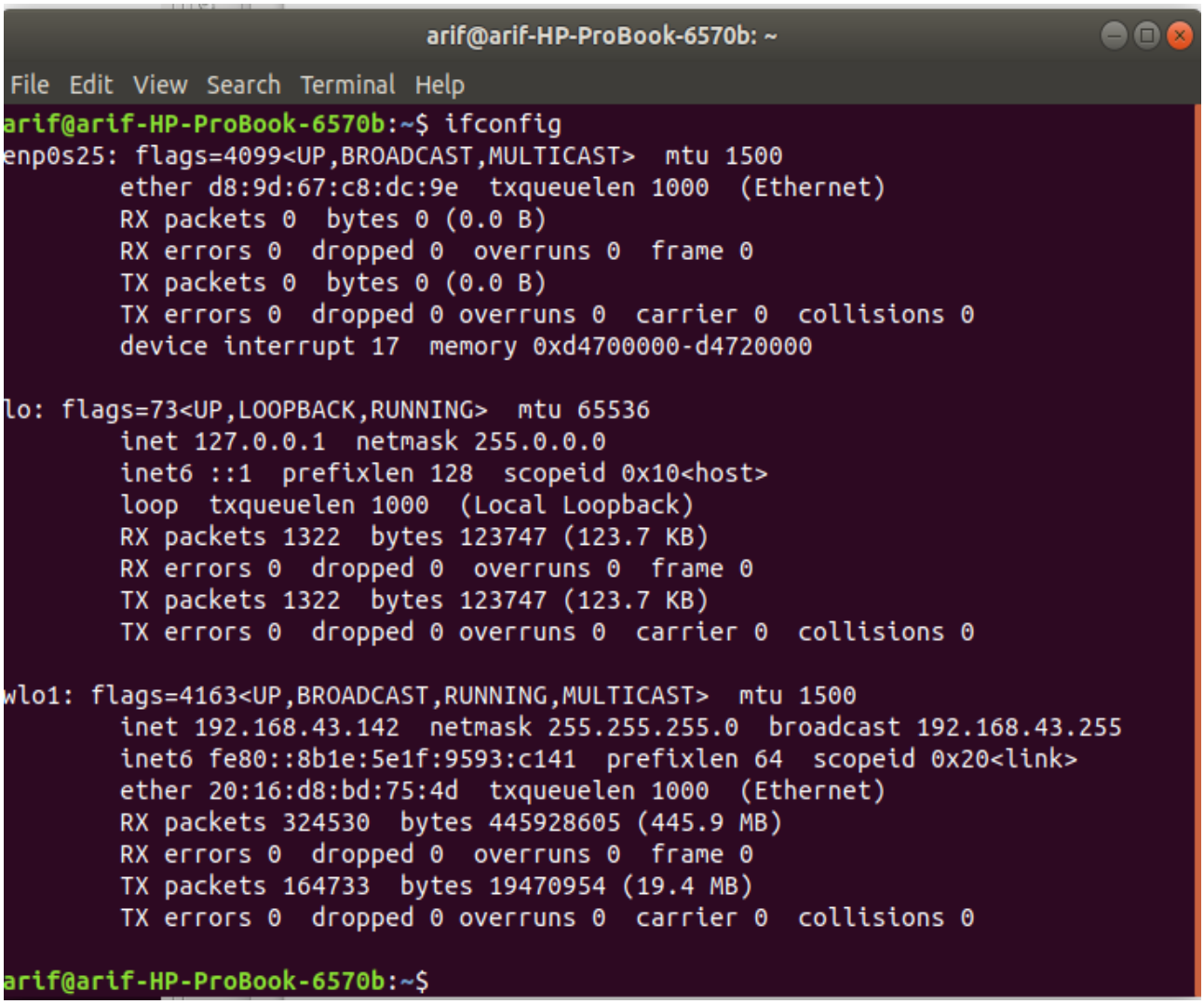
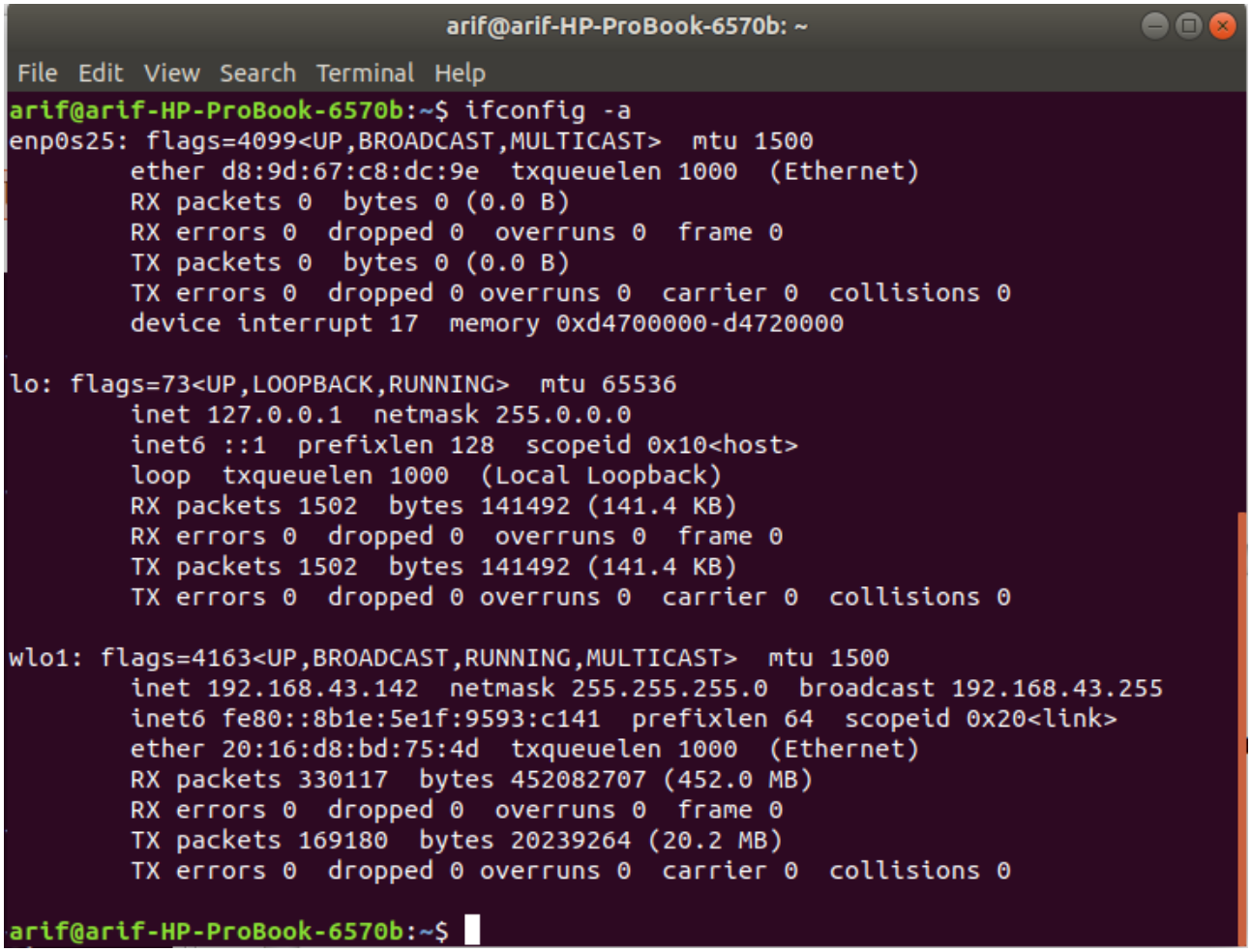
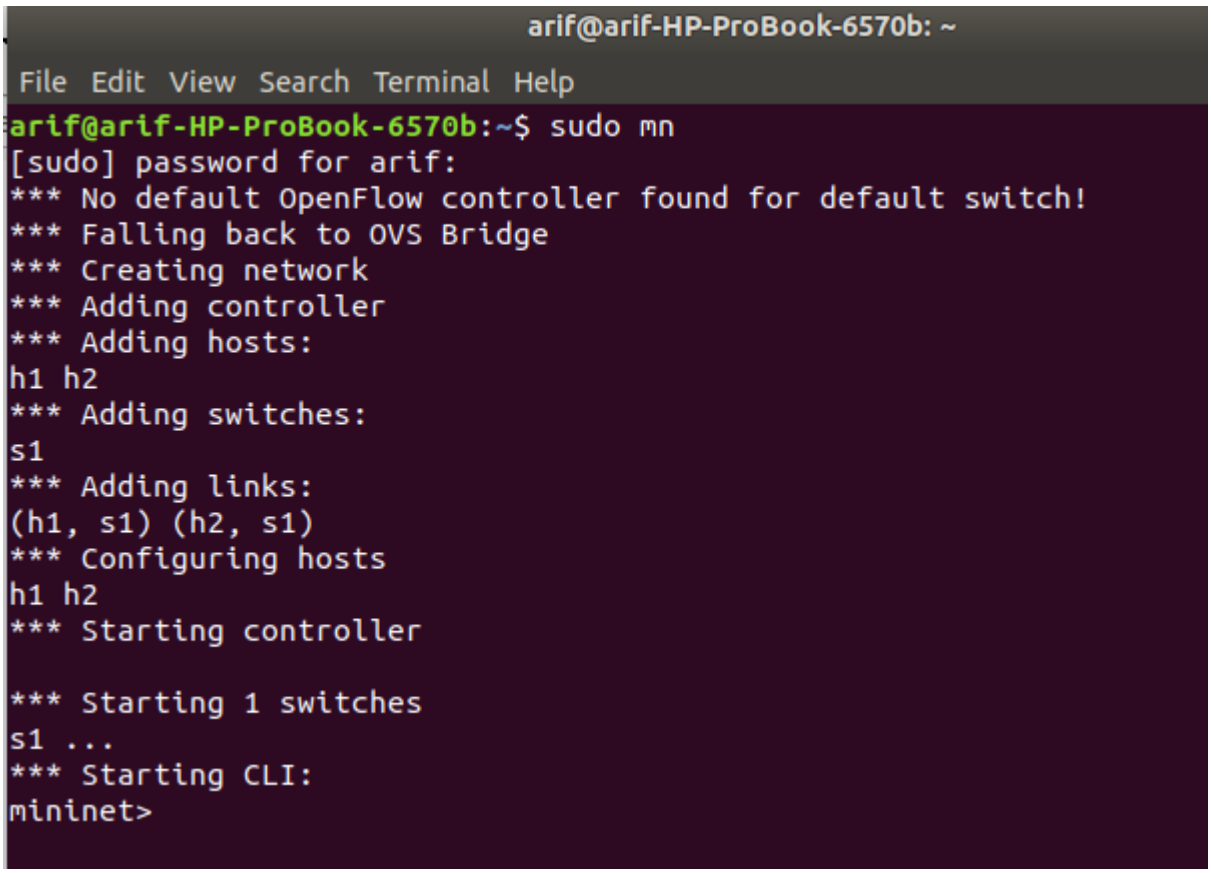
ifconfig : **ifconfig** is a system administration utility in Unix-like operating systems for network interface configuration. The utility is a command-line interface tool and is also used in the system start-up scripts of many operating systems.



ifconfig -a: **ifconfig -a** displays the configuration of all interfaces, both active and inactive.



Virtual Network Creating: Using CLI(sudo mn command) to manage our virtual network. This is default topology and it includes two hosts (h1,h2), OpenFlow Swith(s1).



Help Command: Run the help option to view the list of commands available.

```
mininet> help

Documented commands (type help <topic>):
========================================
EOF    gterm  iperfudp  nodes        pingpair      py      switch
dpctl  help   link      noecho       pingpairfull  quit    time
dump   intfs  links     pingall      ports         sh      x
exit   iperf  net       pingallfull  px            source  xterm

You may also send a command to a node using:
  <node> command {args}
For example:
  mininet> h1 ifconfig

The interpreter automatically substitutes IP addresses
for node names when a node is the first arg, so commands
like
  mininet> h2 ping h3
should work.

Some character-oriented interactive commands require
noecho:
  mininet> noecho h2 vi foo.py
However, starting up an xterm/gterm is generally better:
  mininet> xterm h2

mininet>
```

Nodes Command: Its command shows the nodes with simple mininet command.

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

Net Command: This command shows the links of all command.

```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0
mininet>
```

Dump Command: This command displays the all summary information of all devices.

```
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=14220>
<Host h2: h2-eth0:10.0.0.2 pid=14222>
<OVSBridge s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=14227>
mininet>
```

h1 ifconfig -a: Its display the IP address, broadcast address and MAC address of the host h1.

```
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::ec85:e9ff:fe9e:3a9f  prefixlen 64  scopeid 0x20<link>
        ether ee:85:e9:9e:3a:9f  txqueuelen 1000  (Ethernet)
        RX packets 44  bytes 4608 (4.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 13  bytes 1006 (1.0 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 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>
```

ping command: Testing connectivity between the host by using ping command.

```
mininet> h1 ping 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=26.8 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.509 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.084 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.083 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.107 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.079 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.080 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.099 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.083 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.083 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=0.080 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=0.105 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=0.100 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=0.102 ms
64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=0.100 ms
64 bytes from 10.0.0.2: icmp_seq=16 ttl=64 time=0.098 ms
64 bytes from 10.0.0.2: icmp_seq=17 ttl=64 time=0.097 ms
64 bytes from 10.0.0.2: icmp_seq=18 ttl=64 time=0.164 ms
64 bytes from 10.0.0.2: icmp_seq=19 ttl=64 time=0.107 ms
```

Pingall command: It will make each host in the network ping every other host in the network.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet>
```

Creating topology : topo =single,4 that means controller , switch is one and host are 4 like h1,h2,h3,h4.

```
arif@arif-HP-ProBook-6570b:~$ sudo mn --topo=single,4
[sudo] password for arif:
*** No default OpenFlow controller found for default switch!
*** Falling back to OVS Bridge
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller

*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

Exit Command:

```
mininet> exit
*** Stopping 0 controllers

*** Stopping 4 links
....
*** Stopping 1 switches
s1
*** Stopping 4 hosts
h1 h2 h3 h4
*** Done
completed in 83.536 seconds
arif@arif-HP-ProBook-6570b:~$
```

Cleanup command: If Mininet crashes for some reason, clean it up.

```
arif@arif-HP-ProBook-6570b:~$ sudo mn -c
*** 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-openfl
owd 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 --timeout=1 list-br
*** Removing all links of the pattern foo-ethX
ip link show | egrep -o '([-_.[:alnum:]]+-eth[[:digit:]]+)'
ip link show
*** Killing stale mininet node processes
pkill -9 -f mininet:
*** Shutting down stale tunnels
pkill -9 -f Tunnel=Ethernet
pkill -9 -f .ssh/mn
rm -f ~/.ssh/mn/*
*** Cleanup complete.
arif@arif-HP-ProBook-6570b:~$
```

Discussion: Here we learnt about how to works mininet command. Here maximum mininet command are used and shows how each works separately.

Using mininet, we are quickly create a realistic virtual network running actual kernel, switch and software application code on a personal computer. Mininet allows the user to quickly create, interact with, customize and share a software-defined network (SDN) prototype to simulate a network topology that uses OpenFlow switches.