Mawlana Bhashani Science & Technology University

Lab Report No : 04

Course Code : ICT3208

Course Title : Computer Network Lab

Report Name : SDN Controllers and Mininet

Submitted by

Name : Maskur Al Shal Sabil

Id: IT18021

3rd year 2nd Semester

Session 2017-2018

Dept of ICT

MBSTU

Submitted To

Nazrul Islam

Assistant Professor

Dept of ICT

MBSTU

1. Objectives

The objective of the lab 4 is to:

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

2. Theory

2.1. 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

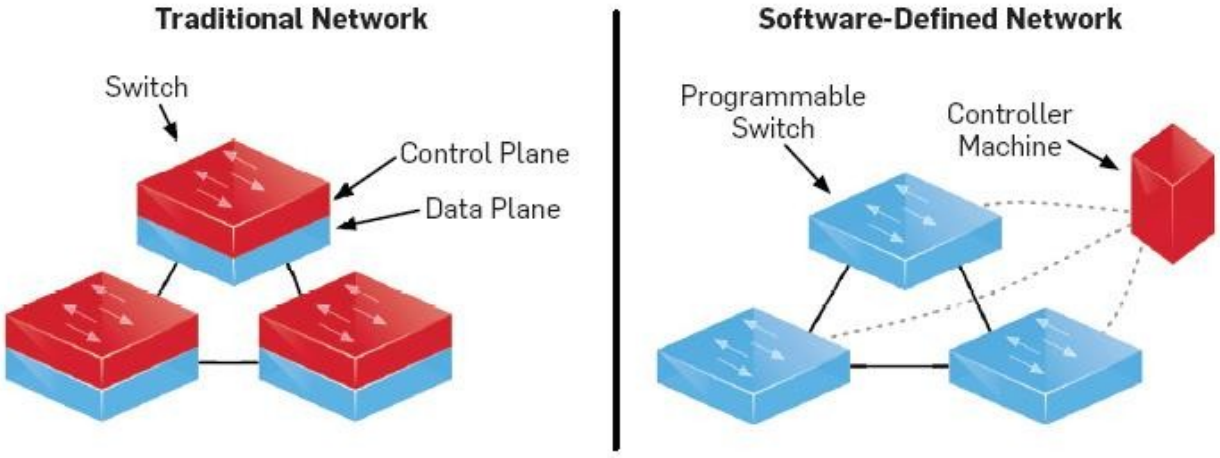
parameters.

2.2. 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



Software-Defined vs. Traditional Networking: The key difference between traditional and

software-defined networking is how SDNs handle data packets. In a traditional network, the way a switch handles an incoming data packet is written into its firmware. Most switches — particularly those used in commercial data centers rather than enterprise environments — respond to and route all packets the same way. SDN provides admins with granular control over the way switches handle data, giving them the ability to automatically prioritize or block certain types of packets. This, in turn, allows for greater efficiency without the need to invest in expensive, application-specific network switches.

Benefits of Software-Defined Networking: There are several benefits to the more advanced level of control afforded by implementing SND in a multi-tenant network environment:

Automation: SND allows for automation of complex operational tasks that make networks faster, more efficient and easier to manage.

Increased uptime: SDN has proven effective in reducing deployment and configuration errors that can lead to service disruptions.

Less drain on resources: SDN gives administrators control over how their routers and switches operate from a single, virtual workflow. This frees up key staff to focus on more important tasks.

Better visibility: With SDN, system administrator's gain improved visibility into overall network function, allowing them to allocate resources more effectively.

Cost savings: SND can lead to significant overall costs savings. It also reduces the amount of spending required on infrastructure by allowing data centers to get the most use of their existing devices.

2.2.1. 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.

Ryu is a component-based software defined networking framework. Ryu provides software components with well-defined API that make it easy for developers to create new network management and control applications. Ryu supports various protocols for managing network devices, such as OpenFlow, Netconf, OF-config, etc. About OpenFlow, Ryu supports fully 1.0,

1.2, 1.3, 1.4, 1.5 and Nicira Extensions. All of the code is freely available under the Apache 2.0 license.

2.2.2. 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

TIP: For getting extra space in your USB-please use the following tips:

Empty the trash

Delete the Android related staff

Delete the extras for other courses

Delete the already installed package sources

Install iperf

1. Open the Synaptic Package Manager (Navigator ->System-> Synaptic Package Manager)

2. Setup the proxy:

o Click on settings-> Preference -> Network

o Click on manual proxy configuration

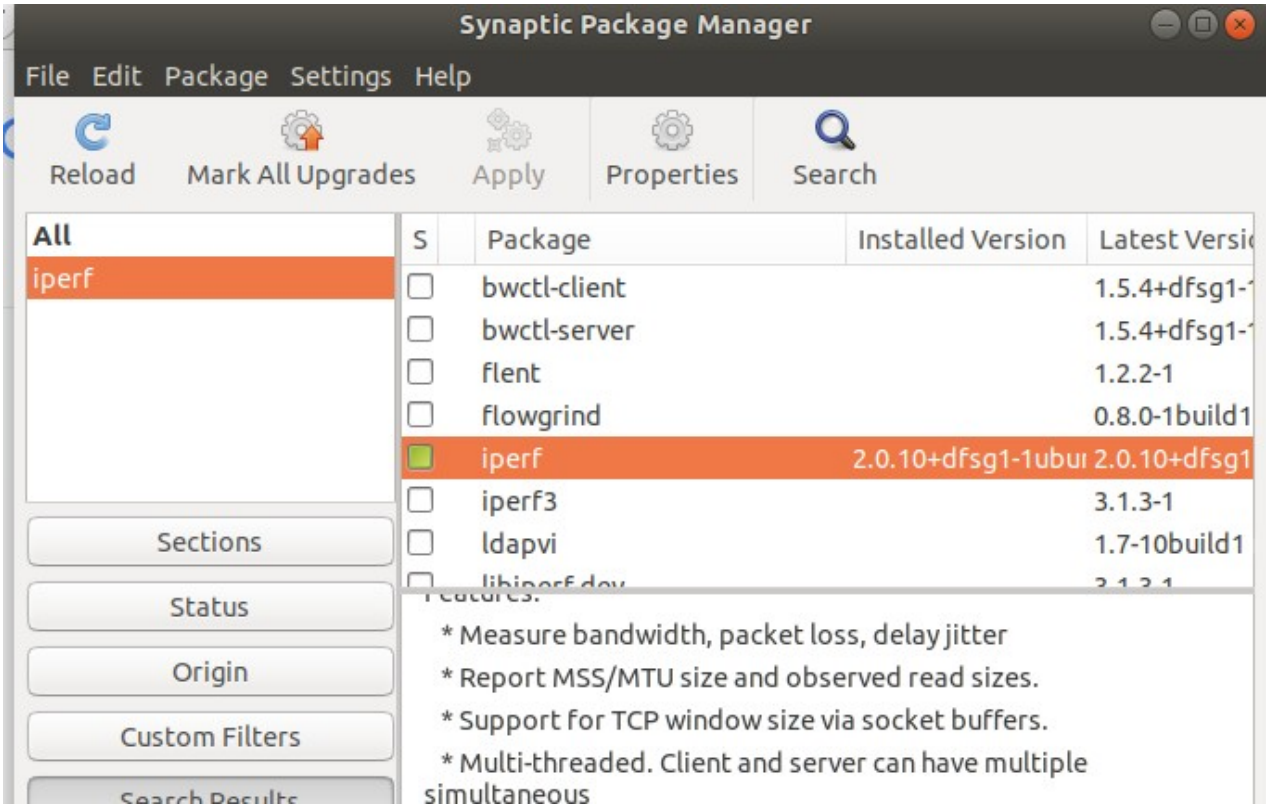
o HTT and FTP Proxy: proxy.rmit.edu.au Port: 8080

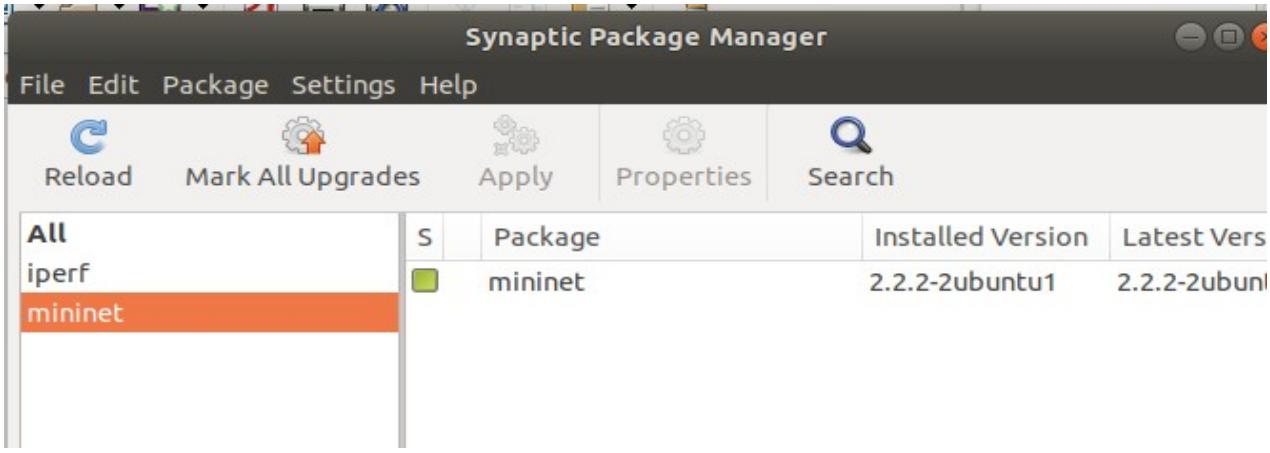
3. Search for Quick filter `iperf`

4. Click on Mark for installation

5. Then click on Apply and wait until the package is installed

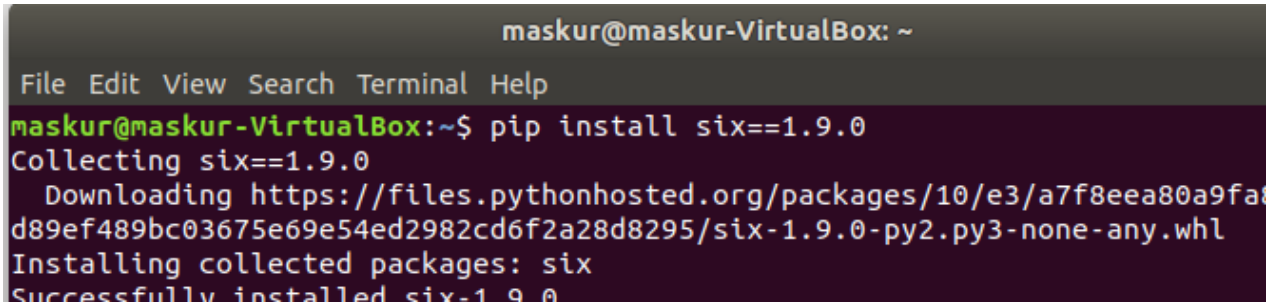
```
File  Edit  View  Search  Terminal  Help
maskur@maskur-VirtualBox:~$ sudo apt install synaptic
[sudo] password for maskur:
Reading package lists... Done
Building dependency tree
Reading state information... Done
```



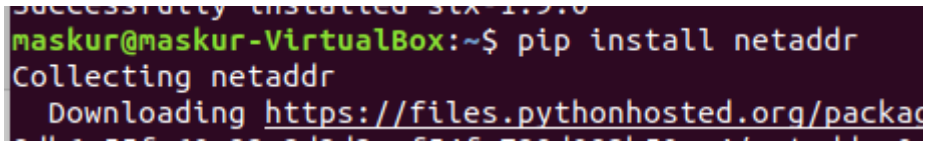


3. Download the following packages:

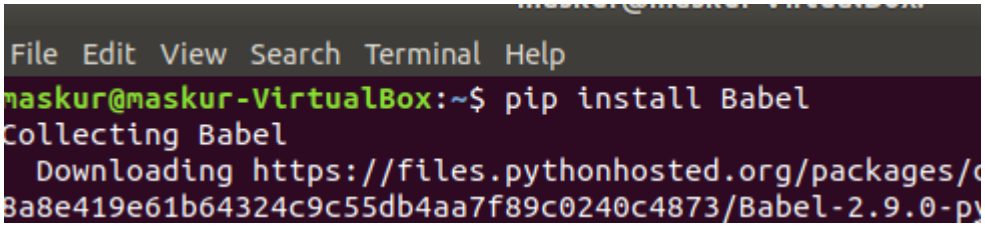
o Six package: https://pypi.python.org/pypi/six/1.9.0



o Netaddr package: https://pypi.python.org/pypi/netaddr#downloads

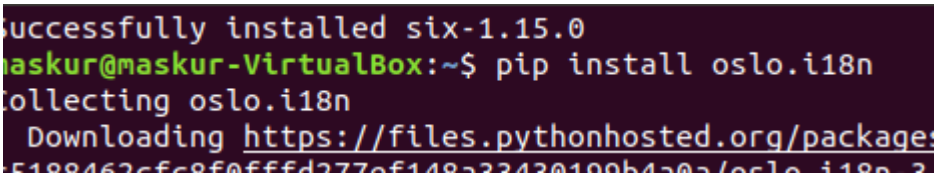


o Babel package: https://pypi.python.org/pypi/Babel

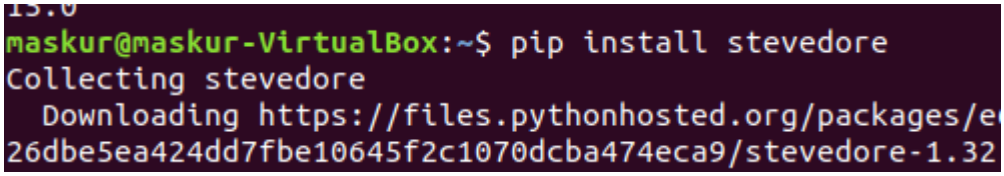


o Six package: https://pypi.python.org/pypi/six

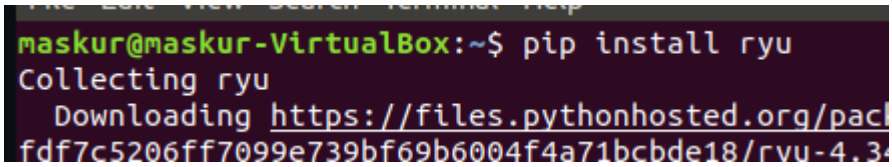
o Oslo package: https://pypi.python.org/pypi/oslo.i18n



o Stevedore package: https://pypi.python.org/pypi/stevedore



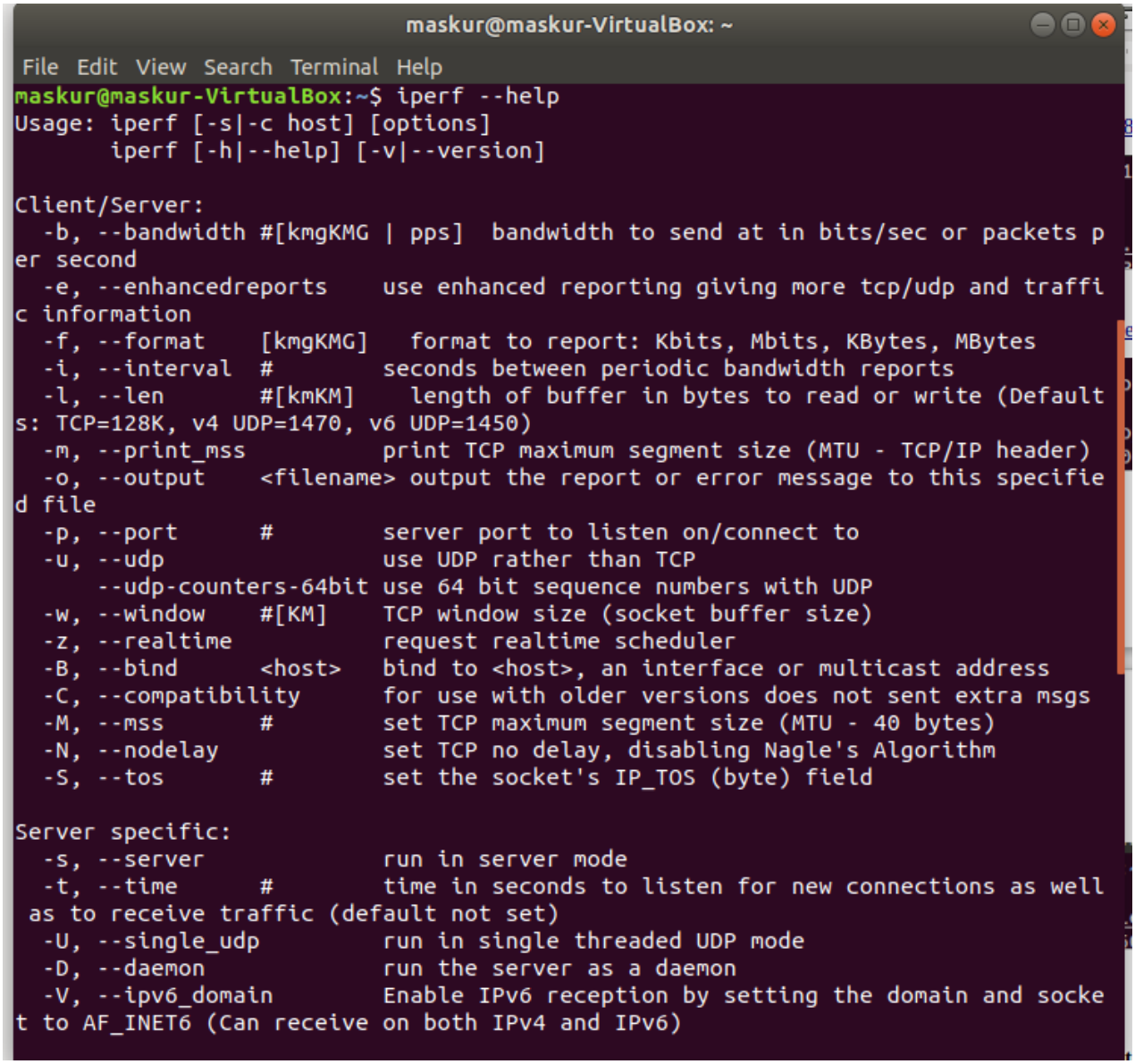
o RYU package: https://github.com/osrg/ryu



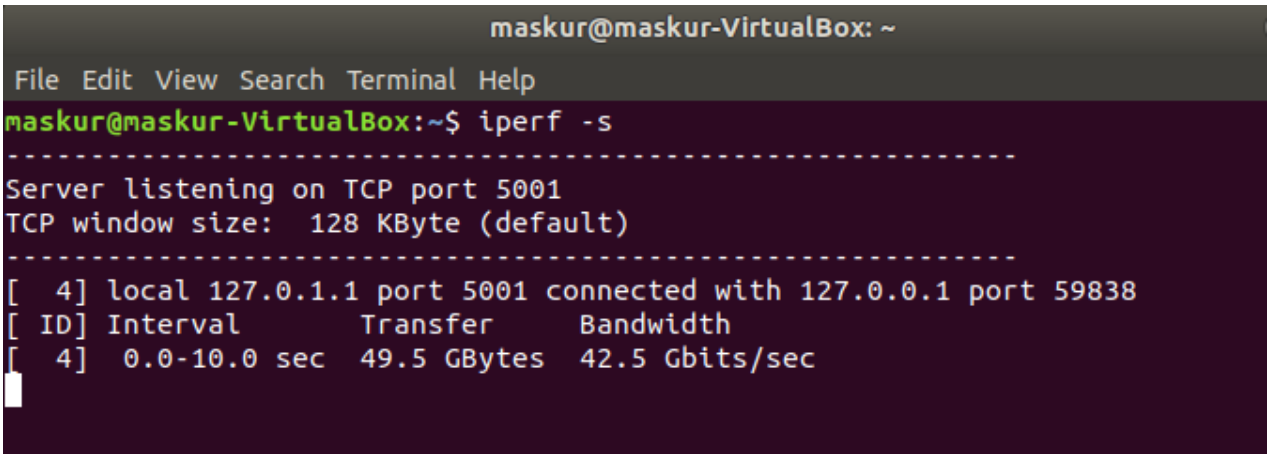
4. Exercises

Exercise 4.1.1: Open a Linux terminal, and execute the command line iperf --help.

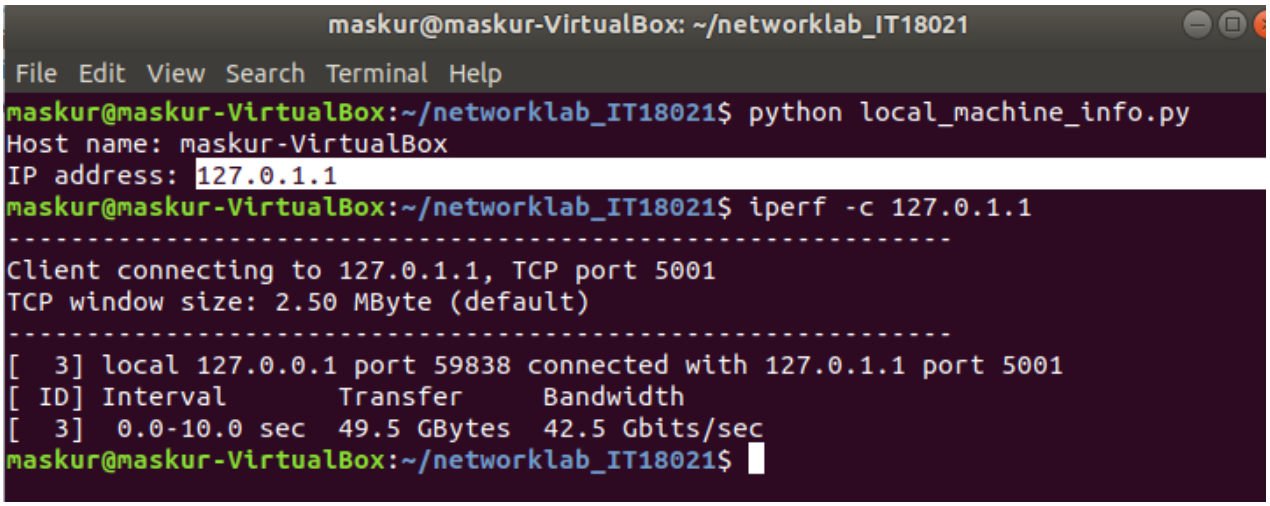
Provide four configuration options of iperf.



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?

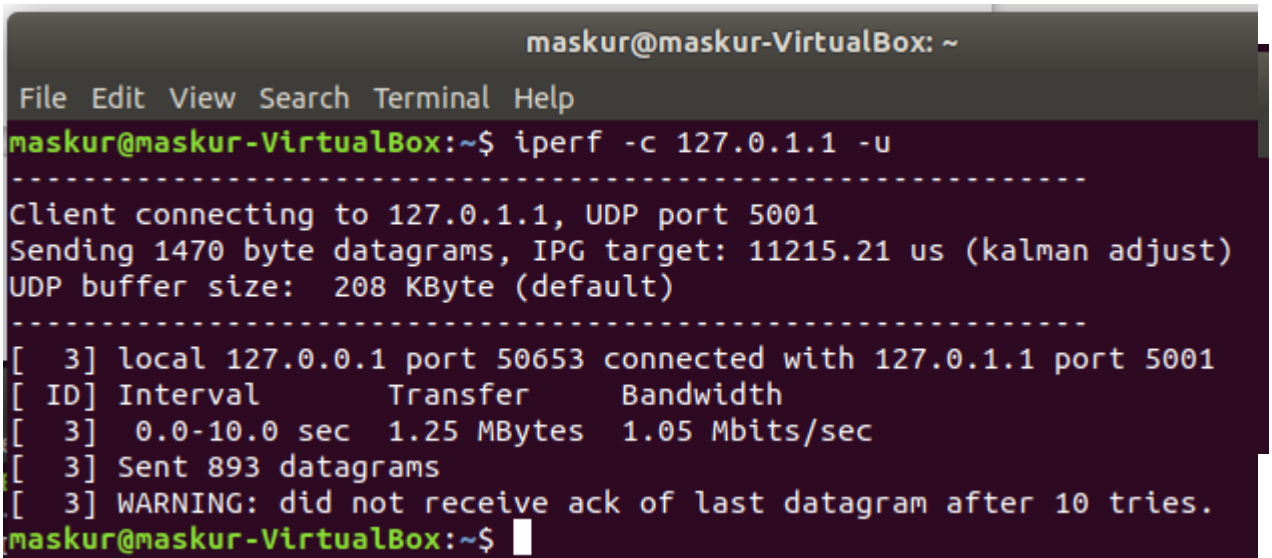


Remark :



Here it has a time interval of 10sec and the badndwitdth is 42.5 g/s. With tranfer 49.5 gb

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.



Comparison with the tcp :

	Interval	Transfer	Bandwidth/ throughput
TCP	10sec	49.5 Gbytes	42.5 Gbits/sec
UDP	10sec	1.25 Mbytes	1.05 Mbits/sec

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?

Here the command is

-l = for setting the length

-t for the time

-b for the bandwidth

-p for the port

```
maskur@maskur-VirtualBox:~$ iperf -c 127.0.1.1 -u -t 20 -p 9900 -b 1 -l 1000
WARNING: delay too large, reducing from 8000.0 to 1.0 seconds.
------------------------------------------------------------
Client connecting to 127.0.1.1, 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 42089 connected with 127.0.1.1 port 9900
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-21.0 sec  1000 Bytes   381 bits/sec
[  3] Sent 1 datagrams
read failed: Connection refused
[  3] WARNING: did not receive ack of last datagram after 2 tries.
maskur@maskur-VirtualBox:~$
```

Exercise 4.2.1: Open two Linux terminals, and execute the command line ifconfig in terminal-1. How many interfaces are present?

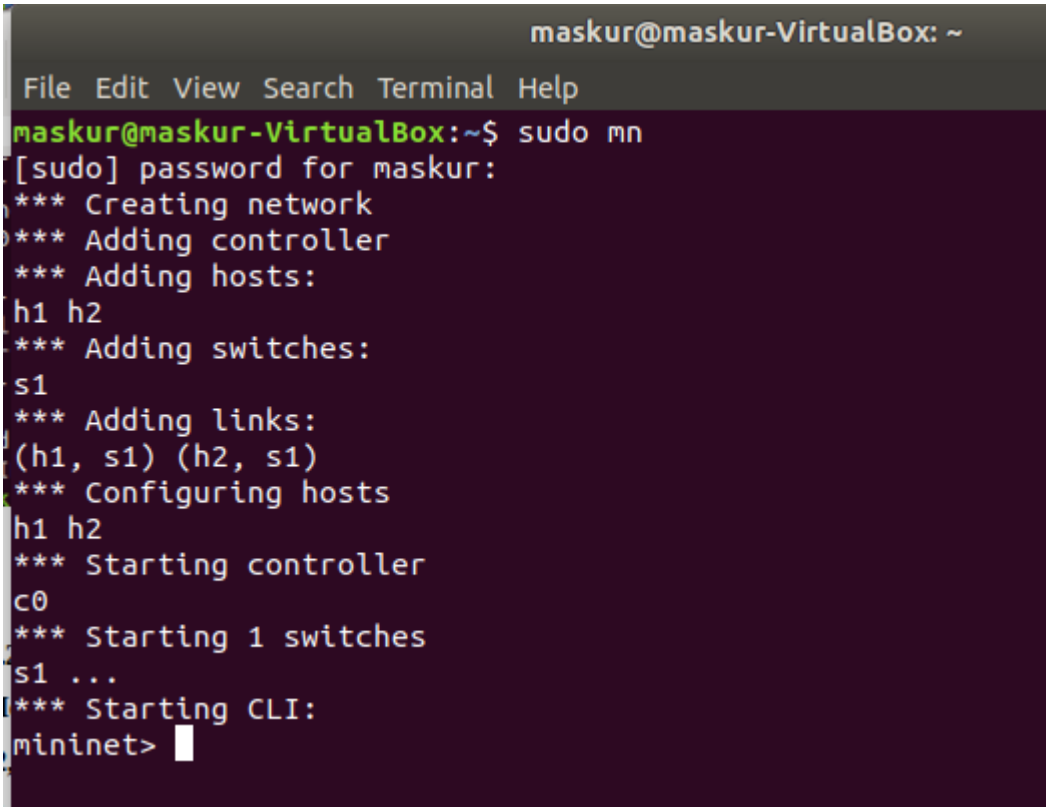
In terminal-2, execute the command line sudo mn, which is the output?

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

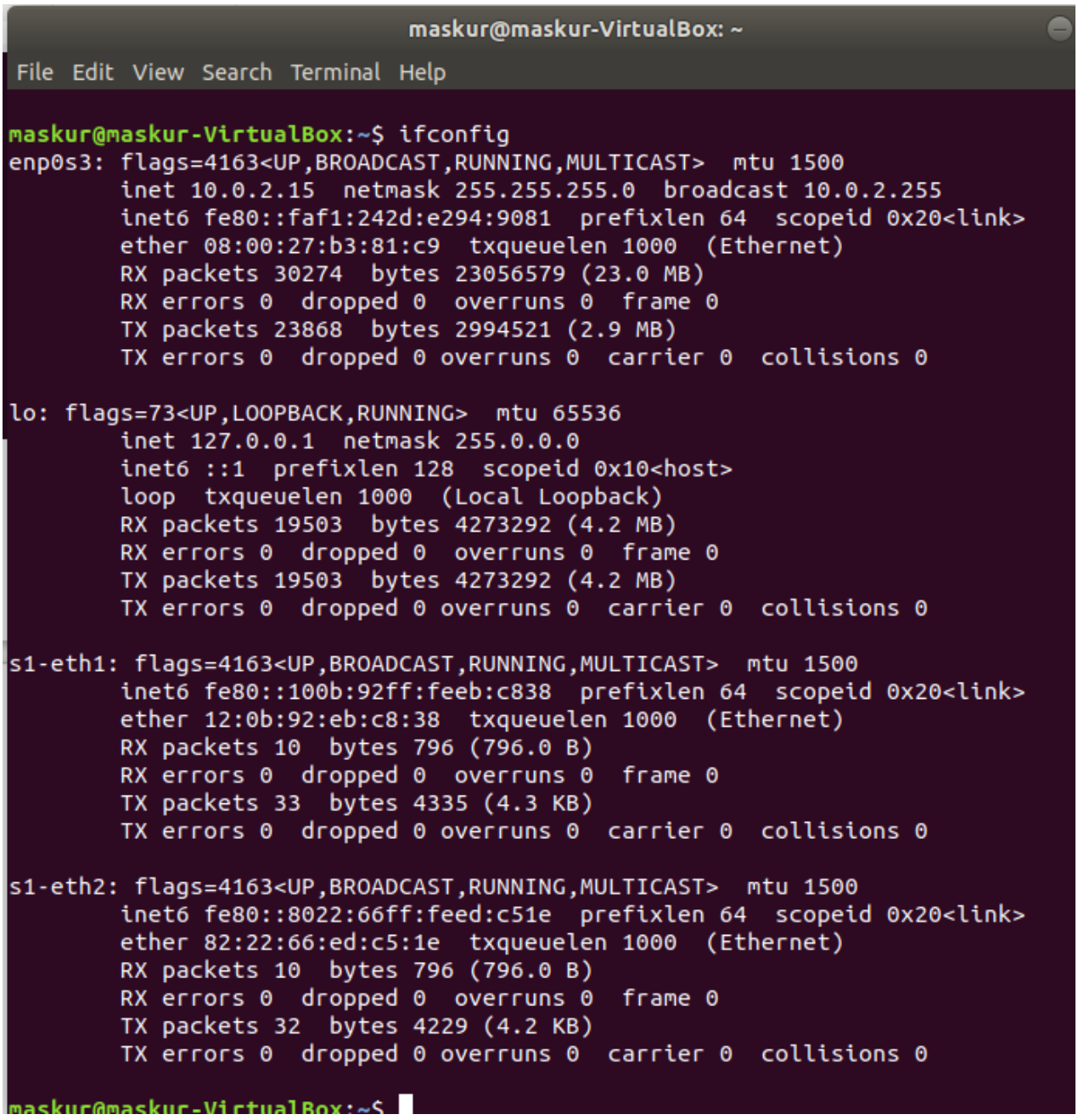
```
maskur@maskur-VirtualBox: ~
File  Edit  View  Search  Terminal  Help
maskur@maskur-VirtualBox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::faf1:242d:e294:9081  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:b3:81:c9  txqueuelen 1000  (Ethernet)
        RX packets 11126  bytes 6583239 (6.5 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 9577  bytes 1226870 (1.2 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 14532  bytes 3935322 (3.9 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 14532  bytes 3935322 (3.9 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Here two interfaces is active " ethernet network peripheral and loopback.

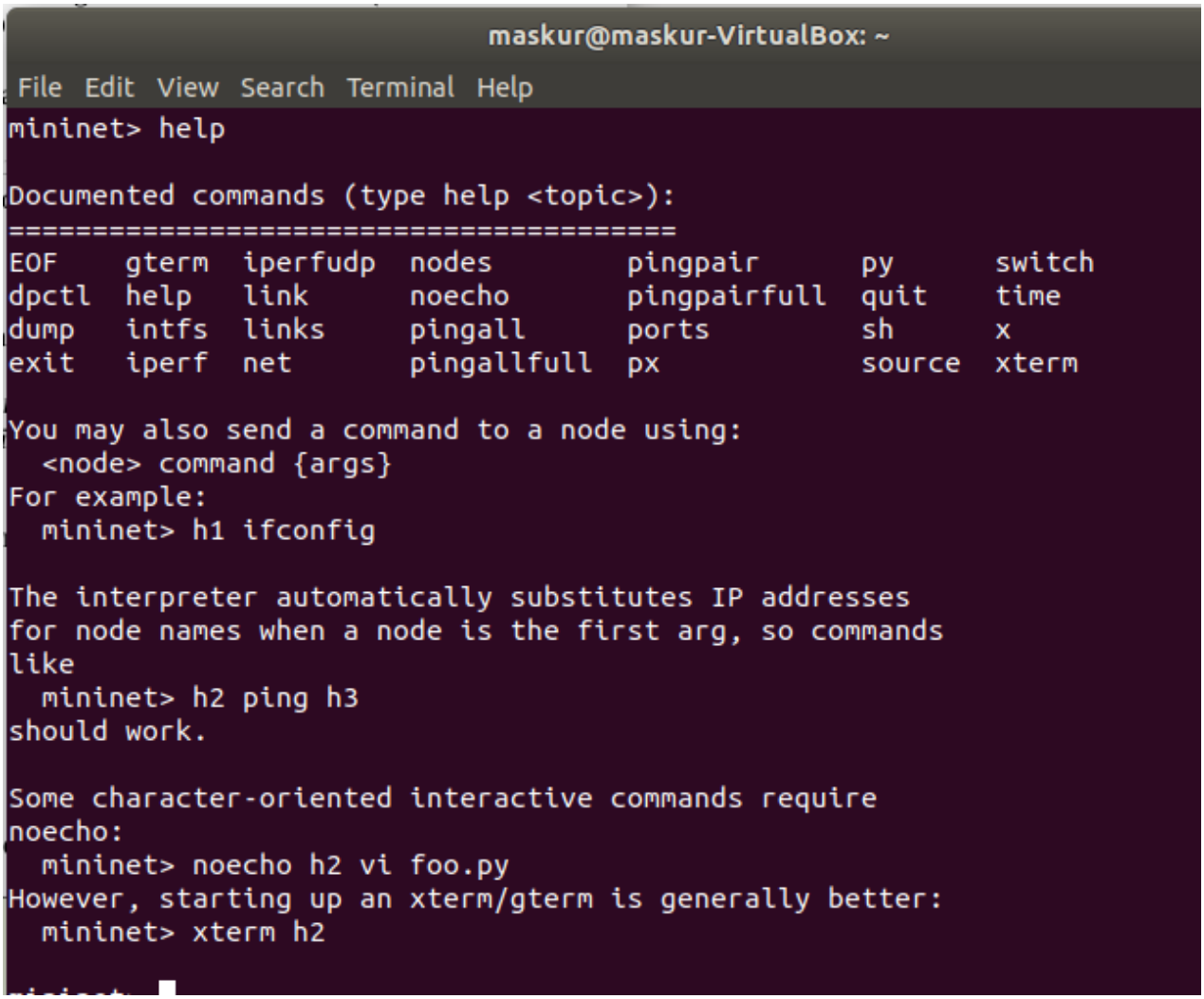
Running sudo mn command in the second terminal :



And again run the command : ifconfig in the first terminal it shows two additional virtual interface " s1-eth1,s1-eth2.


maskur@maskur-VirtualBox: ~
File Edit View Search Terminal Help
maskur@maskur-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::faf1:242d:e294:9081 prefixlen 64 scopeid 0x20<link>
ether 08:00:27:b3:81:c9 txqueuelen 1000 (Ethernet)
RX packets 30274 bytes 23056579 (23.0 MB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 23868 bytes 2994521 (2.9 MB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Local Loopback)
RX packets 19503 bytes 4273292 (4.2 MB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 19503 bytes 4273292 (4.2 MB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
s1-eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet6 fe80::100b:92ff:feeb:c838 prefixlen 64 scopeid 0x20<link>
ether 12:0b:92:eb:c8:38 txqueuelen 1000 (Ethernet)
RX packets 10 bytes 796 (796.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 33 bytes 4335 (4.3 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
s1-eth2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet6 fe80::8022:66ff:feed:c51e prefixlen 64 scopeid 0x20<link>
ether 82:22:66:ed:c5:1e txqueuelen 1000 (Ethernet)
RX packets 10 bytes 796 (796.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 32 bytes 4229 (4.2 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
maskur@maskur-VirtualBox:~$

Exercise 4.2.2: Interacting with mininet; in terminal-2, display the following command



lines and explain what it does:

o mininet> help

o mininet> nodes

o mininet> net

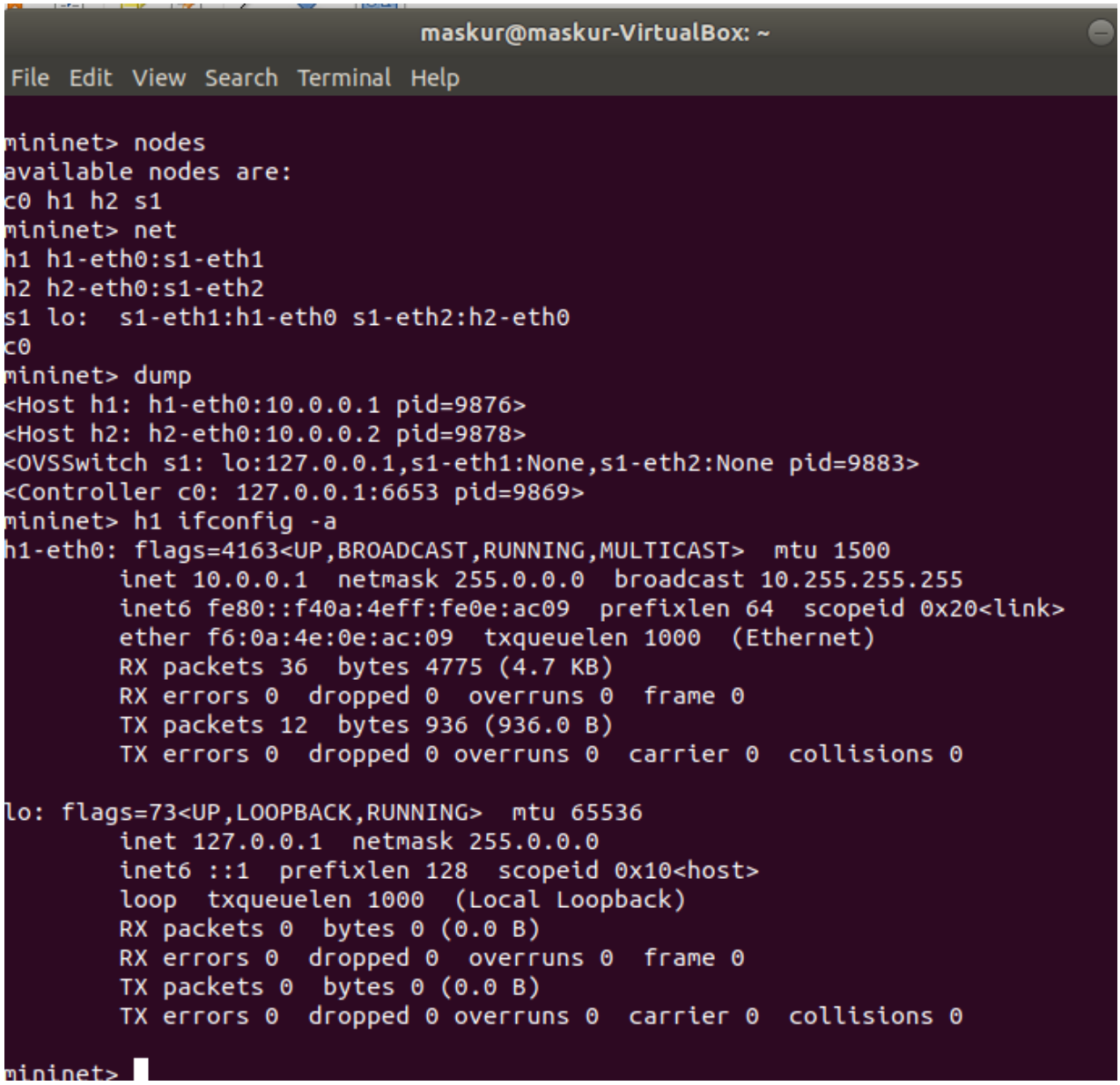
o mininet> dump

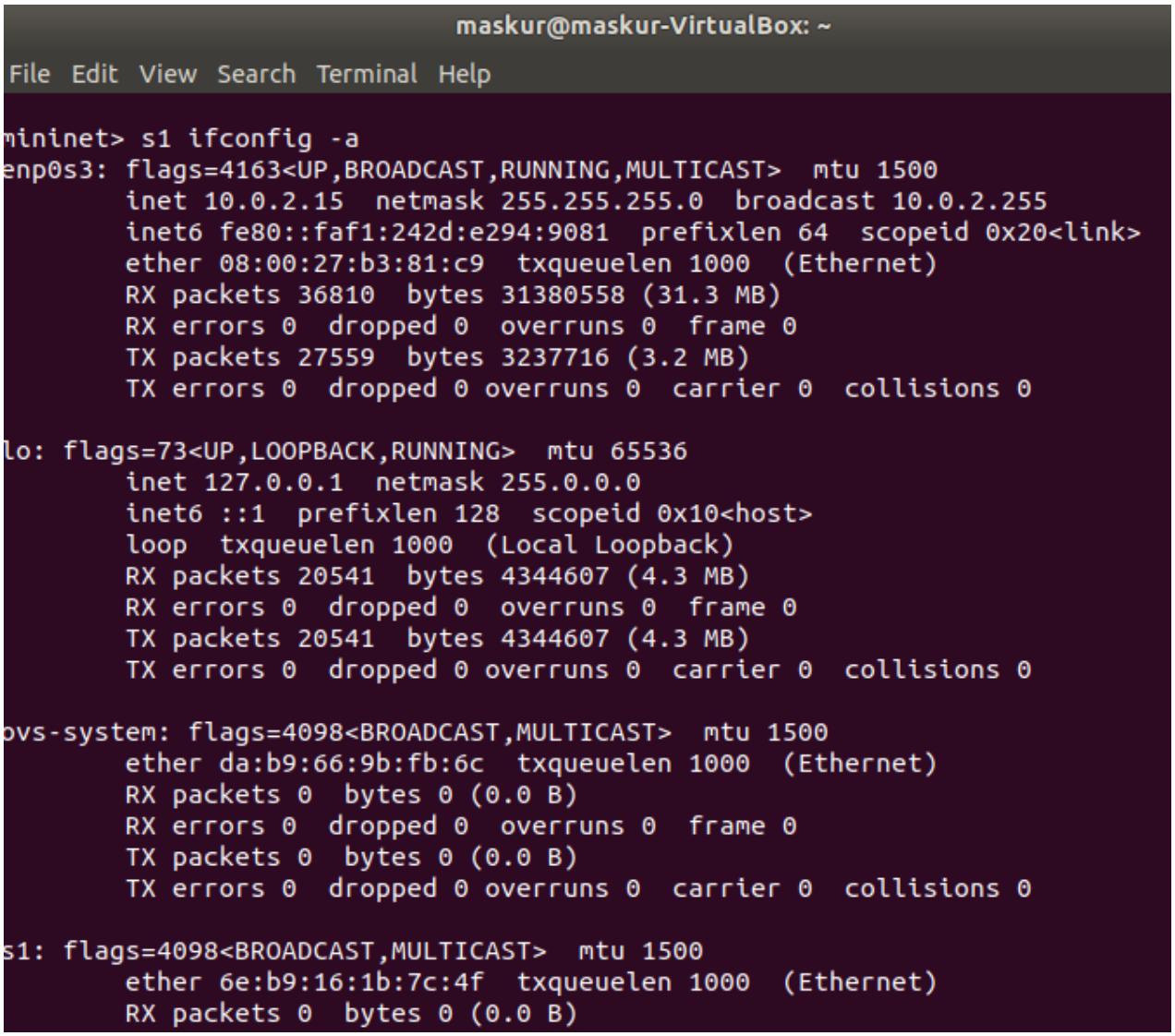
o mininet> h1 ifconfig –a

o mininet> s1 ifconfig –a

o mininet> h1 ping -c 5 h2

o Draw the network topology that is created in mininet:


maskur@maskur-VirtualBox: ~
File Edit View Search Terminal Help
mininet> nodes
available nodes are:
c0 h1 h2 s1
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo: s1-eth1:h1-eth0 s1-eth2:h2-eth0
c0
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=9876>
<Host h2: h2-eth0:10.0.0.2 pid=9878>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=9883>
<Controller c0: 127.0.0.1:6653 pid=9869>
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::f40a:4eff:fe0e:ac09 prefixlen 64 scopeid 0x20<link>
ether f6:0a:4e:0e:ac:09 txqueuelen 1000 (Ethernet)
RX packets 36 bytes 4775 (4.7 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> 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=55.9 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.909 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.171 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.094 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.099 ms

--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4057ms
rtt min/avg/max/mdev = 0.094/11.435/55.903/22.236 ms
mininet>
```

The network topology that is created in mininet is given below :

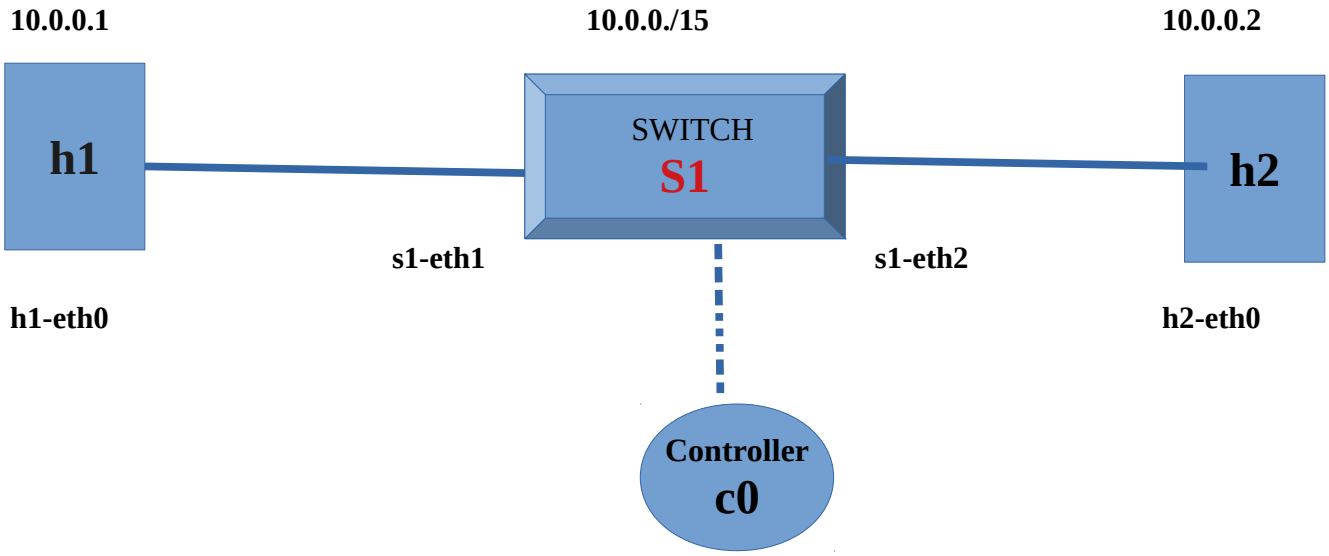


Figure : Mininet default topology

In terminal-1, display the following command line: sudo ovs-vsctl show, what is displayed?

```
maskur@maskur-VirtualBox:~$ ovs-vsctl show
ovs-vsctl: unix:/var/run/openvswitch/db.sock: database connection failed (Per
mission denied)
maskur@maskur-VirtualBox:~$
```

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 562.528 seconds
maskur@maskur-VirtualBox:~$
```

Exercise 4.2.3: In terminal-2, display the following command line: sudo mn --link

tc,bw=10,delay=500ms

```
maskur@maskur-VirtualBox: ~
File  Edit  View  Search  Terminal  Help
maskur@maskur-VirtualBox:~$ sudo mn --link tc,bw=10,delay=500ms
*** 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>
```

o 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=3 ttl=64 time=2014 ms
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=4051 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=3044 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=2001 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=2003 ms

--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4066ms
rtt min/avg/max/mdev = 2001.331/2623.205/4051.372/819.414 ms, pipe 4
```

o 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)
------------------------------------------------------------
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
```

o 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?

Discussion : This SDN lab is more complex to do comparing with the previous. Here in the -iperf command I run the server easily but to connect with the client at first time it was not clear to me about the ip . Which one should I use . After that I use the loopback 127.0.1.1 and it connect . And to check the wireshark it needs another friends computer to make a communication since I was not able to do this . In the next I use the mininet to construct network. I'm still working with this lab report . I Hope I will add more .