Academic Year: 2023-24 **Semester: V**
Class /Branch: TE-DS **Subject: WCN**

Experiment No. 10

1. **Aim:** To simulate Software Defined Network using Mininet.

2. **Software used:** Mininet

3. **Theory:** -

Mininet

Mininet is a tool for software-defined networks. It is an emulator of a network and it is used to visualize the switches and application of software-defined networks in a virtualized environment. It is also used to test the software-defined network devices and those using OpenFlow protocols. The switches used in Mininet are OpenFlow switches. Mininet is majorly used as a learning tool to test, experiment, and learn about software-defined networks. Mininet is preferable because it is very fast and helps us to create customizable topologies. It is also very easy to use.

Methods to Install Mininet:

1. Install Mininet as a virtual machine provided having a virtual box installed already
2. By installing it directly from the source.
3. By installing it from the package.
4. By upgrading the currently existing Mininet.

On Downloading the Mininet we get –

1. Command-line:

The Mininet command line is similar to that of the Linux command line, using it we can interact with the system. First, we need to type the Linux commands on the shell prompt later we type the Mininet command on the Mininet CLI.

Mininet commands:

1. ***nodes:*** *It displays all the mininet nodes available*
2. ***net:*** *It displays the links between the nodes*
3. ***dump:*** *It displays the information about all the nodes.*

2. Mininet User interface:

Actually, there is no exact user interface for the Mininet because everything in the Mininet is controlled either by python scripts or commands. But the Mininet can be controlled using Graphical User Interface while creating and visualizing the topology in the graphic environment.

3. Python interfaces:

Mininet is available as a package and can be imported using python. We can also communicate and control the Mininet nodes by writing the scripts in python.

Features of Mininet:

1. Mininet provides us the space to develop and test our software-defined network applications without the need to set up a physical environment.
2. It gives us a network testbed thereby allowing us to develop and test applications that are using OpenFlow protocols.
3. It can be used without programming also.
4. It also gives us the flexibility to integrate python API, thereby paving the way for creating and experimenting with networks.
5. It also has a CLI (Command Line Interface) which is aware of topology and OpenFlow thereby allowing us to debug or run network tests for our application.

Advantages of Mininet:

1. It is very fast and takes very little time for booting.
2. It is easy to install and use.
3. It saves money because the emulators are cost-effective instead of testing with hardware devices.
4. It is also very easy to connect with real-world network devices.
5. It has high availability.

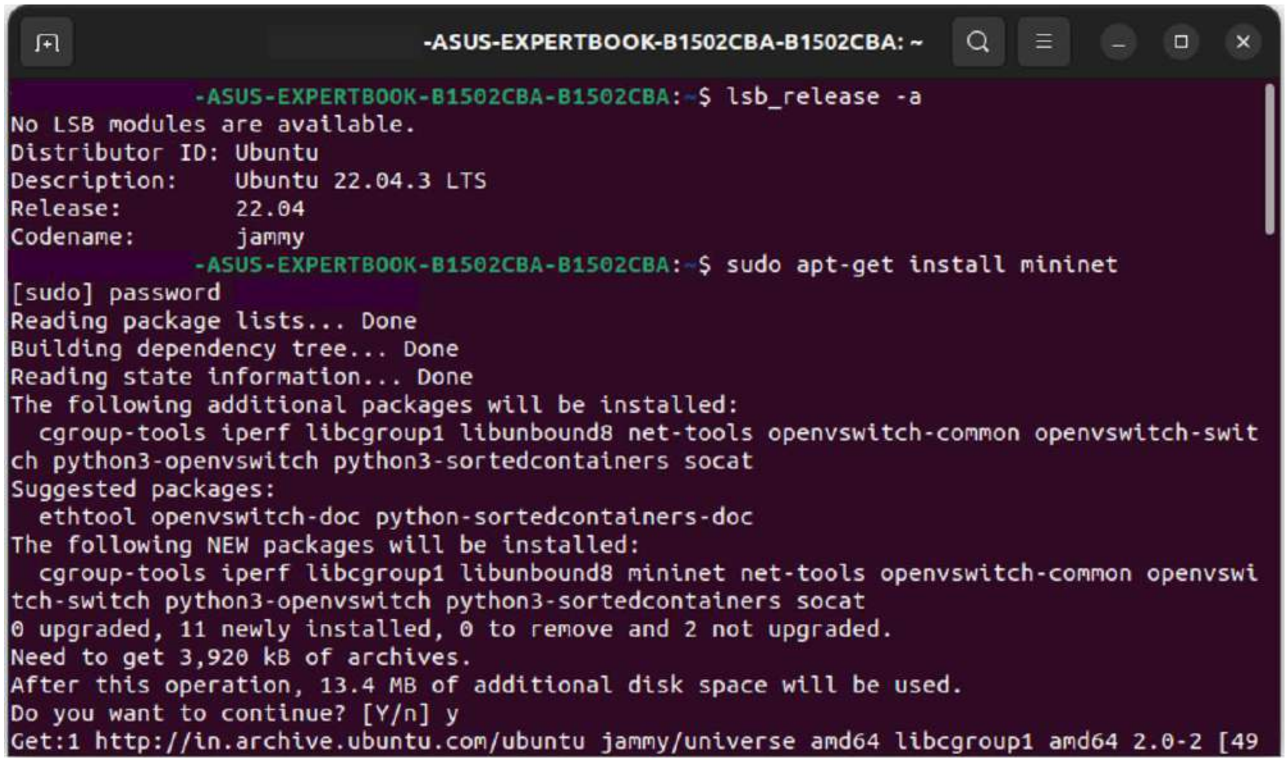
Installing Mininet

Step 1. Launch a Linux terminal by holding the `Ctrl+Alt+T` keys or by clicking on the Linux terminal icon.

Step 2. Make sure that the list of packages from all repositories is up to date by running the following command. When prompted to enter a password, please enter the password of the account you are currently logged into.

Step 3. Install the ***mininet*** package by entering the following command.

1. Type the command "lsb_release -a" into the command line and press enter.
2. The terminal shows the Ubuntu version you're running under "Description" and "Release".

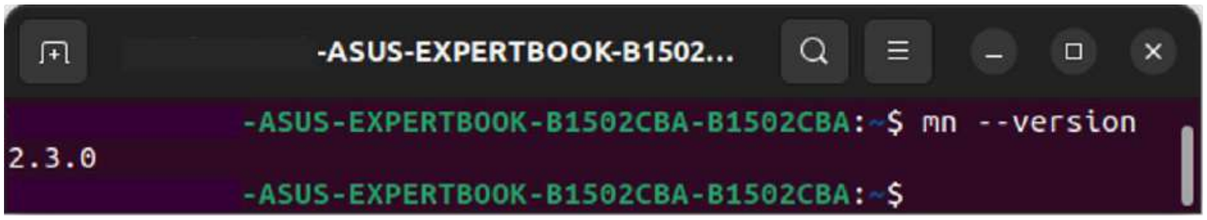


```
[redacted]-ASUS-EXPERTBOOK-B1502CBA-B1502CBA: ~
[redacted]-ASUS-EXPERTBOOK-B1502CBA-B1502CBA:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 22.04.3 LTS
Release:        22.04
Codename:       jammy
[redacted]-ASUS-EXPERTBOOK-B1502CBA-B1502CBA:~$ sudo apt-get install mininet
[sudo] password [redacted]
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  cgroup-tools iperf libcgroup1 libunbound8 net-tools openvswitch-common openvswitch-swit
ch python3-openvswitch python3-sortedcontainers socat
Suggested packages:
  ethtool openvswitch-doc python-sortedcontainers-doc
The following NEW packages will be installed:
  cgroup-tools iperf libcgroup1 libunbound8 mininet net-tools openvswitch-common openvswi
tch-switch python3-openvswitch python3-sortedcontainers socat
0 upgraded, 11 newly installed, 0 to remove and 2 not upgraded.
Need to get 3,920 kB of archives.
After this operation, 13.4 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://in.archive.ubuntu.com/ubuntu jammy/universe amd64 libcgroup1 amd64 2.0-2 [49
```

Press ***Y*** key on your keyboard to proceed with the installation.

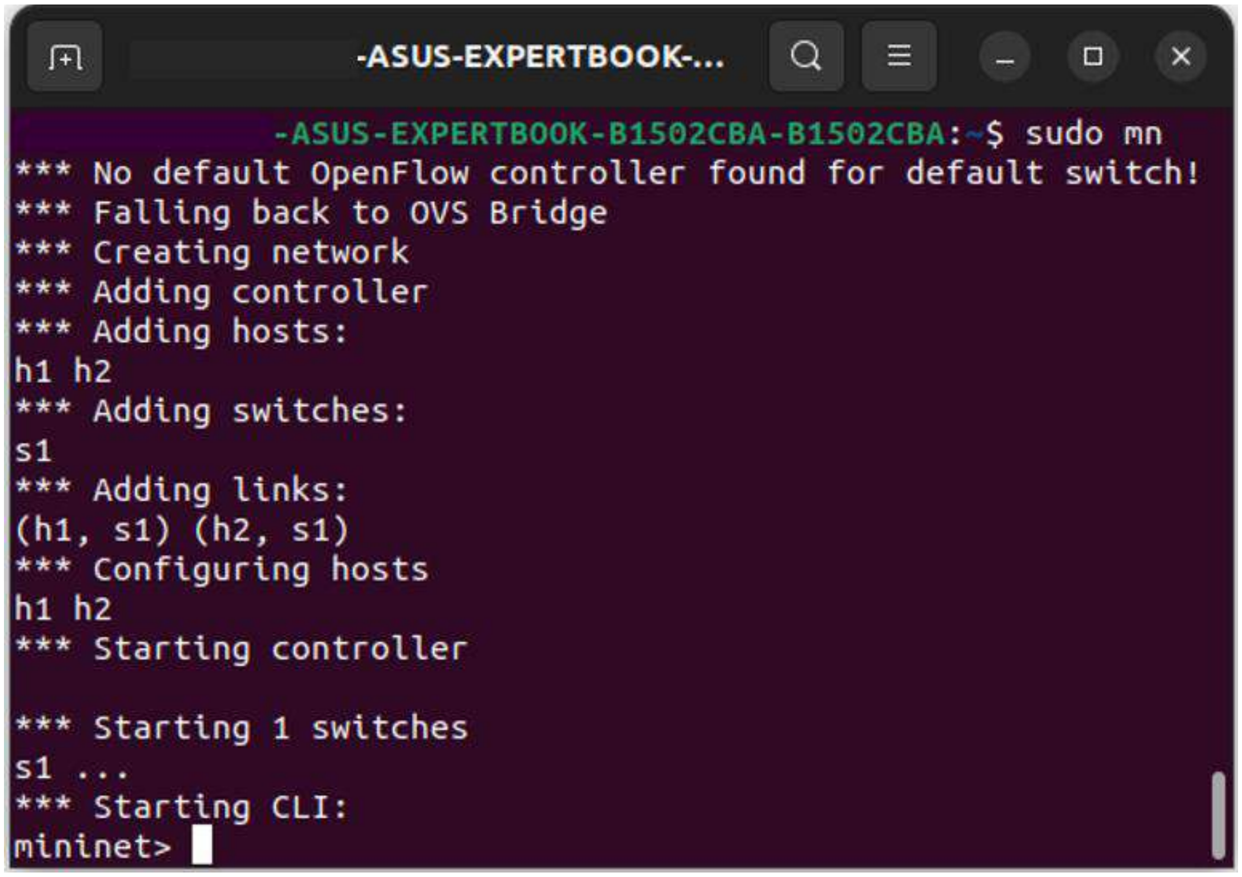
Getting started with Its commands

- To know the version of Mininet. The below command is used.



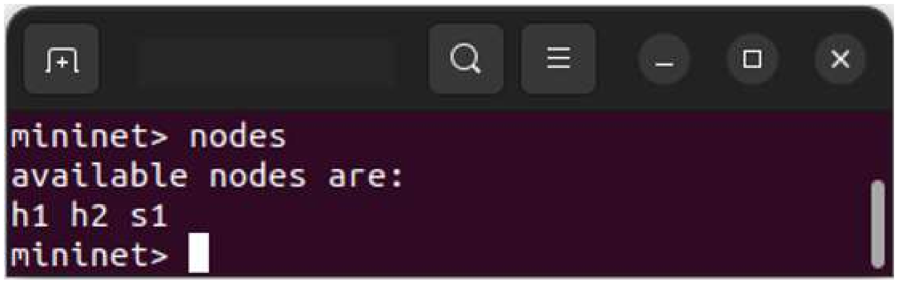
```
[redacted]-ASUS-EXPERTBOOK-B1502...

[redacted]-ASUS-EXPERTBOOK-B1502CBA-B1502CBA:~$ mn --version
2.3.0
[redacted]-ASUS-EXPERTBOOK-B1502CBA-B1502CBA:~$
```

- Next, verify the installation by issuing the following command:



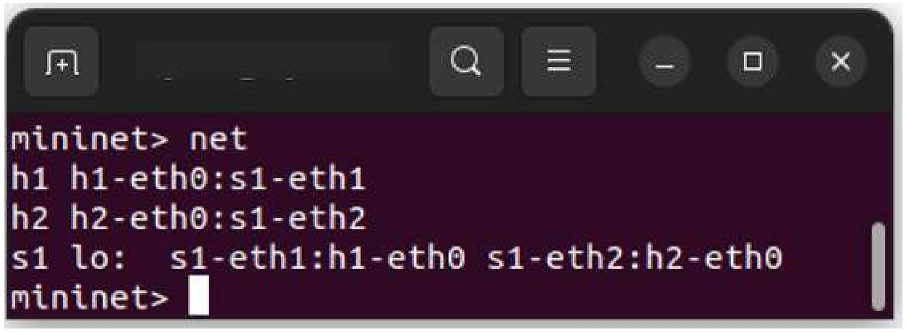
```
[redacted]-ASUS-EXPERTBOOK-...

[redacted]-ASUS-EXPERTBOOK-B1502CBA-B1502CBA:~$ sudo mn
*** 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>
```

- The command to display the nodes present in the network is:



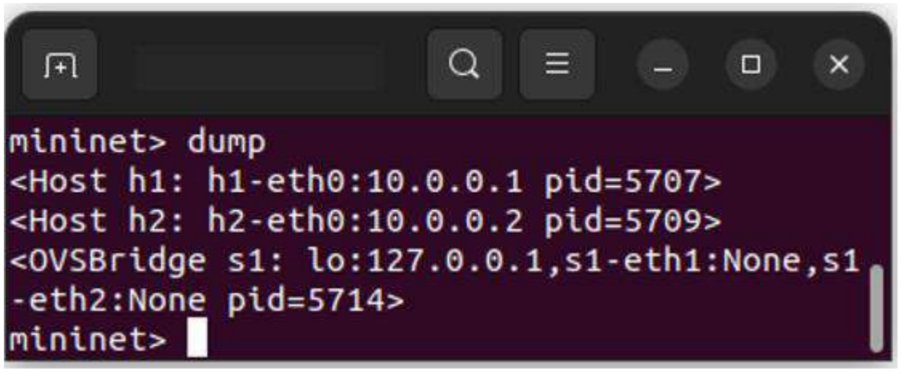
```
mininet> nodes
available nodes are:
h1 h2 s1
mininet>
```

- The command to display and list the links present in the network is:



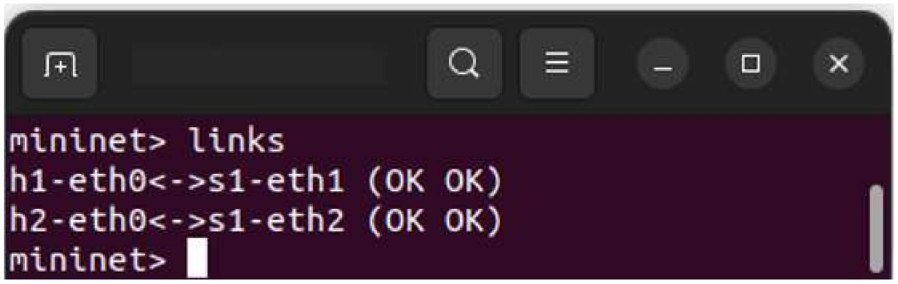
```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0
mininet>
```

- The command to display the IP addresses and the process IDs of the nodes is:



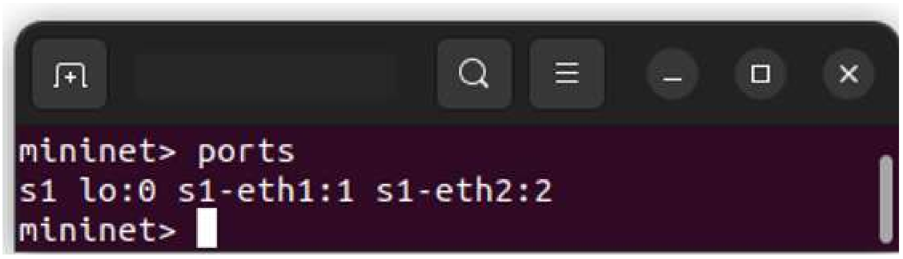
```
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=5707>
<Host h2: h2-eth0:10.0.0.2 pid=5709>
<OVSBridge s1: lo:127.0.0.1,s1-eth1:None,s1
-eth2:None pid=5714>
mininet>
```

- The command to display the links of the network is:



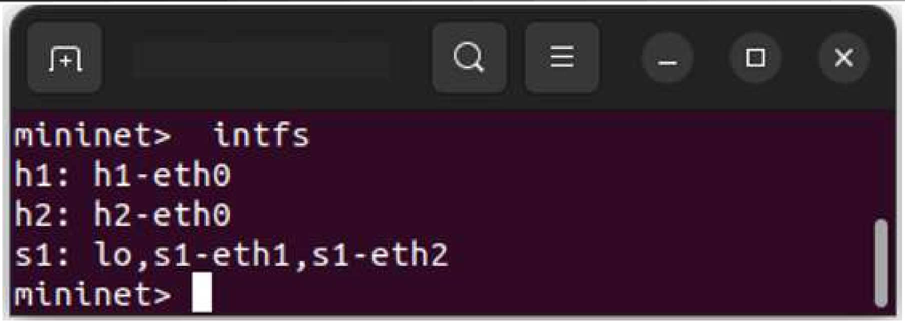
```
mininet> links
h1-eth0<->s1-eth1 (OK OK)
h2-eth0<->s1-eth2 (OK OK)
mininet>
```

- The command to display the ports used in the network is:



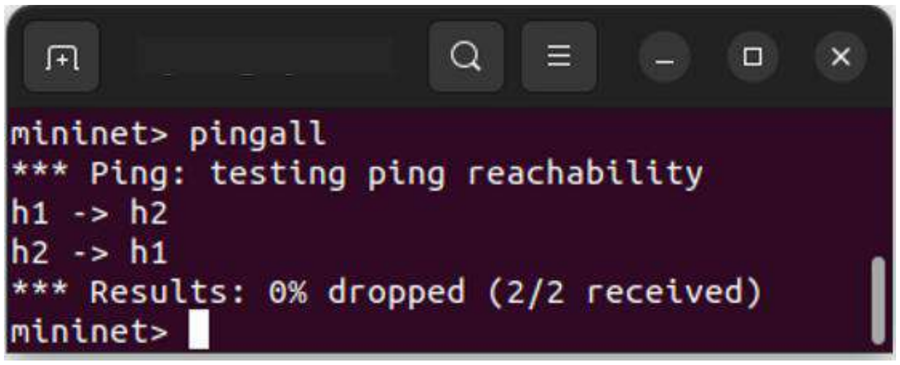
```
mininet> ports
s1 lo:0 s1-eth1:1 s1-eth2:2
mininet>
```

- The command to display the interfaces in the network is:



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

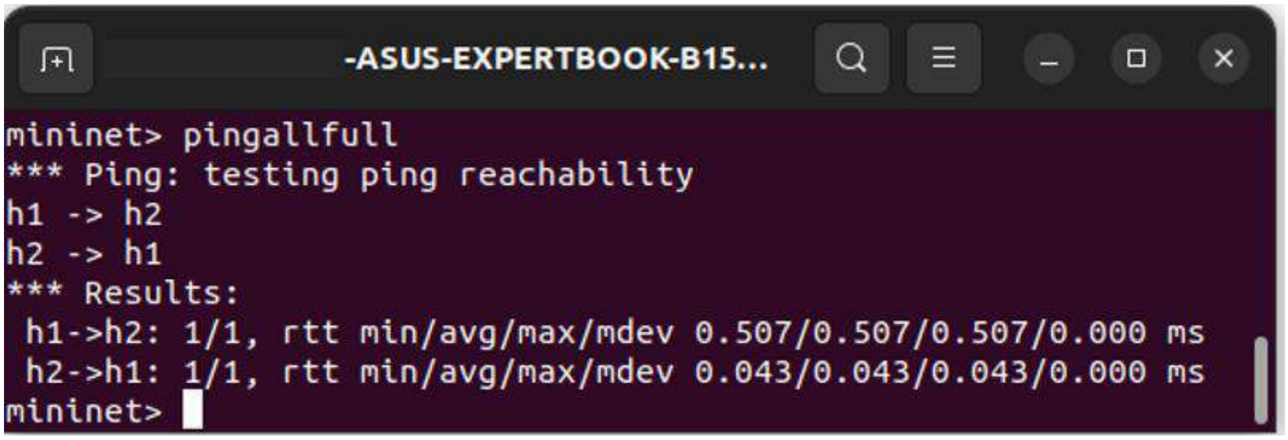
- The command to test the connectivity among hosts is:



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

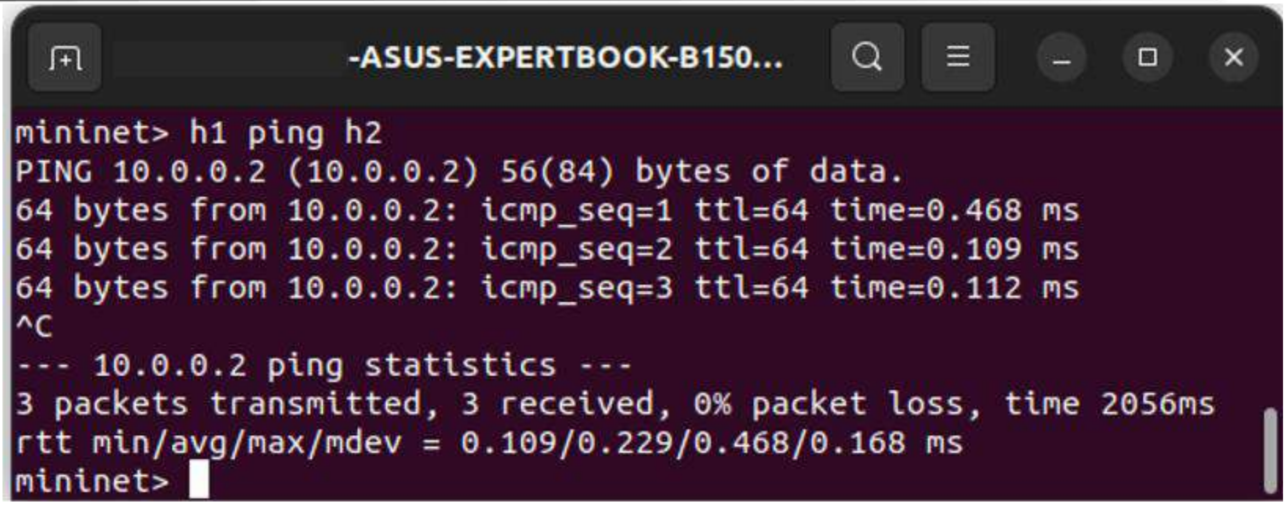
This command will make each host in the network ping every other host in the network. In the network that we have, *h1* will ping *h2*, and *h2* will ping *h1*.

- The command to test the connectivity among hosts by giving all details including RTT (round trip time) etc., is:



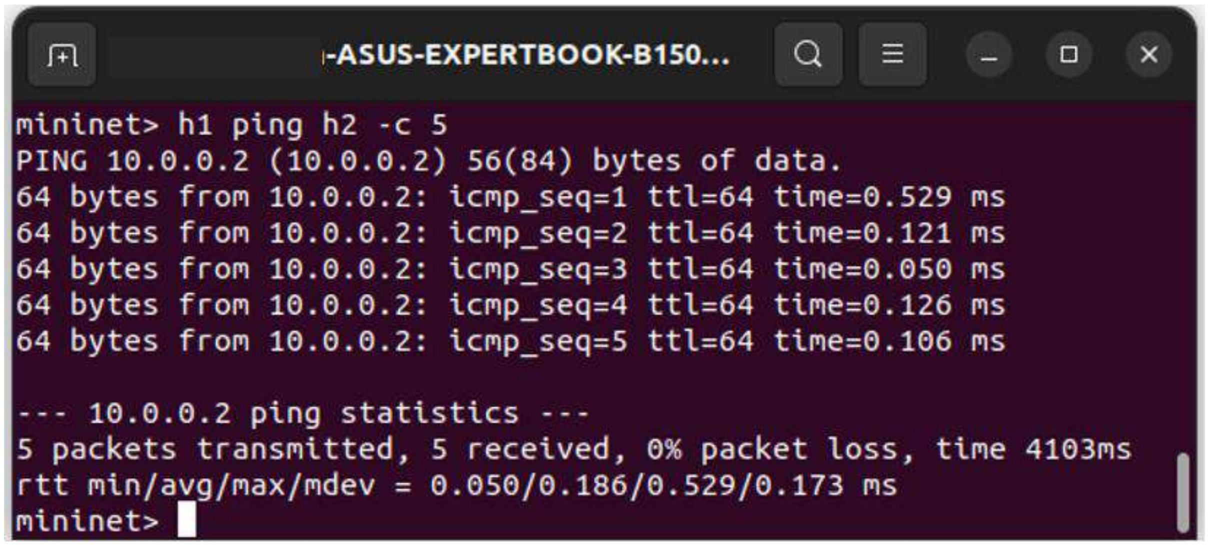
```
mininet> pingallfull
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results:
 h1->h2: 1/1, rtt min/avg/max/mdev 0.507/0.507/0.507/0.000 ms
 h2->h1: 1/1, rtt min/avg/max/mdev 0.043/0.043/0.043/0.000 ms
mininet>
```

- The command to ping a specific host to a targeted host is: (undergoes to execute infinite pings, to stop click CTRL + C.



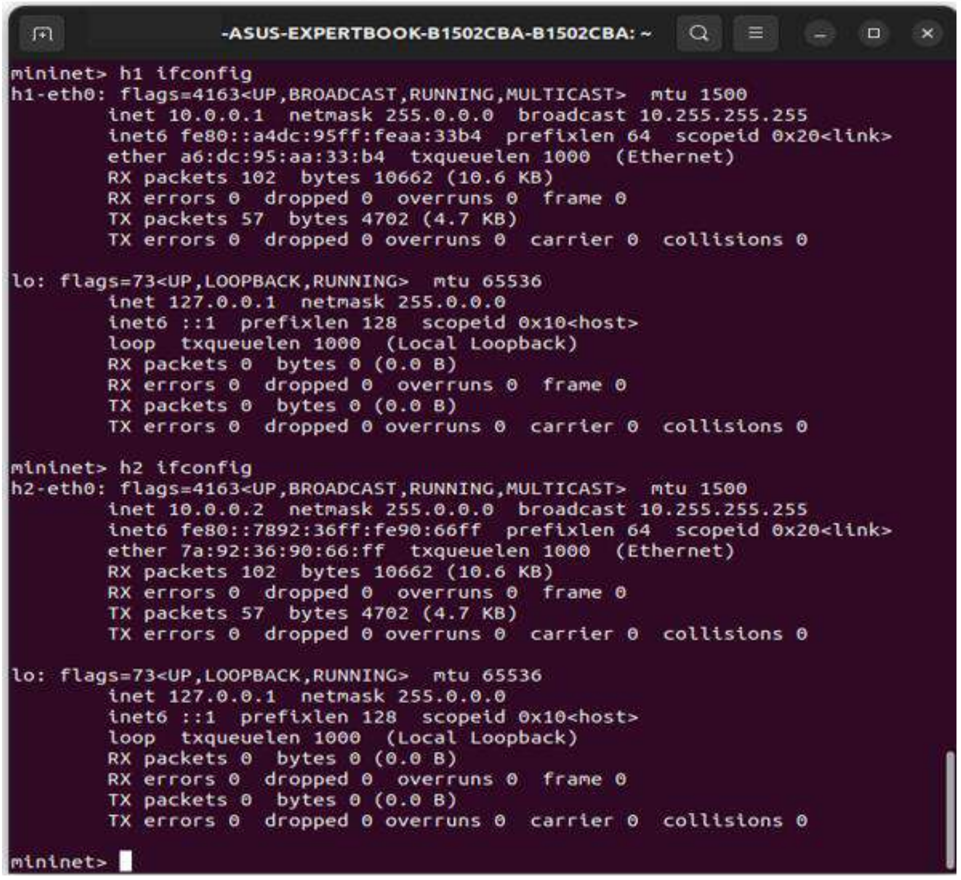
```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.468 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.109 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.112 ms
^C
--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2056ms
rtt min/avg/max/mdev = 0.109/0.229/0.468/0.168 ms
mininet>
```

- The command to ping a specific host to a targeted host with specific number of pings is:



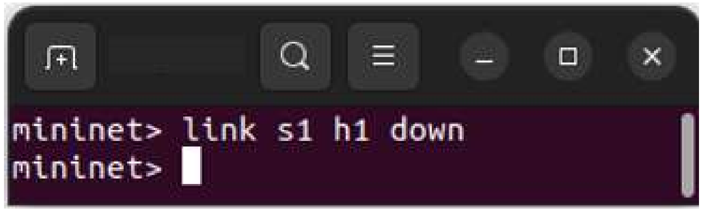
```
mininet> h1 ping h2 -c 5
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.529 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.121 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.050 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.126 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.106 ms

--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4103ms
rtt min/avg/max/mdev = 0.050/0.186/0.529/0.173 ms
mininet>
```

- The command to display the address information of the nodes is:



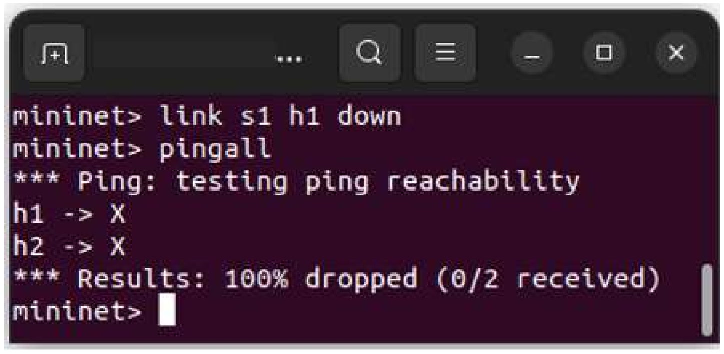
```
mininet> h1 ifconfig
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.1  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::a4dc:95ff:feaa:33b4  prefixlen 64  scopeid 0x20<link>
        ether a6:dc:95:aa:33:b4  txqueuelen 1000  (Ethernet)
        RX packets 102  bytes 10662 (10.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 57  bytes 4702 (4.7 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> h2 ifconfig
h2-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.2  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::7892:36ff:fe90:66ff  prefixlen 64  scopeid 0x20<link>
        ether 7a:92:36:90:66:ff  txqueuelen 1000  (Ethernet)
        RX packets 102  bytes 10662 (10.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 57  bytes 4702 (4.7 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>
```

- The command to down a link is:



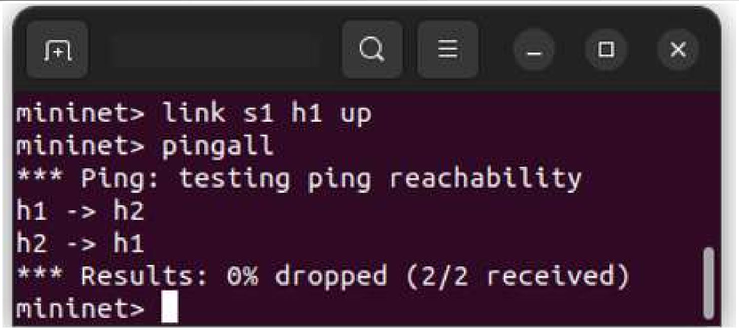
```
mininet> link s1 h1 down
mininet>
```

Ping now fails to connect:



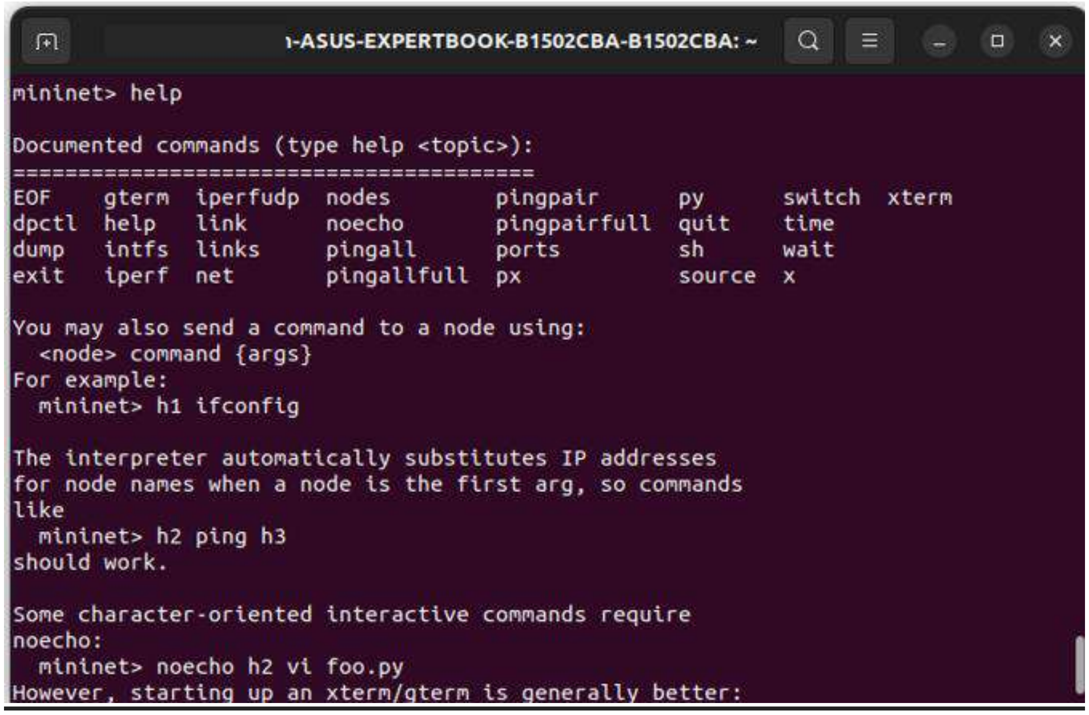
```
mininet> link s1 h1 down
mininet> pingall
*** Ping: testing ping reachability
h1 -> X
h2 -> X
*** Results: 100% dropped (0/2 received)
mininet>
```

- The command to activate a link is:



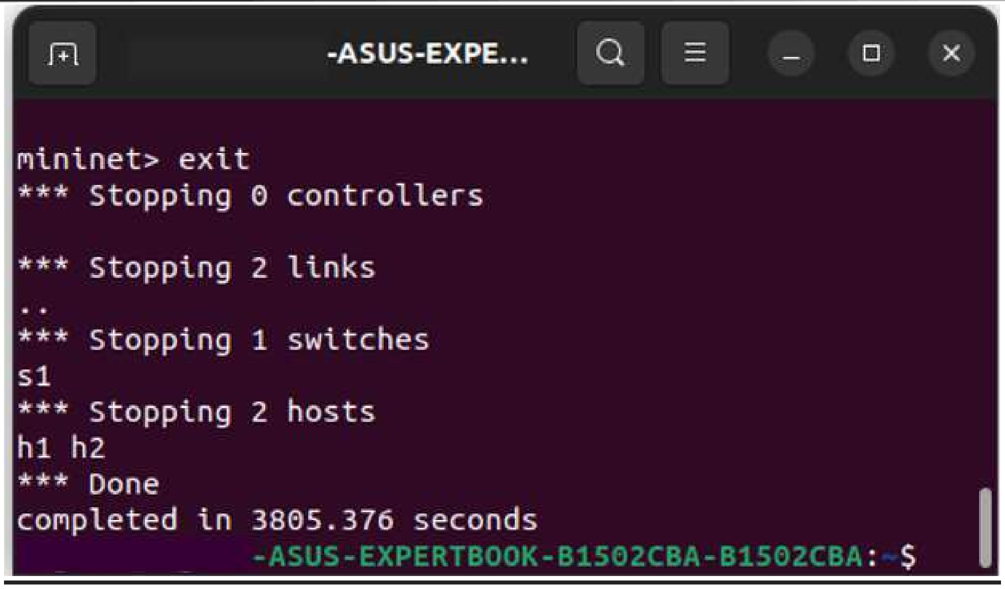
```
mininet> link s1 h1 up
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet> 
```

- The command Help:



```
mininet> help

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

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:
```

- The command Exit:



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

*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 3805.376 seconds
             -ASUS-EXPERTBOOK-B1502CBA-B1502CBA:~$
```

CONCLUSION:

We have successfully simulated Software Defined Network using Mininet.