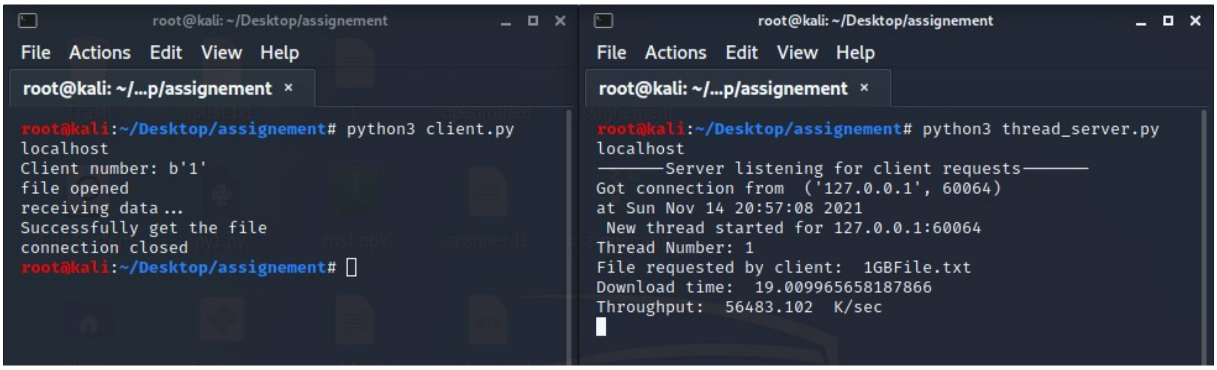
The download time can further be used to calculate the throughput. Throughput is the time taken to transfer the file. As the data is exchanged in 1024 bytes (buffer_size) at a time, the filesize can be calculated by counting the number if times the transfer is made, times the buffer_size.

Throughput = (Filesize*BUFFER_SIZE * 0.001) / download_time K/s

In this case (Figure 15), the download time was: 19 sec and throughput is 56483 K/s Both thread and fork servers took the same amount of time in the case of single request.

### 4.2.2 Multiple Connections

Multiple requests are handled concurrently in both the servers.In case of multi-threading:


root@kali: ~/Desktop/assignement
File Actions Edit View Help
root@kali: ~/...p/assignement ×
root@kali: ~/Desktop/assignement
File Actions Edit View Help
root@kali: ~/...p/assignement ×

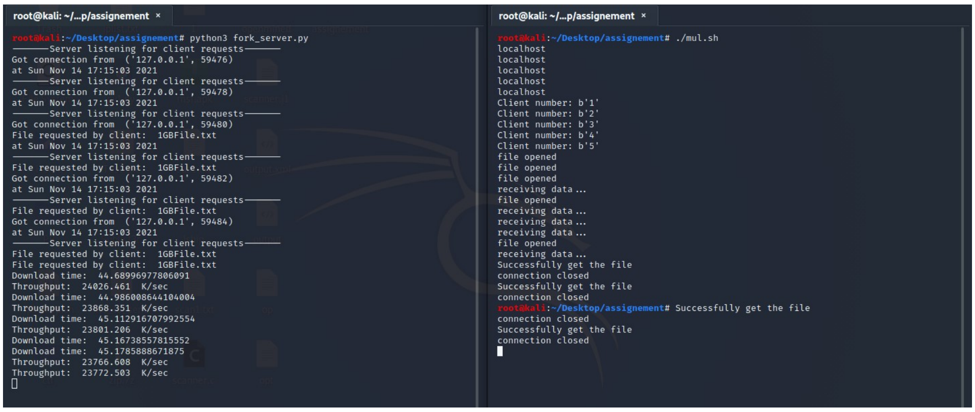
```
root@kali:~/Desktop/assignement# python3 client.py
localhost
Client number: b'1'
file opened
receiving data ...
Successfully get the file
connection closed
root@kali:~/Desktop/assignement# 
```

```
root@kali:~/Desktop/assignement# python3 thread_server.py
localhost
———Server listening for client requests————
Got connection from  ('127.0.0.1', 60064)
at Sun Nov 14 20:57:08 2021
 New thread started for 127.0.0.1:60064
Thread Number: 1
File requested by client:  1GBFile.txt
Download time:  19.009965658187866
Throughput:  56483.102  K/sec
```

Figure 16: Threads Server and 5 parallel clients requests

Download time: 43.49+44.39+44.73+44.74+44.78= 222.13s

Aggregate Throughput : (24686.909+24183.954+24004.306+23998.558+23975.747 )/5 = 120849.474/5= 24169.8948 K/s


root@kali: ~/...p/assignement ×
root@kali: ~/...p/assignement ×

```
root@kali:~/Desktop/assignement# python3 fork_server.py
———Server listening for client requests———
Got connection from  ('127.0.0.1', 59476)
at Sun Nov 14 17:15:03 2021
———Server listening for client requests———
Got connection from  ('127.0.0.1', 59478)
at Sun Nov 14 17:15:03 2021
———Server listening for client requests———
Got connection from  ('127.0.0.1', 59480)
File requested by client:  1GBFile.txt
at Sun Nov 14 17:15:03 2021
———Server listening for client requests———
File requested by client:  1GBFile.txt
Got connection from  ('127.0.0.1', 59482)
at Sun Nov 14 17:15:03 2021
———Server listening for client requests———
File requested by client:  1GBFile.txt
Got connection from  ('127.0.0.1', 59484)
at Sun Nov 14 17:15:03 2021
———Server listening for client requests———
File requested by client:  1GBFile.txt
File requested by client:  1GBFile.txt
Download time:  44.68996977806091
Throughput:  24026.461  K/sec
Download time:  44.986008644104004
Throughput:  23868.351  K/sec
Download time:  45.112916707992554
Throughput:  23801.206  K/sec
Download time:  45.16738557815552
Download time:  45.1785888671875
Throughput:  23766.608  K/sec
Throughput:  23772.503  K/sec
```

```
root@kali:~/Desktop/assignement# ./mul.sh
localhost
localhost
localhost
localhost
localhost
Client number: b'1'
Client number: b'2'
Client number: b'3'
Client number: b'4'
Client number: b'5'
file opened
file opened
file opened
receiving data ...
file opened
receiving data ...
receiving data ...
receiving data ...
file opened
receiving data ...
Successfully get the file
connection closed
Successfully get the file
connection closed
root@kali:~/Desktop/assignement# Successfully get the file
connection closed
Successfully get the file
connection closed
```

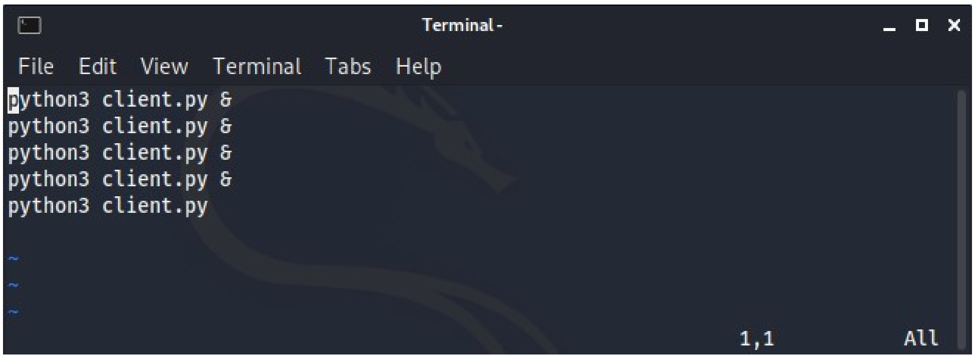
Figure 17: Fork Server and 5 parallel clients requests

Download time: 44.689+44.989+45.122+45.167+45.178= 225.132 s

Aggregate Throughput : 119235.247/5 =23847.0548 K/s

The concurrent requests in both cases is sent by running 5 concurrent client.py codes. Each request is given a thread number or a child number wrt to the type of server and this number is then used as the output filename generated on downloading the file at the client side.

Concurrent requests were sent using the following shell script:

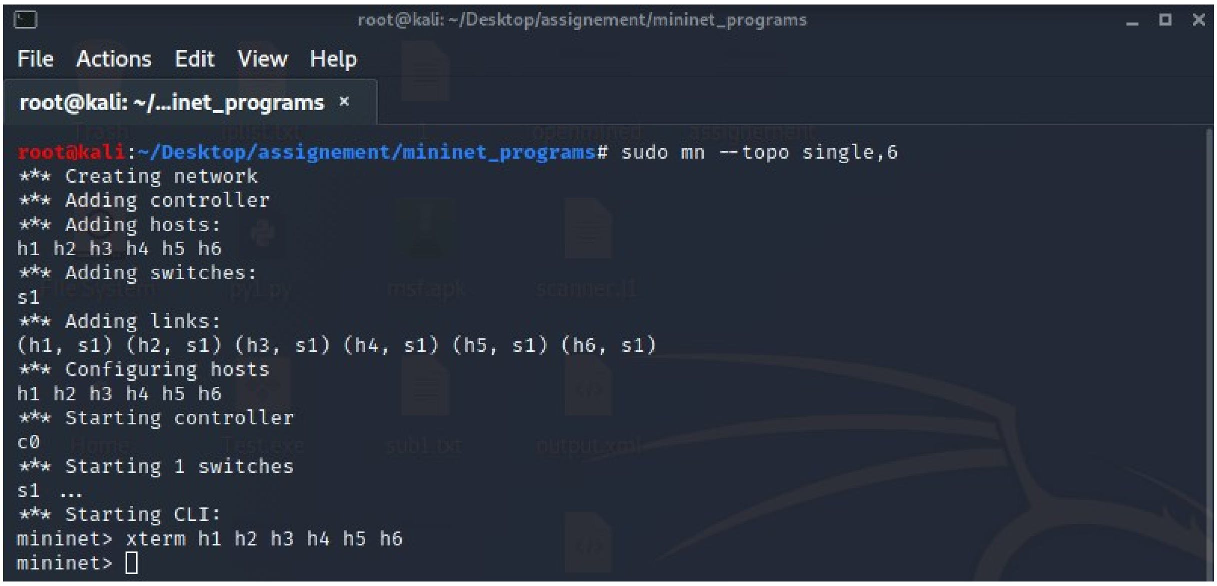


```
python3 client.py &
python3 client.py &
python3 client.py &
python3 client.py &
python3 client.py

~
~
~
                                                    1,1           All
```

Figure 18: 5 clients run concurrently

# 5 Experimenting with Mininet

This part of the assignment is done by installing Mininet in the Kali OS with 10.0.2.15 as the IP address. Once Mininet and Open VSwitch is installed, topologies can be created using the command:



root@kali: ~/Desktop/assignement/mininet_programs
File Actions Edit View Help
root@kali: ~/...inet_programs

```
root@kali:~/Desktop/assignement/mininet_programs# sudo mn --topo single,6
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1) (h6, s1)
*** Configuring hosts
h1 h2 h3 h4 h5 h6
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> xterm h1 h2 h3 h4 h5 h6
mininet> 
```

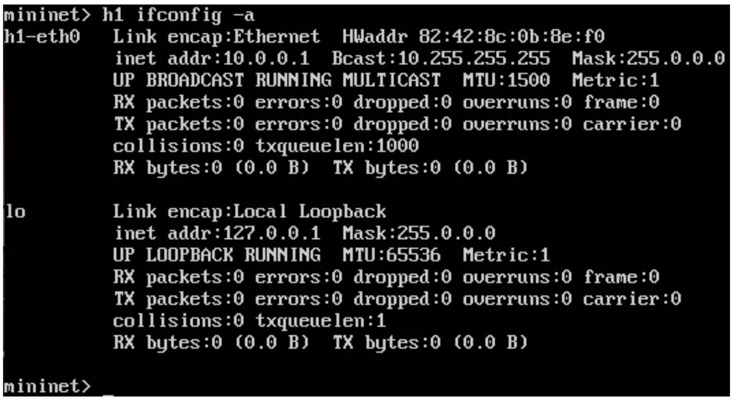
Figure 19: Topology with 6 hosts and 1 switch

## 5.1 Running commands on individual hosts

At the mininet terminal, commands at particular hosts can be run by stating the command with it's name. For example:

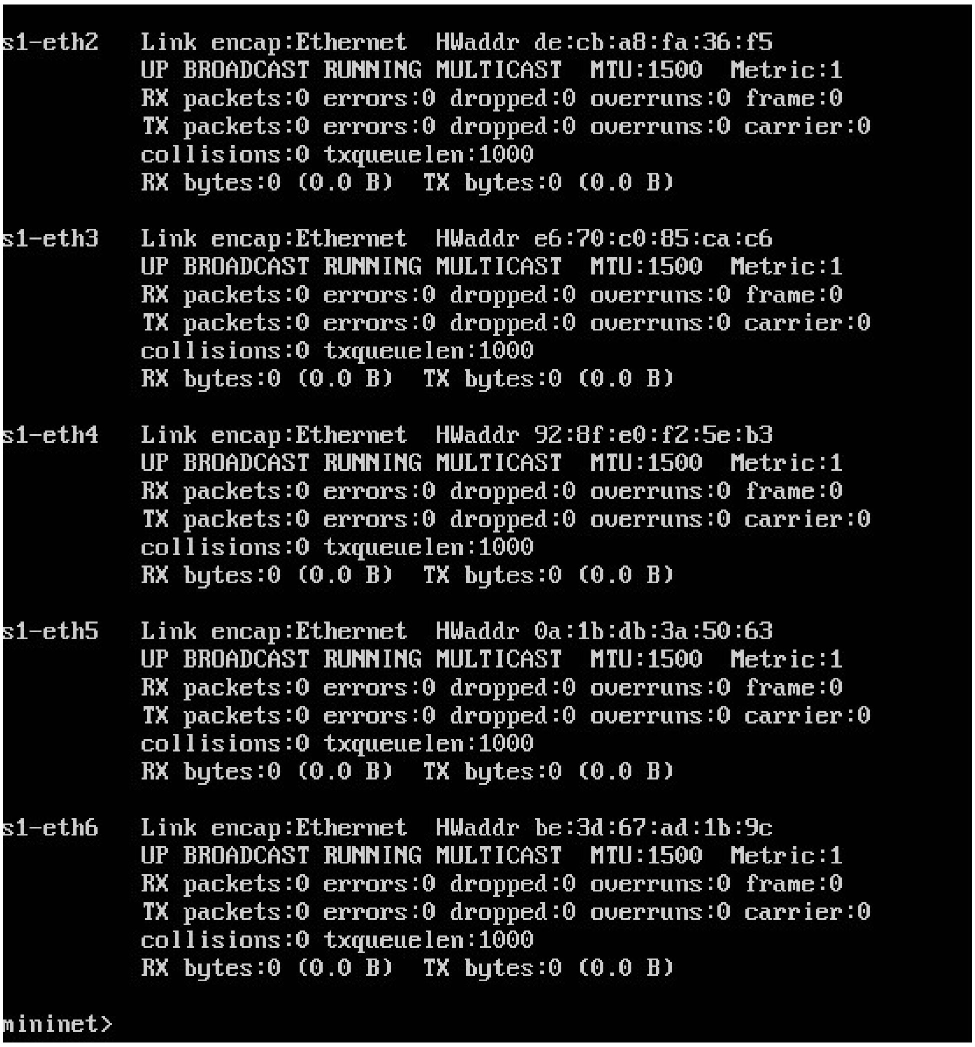
The switch has 6 interfaces, one for each host.

To run individual terminals for each host: xterm h1 h2 h3 h4 h5 h6 command is used.



```
mininet> h1 ifconfig -a
h1-eth0   Link encap:Ethernet  HWaddr 82:42:8c:0b:8e:f0
          inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.0.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

mininet> _
```

Figure 20: ifconfig at h1

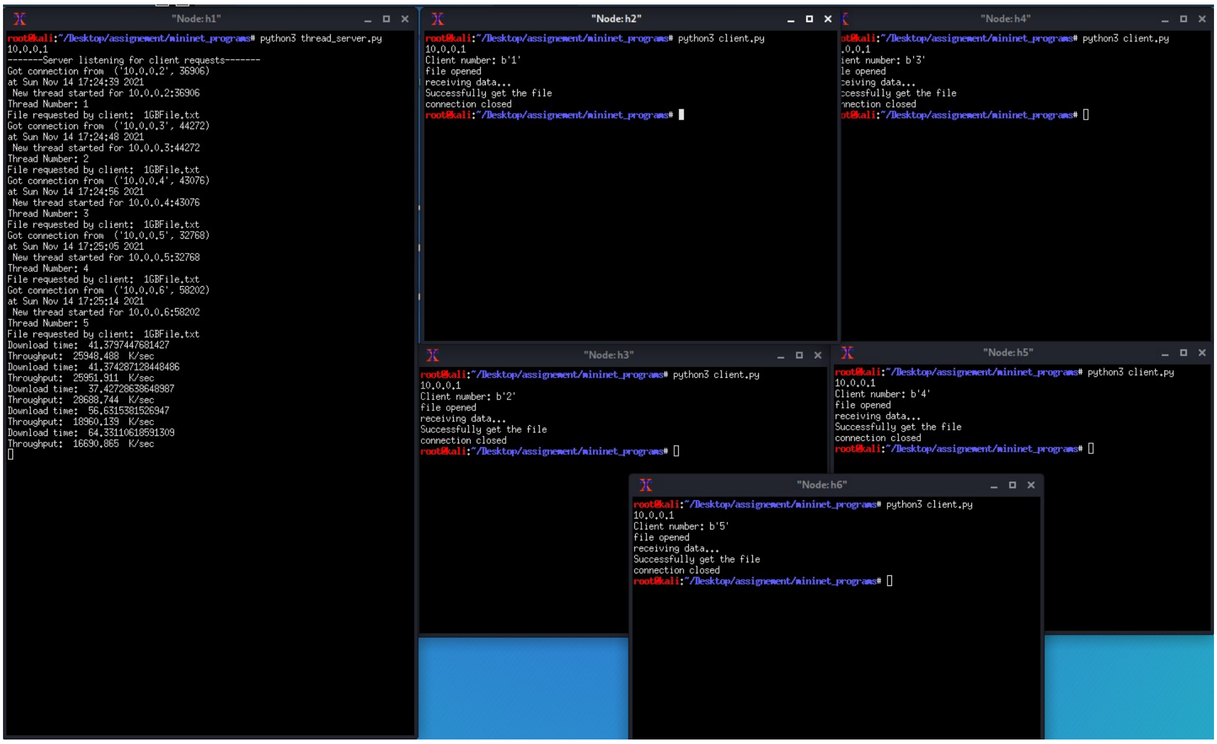


```
s1-eth2   Link encap:Ethernet  HWaddr de:cb:a8:fa:36:f5
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

s1-eth3   Link encap:Ethernet  HWaddr e6:70:c0:85:ca:c6
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

s1-eth4   Link encap:Ethernet  HWaddr 92:8f:e0:f2:5e:b3
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

s1-eth5   Link encap:Ethernet  HWaddr 0a:1b:db:3a:50:63
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

s1-eth6   Link encap:Ethernet  HWaddr be:3d:67:ad:1b:9c
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

mininet>
```

Figure 21: ifconfig at s1

## 5.2 Use Case: 1 server and 5 clients

h1 is made the server by running the thread_server.py and all other hosts make a request for downloading the 1GB file to the server h1.



```
root@kali:~/Desktop/assignement/mininet_programs# python3 thread_server.py
10.0.0.1
-------Server listening for client requests-------
Got connection from  ('10.0.0.2', 36906)
at Sun Nov 14 17:24:39 2021
 New thread started for 10.0.0.2:36906
Thread Number: 1
File requested by client:  1GBFile.txt
Got connection from  ('10.0.0.3', 44272)
at Sun Nov 14 17:24:48 2021
 New thread started for 10.0.0.3:44272
Thread Number: 2
File requested by client:  1GBFile.txt
Got connection from  ('10.0.0.4', 43076)
at Sun Nov 14 17:24:56 2021
 New thread started for 10.0.0.4:43076
Thread Number: 3
File requested by client:  1GBFile.txt
Got connection from  ('10.0.0.5', 32768)
at Sun Nov 14 17:25:05 2021
 New thread started for 10.0.0.5:32768
Thread Number: 4
File requested by client:  1GBFile.txt
Got connection from  ('10.0.0.6', 58202)
at Sun Nov 14 17:25:14 2021
 New thread started for 10.0.0.6:58202
Thread Number: 5
File requested by client:  1GBFile.txt
Download time:  41.3797447681427
Throughput:  25948.488  K/sec
Download time:  41.374287128448486
Throughput:  25951.911  K/sec
Download time:  37.42728638648987
Throughput:  28688.744  K/sec
Download time:  56.6315381526947
Throughput:  18960.139  K/sec
Download time:  64.33110618591309
Throughput:  16690.865  K/sec
```

```
root@kali:~/Desktop/assignement/mininet_programs# python3 client.py
10.0.0.1
Client number: b'1'
file opened
receiving data...
Successfully get the file
connection closed
root@kali:~/Desktop/assignement/mininet_programs#
```

```
ot@kali:~/Desktop/assignement/mininet_programs# python3 client.py
.0.0.1
ient number: b'3'
le opened
ceiving data...
ccessfully get the file
nnection closed
ot@kali:~/Desktop/assignement/mininet_programs#
```

```
root@kali:~/Desktop/assignement/mininet_programs# python3 client.py
10.0.0.1
Client number: b'2'
file opened
receiving data...
Successfully get the file
connection closed
root@kali:~/Desktop/assignement/mininet_programs#
```

```
root@kali:~/Desktop/assignement/mininet_programs# python3 client.py
10.0.0.1
Client number: b'4'
file opened
receiving data...
Successfully get the file
connection closed
root@kali:~/Desktop/assignement/mininet_programs#
```

```
root@kali:~/Desktop/assignement/mininet_programs# python3 client.py
10.0.0.1
Client number: b'5'
file opened
receiving data...
Successfully get the file
connection closed
root@kali:~/Desktop/assignement/mininet_programs#
```

Figure 22: xterm for each host

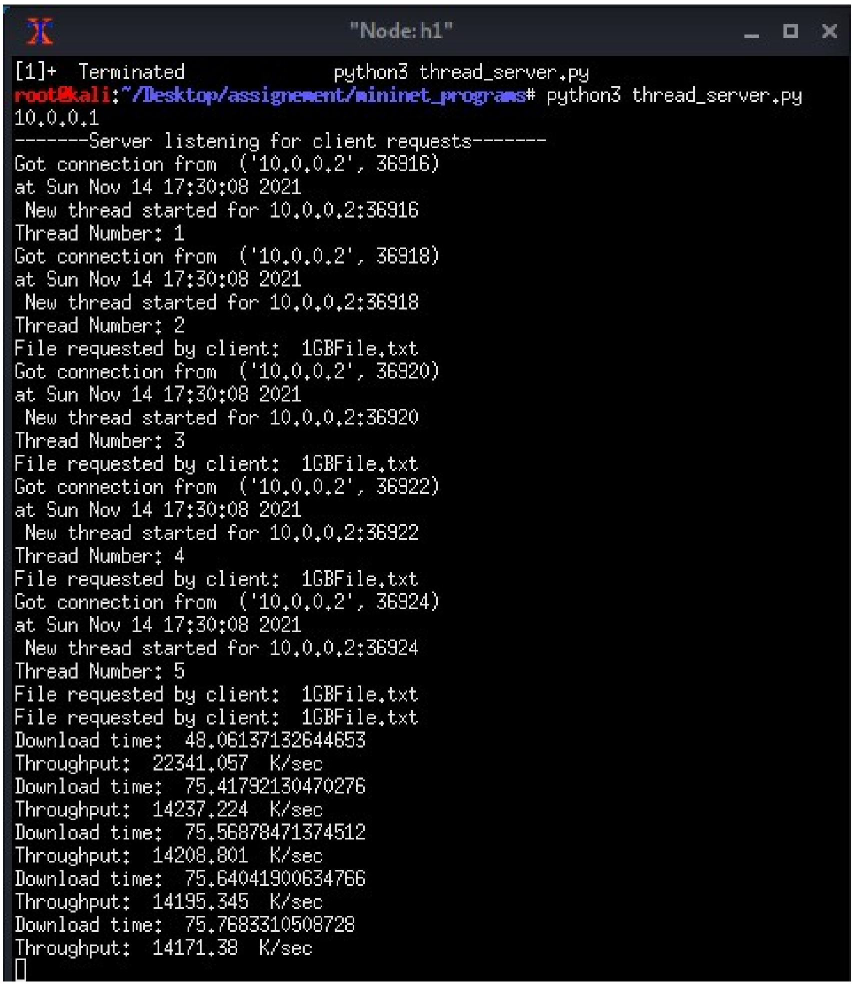
Download time: 241.134 s

Throughput:116240.147/5 = 23248.0294 K/s

If the requests are run parallely from the same host (figure 23):

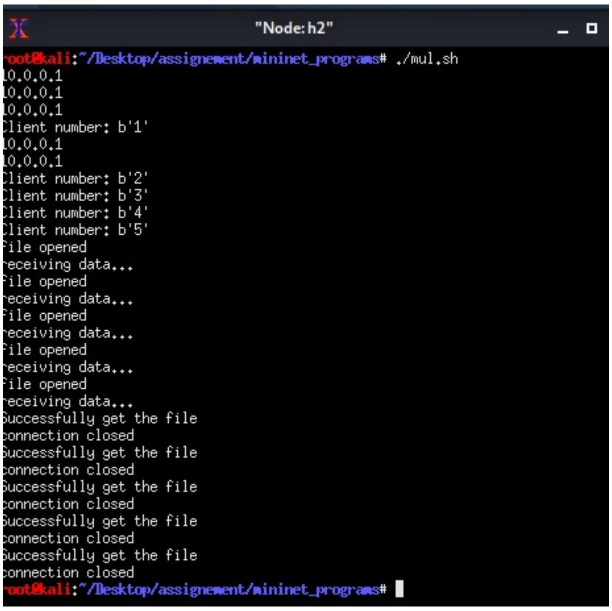
Download time: 350.451 s

Throughput:79152.402/5 = 15830.4804 K/s



```
[1]+  Terminated              python3 thread_server.py
root@kali:~/Desktop/assignement/mininet_programs# python3 thread_server.py
10.0.0.1
-------Server listening for client requests-------
Got connection from  ('10.0.0.2', 36916)
at Sun Nov 14 17:30:08 2021
 New thread started for 10.0.0.2:36916
Thread Number: 1
Got connection from  ('10.0.0.2', 36918)
at Sun Nov 14 17:30:08 2021
 New thread started for 10.0.0.2:36918
Thread Number: 2
File requested by client:  1GBFile.txt
Got connection from  ('10.0.0.2', 36920)
at Sun Nov 14 17:30:08 2021
 New thread started for 10.0.0.2:36920
Thread Number: 3
File requested by client:  1GBFile.txt
Got connection from  ('10.0.0.2', 36922)
at Sun Nov 14 17:30:08 2021
 New thread started for 10.0.0.2:36922
Thread Number: 4
File requested by client:  1GBFile.txt
Got connection from  ('10.0.0.2', 36924)
at Sun Nov 14 17:30:08 2021
 New thread started for 10.0.0.2:36924
Thread Number: 5
File requested by client:  1GBFile.txt
File requested by client:  1GBFile.txt
Download time:  48.06137132644653
Throughput:  22341.057  K/sec
Download time:  75.41792130470276
Throughput:  14237.224  K/sec
Download time:  75.56878471374512
Throughput:  14208.801  K/sec
Download time:  75.64041900634766
Throughput:  14195.345  K/sec
Download time:  75.7683310508728
Throughput:  14171.38  K/sec
```

Figure 23: 5 concurrent requests from h1

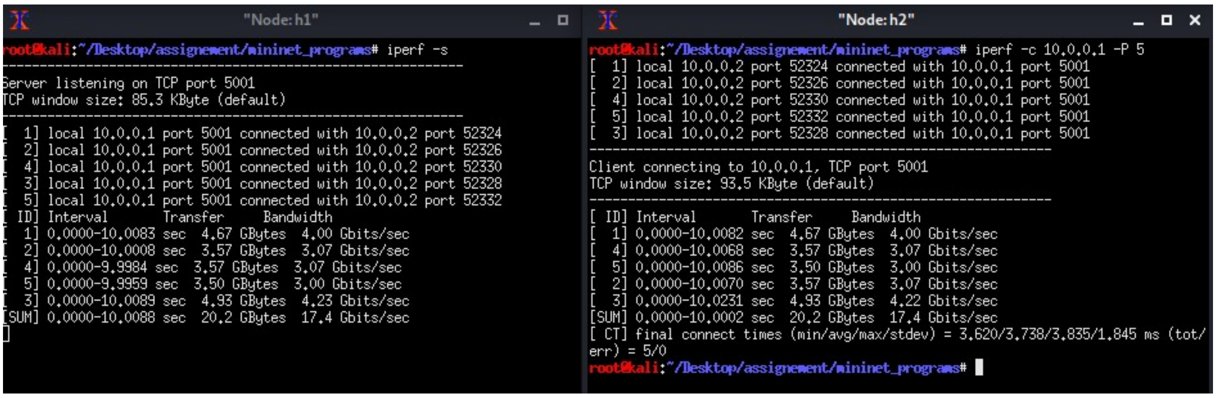


```
oot@kali:~/Desktop/assignement/mininet_programs# ./mul.sh
10.0.0.1
10.0.0.1
10.0.0.1
lient number: b'1'
10.0.0.1
10.0.0.1
lient number: b'2'
lient number: b'3'
lient number: b'4'
lient number: b'5'
ile opened
eceiving data...
ile opened
eceiving data...
ile opened
eceiving data...
ile opened
eceiving data...
ile opened
eceiving data...
uccessfully get the file
onnection closed
uccessfully get the file
onnection closed
uccessfully get the file
onnection closed
uccessfully get the file
onnection closed
uccessfully get the file
onnection closed
oot@kali:~/Desktop/assignement/mininet_programs#
```

Figure 24: xterm for each host

### 5.3 Iperf3

Iperf3 -s is used to make a host (at h1) as server and iperf3 -c is used at the client side at h2 and TCP packets are transfered from the server,h1 to h2. -P option makes 5 parallel connection



```
oot@kali:~/Desktop/assignement/mininet_programs# iperf -s
------------------------------------------------------------
erver listening on TCP port 5001
CP window size: 85.3 KByte (default)
------------------------------------------------------------
[  1] local 10.0.0.1 port 5001 connected with 10.0.0.2 port 52324
[  2] local 10.0.0.1 port 5001 connected with 10.0.0.2 port 52326
[  4] local 10.0.0.1 port 5001 connected with 10.0.0.2 port 52330
[  3] local 10.0.0.1 port 5001 connected with 10.0.0.2 port 52328
[  5] local 10.0.0.1 port 5001 connected with 10.0.0.2 port 52332
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-10.0083 sec  4.67 GBytes  4.00 Gbits/sec
[  2] 0.0000-10.0008 sec  3.57 GBytes  3.07 Gbits/sec
[  4] 0.0000-9.9984 sec  3.57 GBytes  3.07 Gbits/sec
[  5] 0.0000-9.9959 sec  3.50 GBytes  3.00 Gbits/sec
[  3] 0.0000-10.0089 sec  4.93 GBytes  4.23 Gbits/sec
[SUM] 0.0000-10.0088 sec  20.2 GBytes  17.4 Gbits/sec
```

```
root@kali:~/Desktop/assignement/mininet_programs# iperf -c 10.0.0.1 -P 5
[  1] local 10.0.0.2 port 52324 connected with 10.0.0.1 port 5001
[  2] local 10.0.0.2 port 52326 connected with 10.0.0.1 port 5001
[  4] local 10.0.0.2 port 52330 connected with 10.0.0.1 port 5001
[  5] local 10.0.0.2 port 52332 connected with 10.0.0.1 port 5001
[  3] local 10.0.0.2 port 52328 connected with 10.0.0.1 port 5001
------------------------------------------------------------
Client connecting to 10.0.0.1, TCP port 5001
TCP window size: 93.5 KByte (default)
------------------------------------------------------------
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-10.0082 sec  4.67 GBytes  4.00 Gbits/sec
[  4] 0.0000-10.0068 sec  3.57 GBytes  3.07 Gbits/sec
[  5] 0.0000-10.0086 sec  3.50 GBytes  3.00 Gbits/sec
[  2] 0.0000-10.0070 sec  3.57 GBytes  3.07 Gbits/sec
[  3] 0.0000-10.0231 sec  4.93 GBytes  4.22 Gbits/sec
[SUM] 0.0000-10.0002 sec  20.2 GBytes  17.4 Gbits/sec
[ CT] final connect times (min/avg/max/stdev) = 3.620/3.738/3.835/1.845 ms (tot/
err) = 5/0
root@kali:~/Desktop/assignement/mininet_programs#
```

Figure 25: 5 concurrent requests from h1 from iperf3

The tool gives the total transferred bytes and bandwidth

Total data: 20.2 GBytes Bandwidth : 17.4 Gbits/sec

## 6 Conclusions

The throughput on a normal network was 23847.0548 KB/s i.e 0.0238 GB/s

the throughput in mininet: 17.4 Gbps or 139.2 GB/s

Throughput difference shows that mininet is better than the normal network configuration, as it uses SDN i.e, Software defined network.

## 7 References

http://mininet.org/download/

https://www.geeksforgeeks.org/alias-secondary-ip-address/

https://programming.vip/docs/mininet-use-iperf-tool-to-test-bandwidth.html

https://seis.bristol.ac.uk/ sy13201/sdn_lab/

https://webcms3.cse.unsw.edu.au/static/uploads/course/COMP3331/16s1/a78e6660ae6c4193d3a3bc88