Network Simulation Assignment

Q). Perform basic network simulation on any of the below mentioned tools.
(NS3, ONE, OMNET++, MiniNet, Scilab, Octave)

MiniNet:

Procedure:

1.) Install MiniNet :

i. Installation of Virtual Box/Xming/putty.

- Virtual Box :-https://www.virtualbox.org/wiki/Downloads
- Putty:- https://www.chiark.greenend.uk/putty/latest.html
- Xming:-https://sourceforge.net/projects/xming/download

ii. Download/extract the mininet image

- Download the MiniNet virtual machine image from https://github.com/mininet/mininet/releases/
- This file is compressed zip file archive containg two files so,after downloading it,decompress it and save it to hard drive

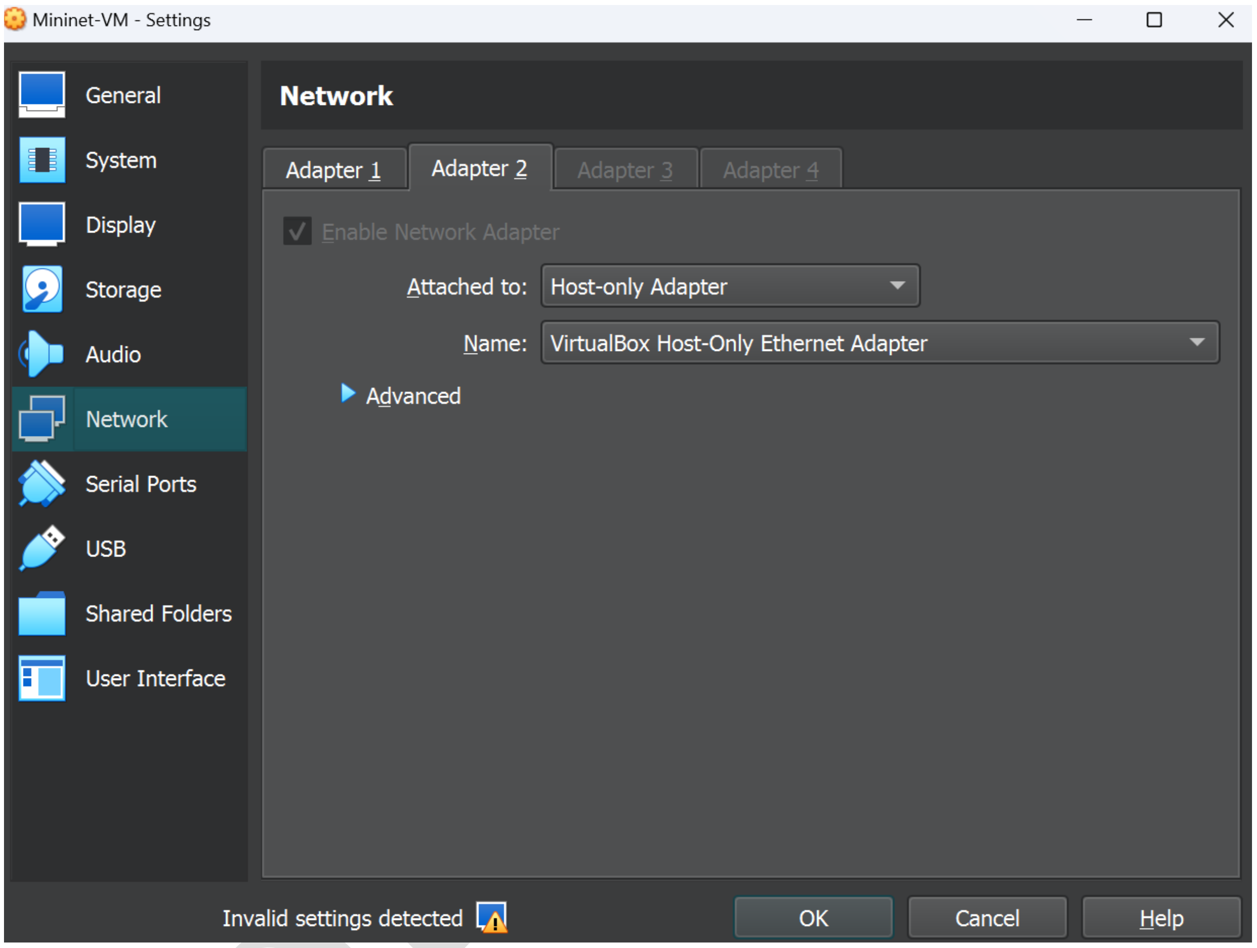
iii. Import the virtual machine into virtualbox

- Open VirtualBox and click on the "Import Appliance" option in the "File" menu.
- In the "Import Virtual Appliance" window, click on the "Choose" button and select the Mininet image file you downloaded.
- Click on the "Next" button and follow the instructions to configure the virtual machine settings (e.g., memory size, network settings, etc.) as per your requirements.
- Once you have configured the settings, click on the "Import" button to start the import process

iv. Add a host-only network interface to your VM :

- click `setting`

- click `Network`, don't change the Adapter 1(it will be initial ethernet 0). click Adapter 2 and check `Enable Network Adapter`. Then choose host-only like below:



Enable Network Adapter ✓

Attached to: Host-only Adapter

Name: VirtualBox Host-Only Ethernet Adapter

- click "ok"

v. start the mininet virtual machine and Log in to the VM, using the following name and password:

```
mininet-vm login: mininet Password: mininet
```

vi. Configure a new virtual ethernet interface on the virtual machine

- Run the ifconfig -a command to see all the interface available on Mininet virtual machine.
- Run sudo dhclient eth1 to assign ip address from the dhcp server that is running on the host-only interface connected to eth1
- Run the ifconfig command and you will see that now eth1 has an ip adress

vii. Acces VM via SSH

- Run the xming
- Run the putty app and click on ssh->x11->enable x11

- Click on session and on hast name section add ip address of eth0 of virtual box (for checking the ip address run ifconfig on mininet vm box)

2.) As Mininet provides both a command-line interface (CLI) and a graphical user interface (GUI) for creating and managing network topologies.

Lets first create a network using mininet gui (i.e MiniEdit)

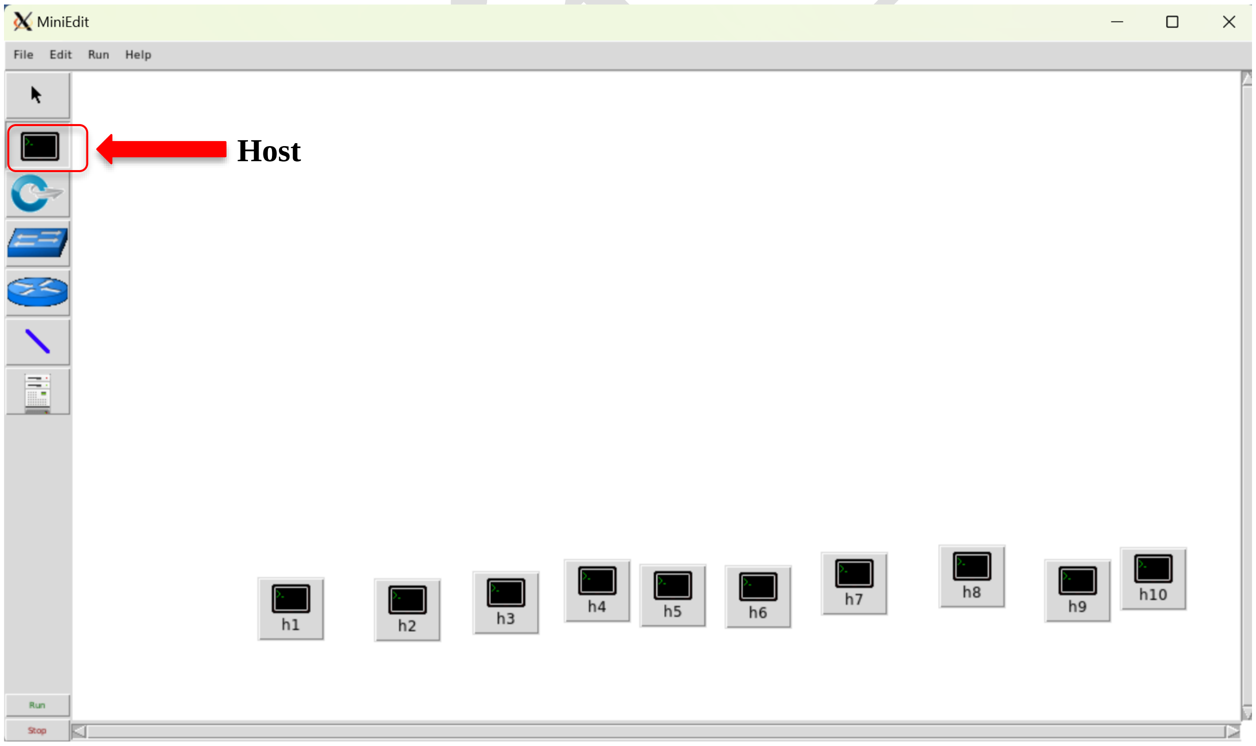
Procedure :

a) The MiniEdit script is located in Mininet's *examples* folder. To run MiniEdit, execute the command:

```
✓  $ sudo ~/mininet/examples/miniedit.py
```

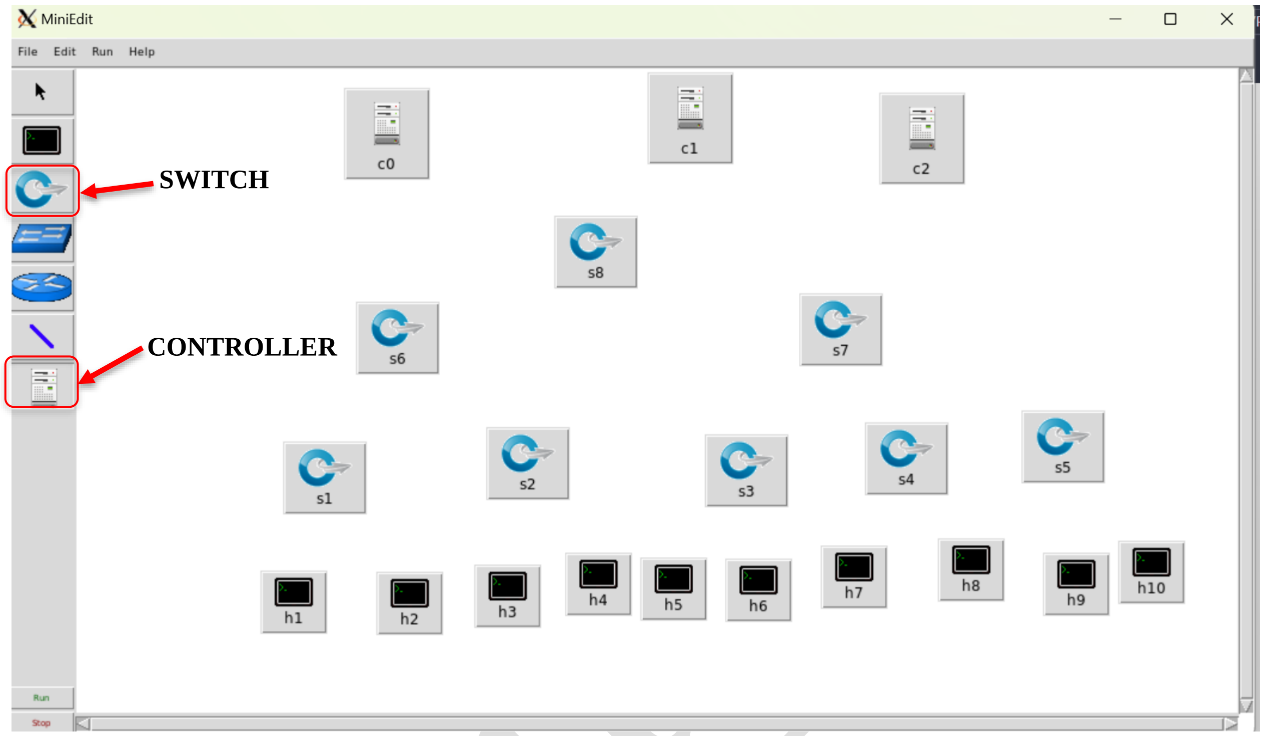
(Mininet needs to run with *root* privileges so we started MiniEdit using the *sudo* command.)

b) First we will add some hosts to network scenario. Click on the *Host* icon, which is shown below :

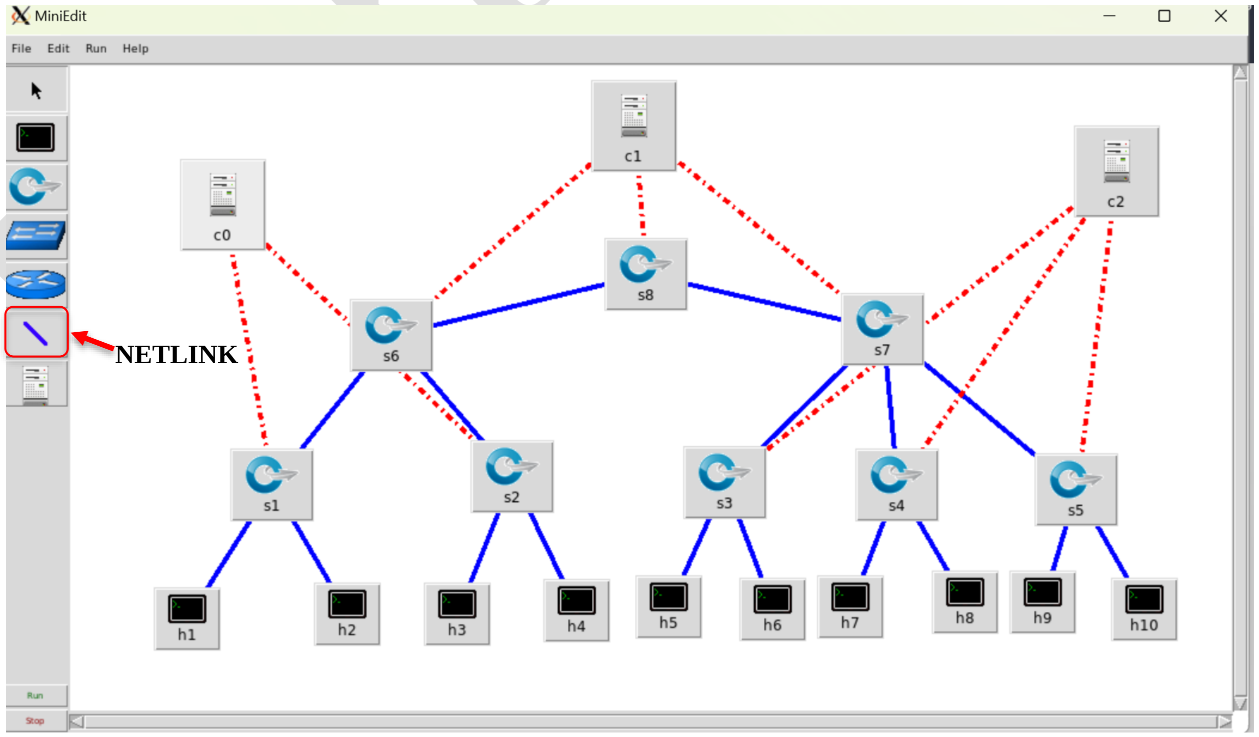


As long as the *Host* tool is active, you can add more hosts. Keep clicking at each spot on the canvas where you want a host to appear. In this example, we will add ten hosts.

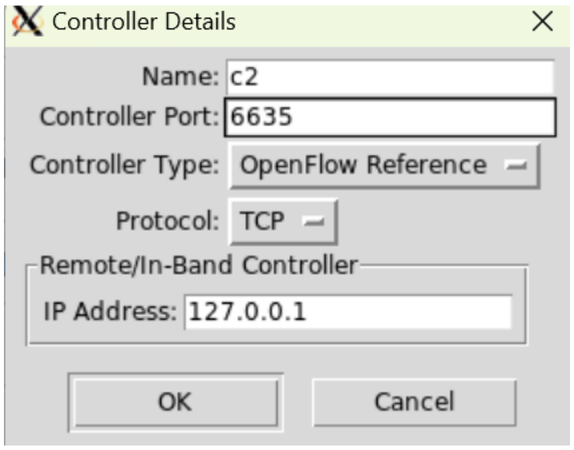
c) Add eight switches and three controllers using the same method: Click on the *Switch* tool and add switches, then click on the *Controller* tool and add controllers.



d) Next, add links between the nodes on the canvas. Click on the *NetLink* tool, then click on a node and drag the link over to another node.

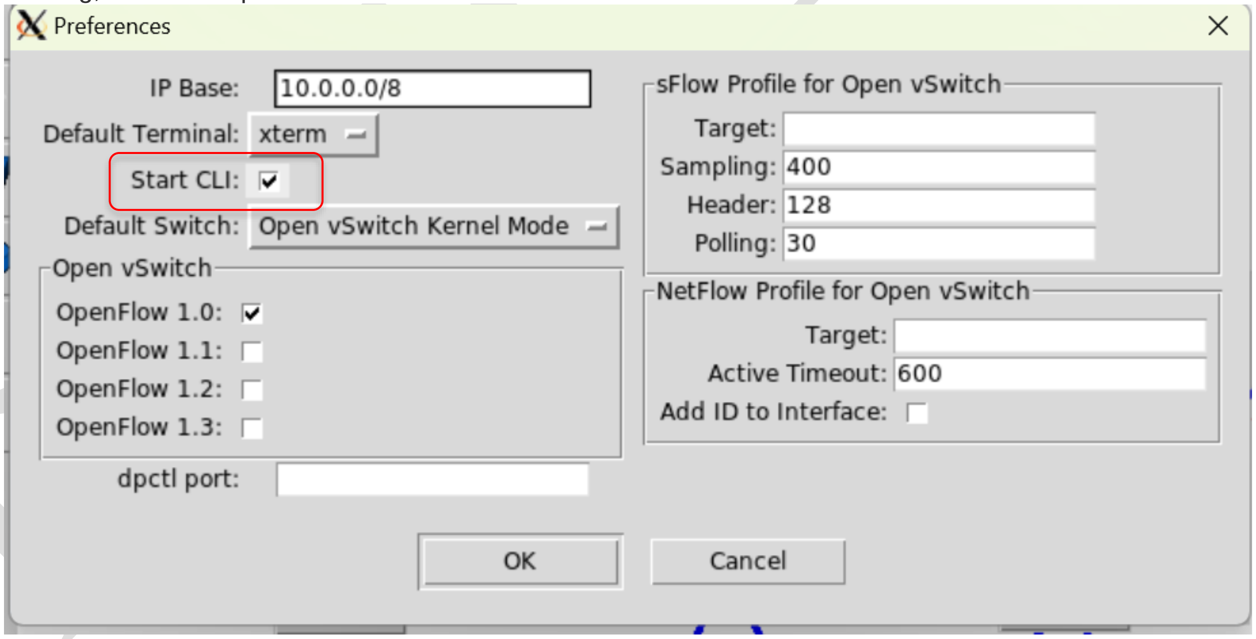


e) **Configure the controllers:-**We have three controllers.Right-click on each controller and select *Properties* from the menu that appears. The default port number for each controller is 6633. Change this so the port numbers used by controllers *c0*, *c1*, and *c2* are 6633, 6634, and 6635, respectively.



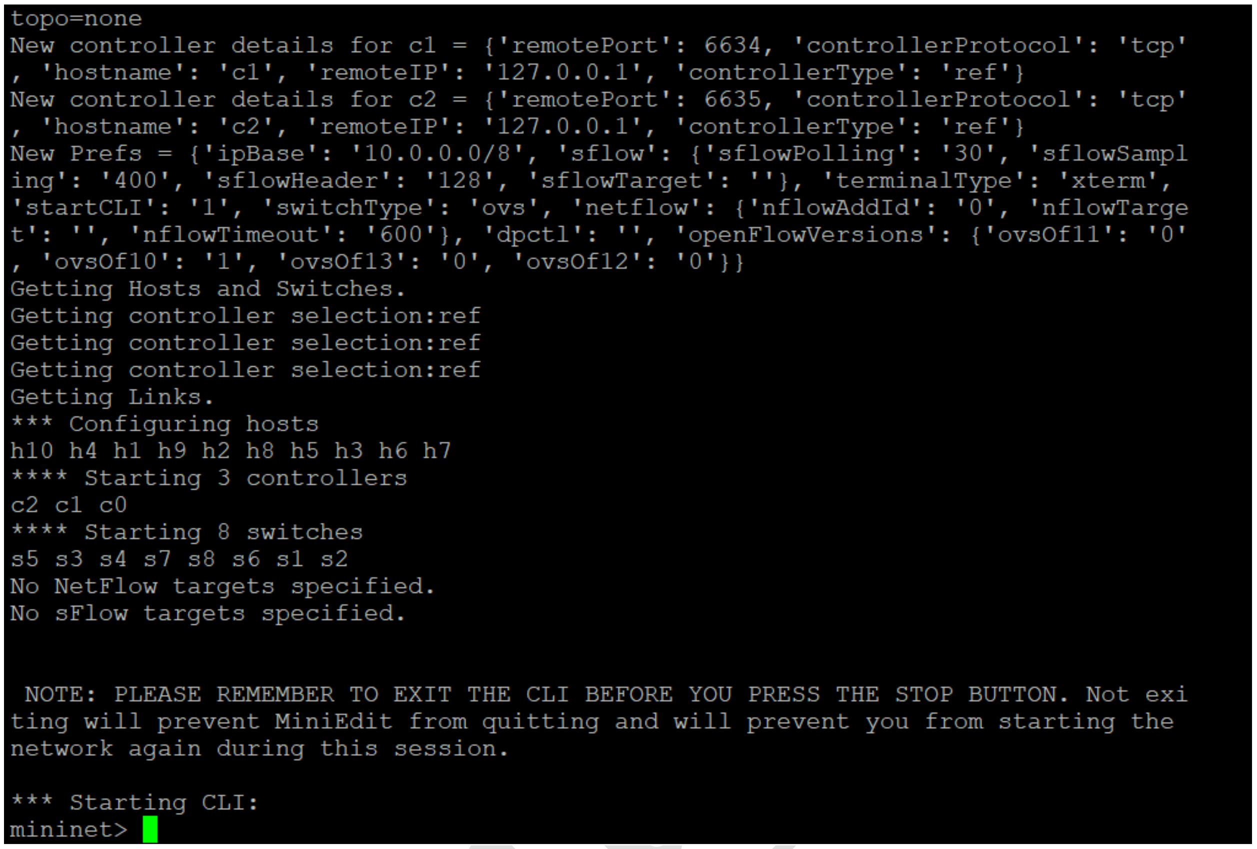
Name: c2
Controller Port: 6635
Controller Type: OpenFlow Reference
Protocol: TCP
Remote/In-Band Controller
IP Address: 127.0.0.1

f) By default, the MiniEdit console window does not give the user access to the Mininet command line interface. If you want to be able to use the Mininet CLI when a simulation is running,Go to edit->preferences then check the *Start CLI* box.



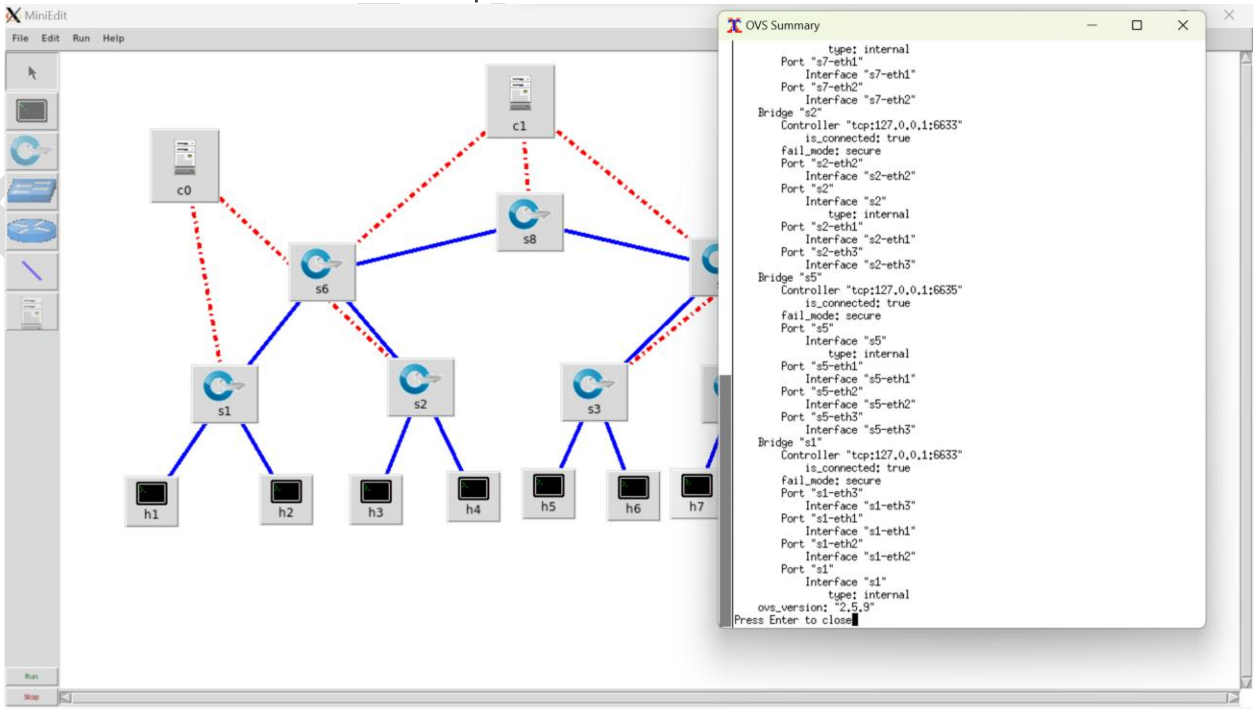
IP Base: 10.0.0.0/8
Default Terminal: xterm
Start CLI: ✓
Default Switch: Open vSwitch Kernel Mode
Open vSwitch
OpenFlow 1.0: ✓
OpenFlow 1.1: ☐
OpenFlow 1.2: ☐
OpenFlow 1.3: ☐
dpctl port:
sFlow Profile for Open vSwitch
Target:
Sampling: 400
Header: 128
Polling: 30
NetFlow Profile for Open vSwitch
Target:
Active Timeout: 600
Add ID to Interface: ☐

g) Run the simulation : -To start the simulation scenario, click the *Run* button on the MiniEdit GUI. In the terminal window from which you started MiniEdit, you will see some messages showing the progress of the simulation startup and then the Miniedit CLI prompt.



```
topo=none
New controller details for c1 = {'remotePort': 6634, 'controllerProtocol': 'tcp'
, 'hostname': 'c1', 'remoteIP': '127.0.0.1', 'controllerType': 'ref'}
New controller details for c2 = {'remotePort': 6635, 'controllerProtocol': 'tcp'
, 'hostname': 'c2', 'remoteIP': '127.0.0.1', 'controllerType': 'ref'}
New Prefs = {'ipBase': '10.0.0.0/8', 'sflow': {'sflowPolling': '30', 'sflowSampl
ing': '400', 'sflowHeader': '128', 'sflowTarget': ''}, 'terminalType': 'xterm',
'startCLI': '1', 'switchType': 'ovs', 'netflow': {'nflowAddId': '0', 'nflowTarge
t': '', 'nflowTimeout': '600'}, 'dpctl': '', 'openFlowVersions': {'ovsOf11': '0'
, 'ovsOf10': '1', 'ovsOf13': '0', 'ovsOf12': '0'}}
Getting Hosts and Switches.
Getting controller selection:ref
Getting controller selection:ref
Getting controller selection:ref
Getting Links.
*** Configuring hosts
h10 h4 h1 h9 h2 h8 h5 h3 h6 h7
**** Starting 3 controllers
c2 c1 c0
**** Starting 8 switches
s5 s3 s4 s7 s8 s6 s1 s2
No NetFlow targets specified.
No sFlow targets specified.


 NOTE: PLEASE REMEMBER TO EXIT THE CLI BEFORE YOU PRESS THE STOP BUTTON. Not exi
ting will prevent MiniEdit from quitting and will prevent you from starting the
network again during this session.

*** Starting CLI:
mininet> 
```

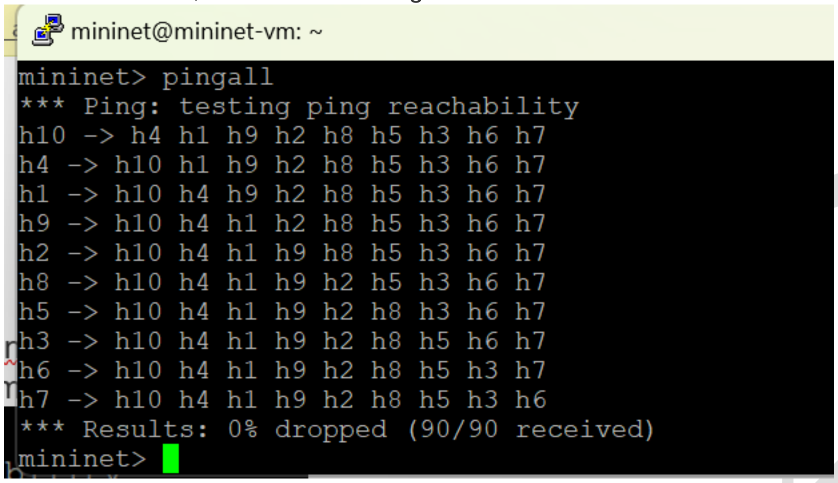
h) First, check the switch configurations in the network simulation to verify that everything is set up correctly. You can run the MiniEdit menu command, *Run → Show OVS Summary* to see an listing of switch configurations. In this case, we can verify that each switch is listening to the correct controller on the correct port.



```
            type: internal
        Port "s7-eth1"
            Interface "s7-eth1"
        Port "s7-eth2"
            Interface "s7-eth2"
    Bridge "s2"
        Controller "tcp:127.0.0.1:6633"
            is_connected: true
        fail_mode: secure
        Port "s2-eth2"
            Interface "s2-eth2"
        Port "s2"
            Interface "s2"
                type: internal
        Port "s2-eth1"
            Interface "s2-eth1"
        Port "s2-eth3"
            Interface "s2-eth3"
    Bridge "s5"
        Controller "tcp:127.0.0.1:6635"
            is_connected: true
        fail_mode: secure
        Port "s5"
            Interface "s5"
                type: internal
        Port "s5-eth1"
            Interface "s5-eth1"
        Port "s5-eth2"
            Interface "s5-eth2"
        Port "s5-eth3"
            Interface "s5-eth3"
    Bridge "s1"
        Controller "tcp:127.0.0.1:6633"
            is_connected: true
        fail_mode: secure
        Port "s1-eth3"
            Interface "s1-eth3"
        Port "s1-eth1"
            Interface "s1-eth1"
        Port "s1-eth2"
            Interface "s1-eth2"
        Port "s1"
            Interface "s1"
                type: internal
    ovs_version: "2.5.9"
Press Enter to close
```

i) Test the connectivity : -

Run the *pingall* command to test the connectivty of the create network . On the MiniEdit console window, enter the following command:

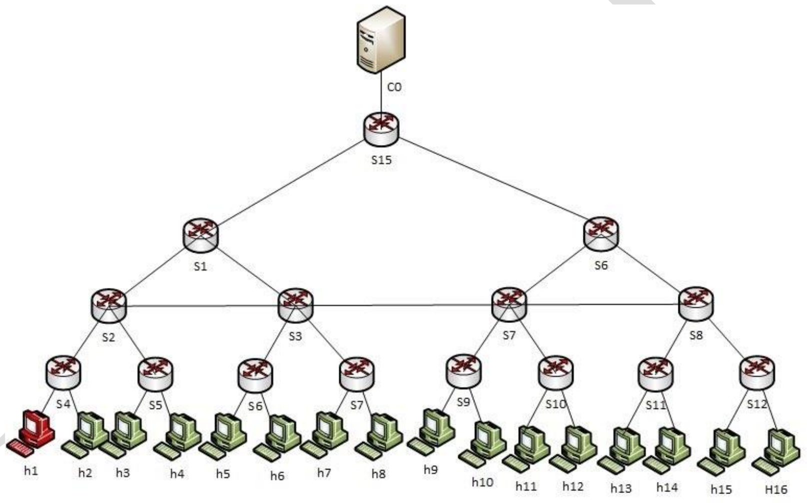


mininet@mininet-vm: ~

```
mininet> pingall
*** Ping: testing ping reachability
h10 -> h4 h1 h9 h2 h8 h5 h3 h6 h7
h4 -> h10 h1 h9 h2 h8 h5 h3 h6 h7
h1 -> h10 h4 h9 h2 h8 h5 h3 h6 h7
h9 -> h10 h4 h1 h2 h8 h5 h3 h6 h7
h2 -> h10 h4 h1 h9 h8 h5 h3 h6 h7
h8 -> h10 h4 h1 h9 h2 h5 h3 h6 h7
h5 -> h10 h4 h1 h9 h2 h8 h3 h6 h7
h3 -> h10 h4 h1 h9 h2 h8 h5 h6 h7
h6 -> h10 h4 h1 h9 h2 h8 h5 h3 h7
h7 -> h10 h4 h1 h9 h2 h8 h5 h3 h6
*** Results: 0% dropped (90/90 received)
mininet>
```

(0% packets dropped means our connection is successful)

Now, Lets create a network using mininet cli interface.



We are going to create this topology using cli interface

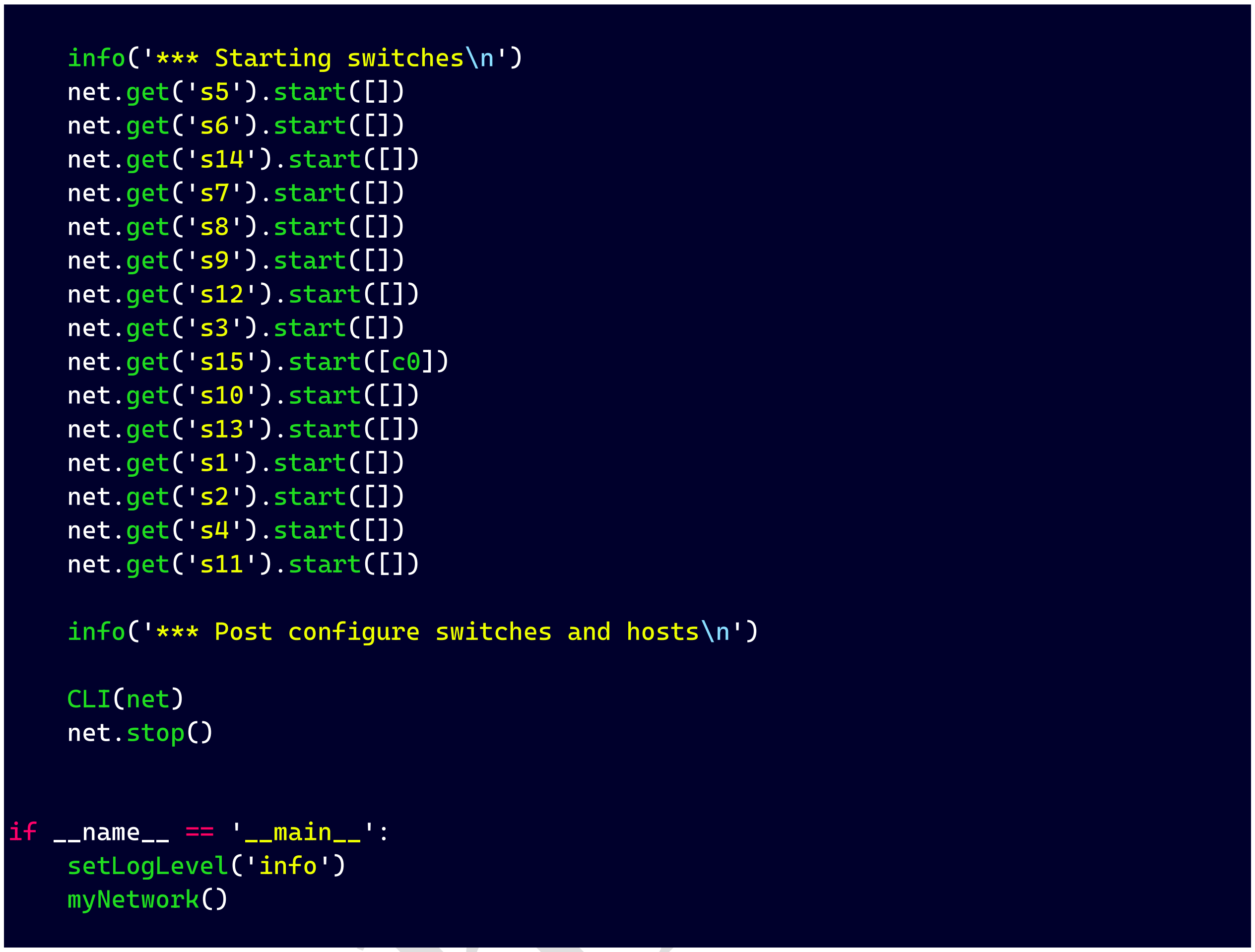
Procedure :-

i) First create a python file on mininet by using this command :

```
✓  vi ~/minint/custom/rohit.py
```

ii)then write the python code as shown below :

```python
from mininet.net import Mininet
from mininet.node import Controller, RemoteController, OVSController
from mininet.node import CPULimitedHost, Host, Node
from mininet.node import OVSKernelSwitch, UserSwitch
from mininet.node import IVSSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel, info
from mininet.link import TCLink, Intf
from subprocess import call


def myNetwork():

    net = Mininet(topo=None,
                  build=False,
                  ipBase='10.0.0.0/8')

    info('*** Adding controller\n')
    c0 = net.addController(name='c0',
                           controller=Controller,
                           protocol='tcp',
                           port=6633)

    info('*** Add switches\n')
    s5 = net.addSwitch('s5', cls=OVSKernelSwitch)
    s6 = net.addSwitch('s6', cls=OVSKernelSwitch)
    s14 = net.addSwitch('s14', cls=OVSKernelSwitch)
    s7 = net.addSwitch('s7', cls=OVSKernelSwitch)
    s8 = net.addSwitch('s8', cls=OVSKernelSwitch)
    s9 = net.addSwitch('s9', cls=OVSKernelSwitch)
    s12 = net.addSwitch('s12', cls=OVSKernelSwitch)
    s3 = net.addSwitch('s3', cls=OVSKernelSwitch)
    s15 = net.addSwitch('s15', cls=OVSKernelSwitch)
    s10 = net.addSwitch('s10', cls=OVSKernelSwitch)
    s13 = net.addSwitch('s13', cls=OVSKernelSwitch)
    s1 = net.addSwitch('s1', cls=OVSKernelSwitch)
    s2 = net.addSwitch('s2', cls=OVSKernelSwitch)
    s4 = net.addSwitch('s4', cls=OVSKernelSwitch)
    s11 = net.addSwitch('s11', cls=OVSKernelSwitch)

    info('*** Add hosts\n')
    h6 = net.addHost('h6', cls=Host, ip='10.0.0.6', defaultRoute=None)
    h7 = net.addHost('h7', cls=Host, ip='10.0.0.7', defaultRoute=None)
    h8 = net.addHost('h8', cls=Host, ip='10.0.0.8', defaultRoute=None)
    h10 = net.addHost('h10', cls=Host, ip='10.0.0.10', defaultRoute=None)
    h14 = net.addHost('h14', cls=Host, ip='10.0.0.14', defaultRoute=None)
    h12 = net.addHost('h12', cls=Host, ip='10.0.0.12', defaultRoute=None)
    h13 = net.addHost('h13', cls=Host, ip='10.0.0.13', defaultRoute=None)
    h16 = net.addHost('h16', cls=Host, ip='10.0.0.16', defaultRoute=None)
    h11 = net.addHost('h11', cls=Host, ip='10.0.0.11', defaultRoute=None)
    h15 = net.addHost('h15', cls=Host, ip='10.0.0.15', defaultRoute=None)
    h1 = net.addHost('h1', cls=Host, ip='10.0.0.1', defaultRoute=None)
    h2 = net.addHost('h2', cls=Host, ip='10.0.0.2', defaultRoute=None)
    h3 = net.addHost('h3', cls=Host, ip='10.0.0.3', defaultRoute=None)
    h9 = net.addHost('h9', cls=Host, ip='10.0.0.9', defaultRoute=None)
    h4 = net.addHost('h4', cls=Host, ip='10.0.0.4', defaultRoute=None)
    h5 = net.addHost('h5', cls=Host, ip='10.0.0.5', defaultRoute=None)

    info('*** Add links\n')
    net.addLink(s15, s1)
    net.addLink(s15, s14)
    net.addLink(s1, s2)
    net.addLink(s1, s3)
    net.addLink(s14, s8)
    net.addLink(s14, s11)
    net.addLink(s2, s4)
    net.addLink(s2, s5)
    net.addLink(s3, s6)
    net.addLink(s3, s7)
    net.addLink(s8, s9)
    net.addLink(s8, s10)
    net.addLink(s11, s12)
    net.addLink(s11, s13)
    net.addLink(s4, h1)
    net.addLink(s4, h2)
    net.addLink(s5, h3)
    net.addLink(s5, h4)
    net.addLink(s6, h5)
    net.addLink(s6, h6)
    net.addLink(s7, h7)
    net.addLink(s7, h8)
    net.addLink(s9, h9)
    net.addLink(s9, h10)
    net.addLink(s10, h11)
    net.addLink(s10, h12)
    net.addLink(s12, h13)
    net.addLink(s12, h14)
    net.addLink(s13, h15)
    net.addLink(s13, h16)
    net.addLink(s2, s3)
    net.addLink(s3, s8)
    net.addLink(s8, s11)

    info('*** Starting network\n')
    net.build()
    info('*** Starting controllers\n')
    for controller in net.controllers:
        controller.start()
```

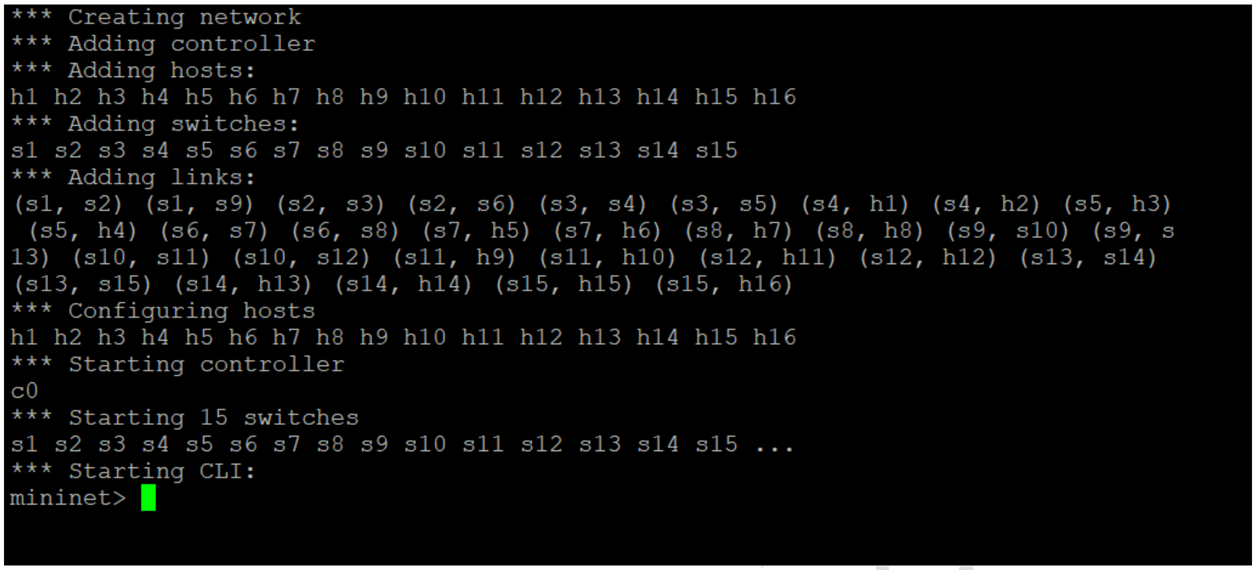


```
    info('*** Starting switches\n')
    net.get('s5').start([])
    net.get('s6').start([])
    net.get('s14').start([])
    net.get('s7').start([])
    net.get('s8').start([])
    net.get('s9').start([])
    net.get('s12').start([])
    net.get('s3').start([])
    net.get('s15').start([c0])
    net.get('s10').start([])
    net.get('s13').start([])
    net.get('s1').start([])
    net.get('s2').start([])
    net.get('s4').start([])
    net.get('s11').start([])

    info('*** Post configure switches and hosts\n')

    CLI(net)
    net.stop()


if __name__ == '__main__':
    setLogLevel('info')
    myNetwork()
```

iii) Start the topology and enter Mininet CLI run by typing this command :

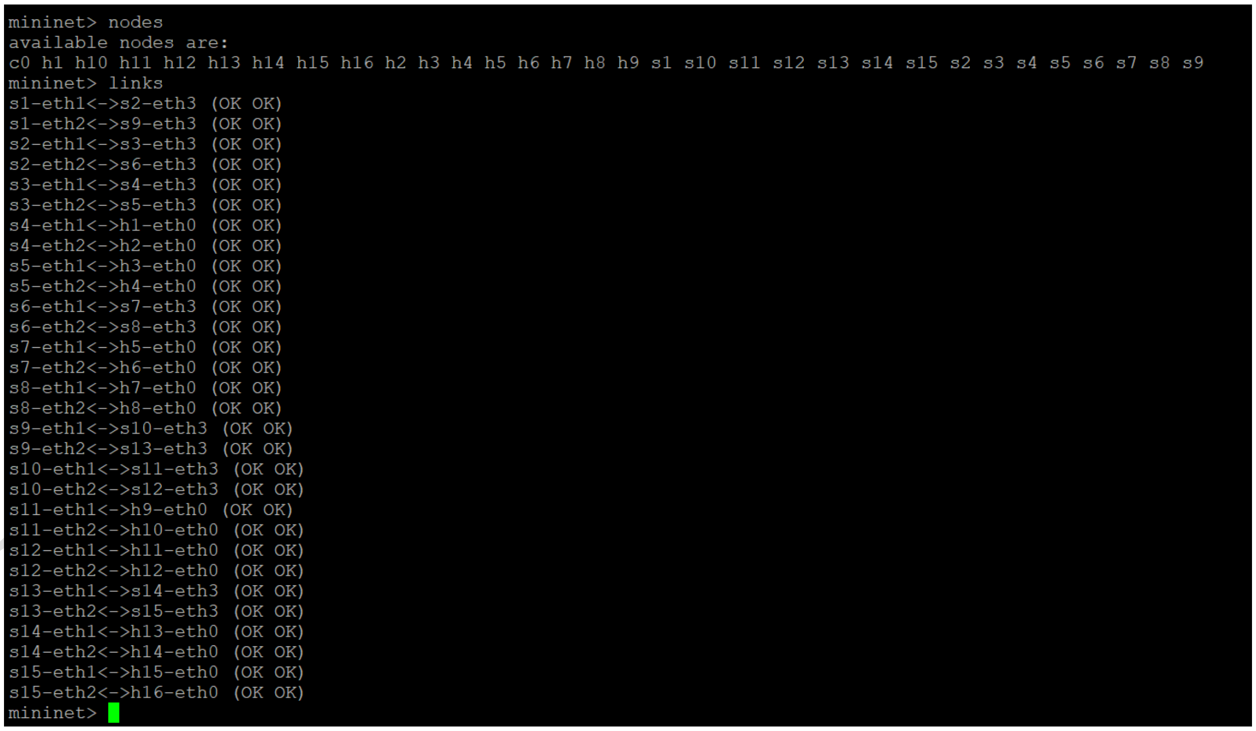
✓
```
mininet@mininet-vm:~$ sudo mn --custom ./mininet/custom/rohit.py --
topo=mytopo
```

iv) Then you will get the ouput as shown below :



```
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 s13 s14 s15
*** Adding links:
(s1, s2) (s1, s9) (s2, s3) (s2, s6) (s3, s4) (s3, s5) (s4, h1) (s4, h2) (s5, h3)
 (s5, h4) (s6, s7) (s6, s8) (s7, h5) (s7, h6) (s8, h7) (s8, h8) (s9, s10) (s9, s
13) (s10, s11) (s10, s12) (s11, h9) (s11, h10) (s12, h11) (s12, h12) (s13, s14)
(s13, s15) (s14, h13) (s14, h14) (s15, h15) (s15, h16)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
*** Starting controller
c0
*** Starting 15 switches
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 s13 s14 s15 ...
*** Starting CLI:
mininet>
```

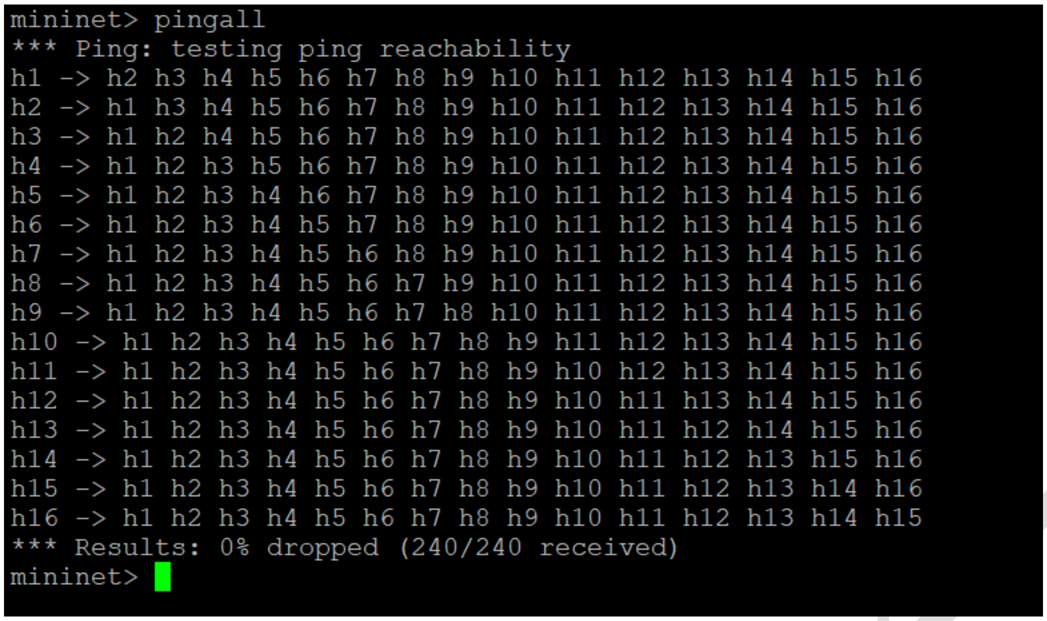
vi)Type "nodes" and "links" command to know all the nodes and links that are created in your topology



```
mininet> nodes
available nodes are:
c0 h1 h10 h11 h12 h13 h14 h15 h16 h2 h3 h4 h5 h6 h7 h8 h9 s1 s10 s11 s12 s13 s14 s15 s2 s3 s4 s5 s6 s7 s8 s9
mininet> links
s1-eth1<->s2-eth3 (OK OK)
s1-eth2<->s9-eth3 (OK OK)
s2-eth1<->s3-eth3 (OK OK)
s2-eth2<->s6-eth3 (OK OK)
s3-eth1<->s4-eth3 (OK OK)
s3-eth2<->s5-eth3 (OK OK)
s4-eth1<->h1-eth0 (OK OK)
s4-eth2<->h2-eth0 (OK OK)
s5-eth1<->h3-eth0 (OK OK)
s5-eth2<->h4-eth0 (OK OK)
s6-eth1<->s7-eth3 (OK OK)
s6-eth2<->s8-eth3 (OK OK)
s7-eth1<->h5-eth0 (OK OK)
s7-eth2<->h6-eth0 (OK OK)
s8-eth1<->h7-eth0 (OK OK)
s8-eth2<->h8-eth0 (OK OK)
s9-eth1<->s10-eth3 (OK OK)
s9-eth2<->s13-eth3 (OK OK)
s10-eth1<->s11-eth3 (OK OK)
s10-eth2<->s12-eth3 (OK OK)
s11-eth1<->h9-eth0 (OK OK)
s11-eth2<->h10-eth0 (OK OK)
s12-eth1<->h11-eth0 (OK OK)
s12-eth2<->h12-eth0 (OK OK)
s13-eth1<->s14-eth3 (OK OK)
s13-eth2<->s15-eth3 (OK OK)
s14-eth1<->h13-eth0 (OK OK)
s14-eth2<->h14-eth0 (OK OK)
s15-eth1<->h15-eth0 (OK OK)
s15-eth2<->h16-eth0 (OK OK)
mininet>
```

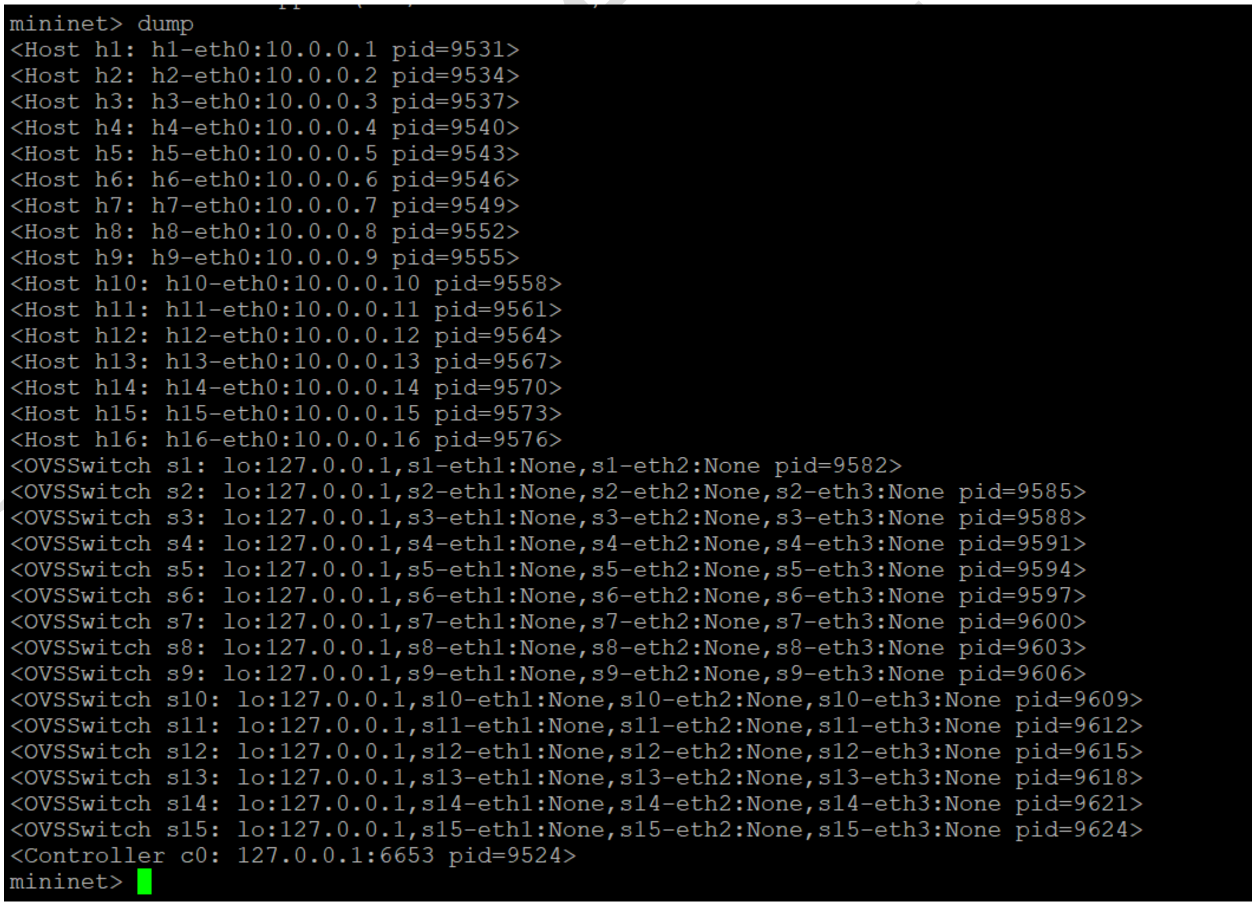
v) Test the simulation :

Now the network is set up. We expect that each virtual machine will be able to communicate with any other virtual machine on the same switch. Mininet provides a built-in command to check the communication between all hosts in the network. Use the Mininet *pingall* command, which will run the ping command on each host and ping every other host, then report the results in the Mininet command line interface:



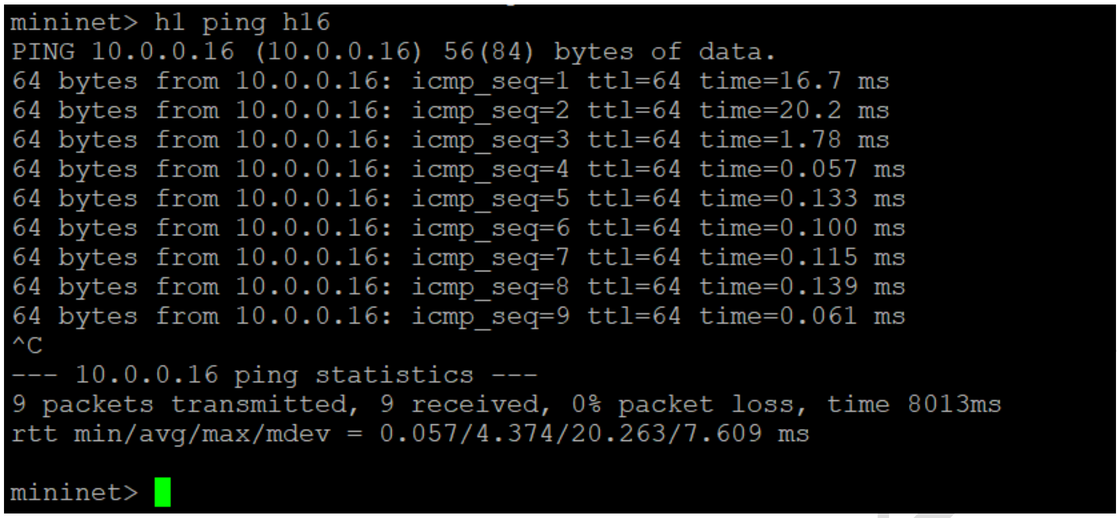
```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10 h11 h12 h13 h14 h15 h16
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10 h11 h12 h13 h14 h15 h16
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10 h11 h12 h13 h14 h15 h16
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h11 h12 h13 h14 h15 h16
h11 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h12 h13 h14 h15 h16
h12 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h13 h14 h15 h16
h13 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h14 h15 h16
h14 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h15 h16
h15 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h16
h16 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15
*** Results: 0% dropped (240/240 received)
mininet>
```

vi) execute the *dump* command to see the IP addresses of each node in the simulated network.



```
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=9531>
<Host h2: h2-eth0:10.0.0.2 pid=9534>
<Host h3: h3-eth0:10.0.0.3 pid=9537>
<Host h4: h4-eth0:10.0.0.4 pid=9540>
<Host h5: h5-eth0:10.0.0.5 pid=9543>
<Host h6: h6-eth0:10.0.0.6 pid=9546>
<Host h7: h7-eth0:10.0.0.7 pid=9549>
<Host h8: h8-eth0:10.0.0.8 pid=9552>
<Host h9: h9-eth0:10.0.0.9 pid=9555>
<Host h10: h10-eth0:10.0.0.10 pid=9558>
<Host h11: h11-eth0:10.0.0.11 pid=9561>
<Host h12: h12-eth0:10.0.0.12 pid=9564>
<Host h13: h13-eth0:10.0.0.13 pid=9567>
<Host h14: h14-eth0:10.0.0.14 pid=9570>
<Host h15: h15-eth0:10.0.0.15 pid=9573>
<Host h16: h16-eth0:10.0.0.16 pid=9576>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=9582>
<OVSSwitch s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None pid=9585>
<OVSSwitch s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None,s3-eth3:None pid=9588>
<OVSSwitch s4: lo:127.0.0.1,s4-eth1:None,s4-eth2:None,s4-eth3:None pid=9591>
<OVSSwitch s5: lo:127.0.0.1,s5-eth1:None,s5-eth2:None,s5-eth3:None pid=9594>
<OVSSwitch s6: lo:127.0.0.1,s6-eth1:None,s6-eth2:None,s6-eth3:None pid=9597>
<OVSSwitch s7: lo:127.0.0.1,s7-eth1:None,s7-eth2:None,s7-eth3:None pid=9600>
<OVSSwitch s8: lo:127.0.0.1,s8-eth1:None,s8-eth2:None,s8-eth3:None pid=9603>
<OVSSwitch s9: lo:127.0.0.1,s9-eth1:None,s9-eth2:None,s9-eth3:None pid=9606>
<OVSSwitch s10: lo:127.0.0.1,s10-eth1:None,s10-eth2:None,s10-eth3:None pid=9609>
<OVSSwitch s11: lo:127.0.0.1,s11-eth1:None,s11-eth2:None,s11-eth3:None pid=9612>
<OVSSwitch s12: lo:127.0.0.1,s12-eth1:None,s12-eth2:None,s12-eth3:None pid=9615>
<OVSSwitch s13: lo:127.0.0.1,s13-eth1:None,s13-eth2:None,s13-eth3:None pid=9618>
<OVSSwitch s14: lo:127.0.0.1,s14-eth1:None,s14-eth2:None,s14-eth3:None pid=9621>
<OVSSwitch s15: lo:127.0.0.1,s15-eth1:None,s15-eth2:None,s15-eth3:None pid=9624>
<Controller c0: 127.0.0.1:6653 pid=9524>
mininet>
```

vii)Type h1 ping h16 command to see that packets are going between first host and last host..



```
mininet> h1 ping h16
PING 10.0.0.16 (10.0.0.16) 56(84) bytes of data.
64 bytes from 10.0.0.16: icmp_seq=1 ttl=64 time=16.7 ms
64 bytes from 10.0.0.16: icmp_seq=2 ttl=64 time=20.2 ms
64 bytes from 10.0.0.16: icmp_seq=3 ttl=64 time=1.78 ms
64 bytes from 10.0.0.16: icmp_seq=4 ttl=64 time=0.057 ms
64 bytes from 10.0.0.16: icmp_seq=5 ttl=64 time=0.133 ms
64 bytes from 10.0.0.16: icmp_seq=6 ttl=64 time=0.100 ms
64 bytes from 10.0.0.16: icmp_seq=7 ttl=64 time=0.115 ms
64 bytes from 10.0.0.16: icmp_seq=8 ttl=64 time=0.139 ms
64 bytes from 10.0.0.16: icmp_seq=9 ttl=64 time=0.061 ms
^C
--- 10.0.0.16 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8013ms
rtt min/avg/max/mdev = 0.057/4.374/20.263/7.609 ms

mininet> 
```

Conclusion

Mininet is a unique open-source network simulator that is developed to support reseatch and education in Software Defined Networking. I found the Mininet command-line interface to be very easy to use. Because Mininet uses network namespaces as its virtualization technology, it can support a large number of virtual nodes without slowing down the simulation.

We used MiniEdit to create and run a simulation of a custom network topology. We showed how to use many of MiniEdit’s features.

MiniEdit seems to be a useful tool for creating custom software-defined network simulation scenarios.