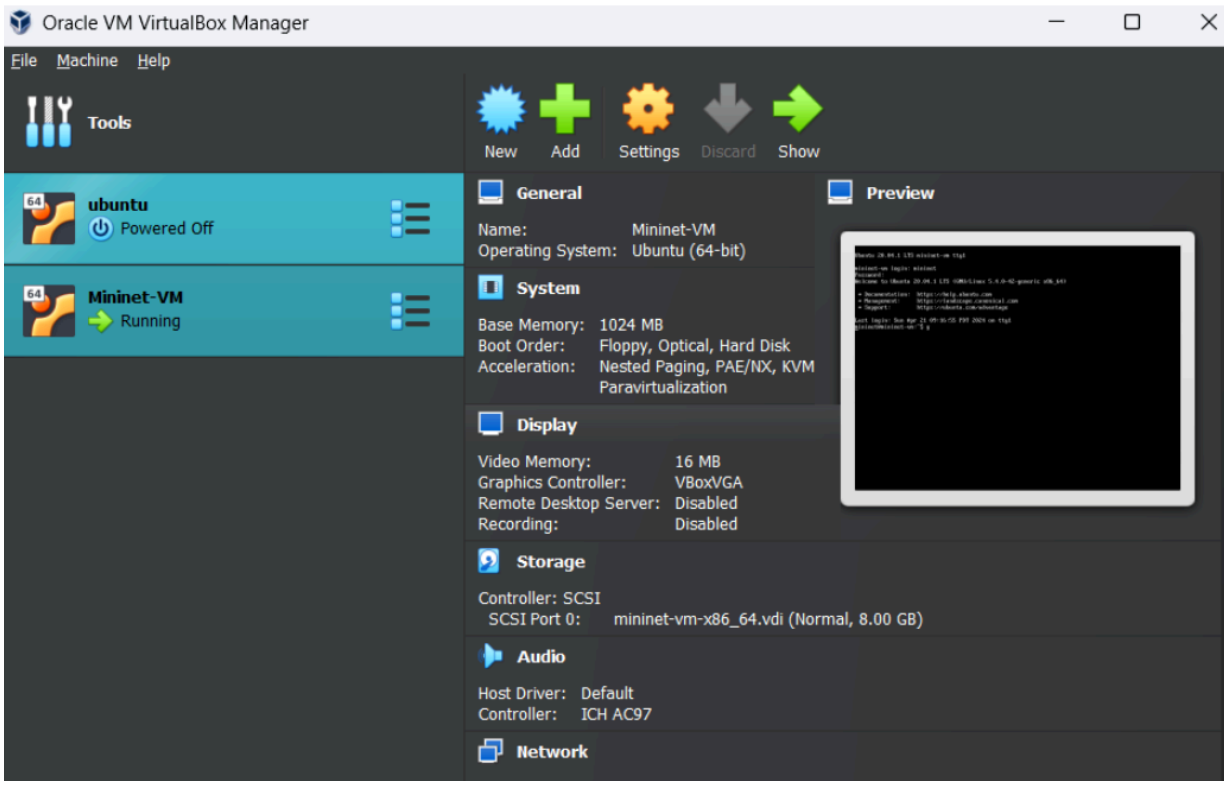
Name:	KEERTI MEHTA
UID:	2022300046
Class:	COMPS-A
Batch:	C
Experiment No:	10

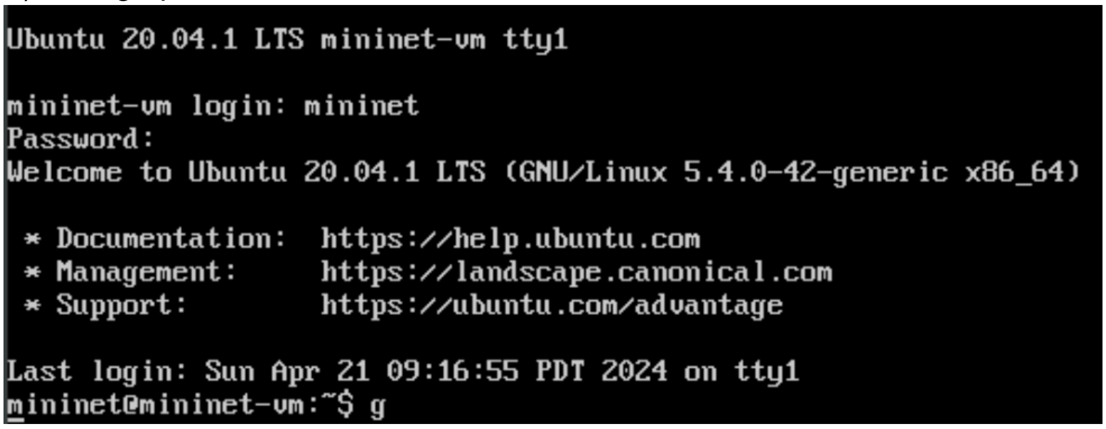
AIM:	Setting up a Virtual Network using Mininet.
OBJECTIVE:	The objective of this lab exercise is to create a realistic virtual network using Mininet, a tool for emulating network environments. By the end of this exercise, students should be able to set up a virtual network, run real kernel, switch, and application code, and understand the basic workflow of Mininet.
THEORY:	Setting up a virtual network using Mininet involves creating a simulated network environment on a single machine. Mininet leverages lightweight virtualization to emulate network nodes, switches, routers, and links, allowing developers to prototype and test network configurations without physical hardware. By defining the network topology through Python scripts or using Mininet's command-line interface, users can simulate complex network scenarios, evaluate performance, and troubleshoot network issues in a controlled environment. Mininet's flexibility and scalability make it a valuable tool for network research, education, and development, enabling rapid iteration and experimentation before deploying configurations on real-world networks.
Step 1: Downloading Mininet and Running it in Virtual Box	



```
Name:               Mininet-VM
Operating System:   Ubuntu (64-bit)

Base Memory:   1024 MB
Boot Order:    Floppy, Optical, Hard Disk
Acceleration:  Nested Paging, PAE/NX, KVM
               Paravirtualization

Video Memory:            16 MB
Graphics Controller:     VBoxVGA
Remote Desktop Server:   Disabled
Recording:               Disabled

Controller: SCSI
 SCSI Port 0:    mininet-vm-x86_64.vdi (Normal, 8.00 GB)

Host Driver:  Default
Controller:   ICH AC97
```

Step 2 : Installation and Setup

1)Setting up the Virtual Machine for Mininet.

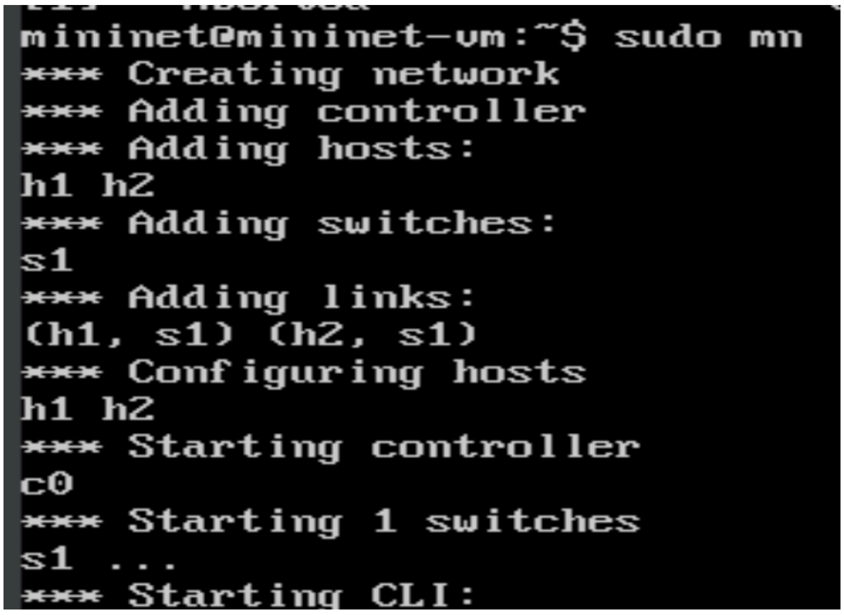


```
Ubuntu 20.04.1 LTS mininet-vm tty1

mininet-vm login: mininet
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-42-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

Last login: Sun Apr 21 09:16:55 PDT 2024 on tty1
mininet@mininet-vm:~$ g
```

- mininet -vm ttyl : This command suggests starting a Mininet virtual network simulation using a virtual machine and potentially with some custom or specific configuration indicated by "ttyl".

Step 3 : Sample Workflow

Interacting with hosts and switches

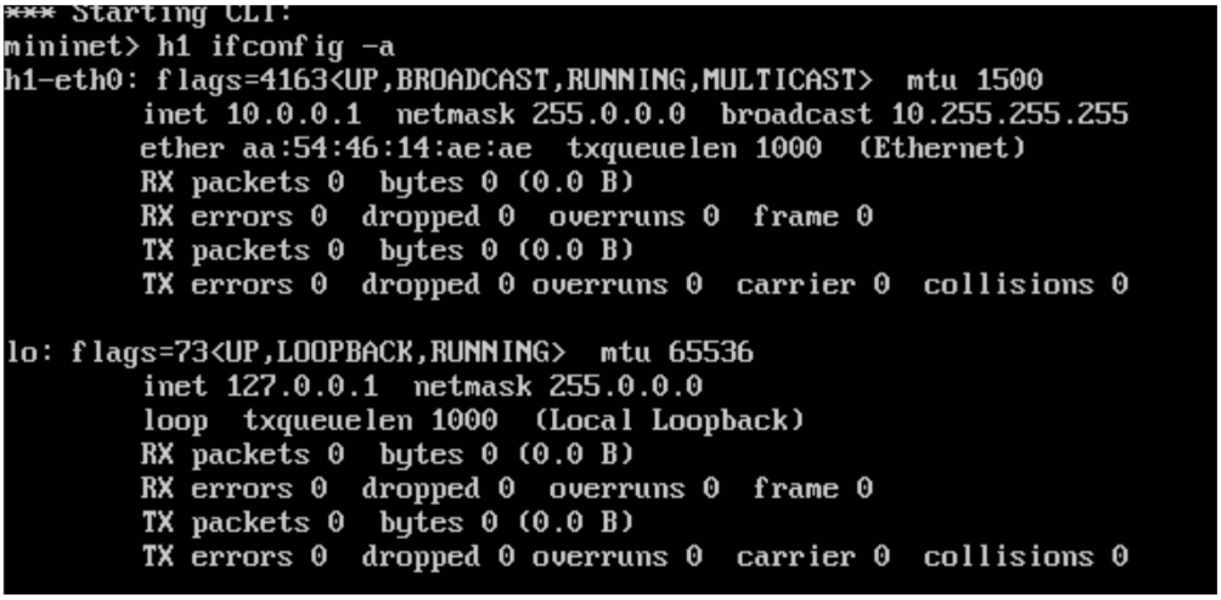


```
mininet@mininet-vm:~$ sudo mn
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
```

Step 4: Walkthrough

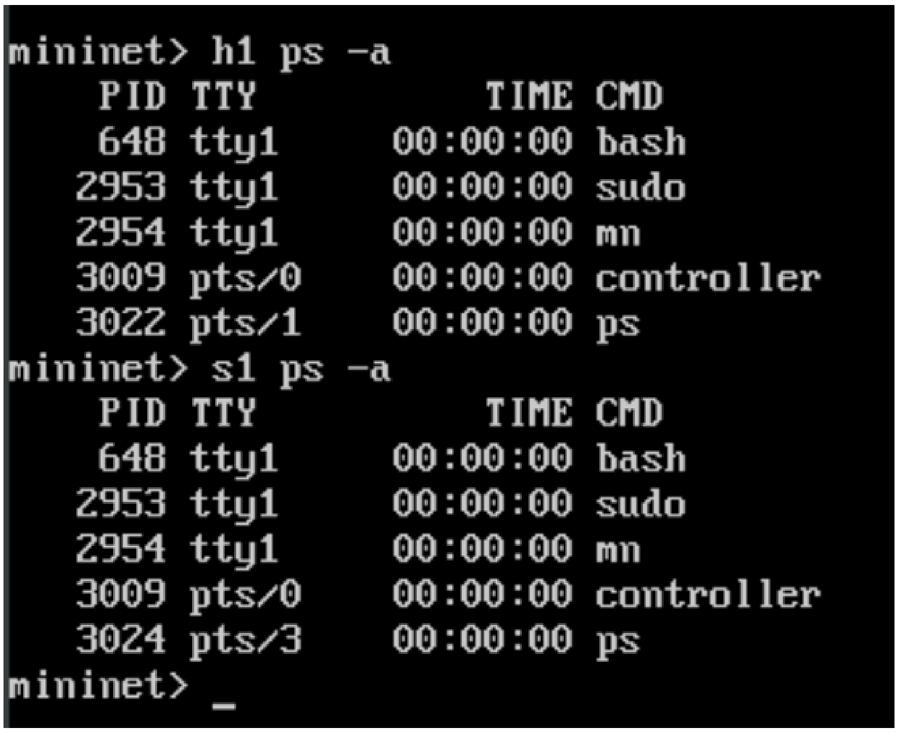
PART 1) EVERYDAY MININET USAGE

If the first string typed into the Mininet CLI is a host, switch or controller name, the command is executed on that node. Run a command on a host process:



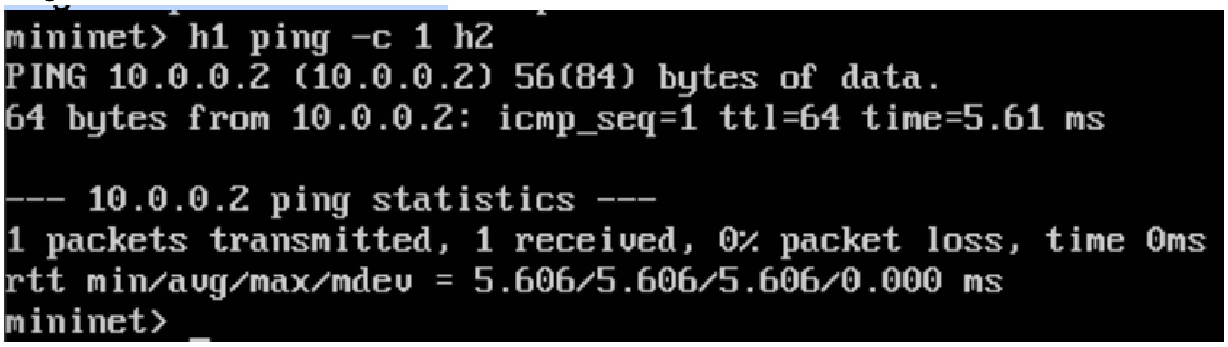
```
*** Starting CLI:
mininet> h1 ifconfig -a
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.1  netmask 255.0.0.0  broadcast 10.255.255.255
        ether aa:54:46:14:ae:ae  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        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
```

Note that only the network is virtualized; each host process sees the same set of processes and directories. For example, print the process list from a host process:



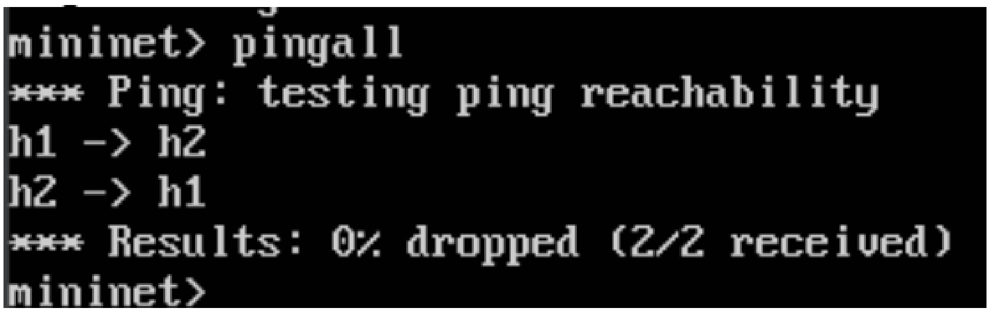
```
mininet> h1 ps -a
    PID TTY          TIME CMD
    648 tty1     00:00:00 bash
   2953 tty1     00:00:00 sudo
   2954 tty1     00:00:00 mn
   3009 pts/0    00:00:00 controller
   3022 pts/1    00:00:00 ps
mininet> s1 ps -a
    PID TTY          TIME CMD
    648 tty1     00:00:00 bash
   2953 tty1     00:00:00 sudo
   2954 tty1     00:00:00 mn
   3009 pts/0    00:00:00 controller
   3024 pts/3    00:00:00 ps
mininet> _
```

Ping from one host to another:



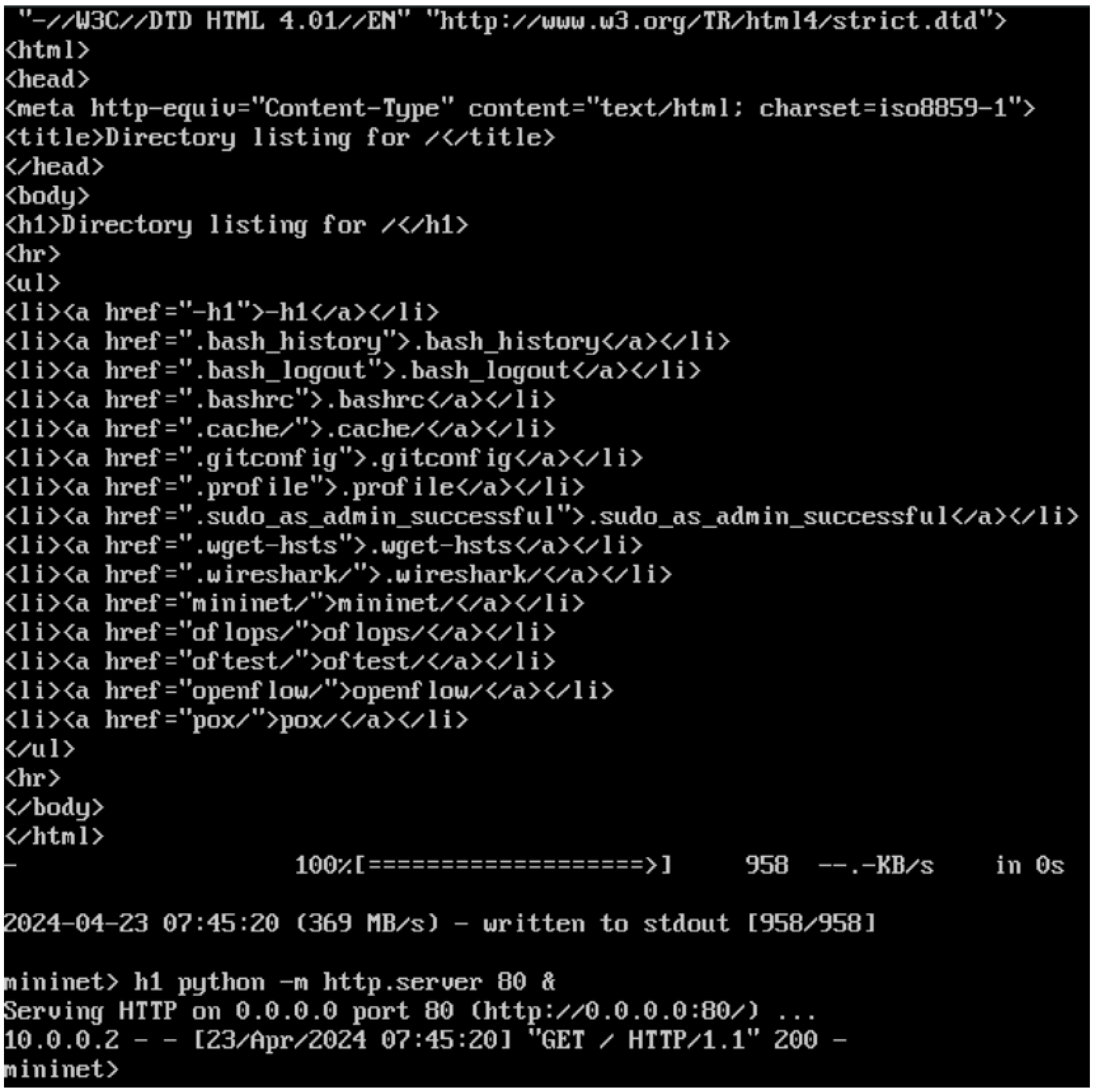
```
mininet> h1 ping -c 1 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=5.61 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 5.606/5.606/5.606/0.000 ms
mininet> _
```

An easier way to run this test is to use the Mininet CLI built-in pingall command, which does an all-pairs ping:



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

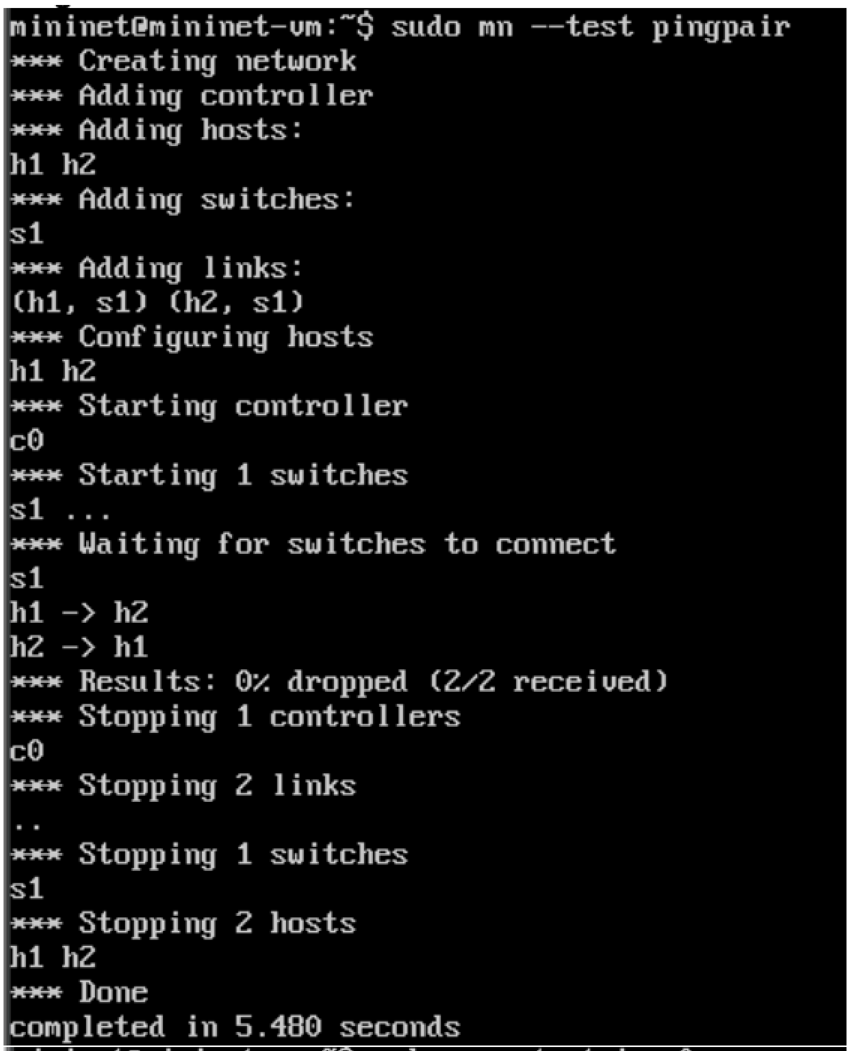
Running a simple web server and client:



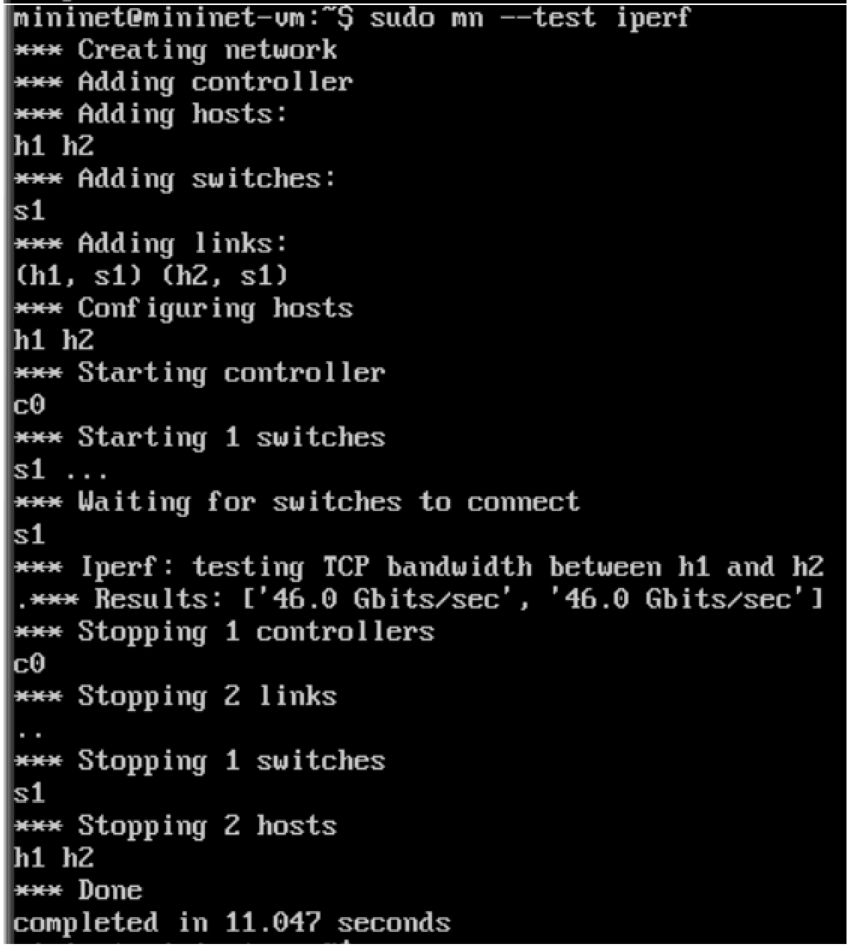
```
 "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso8859-1">
<title>Directory listing for /</title>
</head>
<body>
<h1>Directory listing for /</h1>
<hr>
<ul>
<li><a href="-h1">-h1</a></li>
<li><a href=".bash_history">.bash_history</a></li>
<li><a href=".bash_logout">.bash_logout</a></li>
<li><a href=".bashrc">.bashrc</a></li>
<li><a href=".cache/">.cache/</a></li>
<li><a href=".gitconfig">.gitconfig</a></li>
<li><a href=".profile">.profile</a></li>
<li><a href=".sudo_as_admin_successful">.sudo_as_admin_successful</a></li>
<li><a href=".wget-hsts">.wget-hsts</a></li>
<li><a href=".wireshark/">.wireshark/</a></li>
<li><a href="mininet/">mininet/</a></li>
<li><a href="oflops/">oflops/</a></li>
<li><a href="oftest/">oftest/</a></li>
<li><a href="openflow/">openflow/</a></li>
<li><a href="pox/">pox/</a></li>
</ul>
<hr>
</body>
</html>
-                    100%[===================>]     958  --.-KB/s    in 0s

2024-04-23 07:45:20 (369 MB/s) - written to stdout [958/958]

mininet> h1 python -m http.server 80 &
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.0.0.2 - - [23/Apr/2024 07:45:20] "GET / HTTP/1.1" 200 -
mininet>
```

PART 2) ADVANCED STARTUP OPTIONS

This command created a minimal topology, started up the OpenFlow reference controller, ran an all-pairs ping test, and tore down both the topology and the controller.

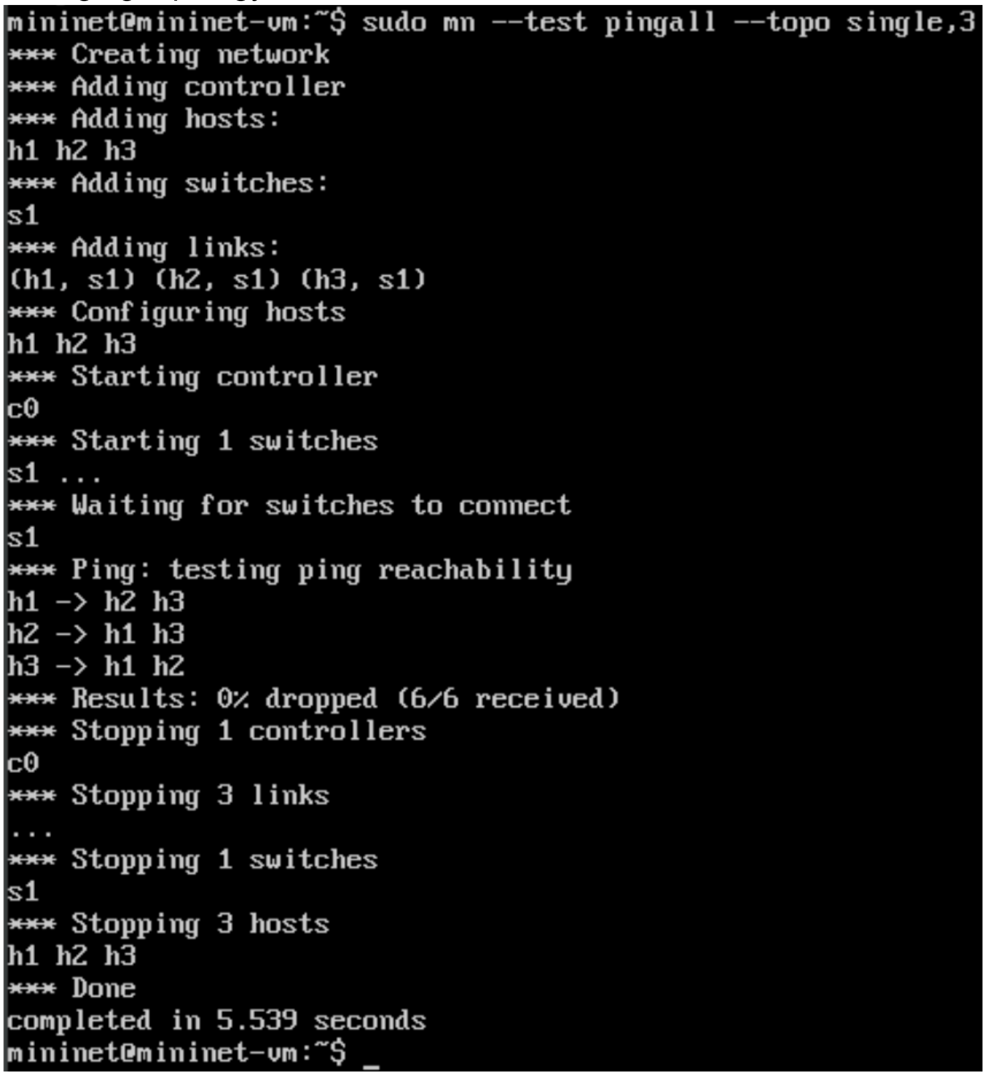


```
mininet@mininet-vm:~$ sudo mn --test pingpair
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Waiting for switches to connect
s1
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 5.480 seconds
```



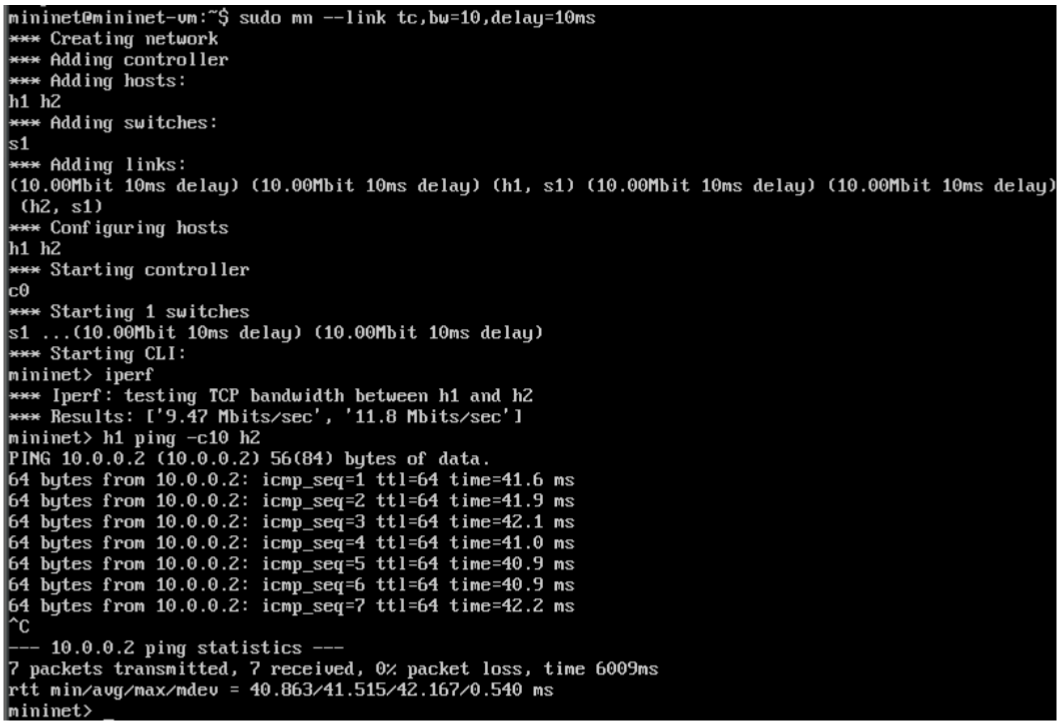
```
mininet@mininet-vm:~$ sudo mn --test iperf
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Waiting for switches to connect
s1
*** Iperf: testing TCP bandwidth between h1 and h2
.*** Results: ['46.0 Gbits/sec', '46.0 Gbits/sec']
*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 11.047 seconds
```

Changing topology and size:



```
mininet@mininet-vm:~$ sudo mn --test pingall --topo single,3
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Waiting for switches to connect
s1
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
*** Stopping 1 controllers
c0
*** Stopping 3 links
...
*** Stopping 1 switches
s1
*** Stopping 3 hosts
h1 h2 h3
*** Done
completed in 5.539 seconds
mininet@mininet-vm:~$ _
```

Link variations can be done as follows:



```
mininet@mininet-vm:~$ sudo mn --link tc,bw=10,delay=10ms
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(10.00Mbit 10ms delay) (10.00Mbit 10ms delay) (h1, s1) (10.00Mbit 10ms delay) (10.00Mbit 10ms delay)
 (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...(10.00Mbit 10ms delay) (10.00Mbit 10ms delay)
*** Starting CLI:
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['9.47 Mbits/sec', '11.8 Mbits/sec']
mininet> h1 ping -c10 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=41.6 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=41.9 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=42.1 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=41.0 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=40.9 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=40.9 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=42.2 ms
^C
--- 10.0.0.2 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6009ms
rtt min/avg/max/mdev = 40.863/41.515/42.167/0.540 ms
mininet> _
```

CONCLUSION:	I learnt about the capability of Mininet in emulating realistic virtual networks.I also understood setting up of a virtual network, executing real kernel, switch, and application code, and fundamental workflow of Mininet. I also learnt to simulate and analyze complex network scenarios in a controlled environment.