KLE Technological University

COMPUTER NETWORKS-2

Submitted By:

Name	**:**	**Apoorva Jinde**
USN	**:**	**01FE18BCS044**
Division	**:**	**A**
Roll No	**:**	**144**

Under the guidance of

Ms Vijaylakshmi Mam

.

SCHOOL OF COMPUTER SCIENCE & ENGINEERING

HUBLI – 580 031 (India).

Academic year 2020-2021

I. Static Routing

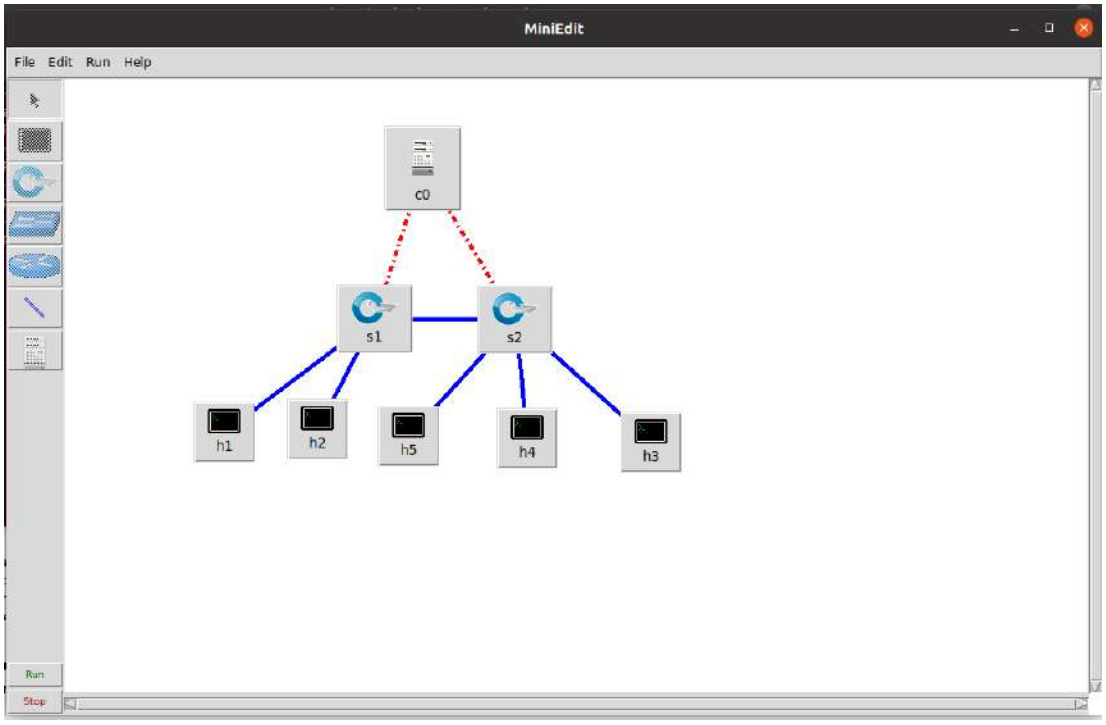
Routers forward packets using either route information from route table entries that you manually configure or the route information that is calculated using dynamic routing algorithms.

Static routes, which define explicit paths between two routers, cannot be automatically updated; you must manually reconfigure static routes when network changes occur.

We should use static routes in environments where network traffic is predictable and where the network design is simple. We should not use static routes in large, constantly changing networks because static routes cannot react to network changes.

Implementation

1) Create the below shown topology in the miniedit software and save it as .mn extension.



2) Export file to Python level 2 script file to get the following python script.

```
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')
    s1 = net.addSwitch('s1', cls=OVSKernelSwitch)
    s2 = net.addSwitch('s2', cls=OVSKernelSwitch)

    info( '*** Add hosts\n')
    h2 = net.addHost('h2', cls=Host, ip='10.0.0.2/24', defaultRoute=None)
    h1 = net.addHost('h1', cls=Host, ip='10.0.0.1/24', defaultRoute=None)
    h3 = net.addHost('h3', cls=Host, ip='10.0.0.129/24', defaultRoute=None)
    h5 = net.addHost('h5', cls=Host, ip='10.0.0.131/24', defaultRoute=None)
    h4 = net.addHost('h4', cls=Host, ip='10.0.0.130/24', defaultRoute=None)

    info( '*** Add links\n')
    net.addLink(s1, h1)
    net.addLink(s1, h2)
    net.addLink(s2, h3)
    net.addLink(s2, h4)
    net.addLink(s2, h5)
    net.addLink(s1, s2)

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

    info( '*** Starting switches\n')
    net.get('s1').start([c0])
    net.get('s2').start([c0])

    info( '*** Post configure switches and hosts\n')
    s1.cmd('ifconfig s1 10.0.0.0/24')
    s2.cmd('ifconfig s2 10.0.0.128/24')

    CLI(net)
    net.stop()

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

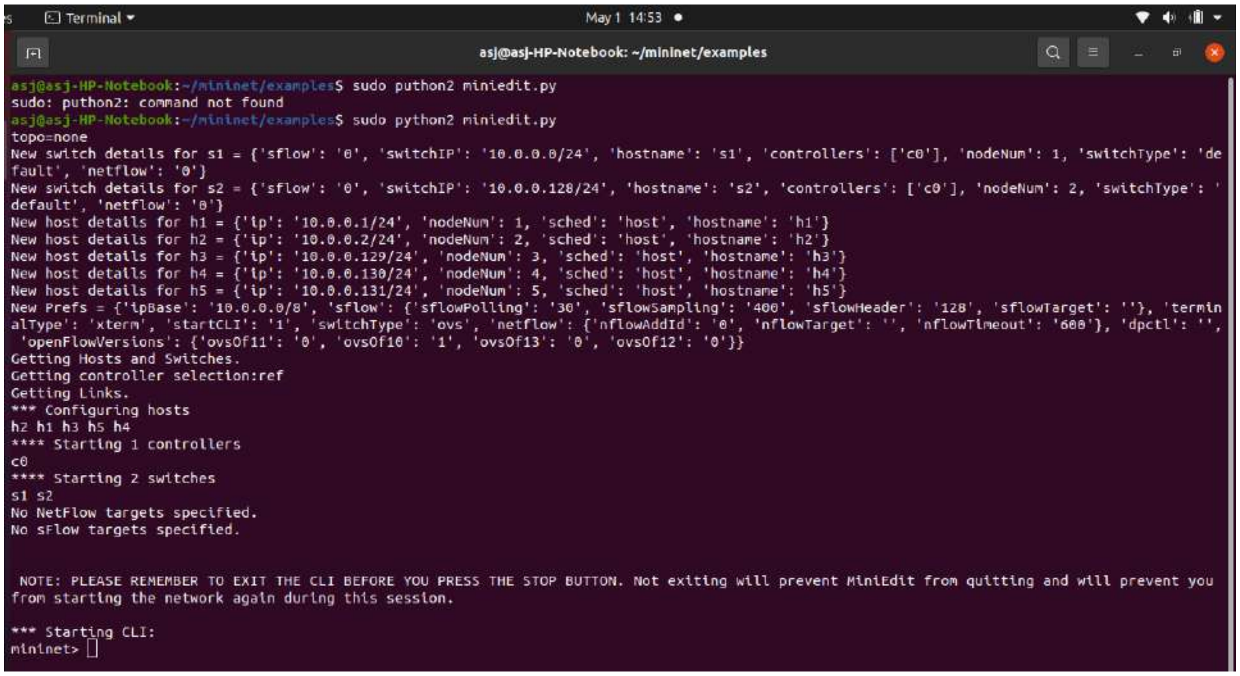
3) Run the python file in the terminal using the command

sudo python (filename).py

or

run the custom topology clicking the run in miniedit

You will see the log info messages for the topology being created followed by the mininet prompt .



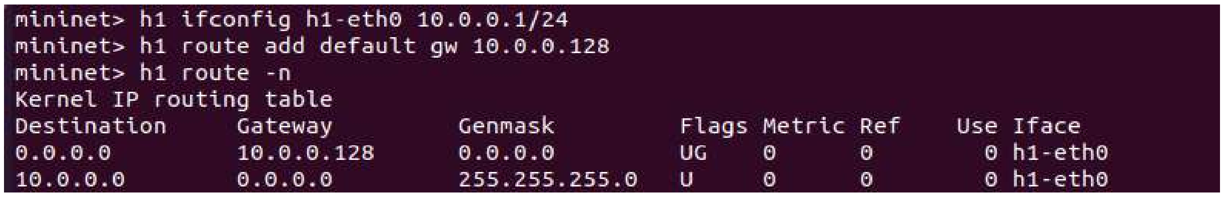
```
asj@asj-HP-Notebook:~/mininet/examples$ sudo puthon2 miniedit.py
sudo: puthon2: command not found
asj@asj-HP-Notebook:~/mininet/examples$ sudo python2 miniedit.py
topo=none
New switch details for s1 = {'sflow': '0', 'switchIP': '10.0.0.0/24', 'hostname': 's1', 'controllers': ['c0'], 'nodeNum': 1, 'switchType': 'de
fault', 'netflow': '0'}
New switch details for s2 = {'sflow': '0', 'switchIP': '10.0.0.128/24', 'hostname': 's2', 'controllers': ['c0'], 'nodeNum': 2, 'switchType': '
default', 'netflow': '0'}
New host details for h1 = {'ip': '10.0.0.1/24', 'nodeNum': 1, 'sched': 'host', 'hostname': 'h1'}
New host details for h2 = {'ip': '10.0.0.2/24', 'nodeNum': 2, 'sched': 'host', 'hostname': 'h2'}
New host details for h3 = {'ip': '10.0.0.129/24', 'nodeNum': 3, 'sched': 'host', 'hostname': 'h3'}
New host details for h4 = {'ip': '10.0.0.130/24', 'nodeNum': 4, 'sched': 'host', 'hostname': 'h4'}
New host details for h5 = {'ip': '10.0.0.131/24', 'nodeNum': 5, 'sched': 'host', 'hostname': 'h5'}
New Prefs = {'ipBase': '10.0.0.0/8', 'sflow': {'sflowPolling': '30', 'sflowSampling': '400', 'sflowHeader': '128', 'sflowTarget': ''}, 'termin
alType': 'xterm', 'startCLI': '1', 'switchType': 'ovs', 'netflow': {'nflowAddId': '0', 'nflowTarget': '', 'nflowTimeout': '600'}, 'dpctl': '',
 'openFlowVersions': {'ovsOf11': '0', 'ovsOf10': '1', 'ovsOf13': '0', 'ovsOf12': '0'}}
Getting Hosts and Switches.
Getting controller selection:ref
Getting Links.
*** Configuring hosts
h2 h1 h3 h5 h4
**** Starting 1 controllers
c0
**** Starting 2 switches
s1 s2
No NetFlow targets specified.
No sFlow targets specified.


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

*** Starting CLI:
mininet>
```

4) Steps performed for static routing:

a) The command **ifconfig** to configure the connection

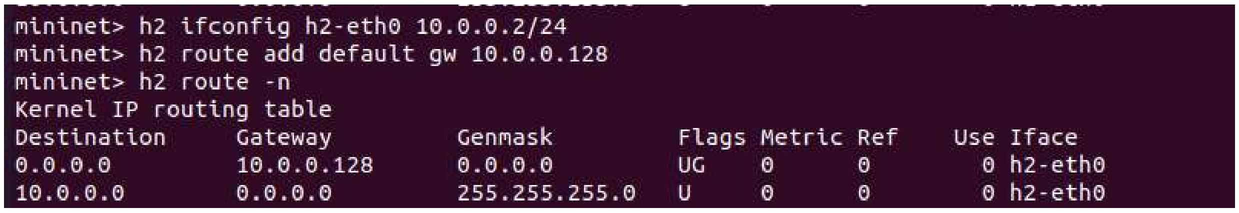
b) The command **ip route add default via <gateway>** is executed at all the hosts

h1 ifconfig h1-eth0 10.0.0.1/24
h1 route add default gw 10.0.0.128
h1 route –n



```
mininet> h1 ifconfig h1-eth0 10.0.0.1/24
mininet> h1 route add default gw 10.0.0.128
mininet> h1 route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.0.0.128      0.0.0.0         UG    0      0        0 h1-eth0
10.0.0.0        0.0.0.0         255.255.255.0   U     0      0        0 h1-eth0
```

h2 ifconfig h2-eth0 10.0.0.2/24
h2 route add default gw 10.0.0.128
h2 route –n



```
mininet> h2 ifconfig h2-eth0 10.0.0.2/24
mininet> h2 route add default gw 10.0.0.128
mininet> h2 route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.0.0.128      0.0.0.0         UG    0      0        0 h2-eth0
10.0.0.0        0.0.0.0         255.255.255.0   U     0      0        0 h2-eth0
```

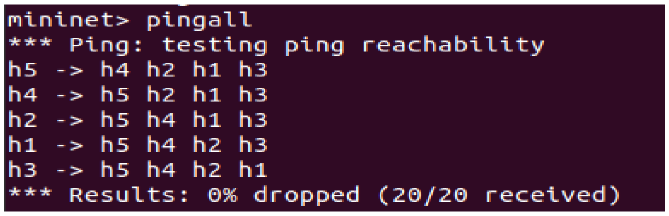
Similar actions are performed for the other subnet to connect statically.

h3 ifconfig h3-eth0 10.0.0.129/24
h3 route add default gw 10.0.0.0
h3 route -n

h4 ifconfig h4-eth0 10.0.0.130/24
h4 route add default gw 10.0.0.0
h4 route -n

h5 ifconfig h5-eth0 10.0.0.131/24
h5 route add default gw 10.0.0.0
h5 route –n

5) To confirm the connection check by using ping command.
All the hosts are pinged using pingall to test reachability

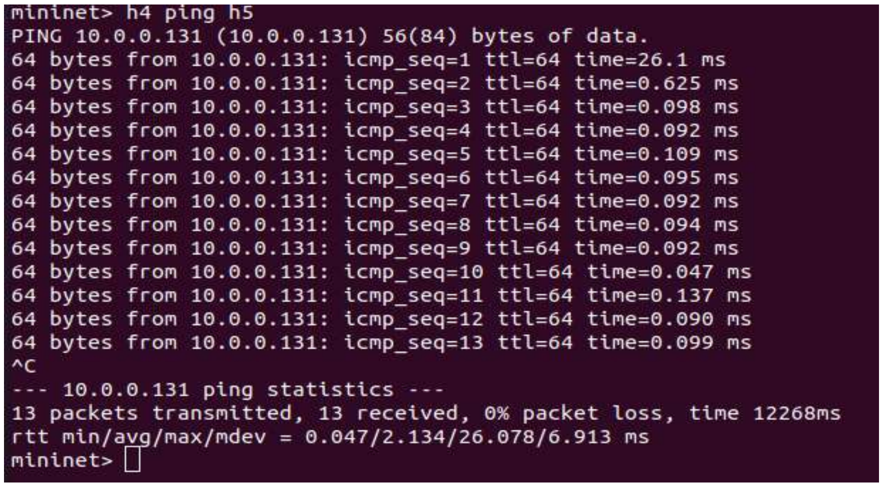


```
mininet> pingall
*** Ping: testing ping reachability
h5 -> h4 h2 h1 h3
h4 -> h5 h2 h1 h3
h2 -> h5 h4 h1 h3
h1 -> h5 h4 h2 h3
h3 -> h5 h4 h2 h1
*** Results: 0% dropped (20/20 received)
```

Results: No drop in the packet as our configuration is successful

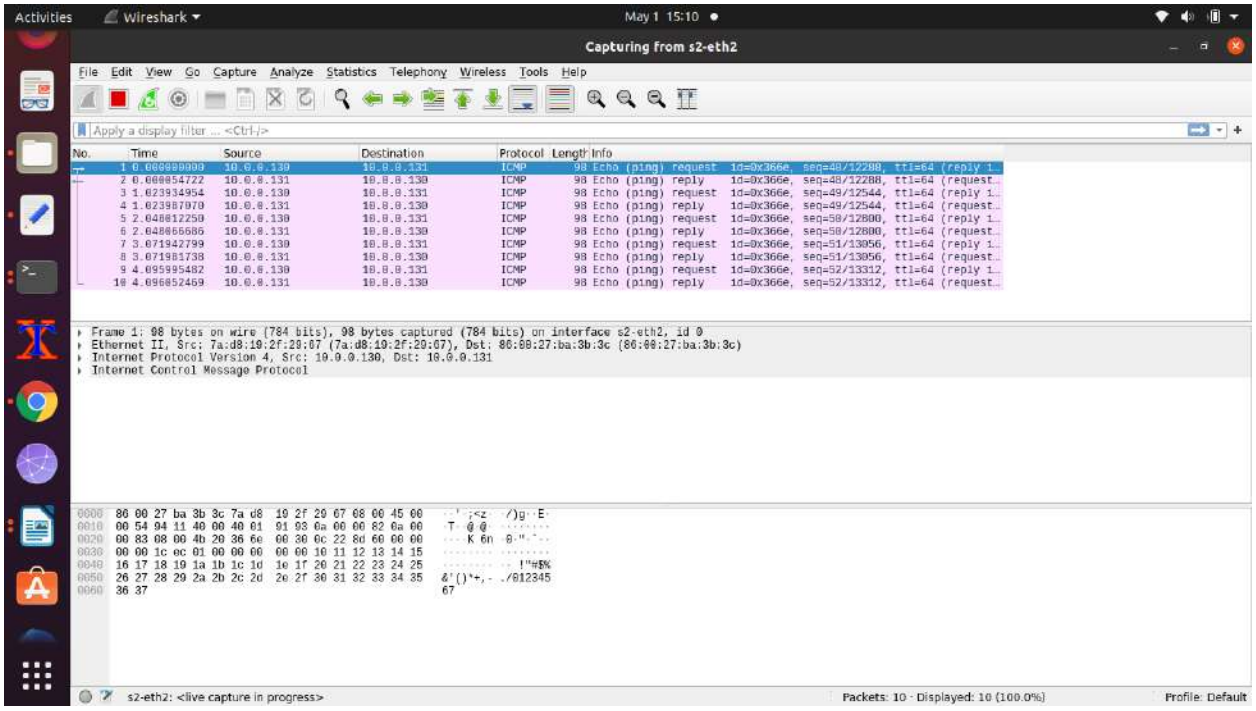
6) Ping the hosts within the subnet.

H4 ping H5 to check the connectivity between hosts within subnet.



```
mininet> h4 ping h5
PING 10.0.0.131 (10.0.0.131) 56(84) bytes of data.
64 bytes from 10.0.0.131: icmp_seq=1 ttl=64 time=26.1 ms
64 bytes from 10.0.0.131: icmp_seq=2 ttl=64 time=0.625 ms
64 bytes from 10.0.0.131: icmp_seq=3 ttl=64 time=0.098 ms
64 bytes from 10.0.0.131: icmp_seq=4 ttl=64 time=0.092 ms
64 bytes from 10.0.0.131: icmp_seq=5 ttl=64 time=0.109 ms
64 bytes from 10.0.0.131: icmp_seq=6 ttl=64 time=0.095 ms
64 bytes from 10.0.0.131: icmp_seq=7 ttl=64 time=0.092 ms
64 bytes from 10.0.0.131: icmp_seq=8 ttl=64 time=0.094 ms
64 bytes from 10.0.0.131: icmp_seq=9 ttl=64 time=0.092 ms
64 bytes from 10.0.0.131: icmp_seq=10 ttl=64 time=0.047 ms
64 bytes from 10.0.0.131: icmp_seq=11 ttl=64 time=0.137 ms
64 bytes from 10.0.0.131: icmp_seq=12 ttl=64 time=0.090 ms
64 bytes from 10.0.0.131: icmp_seq=13 ttl=64 time=0.099 ms
^C
--- 10.0.0.131 ping statistics ---
13 packets transmitted, 13 received, 0% packet loss, time 12268ms
rtt min/avg/max/mdev = 0.047/2.134/26.078/6.913 ms
mininet> 
```

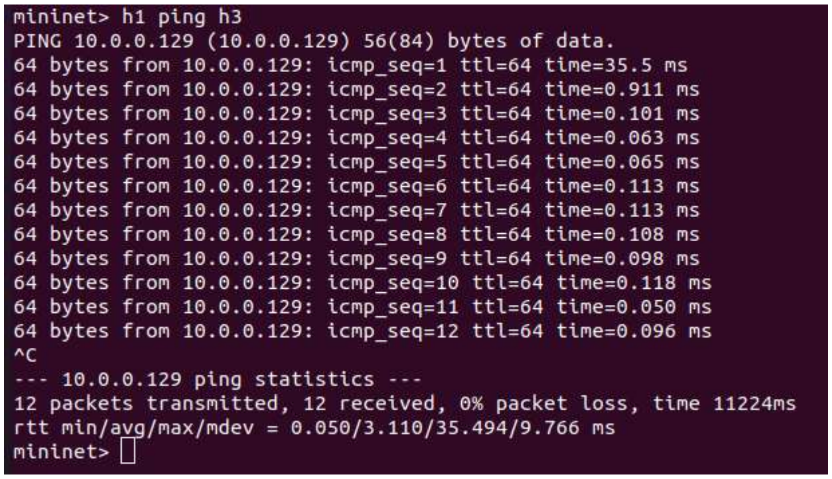
7) In the new terminal we open wireshark and record the flow of packets of the switch.
As seen above we had pinged h4 and h5 i.e. source is 10.0.0.130 and destination is 10.0.0.131


Activities
Wireshark
May 1 15:10
Capturing from s2-eth2
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Packets: 10 · Displayed: 10 (100.0%)
Profile: Default

```
No.  Time         Source      Destination  Protocol Length Info
   1 0.000000000  10.0.0.130  10.0.0.131   ICMP     98 Echo (ping) request  id=0x366e, seq=48/12288, ttl=64 (reply i…
   2 0.000054722  10.0.0.131  10.0.0.130   ICMP     98 Echo (ping) reply    id=0x366e, seq=48/12288, ttl=64 (request…
   3 1.023934954  10.0.0.130  10.0.0.131   ICMP     98 Echo (ping) request  id=0x366e, seq=49/12544, ttl=64 (reply i…
   4 1.023987070  10.0.0.131  10.0.0.130   ICMP     98 Echo (ping) reply    id=0x366e, seq=49/12544, ttl=64 (request…
   5 2.048012250  10.0.0.130  10.0.0.131   ICMP     98 Echo (ping) request  id=0x366e, seq=50/12800, ttl=64 (reply i…
   6 2.048066686  10.0.0.131  10.0.0.130   ICMP     98 Echo (ping) reply    id=0x366e, seq=50/12800, ttl=64 (request…
   7 3.071942799  10.0.0.130  10.0.0.131   ICMP     98 Echo (ping) request  id=0x366e, seq=51/13056, ttl=64 (reply i…
   8 3.071981738  10.0.0.131  10.0.0.130   ICMP     98 Echo (ping) reply    id=0x366e, seq=51/13056, ttl=64 (request…
   9 4.095995482  10.0.0.130  10.0.0.131   ICMP     98 Echo (ping) request  id=0x366e, seq=52/13312, ttl=64 (reply i…
  10 4.096052469  10.0.0.131  10.0.0.130   ICMP     98 Echo (ping) reply    id=0x366e, seq=52/13312, ttl=64 (request…

Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface s2-eth2, id 0
Ethernet II, Src: 7a:d8:19:2f:29:67 (7a:d8:19:2f:29:67), Dst: 86:00:27:ba:3b:3c (86:00:27:ba:3b:3c)
Internet Protocol Version 4, Src: 10.0.0.130, Dst: 10.0.0.131
Internet Control Message Protocol

0000  86 00 27 ba 3b 3c 7a d8  19 2f 29 67 08 00 45 00   ··'·;<z· ·/)g··E·
0010  00 54 94 11 40 00 40 01  91 93 0a 00 00 82 0a 00   ·T··@·@· ········
0020  00 83 08 00 4b 20 36 6e  00 30 0c 22 8d 60 00 00   ····K 6n ·0·"·`··
0030  00 00 1c ec 01 00 00 00  00 00 10 11 12 13 14 15   ········ ········
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ········ ·· !"#$%
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35   &'()*+,- ./012345
0060  36 37                                              67

s2-eth2: <live capture in progress>
```

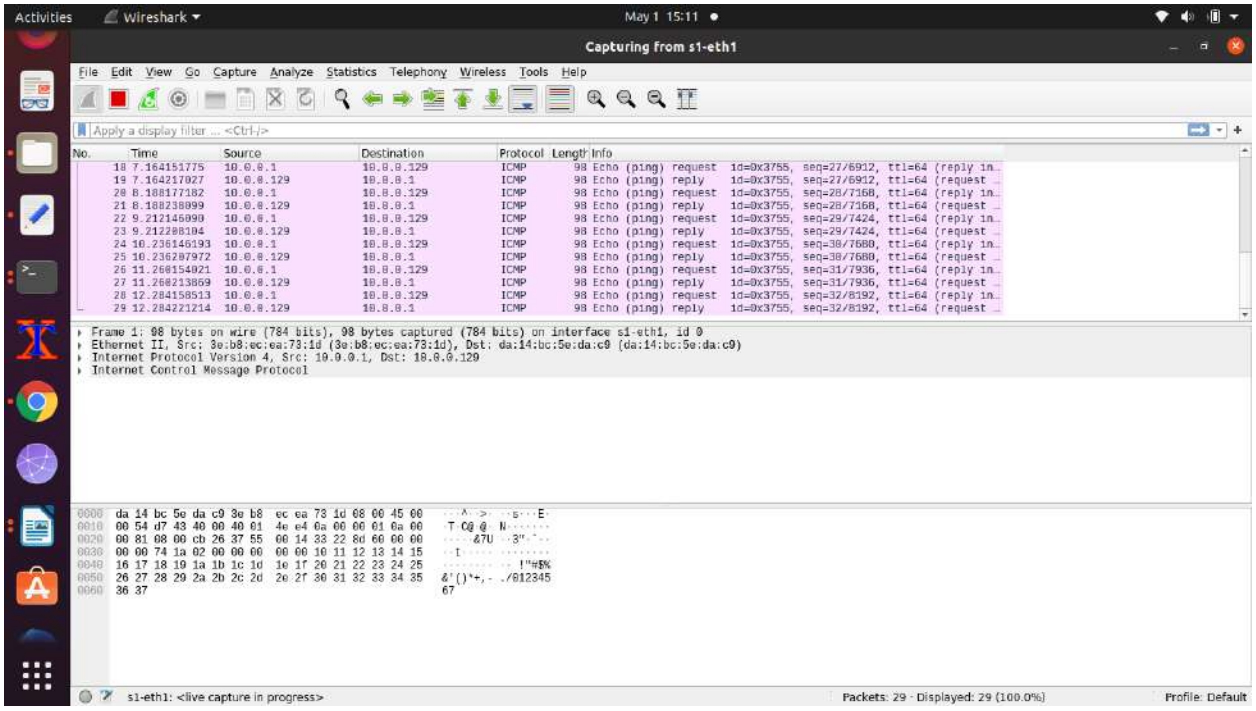
8) Ping the hosts across the subnet.

H1 ping H3 to check the connectivity between hosts across the subnets.



```
mininet> h1 ping h3
PING 10.0.0.129 (10.0.0.129) 56(84) bytes of data.
64 bytes from 10.0.0.129: icmp_seq=1 ttl=64 time=35.5 ms
64 bytes from 10.0.0.129: icmp_seq=2 ttl=64 time=0.911 ms
64 bytes from 10.0.0.129: icmp_seq=3 ttl=64 time=0.101 ms
64 bytes from 10.0.0.129: icmp_seq=4 ttl=64 time=0.063 ms
64 bytes from 10.0.0.129: icmp_seq=5 ttl=64 time=0.065 ms
64 bytes from 10.0.0.129: icmp_seq=6 ttl=64 time=0.113 ms
64 bytes from 10.0.0.129: icmp_seq=7 ttl=64 time=0.113 ms
64 bytes from 10.0.0.129: icmp_seq=8 ttl=64 time=0.108 ms
64 bytes from 10.0.0.129: icmp_seq=9 ttl=64 time=0.098 ms
64 bytes from 10.0.0.129: icmp_seq=10 ttl=64 time=0.118 ms
64 bytes from 10.0.0.129: icmp_seq=11 ttl=64 time=0.050 ms
64 bytes from 10.0.0.129: icmp_seq=12 ttl=64 time=0.096 ms
^C
--- 10.0.0.129 ping statistics ---
12 packets transmitted, 12 received, 0% packet loss, time 11224ms
rtt min/avg/max/mdev = 0.050/3.110/35.494/9.766 ms
mininet> 
```

9) In the new terminal we open wireshark and record the flow of packets of across the subnet.
As seen above we had pinged h1 and h3 i.e. source is 10.0.0.1 and destination is 10.0.0.129


Activities
Wireshark
May 1 15:11
Capturing from s1-eth1
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.  Time          Source       Destination  Protocol Length Info
  18 7.164151775   10.0.0.1     10.0.0.129   ICMP     98 Echo (ping) request  id=0x3755, seq=27/6912, ttl=64 (reply in…
  19 7.164217027   10.0.0.129   10.0.0.1     ICMP     98 Echo (ping) reply    id=0x3755, seq=27/6912, ttl=64 (request …
  20 8.188177182   10.0.0.1     10.0.0.129   ICMP     98 Echo (ping) request  id=0x3755, seq=28/7168, ttl=64 (reply in…
  21 8.188238099   10.0.0.129   10.0.0.1     ICMP     98 Echo (ping) reply    id=0x3755, seq=28/7168, ttl=64 (request …
  22 9.212146090   10.0.0.1     10.0.0.129   ICMP     98 Echo (ping) request  id=0x3755, seq=29/7424, ttl=64 (reply in…
  23 9.212208104   10.0.0.129   10.0.0.1     ICMP     98 Echo (ping) reply    id=0x3755, seq=29/7424, ttl=64 (request …
  24 10.236146193  10.0.0.1     10.0.0.129   ICMP     98 Echo (ping) request  id=0x3755, seq=30/7680, ttl=64 (reply in…
  25 10.236207972  10.0.0.129   10.0.0.1     ICMP     98 Echo (ping) reply    id=0x3755, seq=30/7680, ttl=64 (request …
  26 11.260154021  10.0.0.1     10.0.0.129   ICMP     98 Echo (ping) request  id=0x3755, seq=31/7936, ttl=64 (reply in…
  27 11.260213869  10.0.0.129   10.0.0.1     ICMP     98 Echo (ping) reply    id=0x3755, seq=31/7936, ttl=64 (request …
  28 12.284158513  10.0.0.1     10.0.0.129   ICMP     98 Echo (ping) request  id=0x3755, seq=32/8192, ttl=64 (reply in…
  29 12.284221214  10.0.0.129   10.0.0.1     ICMP     98 Echo (ping) reply    id=0x3755, seq=32/8192, ttl=64 (request …

▸ Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface s1-eth1, id 0
▸ Ethernet II, Src: 3e:b8:ec:ea:73:1d (3e:b8:ec:ea:73:1d), Dst: da:14:bc:5e:da:c9 (da:14:bc:5e:da:c9)
▸ Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.129
▸ Internet Control Message Protocol

0000  da 14 bc 5e da c9 3e b8  ec ea 73 1d 08 00 45 00   ···^··>···s···E·
0010  00 54 d7 43 40 00 40 01  4e e4 0a 00 00 01 0a 00   ·T·C@·@·N·······
0020  00 81 08 00 cb 26 37 55  00 14 33 22 8d 60 00 00   ·····&7U··3"·`··
0030  00 00 74 1a 02 00 00 00  00 00 10 11 12 13 14 15   ··t·············
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ·········· !"#$%
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35   &'()*+,-./012345
0060  36 37                                              67

s1-eth1: <live capture in progress>        Packets: 29 · Displayed: 29 (100.0%)        Profile: Default
```

Conclusion:

The hosts are connected to each other by manually connecting the hosts. In static routing changes are to be done manually depending on the changes in the network. It is useful when the network is not dense or complex.

II. Load Balancing:

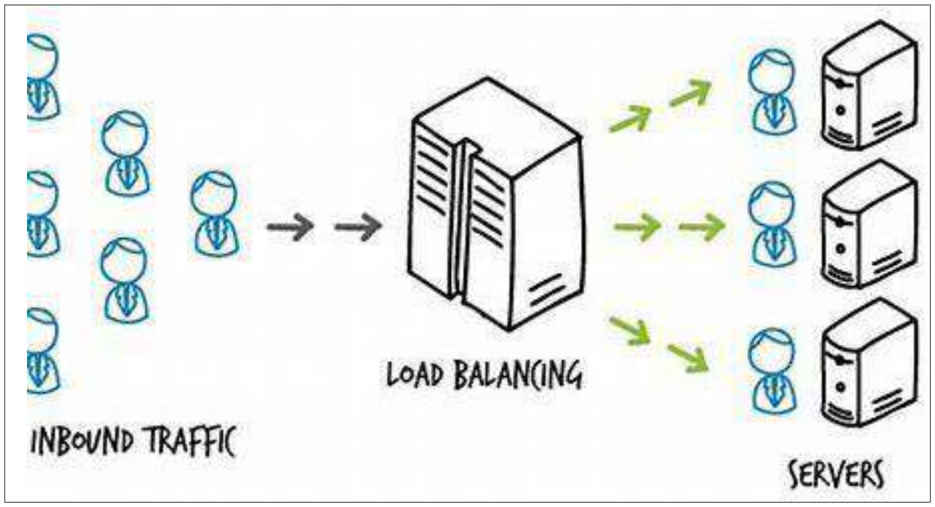
In computing, load balancing refers to **the process of distributing a set of tasks over a set of resources (computing units), with the aim of making their overall processing more efficient.**

Load balancing techniques can optimize the response time for each task, avoiding unevenly overloading compute nodes while other compute nodes are left idle.

In this manner, a load balancer performs the following functions:

- Distributes client requests or network load efficiently across multiple servers
- Ensures high availability and reliability by sending requests only to servers that are online
- Provides the flexibility to add or subtract servers as demand dictates

If a single server goes down, the load balancer redirects traffic to the remaining online servers.



Different load balancing algorithms:

1. Round Robin
2. Weighted round robin.
3. Least Connections.
4. Equal connections

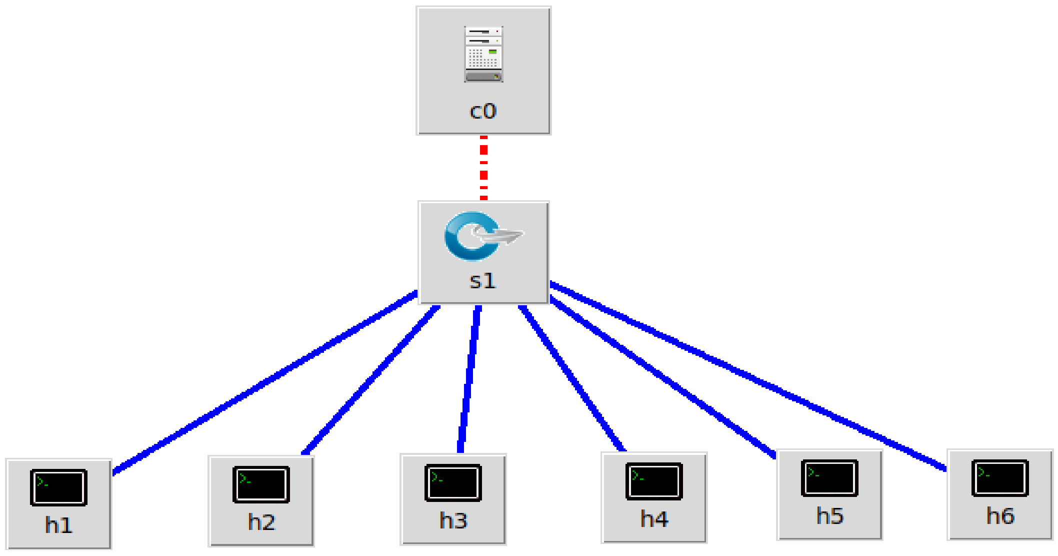
Custom Topology:

The topology consists of one open flow controller **c0,** 1 switches (**s1**) and 3 client machines (**h4, h5, h6**) and 3 servers(**h1,h2,h3**), the ip addresses are as follows:

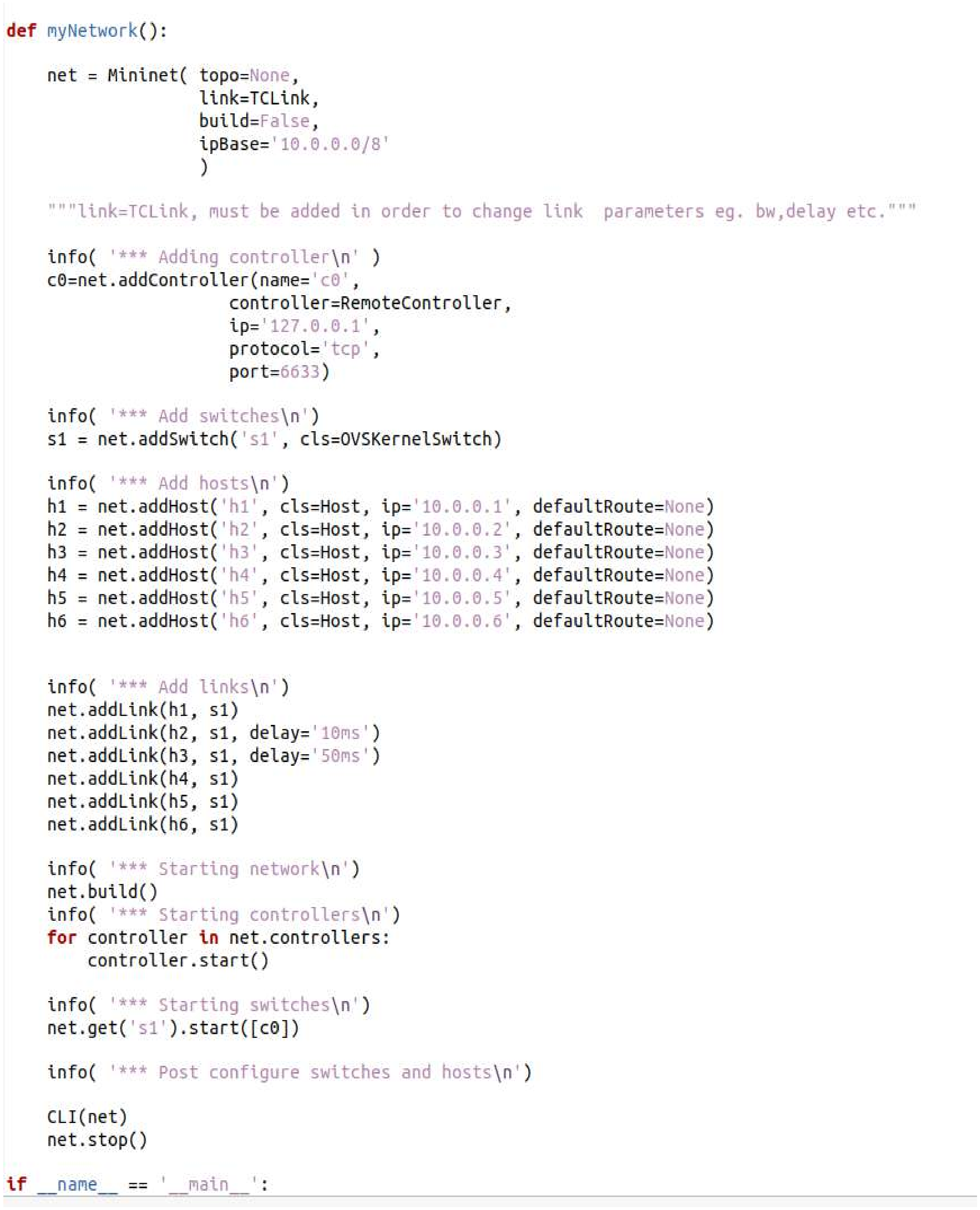
server1: 10.0.0.1
server2: 10.0.0.2
server3: 10.0.0.3

client1: 10.0.0.4
client2: 10.0.0.5
client3: 10.0.0.6

1) We have constructed a simple topology using miniedit tool

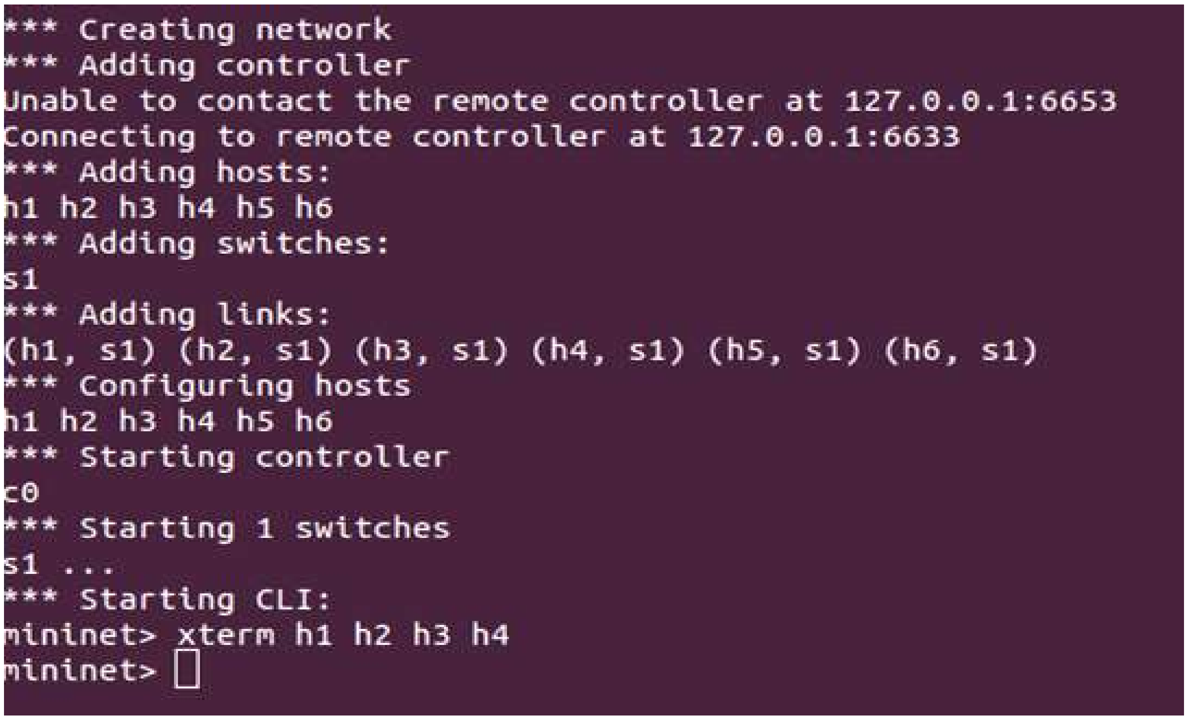


2) The custom topology network is exported as python(.py) file



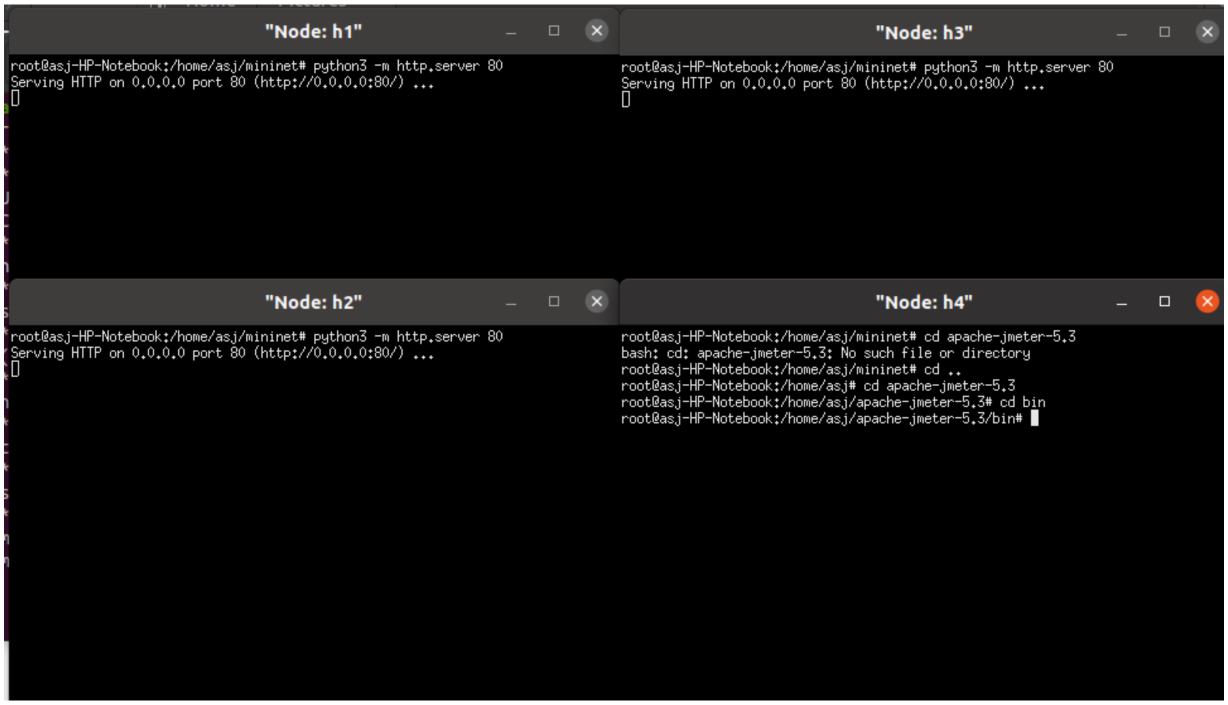
```
def myNetwork():

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

    """link=TCLink, must be added in order to change link  parameters eg. bw,delay etc."""

    info( '*** Adding controller\n' )
    c0=net.addController(name='c0',
                      controller=RemoteController,
                      ip='127.0.0.1',
                      protocol='tcp',
                      port=6633)

    info( '*** Add switches\n')
    s1 = net.addSwitch('s1', cls=OVSKernelSwitch)

    info( '*** Add hosts\n')
    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)
    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)
    h6 = net.addHost('h6', cls=Host, ip='10.0.0.6', defaultRoute=None)


    info( '*** Add links\n')
    net.addLink(h1, s1)
    net.addLink(h2, s1, delay='10ms')
    net.addLink(h3, s1, delay='50ms')
    net.addLink(h4, s1)
    net.addLink(h5, s1)
    net.addLink(h6, s1)

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

    info( '*** Starting switches\n')
    net.get('s1').start([c0])

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

    CLI(net)
    net.stop()

if __name__ == '__main__':
```

3) Running our topology and hence our customized topology is constructed



```
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Connecting to remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3 h4 h5 h6
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1) (h6, s1)
*** Configuring hosts
h1 h2 h3 h4 h5 h6
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> xterm h1 h2 h3 h4
mininet> 
```

4) Setting the h1, h2, h3 as servers



"Node: h1"

```
root@asj-HP-Notebook:/home/asj/mininet# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

"Node: h3"

```
root@asj-HP-Notebook:/home/asj/mininet# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

"Node: h2"

```
root@asj-HP-Notebook:/home/asj/mininet# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

"Node: h4"

```
root@asj-HP-Notebook:/home/asj/mininet# cd apache-jmeter-5.3
bash: cd: apache-jmeter-5.3: No such file or directory
root@asj-HP-Notebook:/home/asj/mininet# cd ..
root@asj-HP-Notebook:/home/asj# cd apache-jmeter-5.3
root@asj-HP-Notebook:/home/asj/apache-jmeter-5.3# cd bin
root@asj-HP-Notebook:/home/asj/apache-jmeter-5.3/bin# 
```

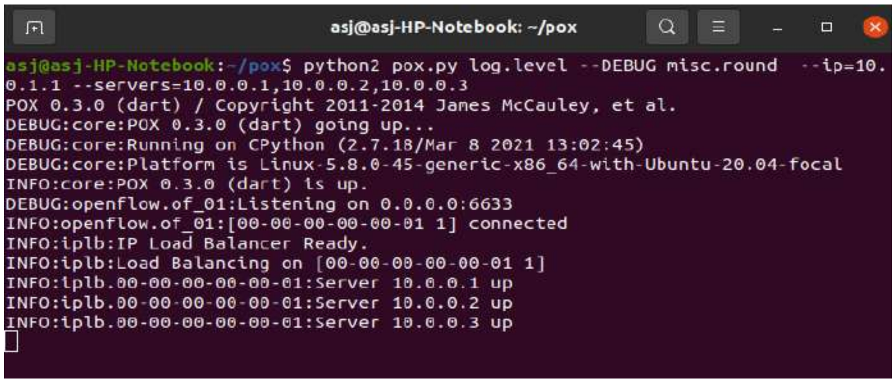
i. Round Robin

Requests are distributed across the servers in a sequential or rotational manner.

For example, the first request goes to the first server, the second one goes to the second server, the third request goes to the third server and it continues further for all the requests.

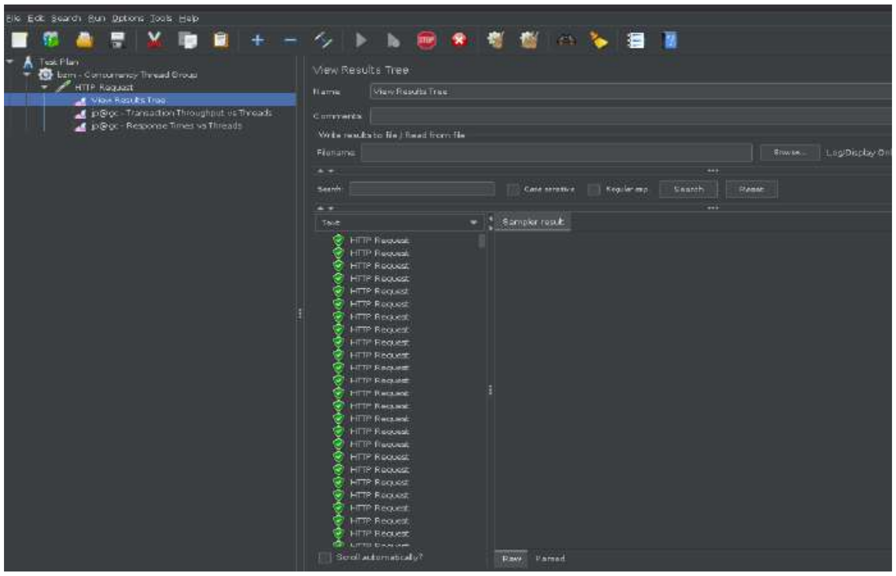
It is easy to implement but it doesn't consider the load already on a server so there is a risk that one of the servers receives a lot of requests and becomes overloaded.

1) Running the POX

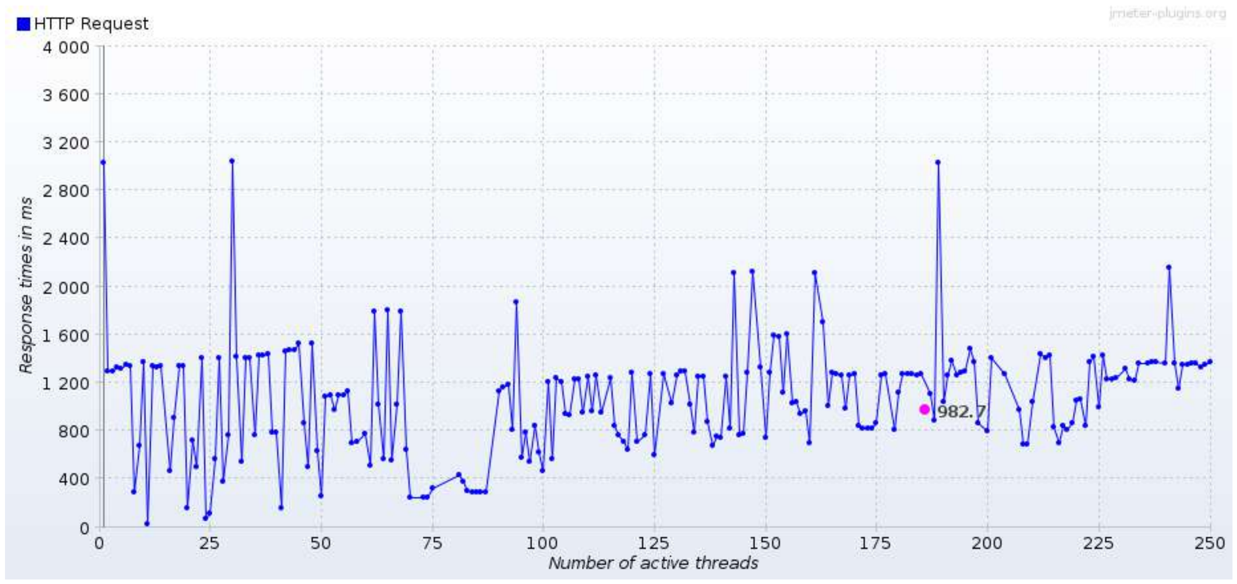


```
asj@asj-HP-Notebook:~/pox$ python2 pox.py log.level --DEBUG misc.round  --ip=10.
0.1.1 --servers=10.0.0.1,10.0.0.2,10.0.0.3
POX 0.3.0 (dart) / Copyright 2011-2014 James McCauley, et al.
DEBUG:core:POX 0.3.0 (dart) going up...
DEBUG:core:Running on CPython (2.7.18/Mar 8 2021 13:02:45)
DEBUG:core:Platform is Linux-5.8.0-45-generic-x86_64-with-Ubuntu-20.04-focal
INFO:core:POX 0.3.0 (dart) is up.
DEBUG:openflow.of_01:Listening on 0.0.0.0:6633
INFO:openflow.of_01:[00-00-00-00-00-01 1] connected
INFO:iplb:IP Load Balancer Ready.
INFO:iplb:Load Balancing on [00-00-00-00-00-01 1]
INFO:iplb.00-00-00-00-00-01:Server 10.0.0.1 up
INFO:iplb.00-00-00-00-00-01:Server 10.0.0.2 up
INFO:iplb.00-00-00-00-00-01:Server 10.0.0.3 up
```

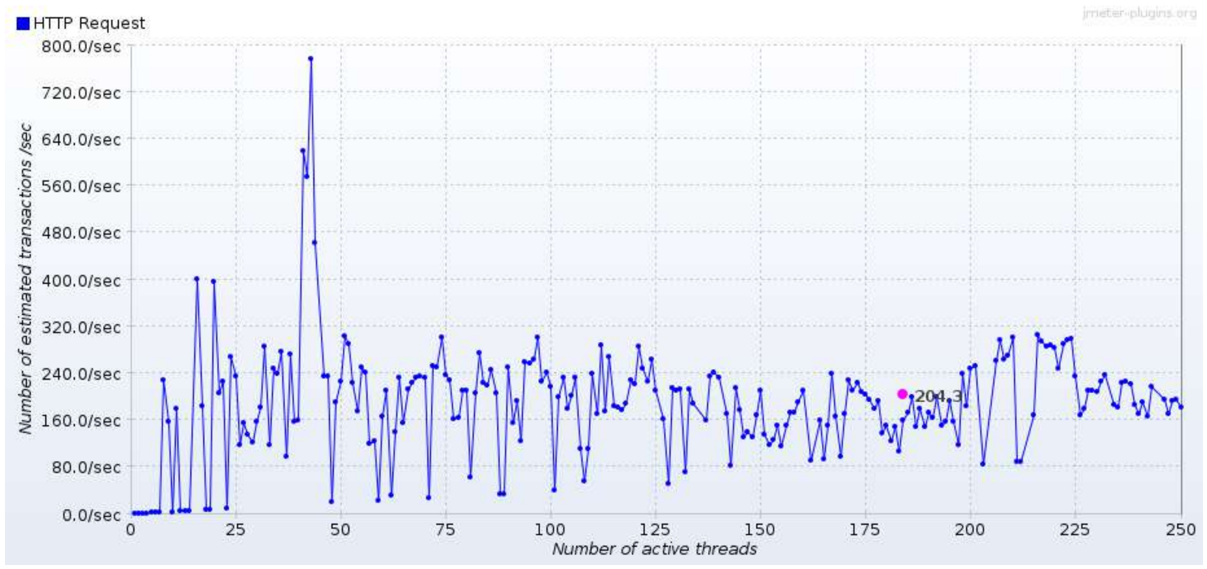
2) Checking the status of different HTTP request



3) Obtaining the Response times vs number of active threads and number of transactions vs number of active threads graphs



Interpretation: We can observe that the response time of our server 10.0.1.1 is almost consistent but we can also observe that the graph consists of upward peaks i.e. high response time which shows it is little inefficient.

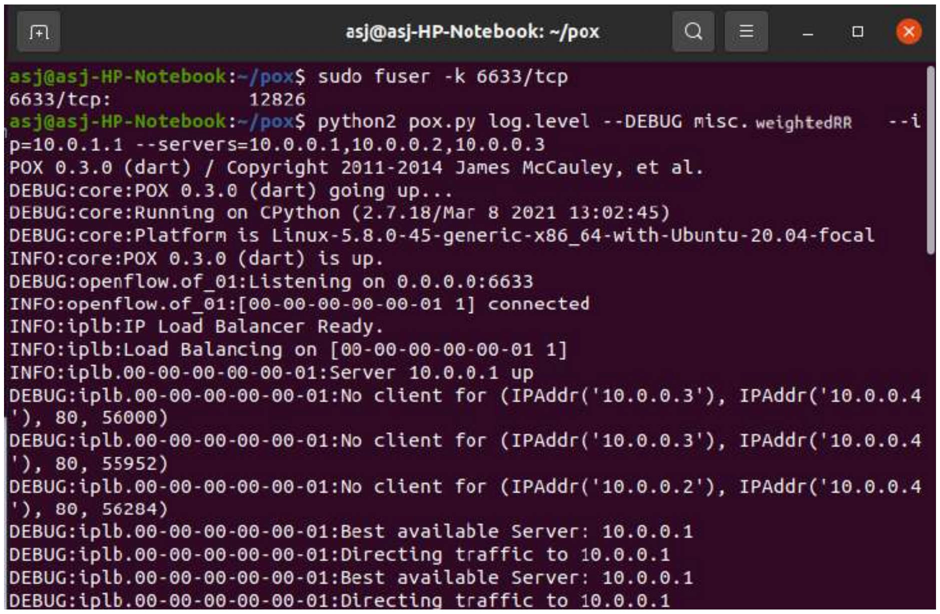


Interpretation: We can observe that the number of transaction our server 10.0.1.1 can handle is almost consistent but we can also observe that the graph consists of downward peaks i.e. less number of transactions which shows it is little inefficient.

ii. Weighted Round Robin

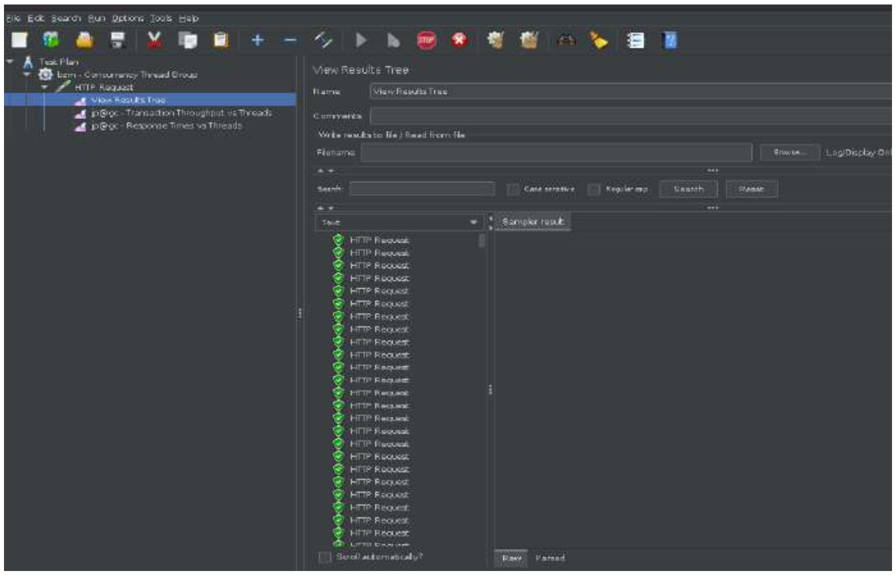
It is much similar to the round-robin technique. The only difference is, each of the resources in a list is provided a weighted score. Depending on the weighted score the request is distributed to these servers. So in this method, some of the servers get a bigger share of the overall request.

1) Running the POX

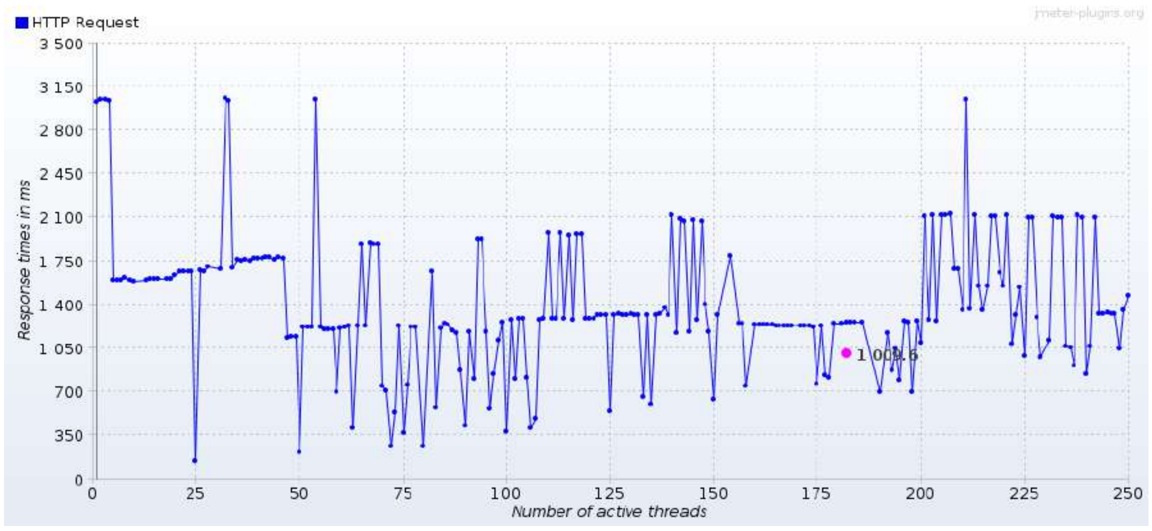


```
asj@asj-HP-Notebook: ~/pox
asj@asj-HP-Notebook:~/pox$ sudo fuser -k 6633/tcp
6633/tcp:            12826
asj@asj-HP-Notebook:~/pox$ python2 pox.py log.level --DEBUG misc.weightedRR   --i
p=10.0.1.1 --servers=10.0.0.1,10.0.0.2,10.0.0.3
POX 0.3.0 (dart) / Copyright 2011-2014 James McCauley, et al.
DEBUG:core:POX 0.3.0 (dart) going up...
DEBUG:core:Running on CPython (2.7.18/Mar 8 2021 13:02:45)
DEBUG:core:Platform is Linux-5.8.0-45-generic-x86_64-with-Ubuntu-20.04-focal
INFO:core:POX 0.3.0 (dart) is up.
DEBUG:openflow.of_01:Listening on 0.0.0.0:6633
INFO:openflow.of_01:[00-00-00-00-00-01 1] connected
INFO:iplb:IP Load Balancer Ready.
INFO:iplb:Load Balancing on [00-00-00-00-00-01 1]
INFO:iplb.00-00-00-00-00-01:Server 10.0.0.1 up
DEBUG:iplb.00-00-00-00-00-01:No client for (IPAddr('10.0.0.3'), IPAddr('10.0.0.4
'), 80, 56000)
DEBUG:iplb.00-00-00-00-00-01:No client for (IPAddr('10.0.0.3'), IPAddr('10.0.0.4
'), 80, 55952)
DEBUG:iplb.00-00-00-00-00-01:No client for (IPAddr('10.0.0.2'), IPAddr('10.0.0.4
'), 80, 56284)
DEBUG:iplb.00-00-00-00-00-01:Best available Server: 10.0.0.1
DEBUG:iplb.00-00-00-00-00-01:Directing traffic to 10.0.0.1
DEBUG:iplb.00-00-00-00-00-01:Best available Server: 10.0.0.1
DEBUG:iplb.00-00-00-00-00-01:Directing traffic to 10.0.0.1
```

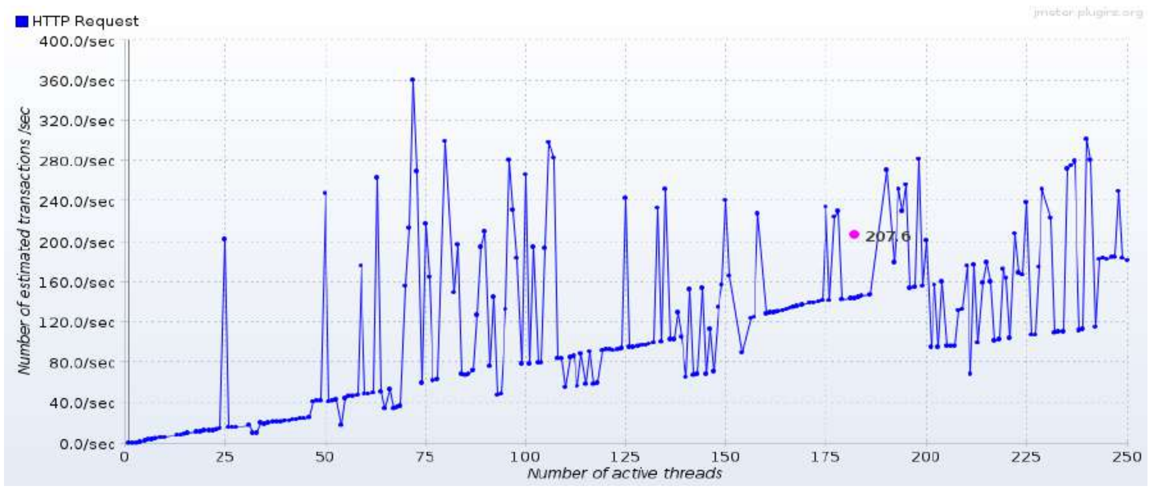
2) Checking the status of different HTTP request



3) Obtaining the Response times vs number of active threads and number of transactions vs number of active threads graphs



Interpretation: We can observe that the response time of our server 10.0.1.1 is inconsistent at the starting but we can observe that the graph is becoming consistent with the value of 1400ms and 2100ms gradually which means with the increase in the number of threads the response time is not increasing instead it is trying to be constant.

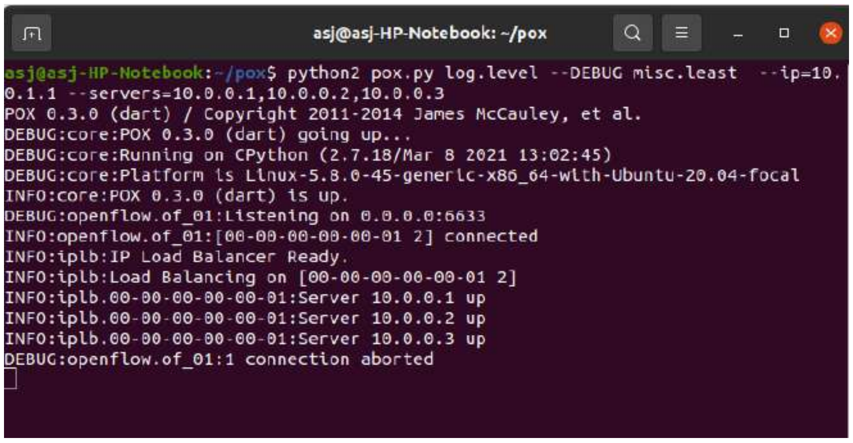


Interpretation: We can observe that the number of transaction our server 10.0.1.1 can handle is increasing i.e. it is able to handle the number of transactions when we are increasing the number of threads. This shows it is trying to increase the its capacity and transact with the threads accordingly.

iii. Least Connection Method

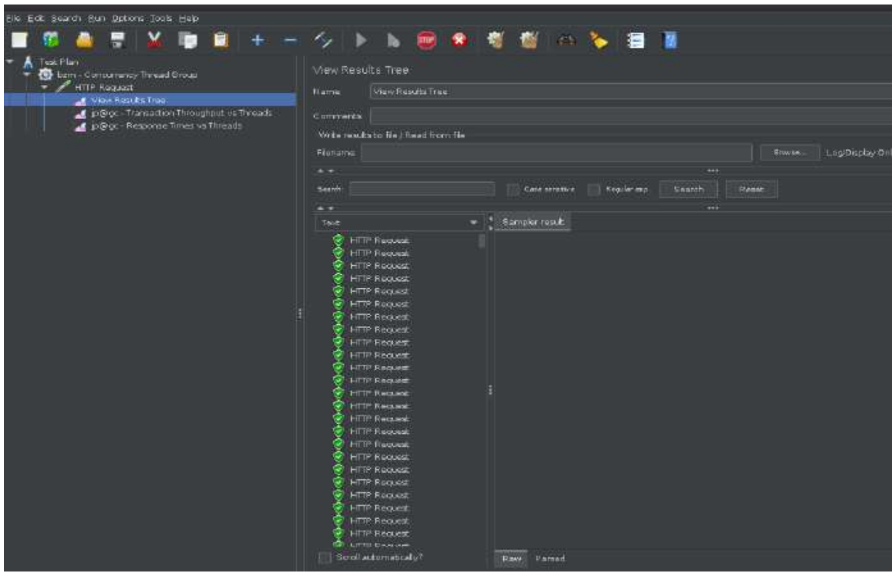
In this method, the request will be directed to the server with the fewest number of requests or active connections. To do this load balancer needs to do some additional computing to identify the server with the least number of connections. This may be a little bit costlier compare to the round-robin method but the evaluation is based on the current load on the server. This algorithm is most useful when there is a huge number of a persistent connection in the traffic unevenly distributed between the servers.

1) Running the POX

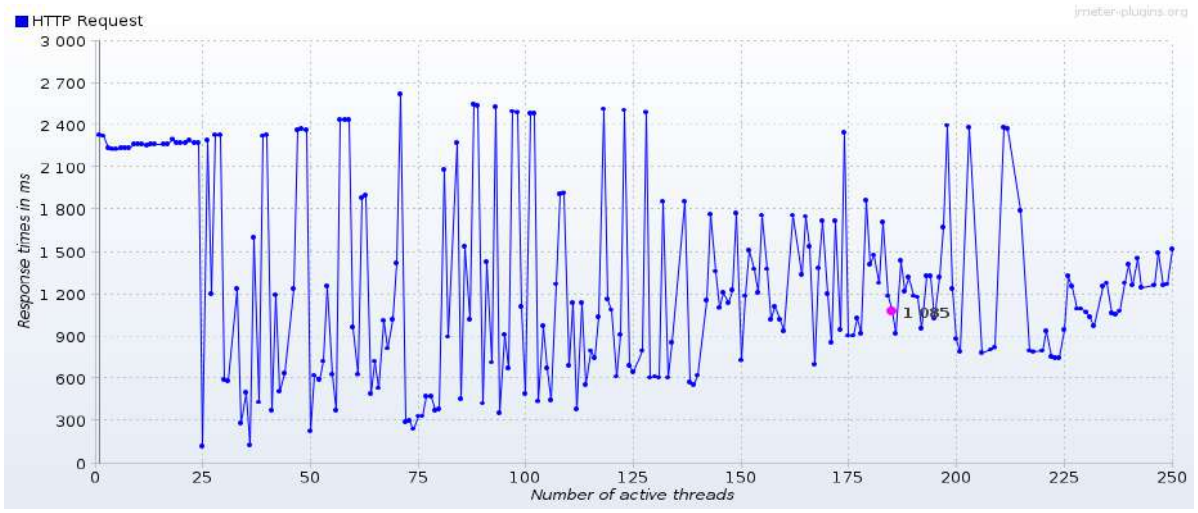


```
asj@asj-HP-Notebook:~/pox$ python2 pox.py log.level --DEBUG misc.least  --ip=10.
0.1.1 --servers=10.0.0.1,10.0.0.2,10.0.0.3
POX 0.3.0 (dart) / Copyright 2011-2014 James McCauley, et al.
DEBUG:core:POX 0.3.0 (dart) going up...
DEBUG:core:Running on CPython (2.7.18/Mar 8 2021 13:02:45)
DEBUG:core:Platform is Linux-5.8.0-45-generic-x86_64-with-Ubuntu-20.04-focal
INFO:core:POX 0.3.0 (dart) is up.
DEBUG:openflow.of_01:Listening on 0.0.0.0:6633
INFO:openflow.of_01:[00-00-00-00-00-01 2] connected
INFO:iplb:IP Load Balancer Ready.
INFO:iplb:Load Balancing on [00-00-00-00-00-01 2]
INFO:iplb.00-00-00-00-00-01:Server 10.0.0.1 up
INFO:iplb.00-00-00-00-00-01:Server 10.0.0.2 up
INFO:iplb.00-00-00-00-00-01:Server 10.0.0.3 up
DEBUG:openflow.of_01:1 connection aborted
```

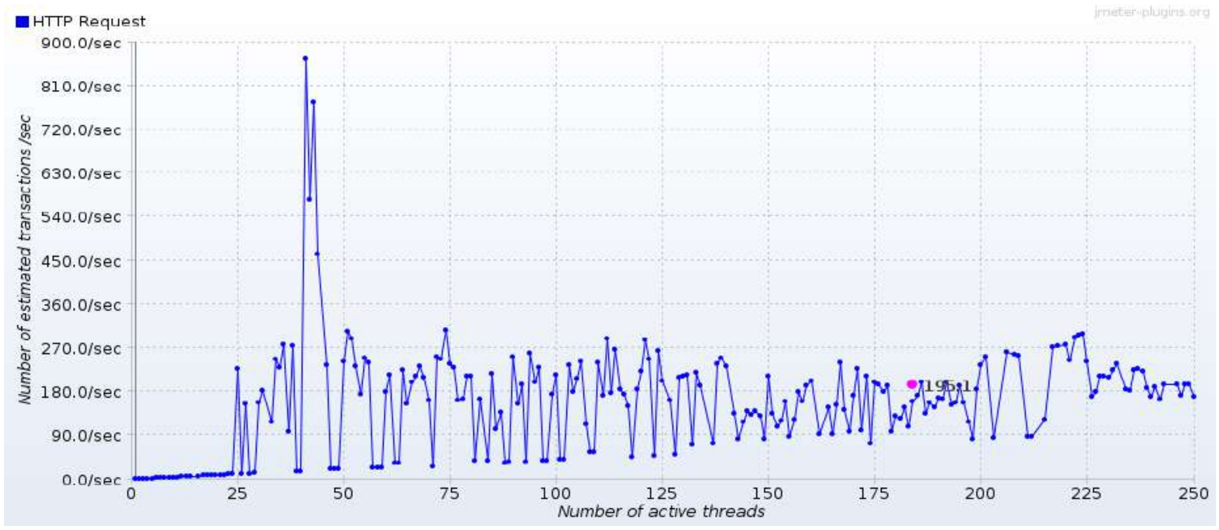
2) Checking the status of different HTTP request



3) Obtaining the Response times vs number of active threads and number of transactions vs number of active threads graphs



Interpretation: We can observe that the response time of our server 10.0.1.1 is though increasing sharply it is not continuing to remain increased for a long time it very speedily decreases again indicating reduced response time of our server and interpreting that our server takes less time to respond

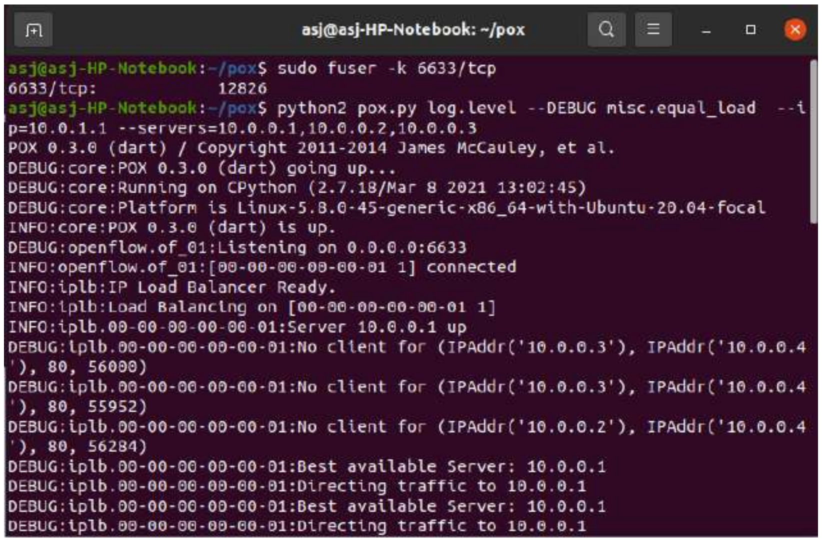


Interpretation: We can observe that the number of transaction our server 10.0.1.1 can handle is consistent i.e. it is able to handle the number of transactions when we are increasing the number of threads. This shows it is trying to increase its capacity and transact with the threads accordingly.

iv. Equal Connections

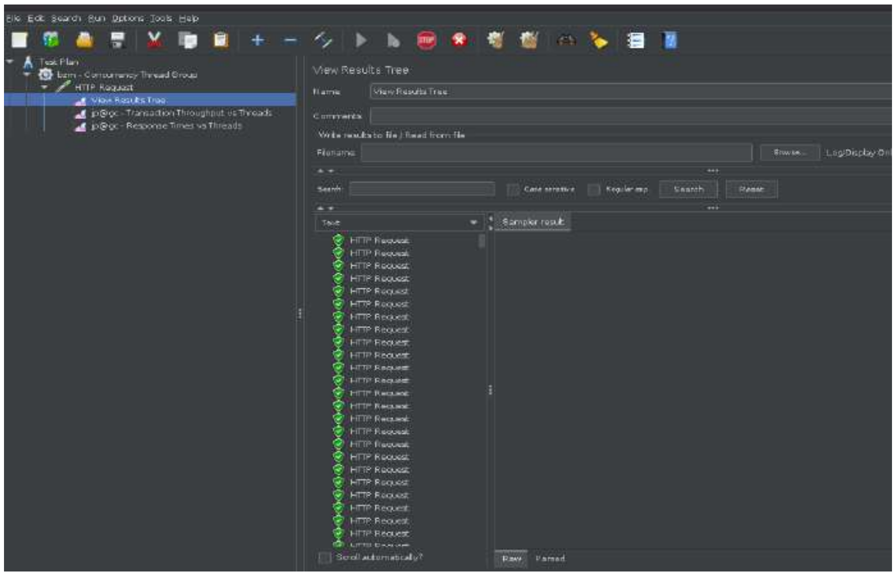
In this method, the requests are distributed equally among the host servers so that all have the equal number of active connections.

1) Running the POX

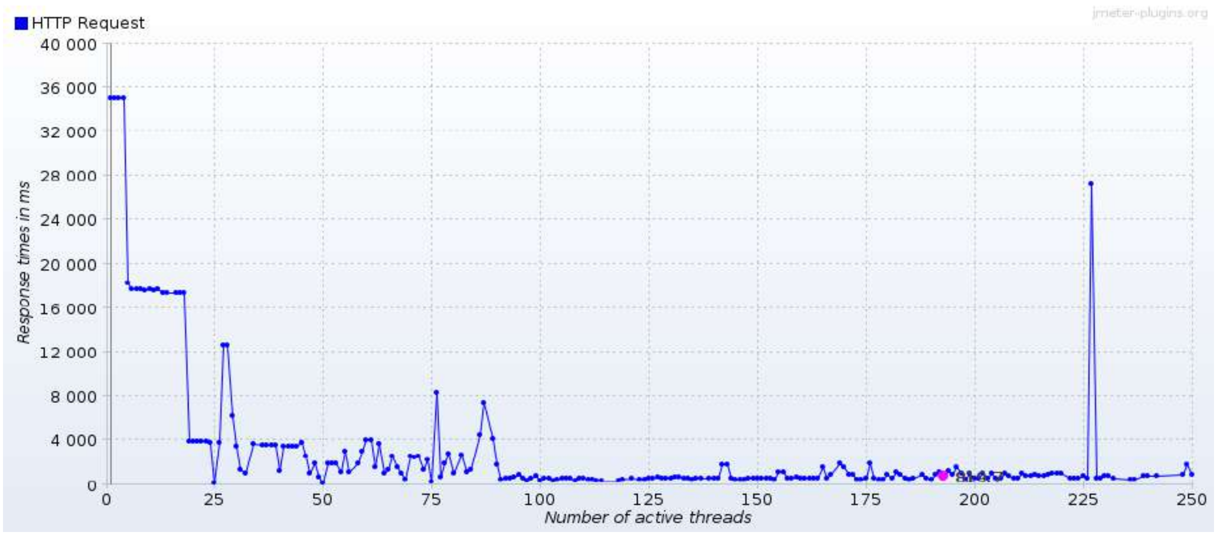


```
asj@asj-HP-Notebook:~/pox$ sudo fuser -k 6633/tcp
6633/tcp:            12826
asj@asj-HP-Notebook:~/pox$ python2 pox.py log.level --DEBUG misc.equal_load  --i
p=10.0.1.1 --servers=10.0.0.1,10.0.0.2,10.0.0.3
POX 0.3.0 (dart) / Copyright 2011-2014 James McCauley, et al.
DEBUG:core:POX 0.3.0 (dart) going up...
DEBUG:core:Running on CPython (2.7.18/Mar 8 2021 13:02:45)
DEBUG:core:Platform is Linux-5.8.0-45-generic-x86_64-with-Ubuntu-20.04-focal
INFO:core:POX 0.3.0 (dart) is up.
DEBUG:openflow.of_01:Listening on 0.0.0.0:6633
INFO:openflow.of_01:[00-00-00-00-00-01 1] connected
INFO:iplb:IP Load Balancer Ready.
INFO:iplb:Load Balancing on [00-00-00-00-00-01 1]
INFO:iplb.00-00-00-00-00-01:Server 10.0.0.1 up
DEBUG:iplb.00-00-00-00-00-01:No client for (IPAddr('10.0.0.3'), IPAddr('10.0.0.4
'), 80, 56000)
DEBUG:iplb.00-00-00-00-00-01:No client for (IPAddr('10.0.0.3'), IPAddr('10.0.0.4
'), 80, 55952)
DEBUG:iplb.00-00-00-00-00-01:No client for (IPAddr('10.0.0.2'), IPAddr('10.0.0.4
'), 80, 56284)
DEBUG:iplb.00-00-00-00-00-01:Best available Server: 10.0.0.1
DEBUG:iplb.00-00-00-00-00-01:Directing traffic to 10.0.0.1
DEBUG:iplb.00-00-00-00-00-01:Best available Server: 10.0.0.1
DEBUG:iplb.00-00-00-00-00-01:Directing traffic to 10.0.0.1
```

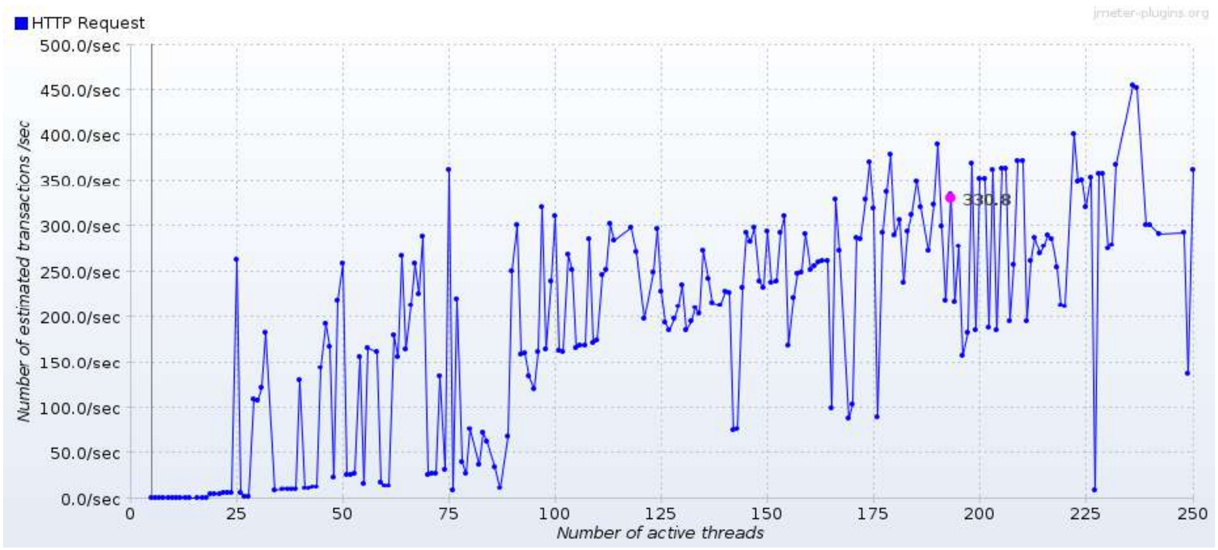
2) Checking the status of different HTTP request



3) Obtaining the Response times vs number of active threads and number of transactions vs number of active threads graphs



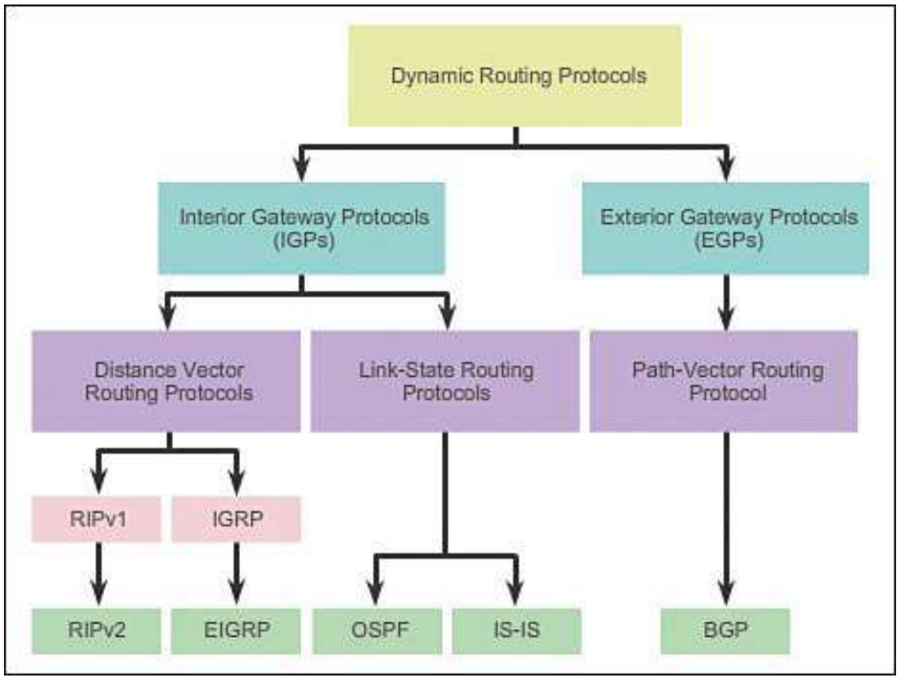
Interpretation: We can observe that the response time of our server 10.0.1.1 is consistently very small. We can also observe that the response time is less which means the server is responding taking very less time.



Interpretation: We can observe that the number of transaction our server 10.0.1.1 can handle is consistent i.e. it is able to handle the number of transactions when we are increasing the number of threads. This shows it is trying to increase its capacity and transact with the threads accordingly.

III. Dynamic Routing

Dynamic routing is also known as **adaptive** routing which change routing table according to the change in topology. Dynamic routing uses complex routing algorithms and it does not provide high security like static routing. When the network change (topology) occurs, it sends the message to router to ensure that changes then the routes are recalculated for sending updated routing information.


Dynamic Routing Protocols
Interior Gateway Protocols (IGPs)
Exterior Gateway Protocols (EGPs)
Distance Vector Routing Protocols
Link-State Routing Protocols
Path-Vector Routing Protocol
RIPv1
IGRP
RIPv2
EIGRP
OSPF
IS-IS
BGP

Routing is the process of selecting best paths in a network.

There are two most well know algorithms for routing as follows:

- Link State Algorithms
- Distance Vector Algorithms

i. Link State Routing

Link state routing is a technique in which each router shares the knowledge of its neighborhood with every other router in the internetwork.

The three keys to understand the Link State Routing algorithm:

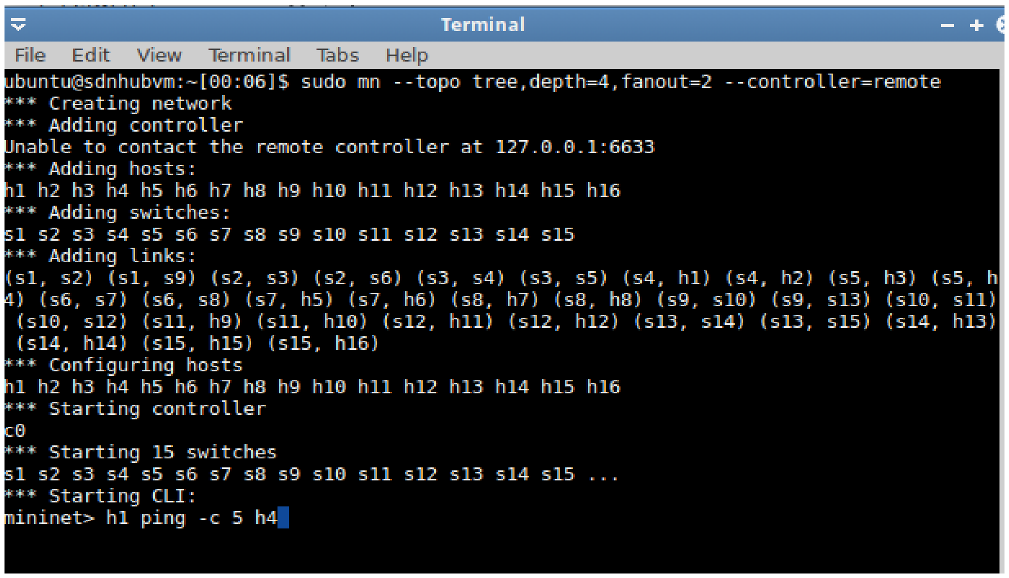
- **Knowledge about the neighborhood:** Instead of sending its routing table, a router sends the information about its neighborhood only. A router broadcast its identities and cost of the directly attached links to other routers.
- **Flooding:** Each router sends the information to every other router on the internetwork except its neighbors. This process is known as Flooding. Every router that receives the packet sends the copies to all its neighbors. Finally, each and every router receives a copy of the same information.
- **Information sharing:** A router sends the information to every other router only when the change occurs in the information.

Implementation:

Each node uses Dijkstra's algorithm on the graph to calculate the optimal routes to all nodes.

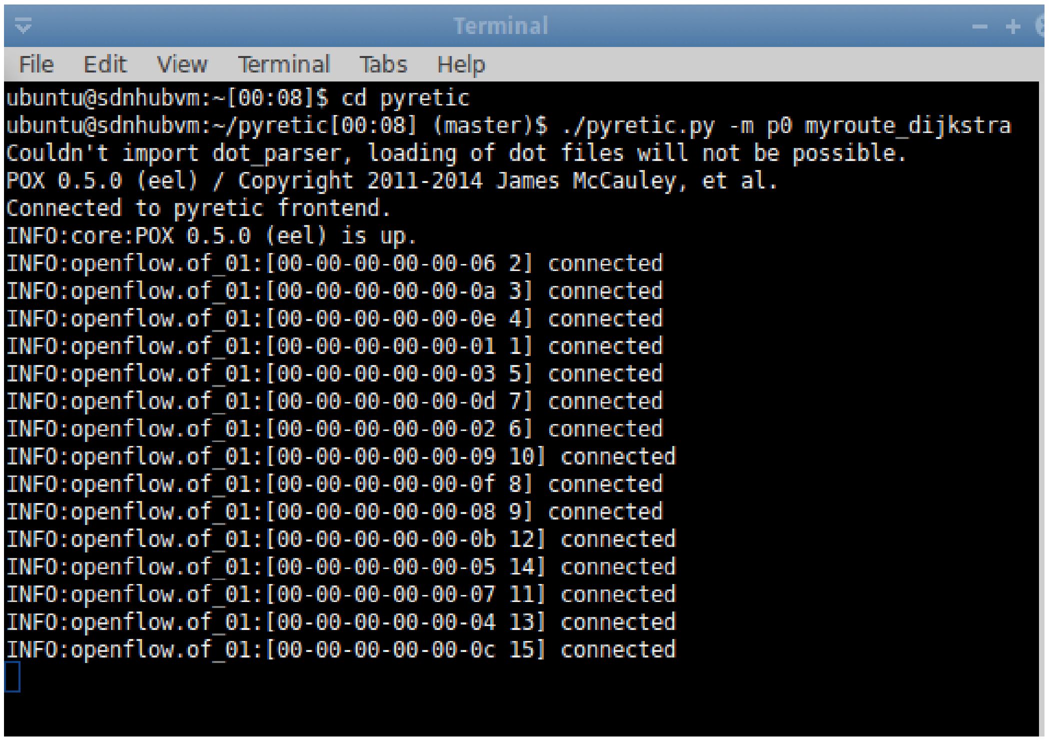
- The Link state routing algorithm is also known as Dijkstra's algorithm which is used to find the shortest path from one node to every other node in the network.
- The Dijkstra's algorithm is an iterative, and it has the property that after k^{th} iteration of the algorithm, the least cost paths are well known for k destination nodes.

1) Creating the topology on which we will also run our Dijkstra's algorithm and pinging two hosts. Here h1 and h4 for better reach.



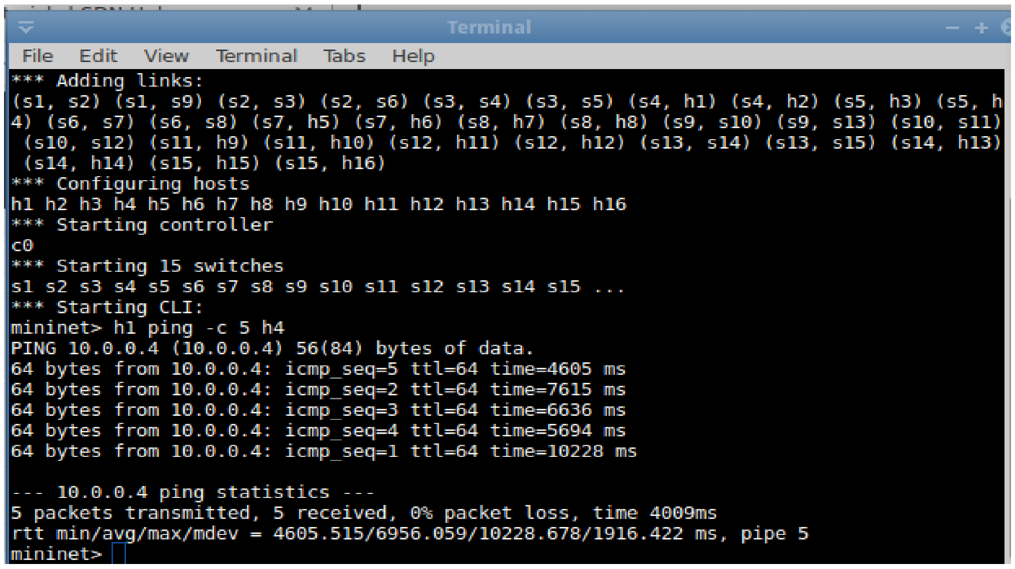
```
ubuntu@sdnhubvm:~[00:06]$ sudo mn --topo tree,depth=4,fanout=2 --controller=remote
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7 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, h
4) (s6, s7) (s6, s8) (s7, h5) (s7, h6) (s8, h7) (s8, h8) (s9, s10) (s9, s13) (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> h1 ping -c 5 h4
```

2) Running our Dijkstra's Algorithm



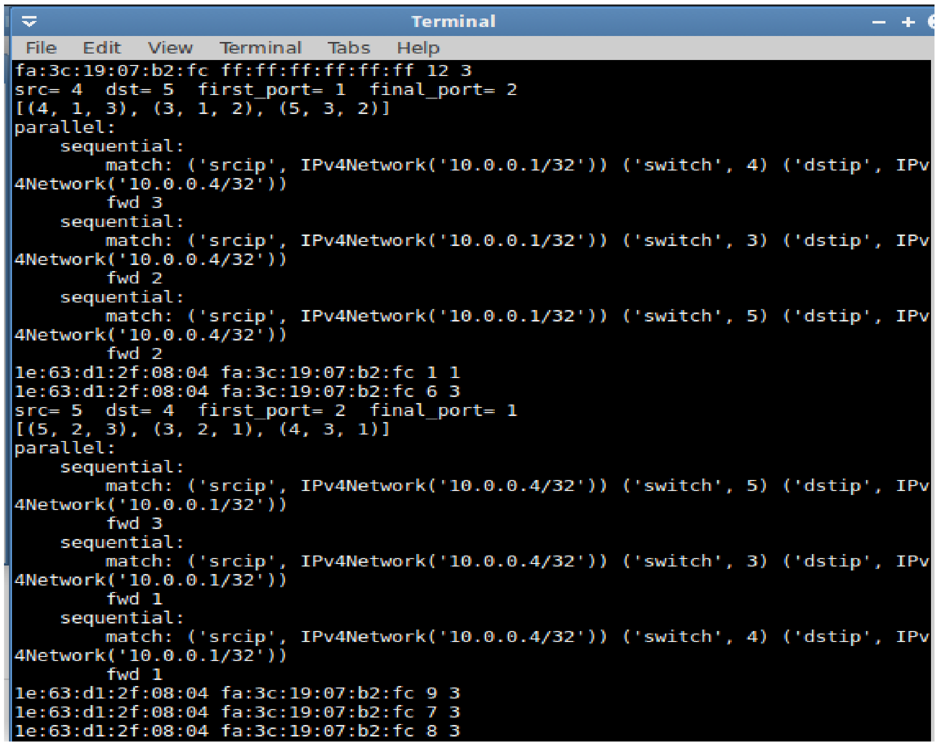
```
ubuntu@sdnhubvm:~[00:08]$ cd pyretic
ubuntu@sdnhubvm:~/pyretic[00:08] (master)$ ./pyretic.py -m p0 myroute_dijkstra
Couldn't import dot_parser, loading of dot files will not be possible.
POX 0.5.0 (eel) / Copyright 2011-2014 James McCauley, et al.
Connected to pyretic frontend.
INFO:core:POX 0.5.0 (eel) is up.
INFO:openflow.of_01:[00-00-00-00-00-06 2] connected
INFO:openflow.of_01:[00-00-00-00-00-0a 3] connected
INFO:openflow.of_01:[00-00-00-00-00-0e 4] connected
INFO:openflow.of_01:[00-00-00-00-00-01 1] connected
INFO:openflow.of_01:[00-00-00-00-00-03 5] connected
INFO:openflow.of_01:[00-00-00-00-00-0d 7] connected
INFO:openflow.of_01:[00-00-00-00-00-02 6] connected
INFO:openflow.of_01:[00-00-00-00-00-09 10] connected
INFO:openflow.of_01:[00-00-00-00-00-0f 8] connected
INFO:openflow.of_01:[00-00-00-00-00-08 9] connected
INFO:openflow.of_01:[00-00-00-00-00-0b 12] connected
INFO:openflow.of_01:[00-00-00-00-00-05 14] connected
INFO:openflow.of_01:[00-00-00-00-00-07 11] connected
INFO:openflow.of_01:[00-00-00-00-00-04 13] connected
INFO:openflow.of_01:[00-00-00-00-00-0c 15] connected
```

3) Observation that there was no packet loss for pinging from h1 to h4



```
*** Adding links:
(s1, s2) (s1, s9) (s2, s3) (s2, s6) (s3, s4) (s3, s5) (s4, h1) (s4, h2) (s5, h3) (s5, h
4) (s6, s7) (s6, s8) (s7, h5) (s7, h6) (s8, h7) (s8, h8) (s9, s10) (s9, s13) (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> h1 ping -c 5 h4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=5 ttl=64 time=4605 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=7615 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=6636 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=5694 ms
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=10228 ms

--- 10.0.0.4 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4009ms
rtt min/avg/max/mdev = 4605.515/6956.059/10228.678/1916.422 ms, pipe 5
mininet>
```

4) The results obtained are as follows .The flow of packet using the dijkstra's algorithm is shown below.



```
fa:3c:19:07:b2:fc ff:ff:ff:ff:ff:ff 12 3
src= 4  dst= 5  first_port= 1  final_port= 2
[(4, 1, 3), (3, 1, 2), (5, 3, 2)]
parallel:
    sequential:
        match: ('srcip', IPv4Network('10.0.0.1/32')) ('switch', 4) ('dstip', IPv
4Network('10.0.0.4/32'))
        fwd 3
    sequential:
        match: ('srcip', IPv4Network('10.0.0.1/32')) ('switch', 3) ('dstip', IPv
4Network('10.0.0.4/32'))
        fwd 2
    sequential:
        match: ('srcip', IPv4Network('10.0.0.1/32')) ('switch', 5) ('dstip', IPv
4Network('10.0.0.4/32'))
        fwd 2
1e:63:d1:2f:08:04 fa:3c:19:07:b2:fc 1 1
1e:63:d1:2f:08:04 fa:3c:19:07:b2:fc 6 3
src= 5  dst= 4  first_port= 2  final_port= 1
[(5, 2, 3), (3, 2, 1), (4, 3, 1)]
parallel:
    sequential:
        match: ('srcip', IPv4Network('10.0.0.4/32')) ('switch', 5) ('dstip', IPv
4Network('10.0.0.1/32'))
        fwd 3
    sequential:
        match: ('srcip', IPv4Network('10.0.0.4/32')) ('switch', 3) ('dstip', IPv
4Network('10.0.0.1/32'))
        fwd 1
    sequential:
        match: ('srcip', IPv4Network('10.0.0.4/32')) ('switch', 4) ('dstip', IPv
4Network('10.0.0.1/32'))
        fwd 1
1e:63:d1:2f:08:04 fa:3c:19:07:b2:fc 9 3
1e:63:d1:2f:08:04 fa:3c:19:07:b2:fc 7 3
1e:63:d1:2f:08:04 fa:3c:19:07:b2:fc 8 3
```

ii. Distance Vector Routing

The Distance vector algorithm is iterative, asynchronous and distributed.

- **Distributed:** It is distributed in that each node receives information from one or more of its directly attached neighbors performs calculation and then distributes the result back to its neighbors.
- **Iterative:** It is iterative in that its process continues until no more information is available to be exchanged between neighbors.
- **Asynchronous:** It does not require that all of its nodes operate in the lock step with each other.

Three Keys to understand the working of Distance Vector Routing Algorithm:

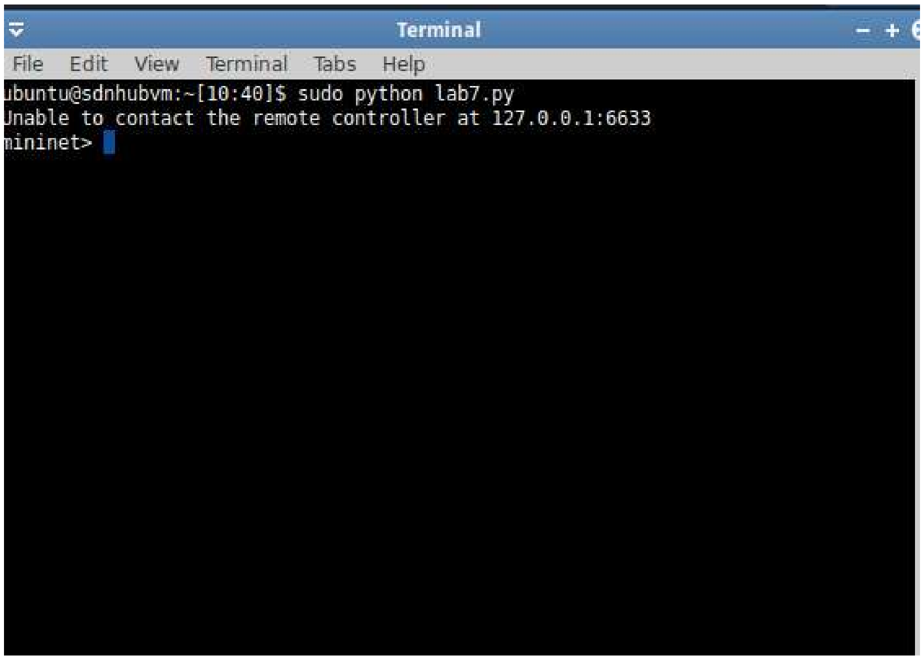
- **Knowledge about the whole network:** Each router shares its knowledge through the entire network. The Router sends its collected knowledge about the network to its neighbors.
- **Routing only to neighbors:** The router sends its knowledge about the network to only those routers which have direct links. The router sends whatever it has about the network through the ports. The information is received by the router and uses the information to update its own routing table.
- **Information sharing at regular intervals:** Within 30 seconds, the router sends the information to the neighboring routers.

Implementation:

Each node uses Bellman Ford algorithm on the graph to calculate the optimal routes to all nodes.

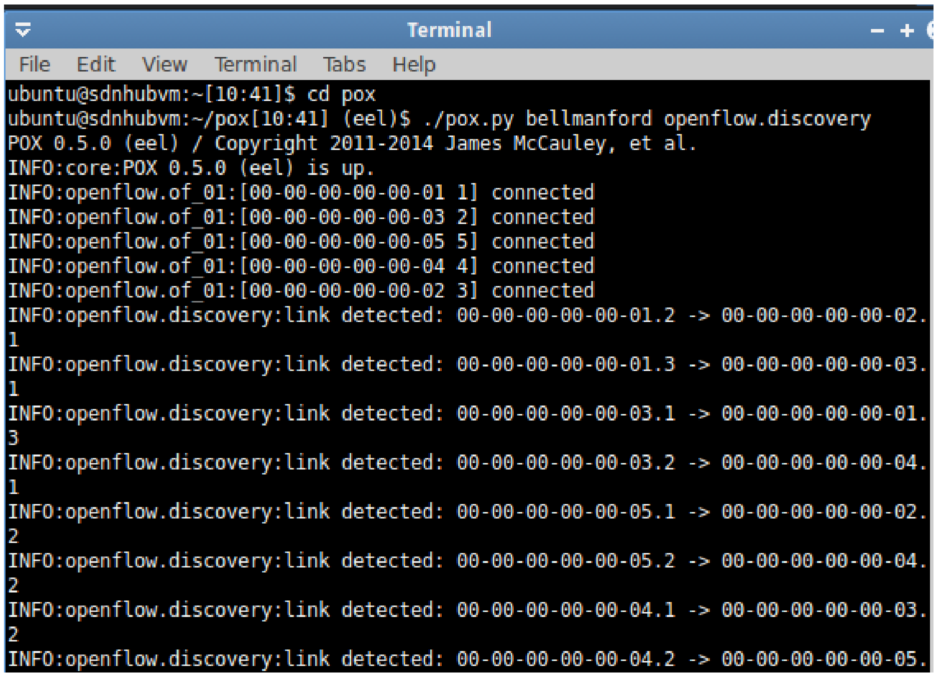
Bellman Ford Basics – Each router maintains a Distance Vector table containing the distance between itself and ALL possible destination nodes. Distances, based on a chosen metric, are computed using information from the neighbors’ distance vectors.

1) Creating the custom topology on which we will run our Bellman Ford algorithm.



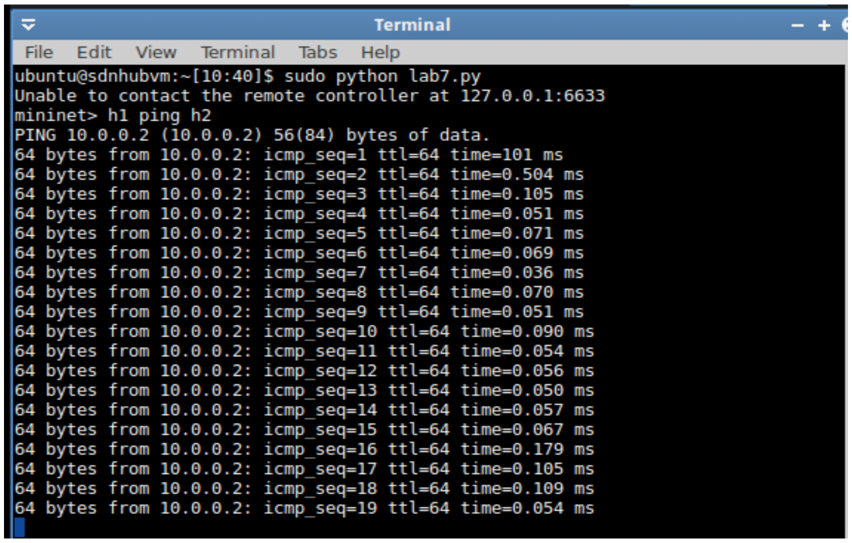
```
ubuntu@sdnhubvm:~[10:40]$ sudo python lab7.py
Unable to contact the remote controller at 127.0.0.1:6633
mininet>
```

2) Running our Bellman Ford Algorithm



```
ubuntu@sdnhubvm:~[10:41]$ cd pox
ubuntu@sdnhubvm:~/pox[10:41] (eel)$ ./pox.py bellmanford openflow.discovery
POX 0.5.0 (eel) / Copyright 2011-2014 James McCauley, et al.
INFO:core:POX 0.5.0 (eel) is up.
INFO:openflow.of_01:[00-00-00-00-00-01 1] connected
INFO:openflow.of_01:[00-00-00-00-00-03 2] connected
INFO:openflow.of_01:[00-00-00-00-00-05 5] connected
INFO:openflow.of_01:[00-00-00-00-00-04 4] connected
INFO:openflow.of_01:[00-00-00-00-00-02 3] connected
INFO:openflow.discovery:link detected: 00-00-00-00-00-01.2 -> 00-00-00-00-00-02.
1
INFO:openflow.discovery:link detected: 00-00-00-00-00-01.3 -> 00-00-00-00-00-03.
1
INFO:openflow.discovery:link detected: 00-00-00-00-00-03.1 -> 00-00-00-00-00-01.
3
INFO:openflow.discovery:link detected: 00-00-00-00-00-03.2 -> 00-00-00-00-00-04.
1
INFO:openflow.discovery:link detected: 00-00-00-00-00-05.1 -> 00-00-00-00-00-02.
2
INFO:openflow.discovery:link detected: 00-00-00-00-00-05.2 -> 00-00-00-00-00-04.
2
INFO:openflow.discovery:link detected: 00-00-00-00-00-04.1 -> 00-00-00-00-00-03.
2
INFO:openflow.discovery:link detected: 00-00-00-00-00-04.2 -> 00-00-00-00-00-05.
```

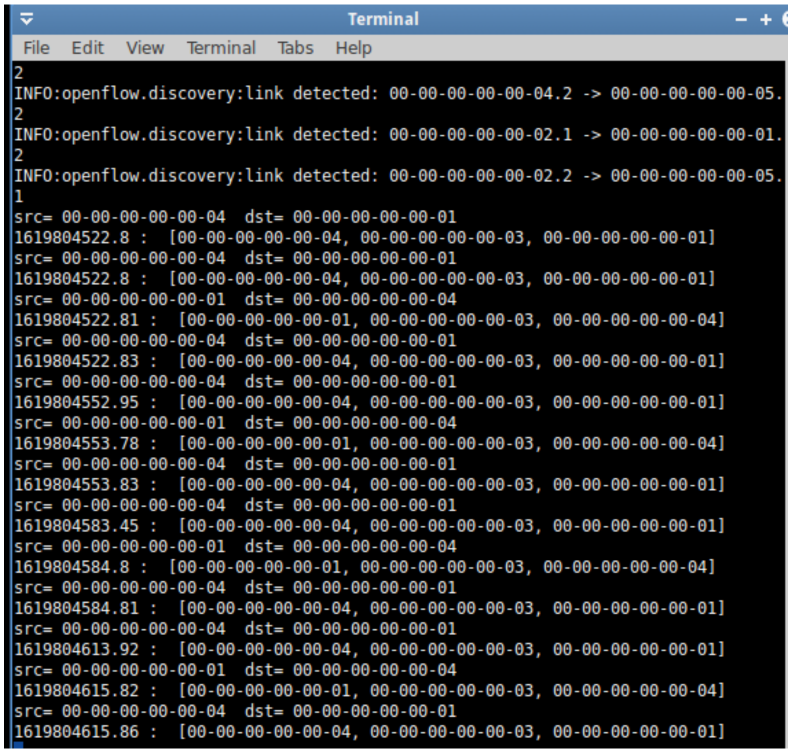
3) Observation that there was no packet loss for pinging from h1 to h2



```
ubuntu@sdnhubvm:~[10:40]$ sudo python lab7.py
Unable to contact the remote controller at 127.0.0.1:6633
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=101 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.504 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.105 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.051 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.071 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.069 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.036 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.070 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.051 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.090 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=0.054 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=0.056 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=0.050 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=0.057 ms
64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=0.067 ms
64 bytes from 10.0.0.2: icmp_seq=16 ttl=64 time=0.179 ms
64 bytes from 10.0.0.2: icmp_seq=17 ttl=64 time=0.105 ms
64 bytes from 10.0.0.2: icmp_seq=18 ttl=64 time=0.109 ms
64 bytes from 10.0.0.2: icmp_seq=19 ttl=64 time=0.054 ms
```

4) The results obtained are as follows .The flow of packet using the Bellman Ford algorithm is shown below.

Interpretation: From the below figure, we can clearly see that the bellman ford finds a way for h1 to h2 via s1, s3 and s4.

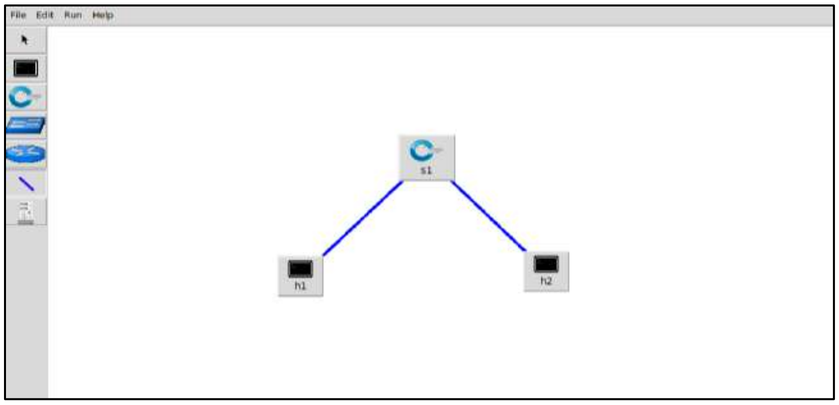


```
2
INFO:openflow.discovery:link detected: 00-00-00-00-00-04.2 -> 00-00-00-00-00-05.
2
INFO:openflow.discovery:link detected: 00-00-00-00-00-02.1 -> 00-00-00-00-00-01.
2
INFO:openflow.discovery:link detected: 00-00-00-00-00-02.2 -> 00-00-00-00-00-05.
1
src= 00-00-00-00-00-04  dst= 00-00-00-00-00-01
1619804522.8 :  [00-00-00-00-00-04, 00-00-00-00-00-03, 00-00-00-00-00-01]
src= 00-00-00-00-00-04  dst= 00-00-00-00-00-01
1619804522.8 :  [00-00-00-00-00-04, 00-00-00-00-00-03, 00-00-00-00-00-01]
src= 00-00-00-00-00-01  dst= 00-00-00-00-00-04
1619804522.81 :  [00-00-00-00-00-01, 00-00-00-00-00-03, 00-00-00-00-00-04]
src= 00-00-00-00-00-04  dst= 00-00-00-00-00-01
1619804522.83 :  [00-00-00-00-00-04, 00-00-00-00-00-03, 00-00-00-00-00-01]
src= 00-00-00-00-00-04  dst= 00-00-00-00-00-01
1619804552.95 :  [00-00-00-00-00-04, 00-00-00-00-00-03, 00-00-00-00-00-01]
src= 00-00-00-00-00-01  dst= 00-00-00-00-00-04
1619804553.78 :  [00-00-00-00-00-01, 00-00-00-00-00-03, 00-00-00-00-00-04]
src= 00-00-00-00-00-04  dst= 00-00-00-00-00-01
1619804553.83 :  [00-00-00-00-00-04, 00-00-00-00-00-03, 00-00-00-00-00-01]
src= 00-00-00-00-00-04  dst= 00-00-00-00-00-01
1619804583.45 :  [00-00-00-00-00-04, 00-00-00-00-00-03, 00-00-00-00-00-01]
src= 00-00-00-00-00-01  dst= 00-00-00-00-00-04
1619804584.8 :  [00-00-00-00-00-01, 00-00-00-00-00-03, 00-00-00-00-00-04]
src= 00-00-00-00-00-04  dst= 00-00-00-00-00-01
1619804584.81 :  [00-00-00-00-00-04, 00-00-00-00-00-03, 00-00-00-00-00-01]
src= 00-00-00-00-00-04  dst= 00-00-00-00-00-01
1619804613.92 :  [00-00-00-00-00-04, 00-00-00-00-00-03, 00-00-00-00-00-01]
src= 00-00-00-00-00-01  dst= 00-00-00-00-00-04
1619804615.82 :  [00-00-00-00-00-01, 00-00-00-00-00-03, 00-00-00-00-00-04]
src= 00-00-00-00-00-04  dst= 00-00-00-00-00-01
1619804615.86 :  [00-00-00-00-00-04, 00-00-00-00-00-03, 00-00-00-00-00-01]
```

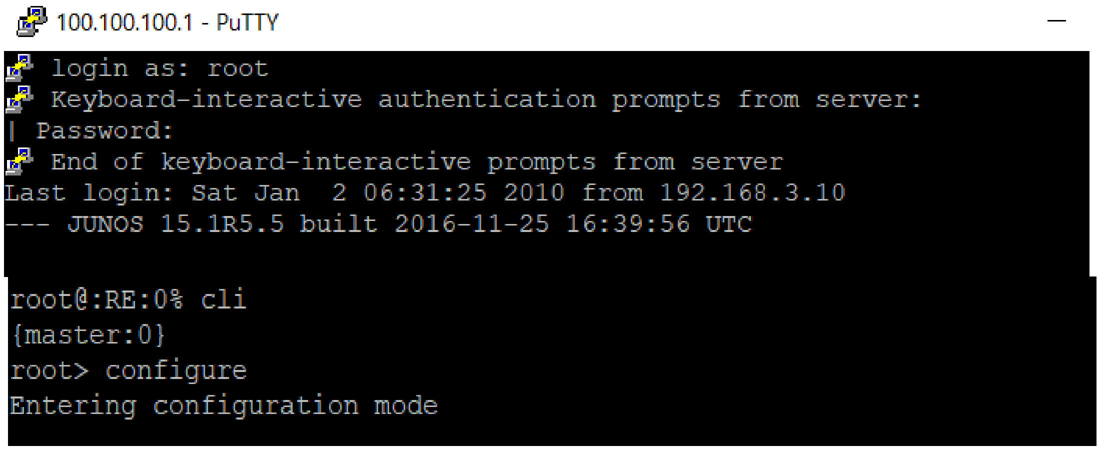
IV. Junos

The Junos operating system (**Junos OS**) used in Juniper Networks network devices creates an environment for accelerating the deployment of services and applications over a single network.
Unlike other network operating systems that share a common name but splinter into many different programs, Junos OS is a single, cohesive operating system that is shared across all network devices and product lines. Junos OS allows Juniper Networks engineers to develop software features once and share these features across all product lines simultaneously, thus reducing the training for each product and interoperability in production environments.

i. VLAN Configuration



Management Device

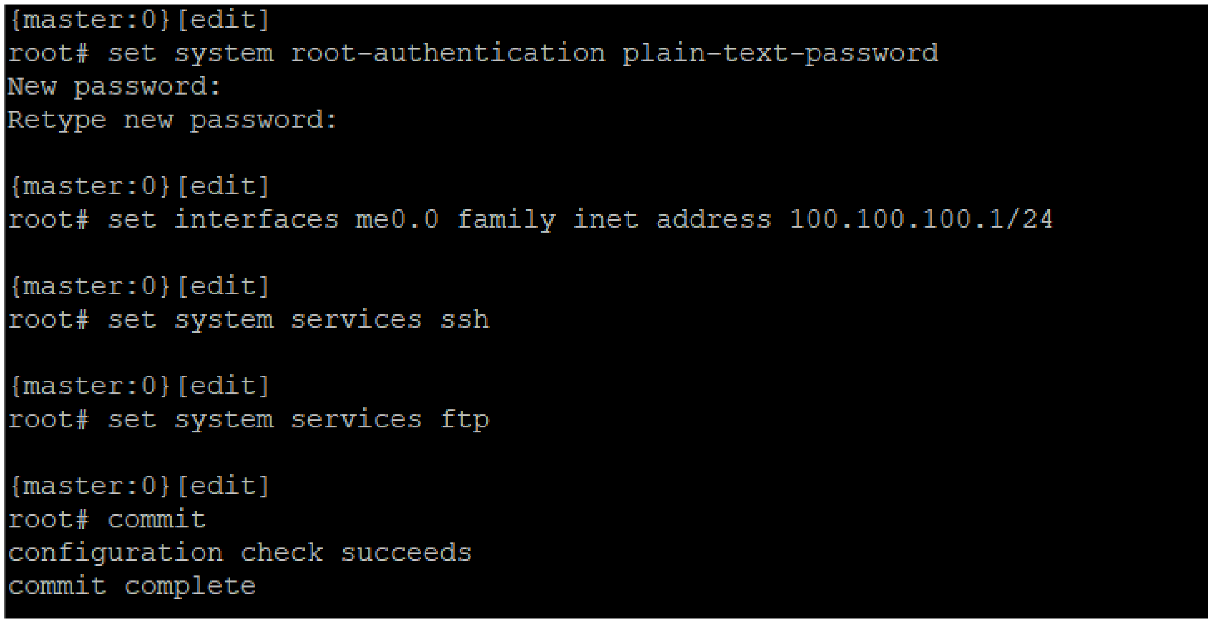


```
login as: root
Keyboard-interactive authentication prompts from server:
| Password:
End of keyboard-interactive prompts from server
Last login: Sat Jan  2 06:31:25 2010 from 192.168.3.10
--- JUNOS 15.1R5.5 built 2016-11-25 16:39:56 UTC

root@:RE:0% cli
{master:0}
root> configure
Entering configuration mode
```

Reset

```
root> configure
root# set system root-authentication plain-text-password
root# set interfaces me0.0 family inet address 100.100.100.1/24
```

root# set system services ssh
root# set system services ftp
root# commit



```
{master:0}[edit]
root# set system root-authentication plain-text-password
New password:
Retype new password:

{master:0}[edit]
root# set interfaces me0.0 family inet address 100.100.100.1/24

{master:0}[edit]
root# set system services ssh

{master:0}[edit]
root# set system services ftp

{master:0}[edit]
root# commit
configuration check succeeds
commit complete
```

SCRIPT :

```
set vlans DATA vlan-id  10
set vlans VOICE vlan-id 20
set interfaces ge-0/0/2 unit 0 family Ethernet switching vlan members DATA
set interfaces ge-0/0/5 unit 0 family Ethernet switching vlan members VOICE
set interfaces vlan unit 10 family inet address 192.168.3.1/24
set interfaces vlan unit 20 family inet address 192.168.4.1/24
set vlans DATA l3-interface vlan.10
set vlans VOICE l3-interface vlan.20
```

ii. OSPF Configuration

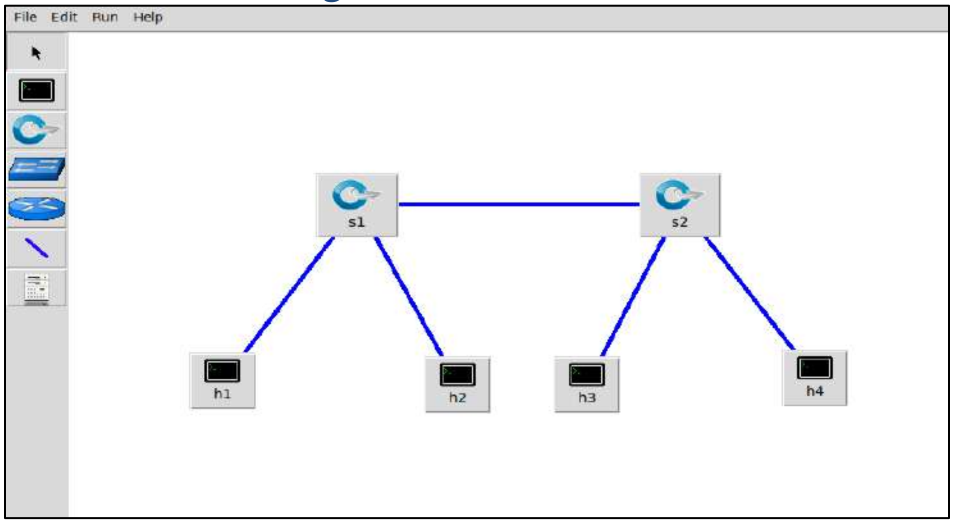


Figure: OSPF topology in miniedit

Scripts:

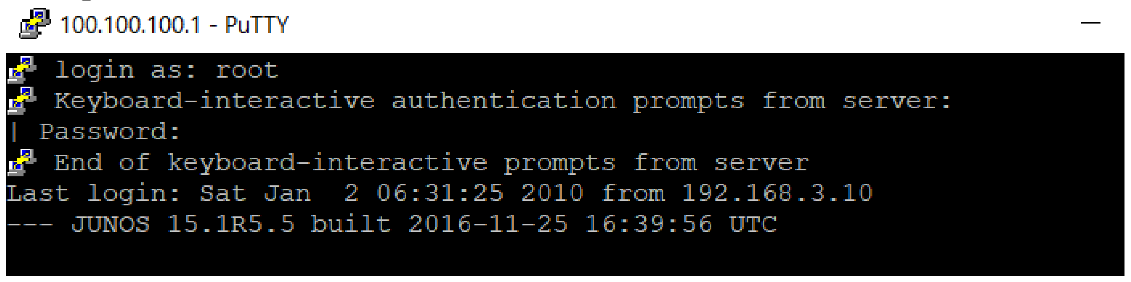
Device 1:

```
set interfaces ge-0/0/0 unit 0 family inet address 10.10.10.1/24
set interfaces ge-0/0/1 unit 0 family inet address 10.10.10.2/24
set interfaces ge-0/0/2 unit 0 family ethernet-switching vlan members VOICE
set interfaces me0 unit 0 family inet address 100.100.100.1/30
set interfaces vlan unit 10 family inet address 192.168.1.1/24
set protocols ospf area 0.0.0.10 interface ge-0/0/0.0 metric 5
set protocols ospf area 0.0.0.10 interface ge-0/0/1.0 metric 10
set protocols ospf area 0.0.0.10 interface vlan.10
set vlans VOICE vlan-id 10
set vlans VOICE l3-interface vlan.10
```

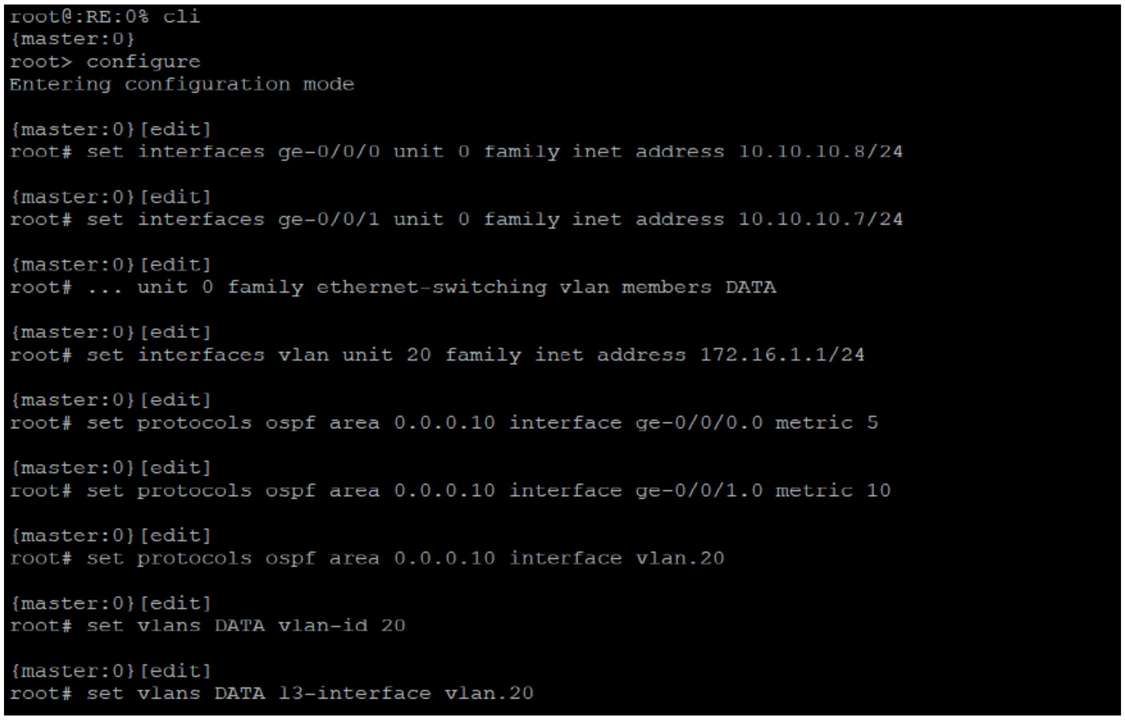
Device 2:

```
set interfaces ge-0/0/0 unit 0 family inet address 10.10.10.8/24
set interfaces ge-0/0/1 unit 0 family inet address 10.10.10.7/24
set interfaces ge-0/0/5 unit 0 family inet address 10.10.10.9/24
set interfaces ge-0/0/2 unit 0 family ethernet-switching vlan members DATA
set interfaces vlan unit 20 family inet address 172.16.1.1/24
set protocols ospf area 0.0.0.10 interface ge-0/0/0.0 metric 5
set protocols ospf area 0.0.0.10 interface ge-0/0/1.0 metric 10
set protocols ospf area 0.0.0.10 interface ge-0/0/5.0 metric 3
set protocols ospf area 0.0.0.10 interface vlan.20
set vlans DATA vlan-id 20
set vlans DATA l3-interface vlan.20
```

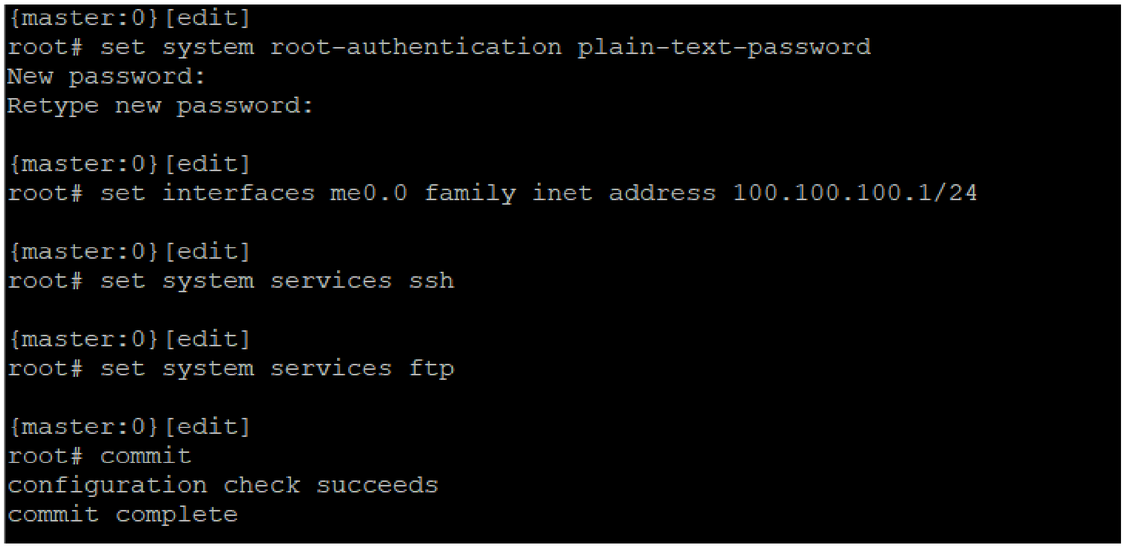
Management Device



```
login as: root
Keyboard-interactive authentication prompts from server:
| Password:
End of keyboard-interactive prompts from server
Last login: Sat Jan  2 06:31:25 2010 from 192.168.3.10
--- JUNOS 15.1R5.5 built 2016-11-25 16:39:56 UTC
```

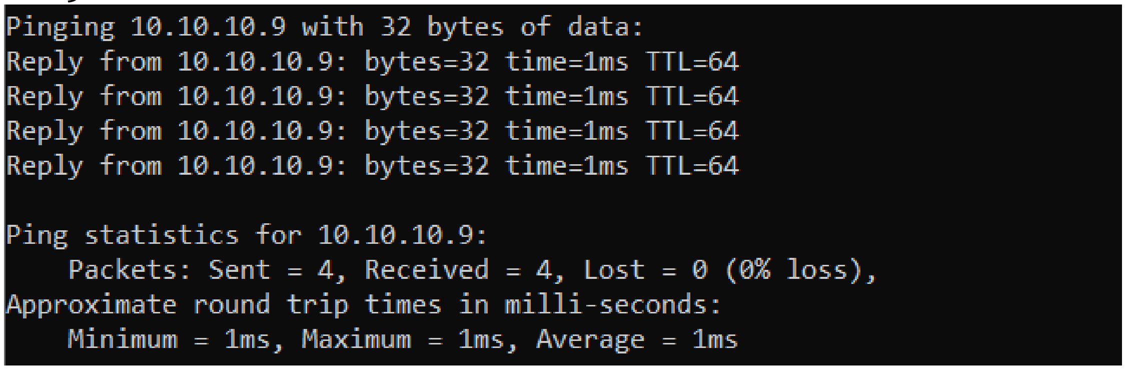


```
root@:RE:0% cli
{master:0}
root> configure
Entering configuration mode

{master:0}[edit]
root# set interfaces ge-0/0/0 unit 0 family inet address 10.10.10.8/24

{master:0}[edit]
root# set interfaces ge-0/0/1 unit 0 family inet address 10.10.10.7/24

{master:0}[edit]
root# ... unit 0 family ethernet-switching vlan members DATA

{master:0}[edit]
root# set interfaces vlan unit 20 family inet address 172.16.1.1/24

{master:0}[edit]
root# set protocols ospf area 0.0.0.10 interface ge-0/0/0.0 metric 5

{master:0}[edit]
root# set protocols ospf area 0.0.0.10 interface ge-0/0/1.0 metric 10

{master:0}[edit]
root# set protocols ospf area 0.0.0.10 interface vlan.20

{master:0}[edit]
root# set vlans DATA vlan-id 20

{master:0}[edit]
root# set vlans DATA l3-interface vlan.20
```



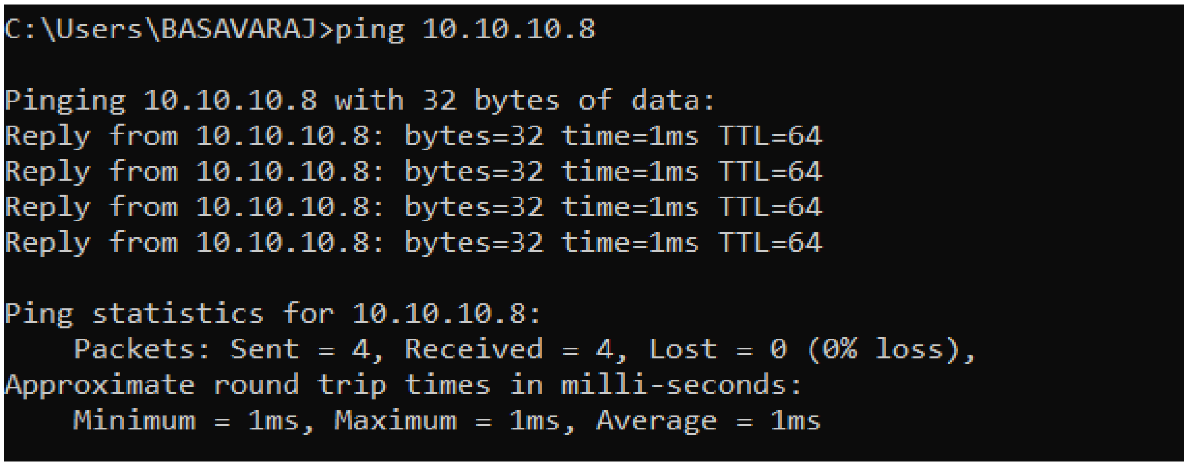
```
{master:0}[edit]
root# set system root-authentication plain-text-password
New password:
Retype new password:

{master:0}[edit]
root# set interfaces me0.0 family inet address 100.100.100.1/24

{master:0}[edit]
root# set system services ssh

{master:0}[edit]
root# set system services ftp

{master:0}[edit]
root# commit
configuration check succeeds
commit complete
```

Port 5



```
Pinging 10.10.10.9 with 32 bytes of data:
Reply from 10.10.10.9: bytes=32 time=1ms TTL=64
Reply from 10.10.10.9: bytes=32 time=1ms TTL=64
Reply from 10.10.10.9: bytes=32 time=1ms TTL=64
Reply from 10.10.10.9: bytes=32 time=1ms TTL=64

Ping statistics for 10.10.10.9:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 1ms, Average = 1ms
```

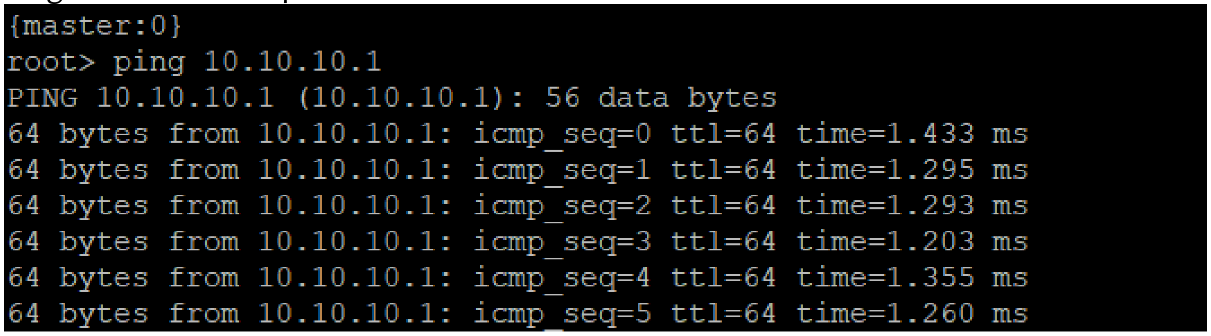
Port 0



```
C:\Users\BASAVARAJ>ping 10.10.10.8

Pinging 10.10.10.8 with 32 bytes of data:
Reply from 10.10.10.8: bytes=32 time=1ms TTL=64
Reply from 10.10.10.8: bytes=32 time=1ms TTL=64
Reply from 10.10.10.8: bytes=32 time=1ms TTL=64
Reply from 10.10.10.8: bytes=32 time=1ms TTL=64

Ping statistics for 10.10.10.8:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 1ms, Average = 1ms
```

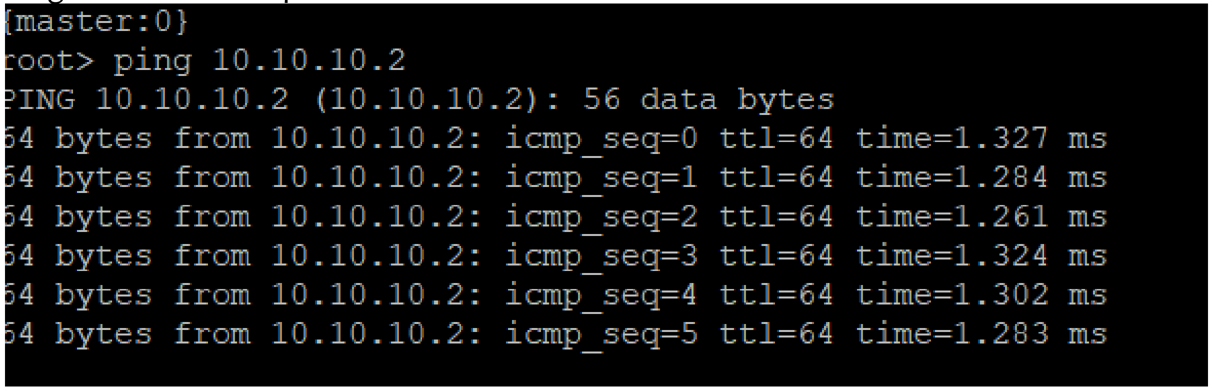
Port 1

Ping from master to port 0

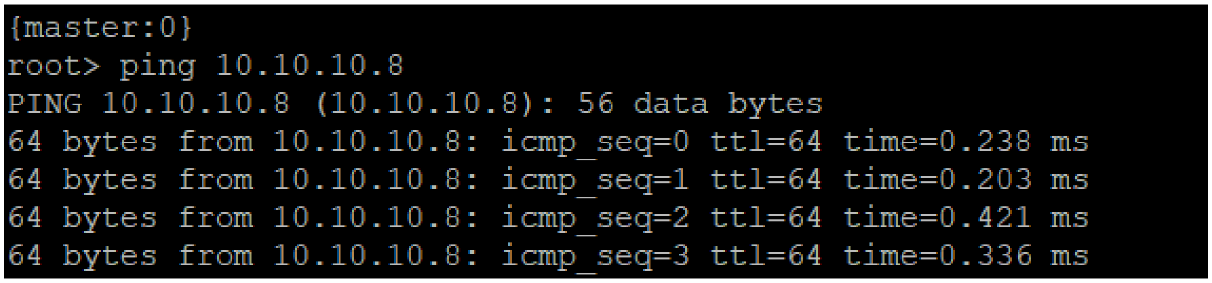


```
{master:0}
root> ping 10.10.10.1
PING 10.10.10.1 (10.10.10.1): 56 data bytes
64 bytes from 10.10.10.1: icmp_seq=0 ttl=64 time=1.433 ms
64 bytes from 10.10.10.1: icmp_seq=1 ttl=64 time=1.295 ms
64 bytes from 10.10.10.1: icmp_seq=2 ttl=64 time=1.293 ms
64 bytes from 10.10.10.1: icmp_seq=3 ttl=64 time=1.203 ms
64 bytes from 10.10.10.1: icmp_seq=4 ttl=64 time=1.355 ms
64 bytes from 10.10.10.1: icmp_seq=5 ttl=64 time=1.260 ms
```

Ping from master to port 1



```
{master:0}
root> ping 10.10.10.2
PING 10.10.10.2 (10.10.10.2): 56 data bytes
64 bytes from 10.10.10.2: icmp_seq=0 ttl=64 time=1.327 ms
64 bytes from 10.10.10.2: icmp_seq=1 ttl=64 time=1.284 ms
64 bytes from 10.10.10.2: icmp_seq=2 ttl=64 time=1.261 ms
64 bytes from 10.10.10.2: icmp_seq=3 ttl=64 time=1.324 ms
64 bytes from 10.10.10.2: icmp_seq=4 ttl=64 time=1.302 ms
64 bytes from 10.10.10.2: icmp_seq=5 ttl=64 time=1.283 ms
```



```
{master:0}
root> ping 10.10.10.8
PING 10.10.10.8 (10.10.10.8): 56 data bytes
64 bytes from 10.10.10.8: icmp_seq=0 ttl=64 time=0.238 ms
64 bytes from 10.10.10.8: icmp_seq=1 ttl=64 time=0.203 ms
64 bytes from 10.10.10.8: icmp_seq=2 ttl=64 time=0.421 ms
64 bytes from 10.10.10.8: icmp_seq=3 ttl=64 time=0.336 ms
```

iii. IBGP Configuration

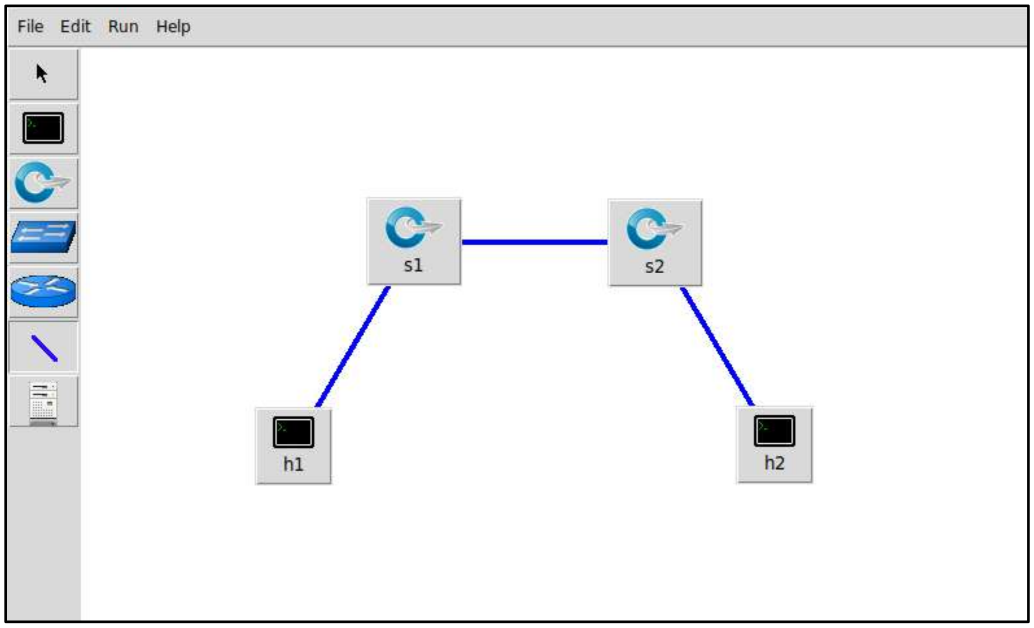


Figure: IBGP topology in miniedit

Scripts:

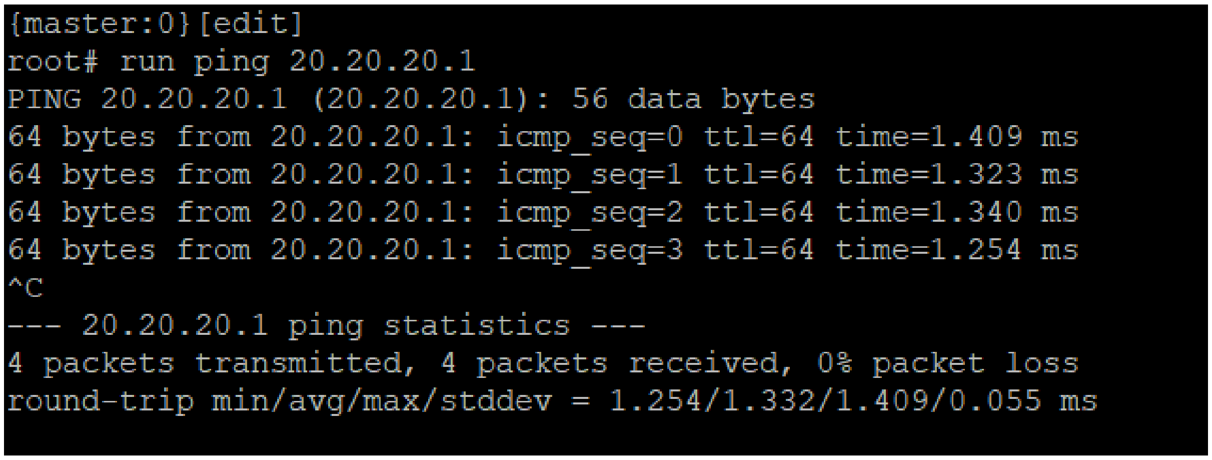
Device 1

```
set interfaces ge-0/0/2 unit 0 family ethernet-switching vlan members study
set interfaces ge-0/0/23 unit 0 family inet address 20.20.20.1/30
set interfaces vlan unit 1 family inet address 192.168.1.1/24
set routing-options autonomous-system 2000
set protocols bgp group BGP_INT type internal
set protocols bgp group BGP_INT export ADVERTISE_INT
set protocols bgp group BGP_INT peer-as 2000
set protocols bgp group BGP_INT neighbor 20.20.20.2
set policy-options policy-statement ADVERTISE_INT term 1 from protocol bgp
set policy-options policy-statement ADVERTISE_INT term 1 then accept
set policy-options policy-statement ADVERTISE_INT term 2 from protocol direct
set policy-options policy-statement ADVERTISE_INT term 2 then accept
set policy-options policy-statement ADVERTISE_INT term 3 then reject
set vlans study vlan-id 1
set vlans study l3-interface vlan.1
```

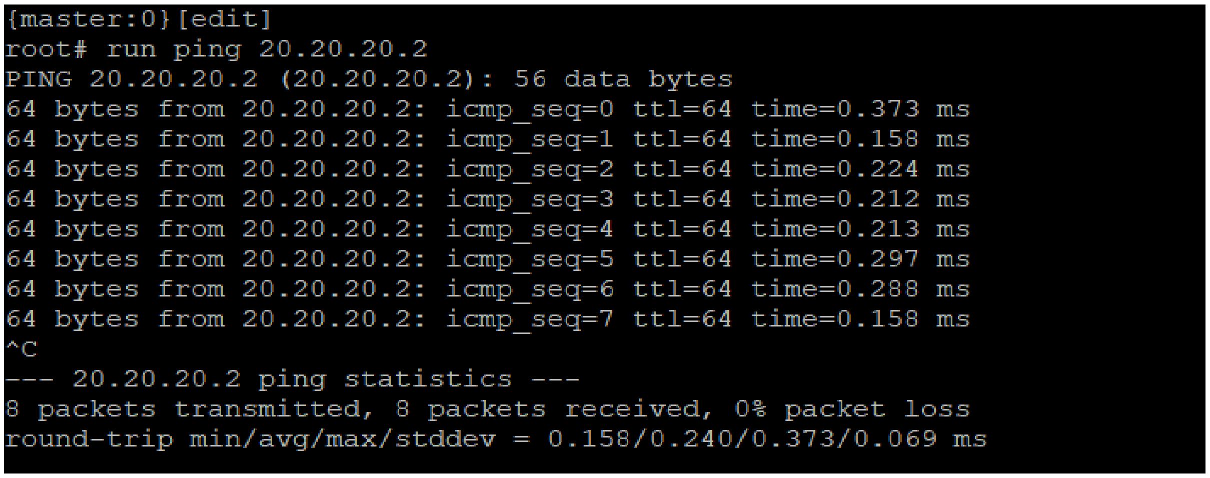
Device 2

```
set interfaces ge-0/0/2 unit 0 family ethernet-switching vlan members DATA
set interfaces ge-0/0/23 unit 0 family inet address 20.20.20.2/30
set interfaces vlan unit 10 family inet address 192.168.3.1/24
set routing-options autonomous-system 2000
```

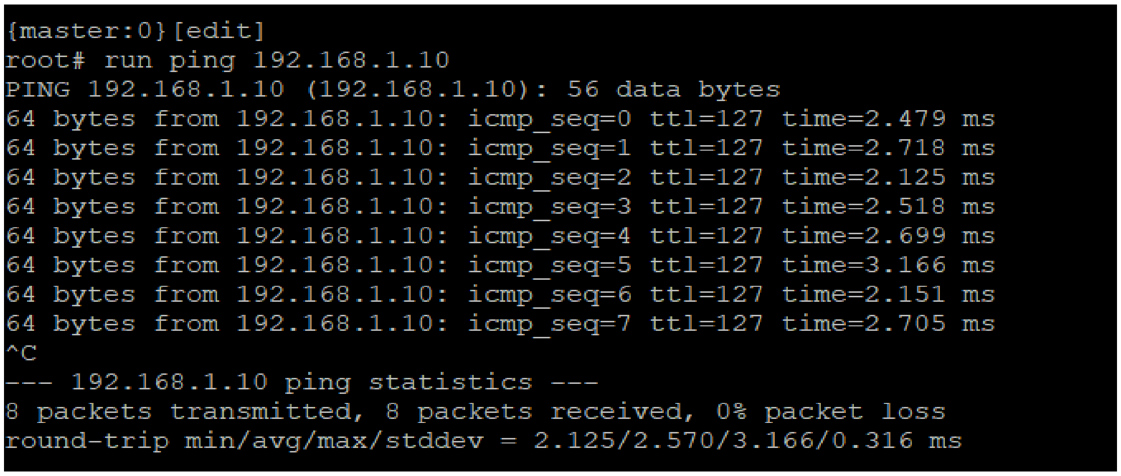
set protocols bgp group INT-BGP type internal
set protocols bgp group INT-BGP export ADVERTISE_INT
set protocols bgp group INT-BGP peer-as 2000
set protocols bgp group INT-BGP neighbor 20.20.20.1
set policy-options policy-statement ADVERTISE_INT term 1 from protocol bgp
set policy-options policy-statement ADVERTISE_INT term 1 then accept
set policy-options policy-statement ADVERTISE_INT term 2 from protocol direct
set policy-options policy-statement ADVERTISE_INT term 2 then accept
set policy-options policy-statement ADVERTISE_INT term 3 then reject
set vlans DATA vlan-id 10
set vlans DATA l3-interface vlan.10



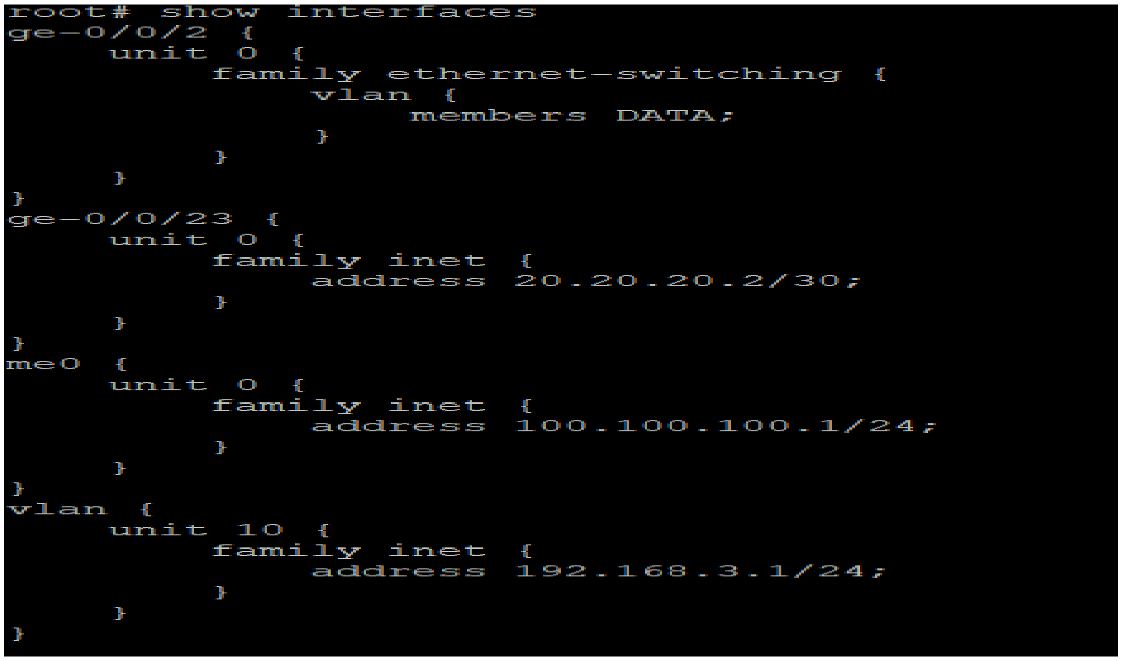
```
{master:0}[edit]
root# run ping 20.20.20.1
PING 20.20.20.1 (20.20.20.1): 56 data bytes
64 bytes from 20.20.20.1: icmp_seq=0 ttl=64 time=1.409 ms
64 bytes from 20.20.20.1: icmp_seq=1 ttl=64 time=1.323 ms
64 bytes from 20.20.20.1: icmp_seq=2 ttl=64 time=1.340 ms
64 bytes from 20.20.20.1: icmp_seq=3 ttl=64 time=1.254 ms
^C
--- 20.20.20.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 1.254/1.332/1.409/0.055 ms
```



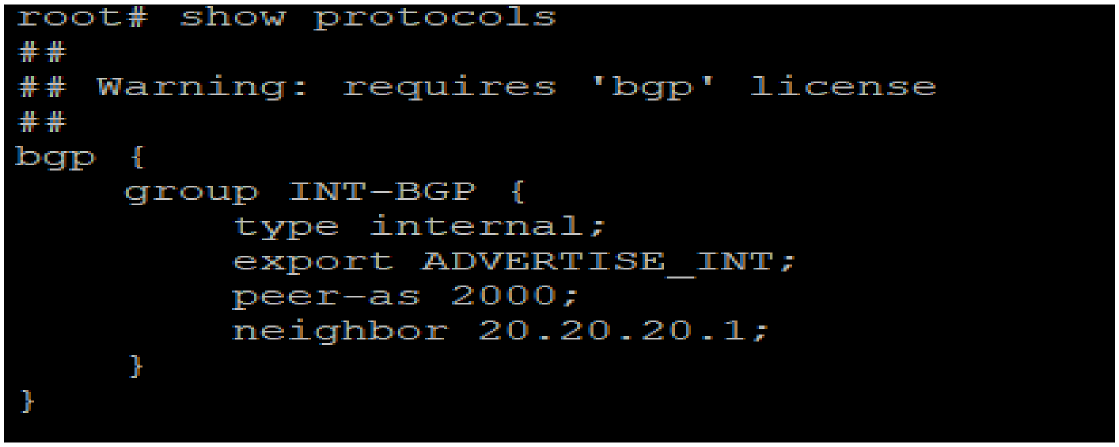
```
{master:0}[edit]
root# run ping 20.20.20.2
PING 20.20.20.2 (20.20.20.2): 56 data bytes
64 bytes from 20.20.20.2: icmp_seq=0 ttl=64 time=0.373 ms
64 bytes from 20.20.20.2: icmp_seq=1 ttl=64 time=0.158 ms
64 bytes from 20.20.20.2: icmp_seq=2 ttl=64 time=0.224 ms
64 bytes from 20.20.20.2: icmp_seq=3 ttl=64 time=0.212 ms
64 bytes from 20.20.20.2: icmp_seq=4 ttl=64 time=0.213 ms
64 bytes from 20.20.20.2: icmp_seq=5 ttl=64 time=0.297 ms
64 bytes from 20.20.20.2: icmp_seq=6 ttl=64 time=0.288 ms
64 bytes from 20.20.20.2: icmp_seq=7 ttl=64 time=0.158 ms
^C
--- 20.20.20.2 ping statistics ---
8 packets transmitted, 8 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.158/0.240/0.373/0.069 ms
```



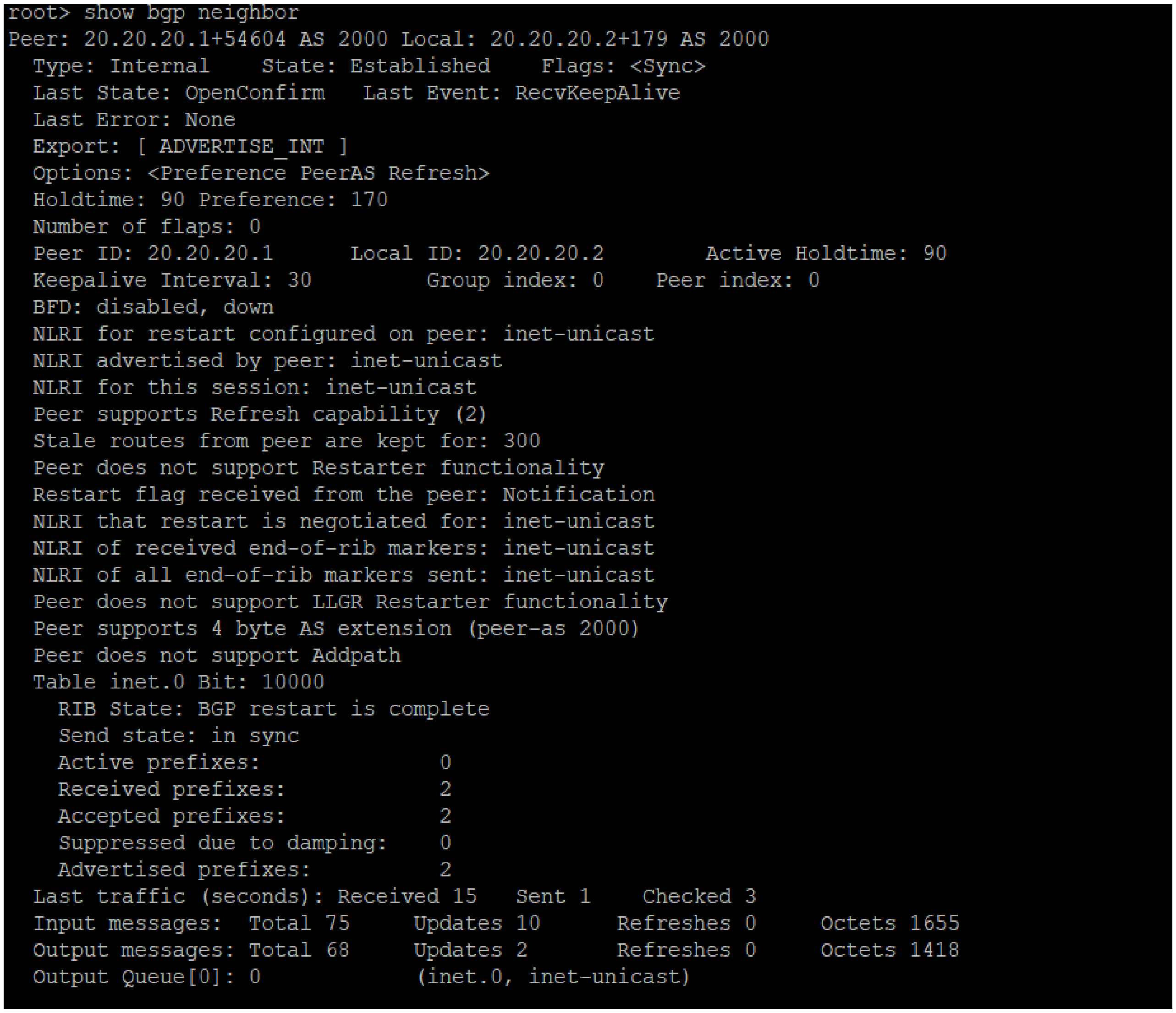
```
{master:0}[edit]
root# run ping 192.168.1.10
PING 192.168.1.10 (192.168.1.10): 56 data bytes
64 bytes from 192.168.1.10: icmp_seq=0 ttl=127 time=2.479 ms
64 bytes from 192.168.1.10: icmp_seq=1 ttl=127 time=2.718 ms
64 bytes from 192.168.1.10: icmp_seq=2 ttl=127 time=2.125 ms
64 bytes from 192.168.1.10: icmp_seq=3 ttl=127 time=2.518 ms
64 bytes from 192.168.1.10: icmp_seq=4 ttl=127 time=2.699 ms
64 bytes from 192.168.1.10: icmp_seq=5 ttl=127 time=3.166 ms
64 bytes from 192.168.1.10: icmp_seq=6 ttl=127 time=2.151 ms
64 bytes from 192.168.1.10: icmp_seq=7 ttl=127 time=2.705 ms
^C
--- 192.168.1.10 ping statistics ---
8 packets transmitted, 8 packets received, 0% packet loss
round-trip min/avg/max/stddev = 2.125/2.570/3.166/0.316 ms
```



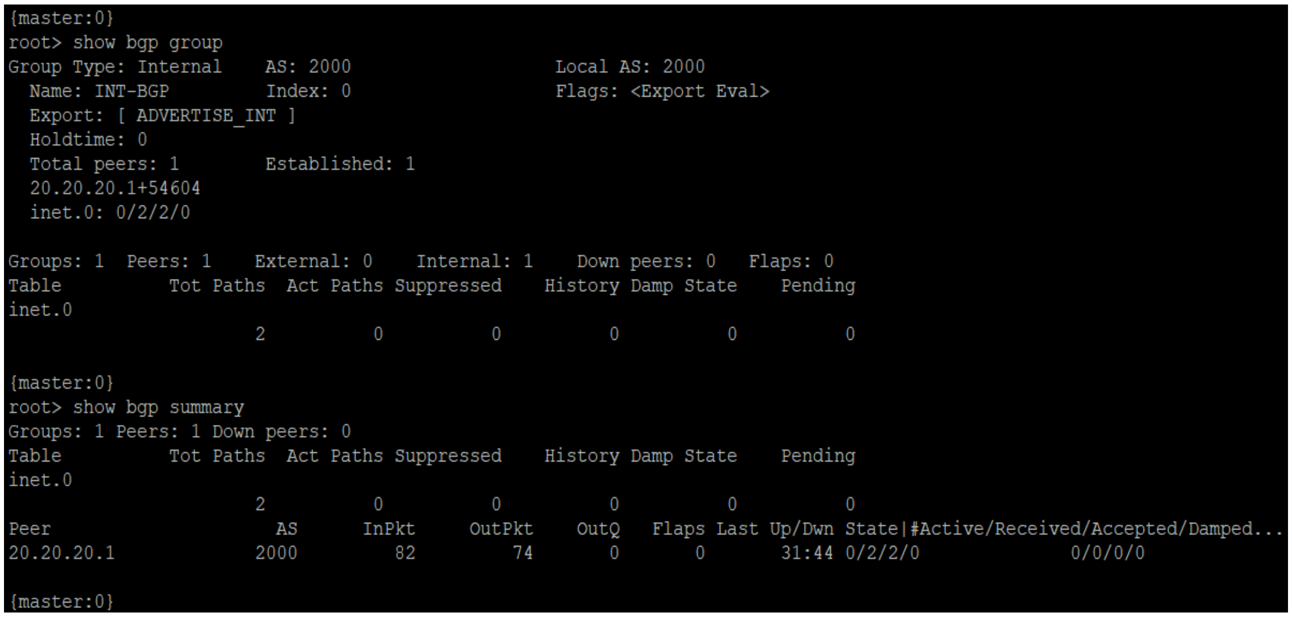
```
root# show interfaces
ge-0/0/2 {
    unit 0 {
        family ethernet-switching {
            vlan {
                members DATA;
            }
        }
    }
}
ge-0/0/23 {
    unit 0 {
        family inet {
            address 20.20.20.2/30;
        }
    }
}
me0 {
    unit 0 {
        family inet {
            address 100.100.100.1/24;
        }
    }
}
vlan {
    unit 10 {
        family inet {
            address 192.168.3.1/24;
        }
    }
}
```



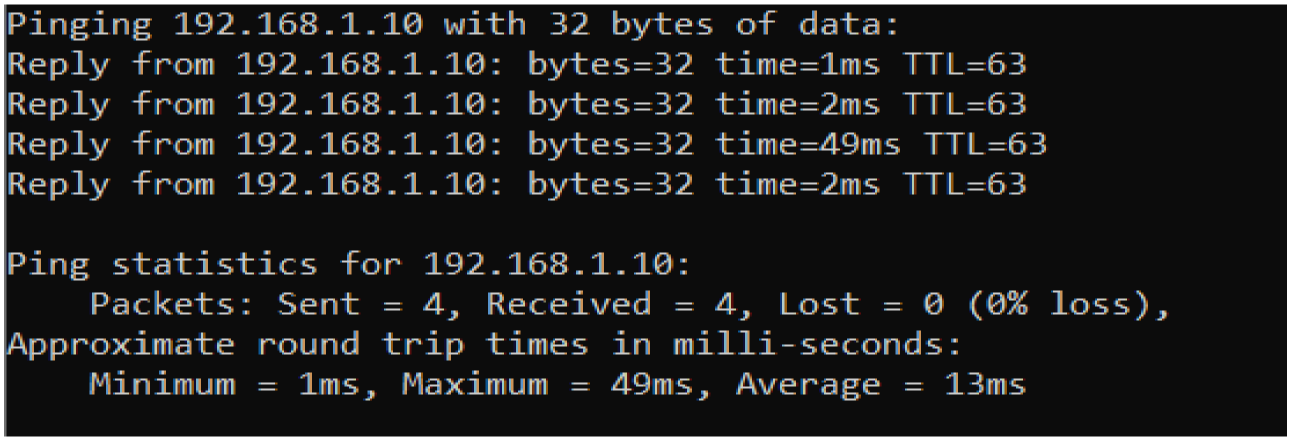
```
root# show protocols
##
## Warning: requires 'bgp' license
##
bgp {
    group INT-BGP {
        type internal;
        export ADVERTISE_INT;
        peer-as 2000;
        neighbor 20.20.20.1;
    }
}
```



```
root> show bgp neighbor
Peer: 20.20.20.1+54604 AS 2000 Local: 20.20.20.2+179 AS 2000
  Type: Internal    State: Established    Flags: <Sync>
  Last State: OpenConfirm   Last Event: RecvKeepAlive
  Last Error: None
  Export: [ ADVERTISE_INT ]
  Options: <Preference PeerAS Refresh>
  Holdtime: 90 Preference: 170
  Number of flaps: 0
  Peer ID: 20.20.20.1      Local ID: 20.20.20.2        Active Holdtime: 90
  Keepalive Interval: 30         Group index: 0    Peer index: 0
  BFD: disabled, down
  NLRI for restart configured on peer: inet-unicast
  NLRI advertised by peer: inet-unicast
  NLRI for this session: inet-unicast
  Peer supports Refresh capability (2)
  Stale routes from peer are kept for: 300
  Peer does not support Restarter functionality
  Restart flag received from the peer: Notification
  NLRI that restart is negotiated for: inet-unicast
  NLRI of received end-of-rib markers: inet-unicast
  NLRI of all end-of-rib markers sent: inet-unicast
  Peer does not support LLGR Restarter functionality
  Peer supports 4 byte AS extension (peer-as 2000)
  Peer does not support Addpath
  Table inet.0 Bit: 10000
    RIB State: BGP restart is complete
    Send state: in sync
    Active prefixes:              0
    Received prefixes:            2
    Accepted prefixes:            2
    Suppressed due to damping:    0
    Advertised prefixes:          2
  Last traffic (seconds): Received 15   Sent 1    Checked 3
  Input messages:  Total 75     Updates 10      Refreshes 0     Octets 1655
  Output messages: Total 68     Updates 2       Refreshes 0     Octets 1418
  Output Queue[0]: 0            (inet.0, inet-unicast)
```



```
{master:0}
root> show bgp group
Group Type: Internal    AS: 2000                   Local AS: 2000
  Name: INT-BGP         Index: 0                   Flags: <Export Eval>
  Export: [ ADVERTISE_INT ]
  Holdtime: 0
  Total peers: 1        Established: 1
  20.20.20.1+54604
  inet.0: 0/2/2/0

Groups: 1  Peers: 1    External: 0    Internal: 1    Down peers: 0   Flaps: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
inet.0
                       2          0          0          0          0          0

{master:0}
root> show bgp summary
Groups: 1 Peers: 1 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
inet.0
                       2          0          0          0          0          0
Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/Received/Accepted/Damped...
20.20.20.1             2000         82         74       0       0       31:44 0/2/2/0              0/0/0/0

{master:0}
```

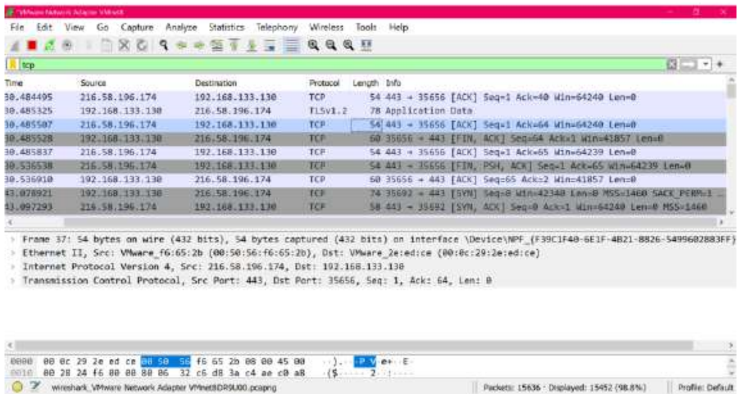


```
Pinging 192.168.1.10 with 32 bytes of data:
Reply from 192.168.1.10: bytes=32 time=1ms TTL=63
Reply from 192.168.1.10: bytes=32 time=2ms TTL=63
Reply from 192.168.1.10: bytes=32 time=49ms TTL=63
Reply from 192.168.1.10: bytes=32 time=2ms TTL=63

Ping statistics for 192.168.1.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 49ms, Average = 13ms
```

V. TP 1: Introduction to Mininet (TCP/IP Networking)

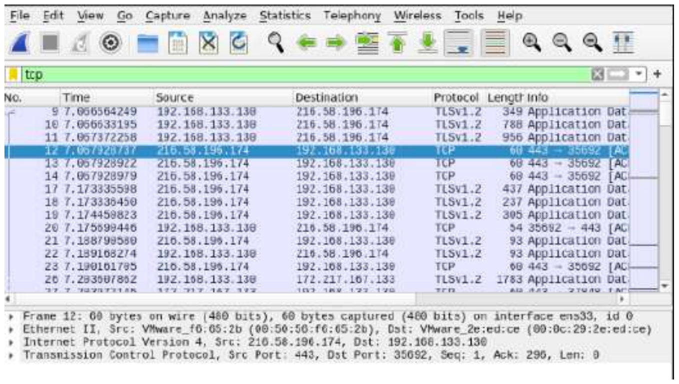
Q1) Start Wireshark on your host (i.e. personal computer) and guest (i.e. virtual machine inside personalcomputer) machines. Then connect a TCP iperf client on your guest machine to iperf.he.net (iperf -c iperf.he.net. Note: If you cannot connect with this command, try to change the server port tolisten on using the -p option: iperf -p port number -c iperf.he.net with port number equal to 5201, 5202, etc.) Compare the traffics captured on your host and guest machines. What are the source and destination IP addresses and port numbers of the packets coming from the server

Solution. There is a difference in destination IP address and port number between two captures.On the host machine: the source IP address is 216.58.196.174 (IP address of iperf.he.net) and the port number is 443, the destination address is 192.168.133.130 (IP address of the host machine)and the port number is35656.**On the guest machine: the source IP address is 216.58.196.174 (IP address of iperf.he.net) and the port number is 443, the destination address is 192.168.133.130 (IP address of the guest machine) and the port number is 35692. We observe that the source IP addresses and port numbers are the same in both cases, whereas the destination port numbers are different. This difference is caused by a NAT between the guest and the host network.**


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp

```
Time       Source           Destination      Protocol Length Info
30.484495  216.58.196.174   192.168.133.130  TCP       54 443 → 35656 [ACK] Seq=1 Ack=40 Win=64240 Len=0
30.485325  192.168.133.130  216.58.196.174   TLSv1.2   78 Application Data
30.485507  216.58.196.174   192.168.133.130  TCP       54 443 → 35656 [ACK] Seq=1 Ack=64 Win=64240 Len=0
30.485528  192.168.133.130  216.58.196.174   TCP       60 35656 → 443 [FIN, ACK] Seq=64 Ack=1 Win=41857 Len=0
30.485837  216.58.196.174   192.168.133.130  TCP       54 443 → 35656 [ACK] Seq=1 Ack=65 Win=64239 Len=0
30.536538  216.58.196.174   192.168.133.130  TCP       54 443 → 35656 [FIN, PSH, ACK] Seq=1 Ack=65 Win=64239 Len=0
30.536910  192.168.133.130  216.58.196.174   TCP       60 35656 → 443 [ACK] Seq=65 Ack=2 Win=41857 Len=0
43.078921  192.168.133.130  216.58.196.174   TCP       74 35692 → 443 [SYN] Seq=0 Win=42340 Len=0 MSS=1460 SACK_PERM=1 …
43.097293  216.58.196.174   192.168.133.130  TCP       58 443 → 35692 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460

> Frame 37: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\NPF_{F39C1F40-6E1F-4B21-8826-5499602883FF}
> Ethernet II, Src: VMware_f6:65:2b (00:50:56:f6:65:2b), Dst: VMware_2e:ed:ce (00:0c:29:2e:ed:ce)
> Internet Protocol Version 4, Src: 216.58.196.174, Dst: 192.168.133.130
> Transmission Control Protocol, Src Port: 443, Dst Port: 35656, Seq: 1, Ack: 64, Len: 0

0000  00 0c 29 2e ed ce 00 50  56 f6 65 2b 08 00 45 00   ..)....P V.e+..E.
0010  00 28 24 f6 00 00 80 06  32 c6 d8 3a c4 ae c0 a8   .($.....2.:....

wireshark_VMware Network Adapter VMnet8DR9U00.pcapng     Packets: 15636 · Displayed: 15452 (98.8%)     Profile: Default
```

Figure : Captured packets onHost Machine

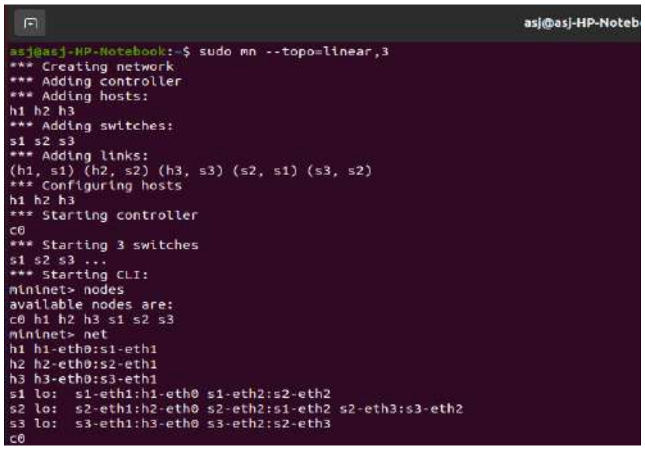

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp

```
No. Time         Source           Destination      Protocol Length Info
  9 7.056564249  192.168.133.130  216.58.196.174   TLSv1.2   349 Application Dat
 10 7.056633195  192.168.133.130  216.58.196.174   TLSv1.2   788 Application Dat
 11 7.057372258  192.168.133.130  216.58.196.174   TLSv1.2   956 Application Dat
 12 7.057928737  216.58.196.174   192.168.133.130  TCP        60 443 → 35692 [AC
 13 7.057928922  216.58.196.174   192.168.133.130  TCP        60 443 → 35692 [AC
 14 7.057928979  216.58.196.174   192.168.133.130  TCP        60 443 → 35692 [AC
 17 7.173335598  216.58.196.174   192.168.133.130  TLSv1.2   437 Application Dat
 18 7.173336450  216.58.196.174   192.168.133.130  TLSv1.2   237 Application Dat
 19 7.174459823  216.58.196.174   192.168.133.130  TLSv1.2   305 Application Dat
 20 7.175699446  192.168.133.130  216.58.196.174   TCP        54 35692 → 443 [AC
 21 7.188798580  216.58.196.174   192.168.133.130  TLSv1.2    93 Application Dat
 22 7.189168274  192.168.133.130  216.58.196.174   TLSv1.2    93 Application Dat
 23 7.190161785  216.58.196.174   192.168.133.130  TCP        60 443 → 35692 [AC
 26 7.283507862  192.168.133.130  172.217.167.133  TLSv1.2  1783 Application Dat

> Frame 12: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface ens33, id 0
> Ethernet II, Src: VMware_f6:65:2b (00:50:56:f6:65:2b), Dst: VMware_2e:ed:ce (00:0c:29:2e:ed:ce)
> Internet Protocol Version 4, Src: 216.58.196.174, Dst: 192.168.133.130
> Transmission Control Protocol, Src Port: 443, Dst Port: 35692, Seq: 1, Ack: 296, Len: 0
```

Figure :Captured packetson Guest Machine

Q2) What are the different links in the network?

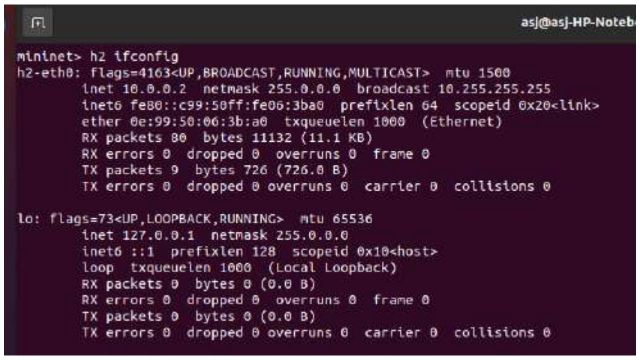
Ans. There are three links in the network
mininet> net
h1-eth0:s1-eth1
h2-eth0:s2-eth1
h3-eth0:s3-eth1



```
asj@asj-HP-Noteb
asj@asj-HP-Notebook:~$ sudo mn --topo=linear,3
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (s2, s1) (s3, s2)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
mininet> nodes
available nodes are:
c0 h1 h2 h3 s1 s2 s3
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s2-eth1
h3 h3-eth0:s3-eth1
s1 lo:  s1-eth1:h1-eth0 s1-eth2:s2-eth2
s2 lo:  s2-eth1:h2-eth0 s2-eth2:s1-eth2 s2-eth3:s3-eth2
s3 lo:  s3-eth1:h3-eth0 s3-eth2:s2-eth3
c0
```

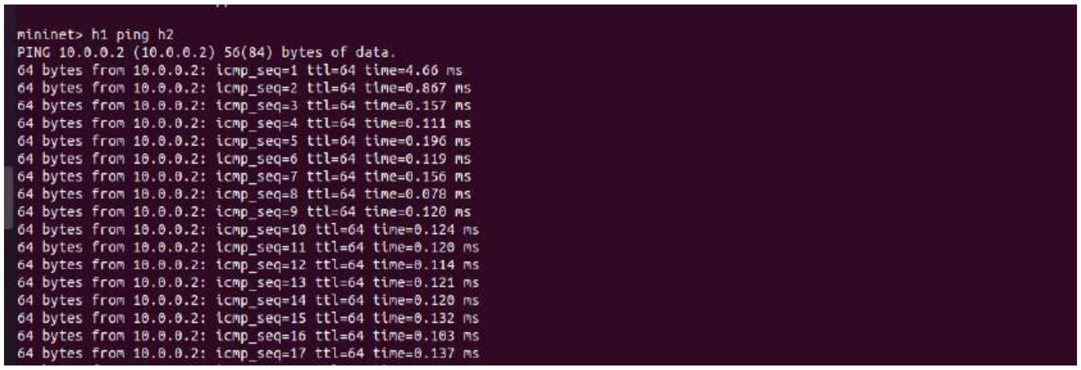
Q3)What command will you use to check the network interfaces of the host h2? How many interfacesdoes host h2 have? What is/are its/their IP addresses?
Ans. mininet>h2 ifconfig
h2 has two interfaces h2-eth0 and lo.

Their IP addresses are 10.0.0.2 and 127.0.0.1, respectively.

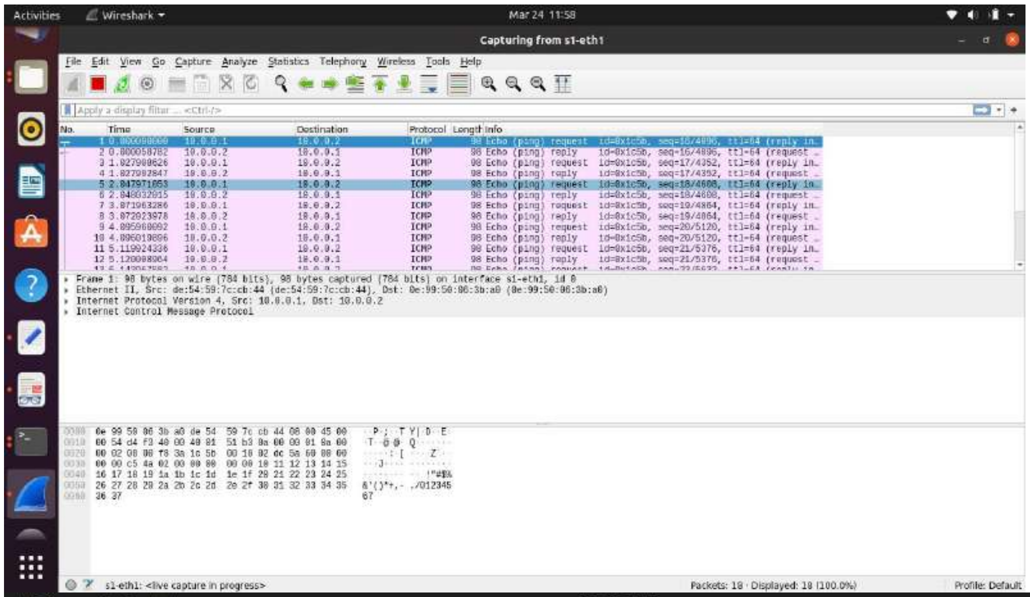


```
asj@asj-HP-Noteb
mininet> h2 ifconfig
h2-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.2  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::c99:50ff:fe06:3ba0  prefixlen 64  scopeid 0x20<link>
        ether 0e:99:50:06:3b:a0  txqueuelen 1000  (Ethernet)
        RX packets 80  bytes 11132 (11.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 9  bytes 726 (726.0 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
```

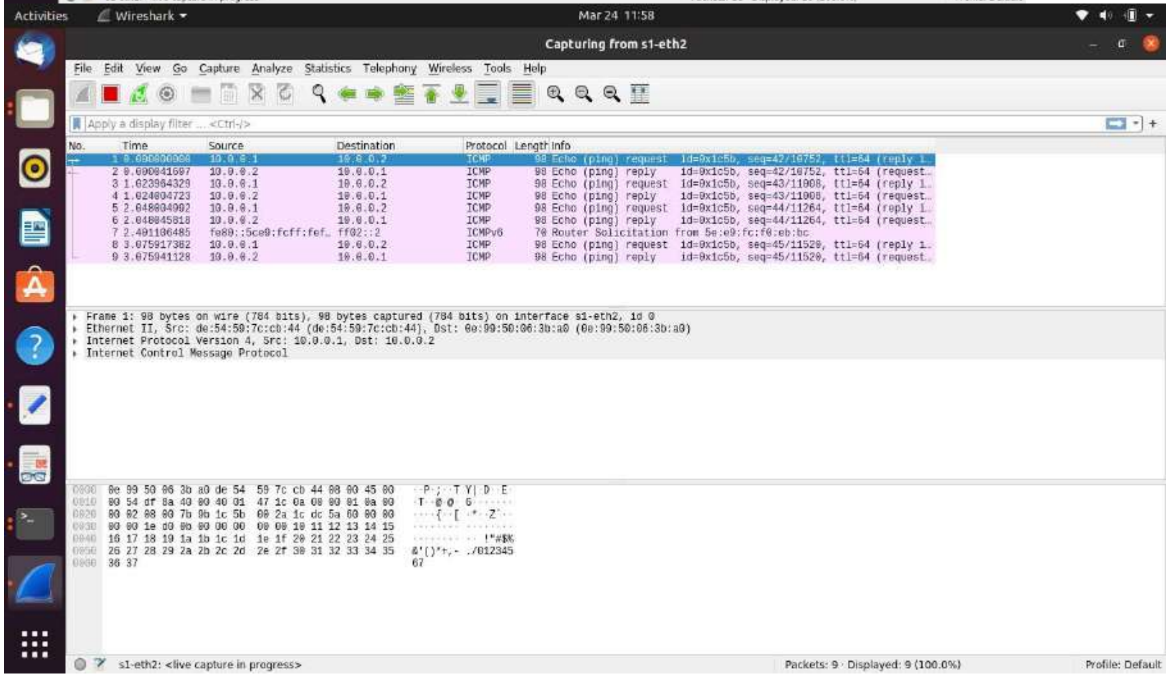
Q4) What is the source IP address of the ICMP echo requests sent from h1 to h2?
Solution. 10.0.0.1



```
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=4.66 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.867 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.157 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.111 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.196 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.119 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.156 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.078 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.120 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.124 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=0.120 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=0.114 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=0.121 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=0.120 ms
64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=0.132 ms
64 bytes from 10.0.0.2: icmp_seq=16 ttl=64 time=0.103 ms
64 bytes from 10.0.0.2: icmp_seq=17 ttl=64 time=0.137 ms
```


Activities
Wireshark
Mar 24 11:58
Capturing from s1-eth1
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

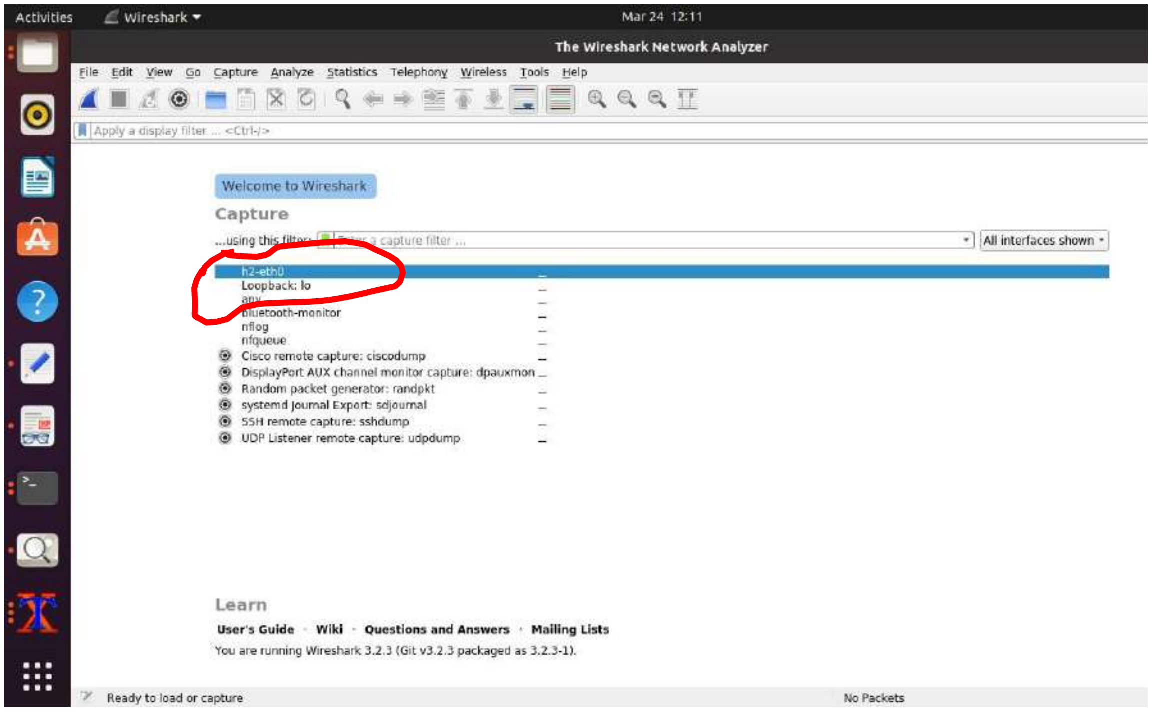
```
No.  Time         Source     Destination  Protocol Length Info
   1 0.000000000  10.0.0.1   10.0.0.2     ICMP      98 Echo (ping) request  id=0x1c5b, seq=16/4096, ttl=64 (reply in…
   2 0.000058782  10.0.0.2   10.0.0.1     ICMP      98 Echo (ping) reply    id=0x1c5b, seq=16/4096, ttl=64 (request …
   3 1.027980626  10.0.0.1   10.0.0.2     ICMP      98 Echo (ping) request  id=0x1c5b, seq=17/4352, ttl=64 (reply in…
   4 1.027992847  10.0.0.2   10.0.0.1     ICMP      98 Echo (ping) reply    id=0x1c5b, seq=17/4352, ttl=64 (request …
   5 2.047971053  10.0.0.1   10.0.0.2     ICMP      98 Echo (ping) request  id=0x1c5b, seq=18/4608, ttl=64 (reply in…
   6 2.048032015  10.0.0.2   10.0.0.1     ICMP      98 Echo (ping) reply    id=0x1c5b, seq=18/4608, ttl=64 (request …
   7 3.071963286  10.0.0.1   10.0.0.2     ICMP      98 Echo (ping) request  id=0x1c5b, seq=19/4864, ttl=64 (reply in…
   8 3.072023978  10.0.0.2   10.0.0.1     ICMP      98 Echo (ping) reply    id=0x1c5b, seq=19/4864, ttl=64 (request …
   9 4.095960002  10.0.0.1   10.0.0.2     ICMP      98 Echo (ping) request  id=0x1c5b, seq=20/5120, ttl=64 (reply in…
  10 4.096019896  10.0.0.2   10.0.0.1     ICMP      98 Echo (ping) reply    id=0x1c5b, seq=20/5120, ttl=64 (request …
  11 5.119924336  10.0.0.1   10.0.0.2     ICMP      98 Echo (ping) request  id=0x1c5b, seq=21/5376, ttl=64 (reply in…
  12 5.120008964  10.0.0.2   10.0.0.1     ICMP      98 Echo (ping) reply    id=0x1c5b, seq=21/5376, ttl=64 (request …

▸ Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface s1-eth1, id 0
▸ Ethernet II, Src: de:54:59:7c:cb:44 (de:54:59:7c:cb:44), Dst: 0e:99:50:06:3b:a0 (0e:99:50:06:3b:a0)
▸ Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.2
▸ Internet Control Message Protocol

0000  0e 99 50 06 3b a0 de 54  59 7c cb 44 08 00 45 00   ··P·;··T Y|·D··E·
0010  00 54 d4 f3 40 00 40 01  51 b3 0a 00 00 01 0a 00   ·T··@·@· Q·······
0020  00 02 08 00 f8 3a 1c 5b  00 10 02 dc 5a 60 00 00   ·····:·[ ····Z`··
0030  00 00 c5 4a 02 00 00 00  00 00 10 11 12 13 14 15   ···J···· ········
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ········ ·· !"#$%
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35   &'()*+,- ./012345
0060  36 37                                              67

s1-eth1: <live capture in progress>          Packets: 18 · Displayed: 18 (100.0%)          Profile: Default
```


Activities
Wireshark
Mar 24 11:58
Capturing from s1-eth2
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No.  Time         Source                Destination  Protocol Length Info
   1 0.000000000  10.0.0.1              10.0.0.2     ICMP      98 Echo (ping) request  id=0x1c5b, seq=42/10752, ttl=64 (reply i…
   2 0.000041607  10.0.0.2              10.0.0.1     ICMP      98 Echo (ping) reply    id=0x1c5b, seq=42/10752, ttl=64 (request…
   3 1.023964329  10.0.0.1              10.0.0.2     ICMP      98 Echo (ping) request  id=0x1c5b, seq=43/11008, ttl=64 (reply i…
   4 1.024004723  10.0.0.2              10.0.0.1     ICMP      98 Echo (ping) reply    id=0x1c5b, seq=43/11008, ttl=64 (request…
   5 2.048004902  10.0.0.1              10.0.0.2     ICMP      98 Echo (ping) request  id=0x1c5b, seq=44/11264, ttl=64 (reply i…
   6 2.048045818  10.0.0.2              10.0.0.1     ICMP      98 Echo (ping) reply    id=0x1c5b, seq=44/11264, ttl=64 (request…
   7 2.401106485  fe80::5ce0:fcff:fef…  ff02::2      ICMPv6    70 Router Solicitation from 5e:e0:fc:f0:eb:bc
   8 3.075917382  10.0.0.1              10.0.0.2     ICMP      98 Echo (ping) request  id=0x1c5b, seq=45/11520, ttl=64 (reply i…
   9 3.075941128  10.0.0.2              10.0.0.1     ICMP      98 Echo (ping) reply    id=0x1c5b, seq=45/11520, ttl=64 (request…

▸ Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface s1-eth2, id 0
▸ Ethernet II, Src: de:54:59:7c:cb:44 (de:54:59:7c:cb:44), Dst: 0e:99:50:06:3b:a0 (0e:99:50:06:3b:a0)
▸ Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.2
▸ Internet Control Message Protocol

0000  0e 99 50 06 3b a0 de 54  59 7c cb 44 08 00 45 00   ··P·;··T Y|·D··E·
0010  00 54 df 8a 40 00 40 01  47 1c 0a 00 00 01 0a 00   ·T··@·@· G·······
0020  00 02 08 00 7b 9b 1c 5b  00 2a 1c dc 5a 60 00 00   ····{··[ ·*··Z`··
0030  00 00 1e d0 0b 00 00 00  00 00 10 11 12 13 14 15   ········ ········
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ········ ·· !"#$%
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35   &'()*+,- ./012345
0060  36 37                                              67

s1-eth2: <live capture in progress>          Packets: 9 · Displayed: 9 (100.0%)          Profile: Default
```

Q5) List down the names of the interfaces that are available for capture.

Solution: h2-eth0,lo



Activities
Wireshark
Mar 24 12:11
The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Welcome to Wireshark
Capture
...using this filter: Enter a capture filter ...
All interfaces shown

```
h2-eth0
Loopback: lo
any
bluetooth-monitor
nflog
nfqueue
Cisco remote capture: ciscodump
DisplayPort AUX channel monitor capture: dpauxmon
Random packet generator: randpkt
systemd Journal Export: sdjournal
SSH remote capture: sshdump
UDP Listener remote capture: udpdump
```

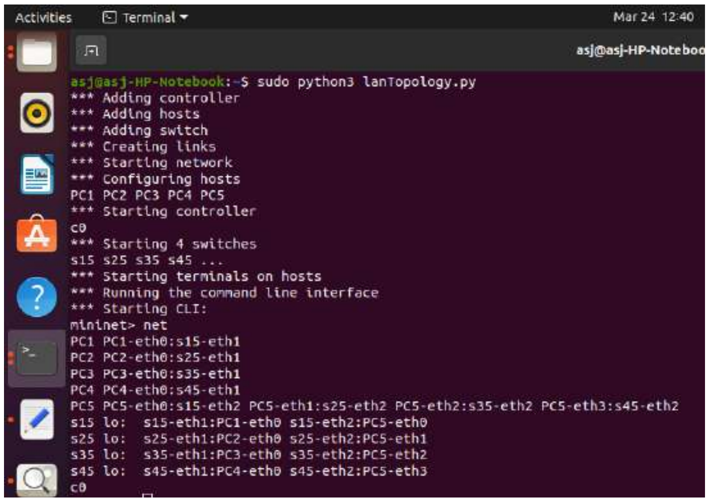
Learn
User's Guide · Wiki · Questions and Answers · Mailing Lists

You are running Wireshark 3.2.3 (Git v3.2.3 packaged as 3.2.3-1).

Ready to load or capture
No Packets

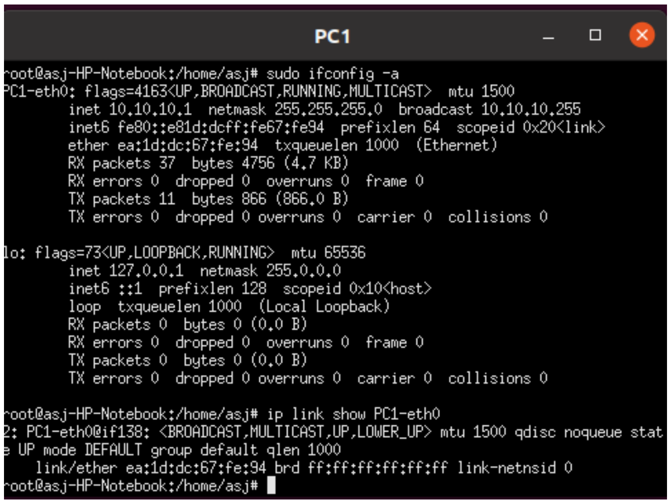
Q6/ How many IPv4 networks does PC4 belong to? Name the type of the network(s).

Solution. 4, Local area networks (LANs)

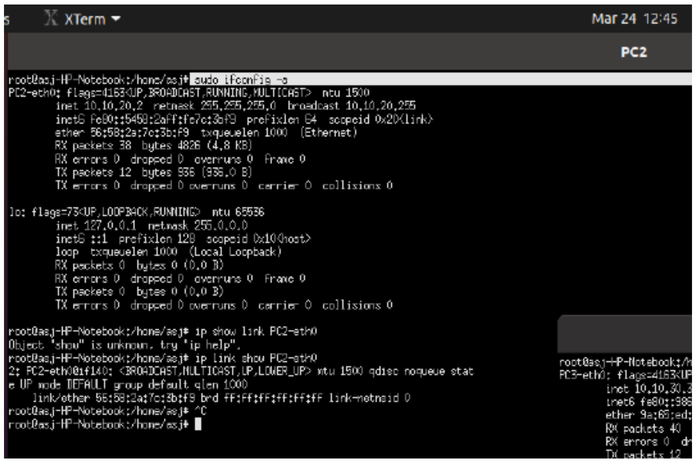


```
asj@asj-HP-Notebook:~$ sudo python3 lanTopology.py
*** Adding controller
*** Adding hosts
*** Adding switch
*** Creating links
*** Starting network
*** Configuring hosts
PC1 PC2 PC3 PC4 PC5
*** Starting controller
c0
*** Starting 4 switches
s15 s25 s35 s45 ...
*** Starting terminals on hosts
*** Running the command line interface
*** Starting CLI:
mininet> net
PC1 PC1-eth0:s15-eth1
PC2 PC2-eth0:s25-eth1
PC3 PC3-eth0:s35-eth1
PC4 PC4-eth0:s45-eth1
PC5 PC5-eth0:s15-eth2 PC5-eth1:s25-eth2 PC5-eth2:s35-eth2 PC5-eth3:s45-eth2
s15 lo:  s15-eth1:PC1-eth0 s15-eth2:PC5-eth0
s25 lo:  s25-eth1:PC2-eth0 s25-eth2:PC5-eth1
s35 lo:  s35-eth1:PC3-eth0 s35-eth2:PC5-eth2
s45 lo:  s45-eth1:PC4-eth0 s45-eth2:PC5-eth3
c0
```

Configuration details of all 5 PCs

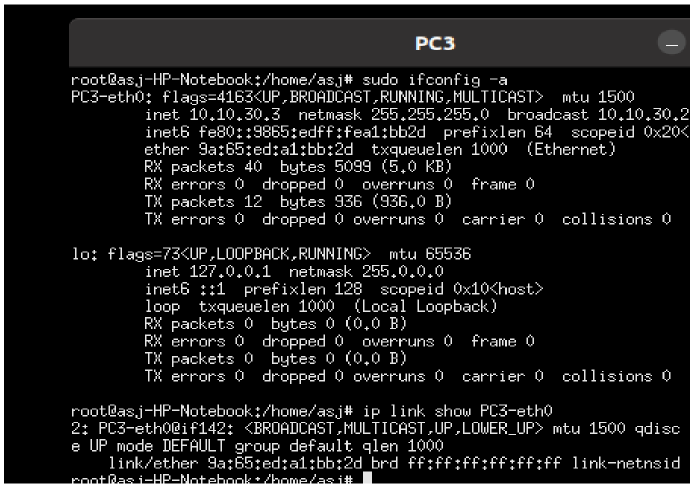


```
root@asj-HP-Notebook:/home/asj# sudo ifconfig -a
PC1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.10.10.1  netmask 255.255.255.0  broadcast 10.10.10.255
        inet6 fe80::e81d:dcff:fe67:fe94  prefixlen 64  scopeid 0x20<link>
        ether ea:1d:dc:67:fe:94  txqueuelen 1000  (Ethernet)
        RX packets 37  bytes 4756 (4.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 11  bytes 866 (866.0 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

root@asj-HP-Notebook:/home/asj# ip link show PC1-eth0
2: PC1-eth0@if138: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue stat
e UP mode DEFAULT group default qlen 1000
    link/ether ea:1d:dc:67:fe:94 brd ff:ff:ff:ff:ff:ff link-netnsid 0
root@asj-HP-Notebook:/home/asj#
```

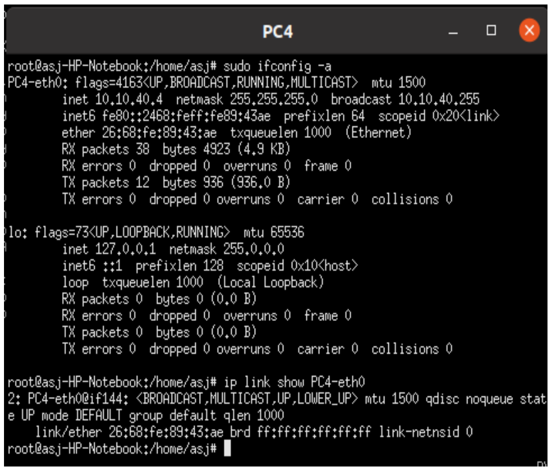


```
root@asj-HP-Notebook:/home/asj# sudo ifconfig -a
PC2-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.10.20.2  netmask 255.255.255.0  broadcast 10.10.20.255
        inet6 fe80::5458:2aff:fe7c:3bf9  prefixlen 64  scopeid 0x20<link>
        ether 56:58:2a:7c:3b:f9  txqueuelen 1000  (Ethernet)
        RX packets 38  bytes 4826 (4.8 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

root@asj-HP-Notebook:/home/asj# ip show link PC2-eth0
Object "show" is unknown, try "ip help".
root@asj-HP-Notebook:/home/asj# ip link show PC2-eth0
2: PC2-eth0@if140: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue stat
e UP mode DEFAULT group default qlen 1000
    link/ether 56:58:2a:7c:3b:f9 brd ff:ff:ff:ff:ff:ff link-netnsid 0
root@asj-HP-Notebook:/home/asj# ^C
root@asj-HP-Notebook:/home/asj#
```

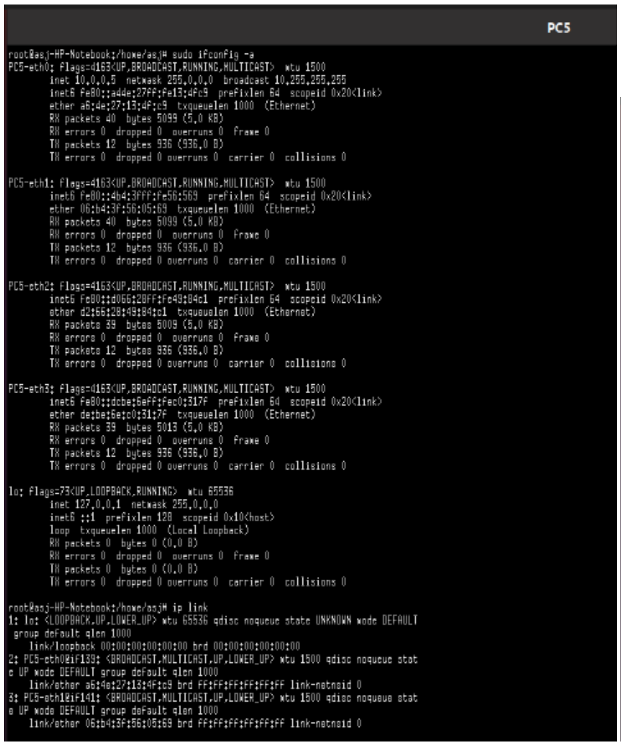
```
root@asj-HP-Notebook:/h
PC3-eth0: flags=4163<UP
        inet 10.10.30.3
        inet6 fe80::986
        ether 9a:65:ed:
        RX packets 40
        RX errors 0  dr
        TX packets 12
```



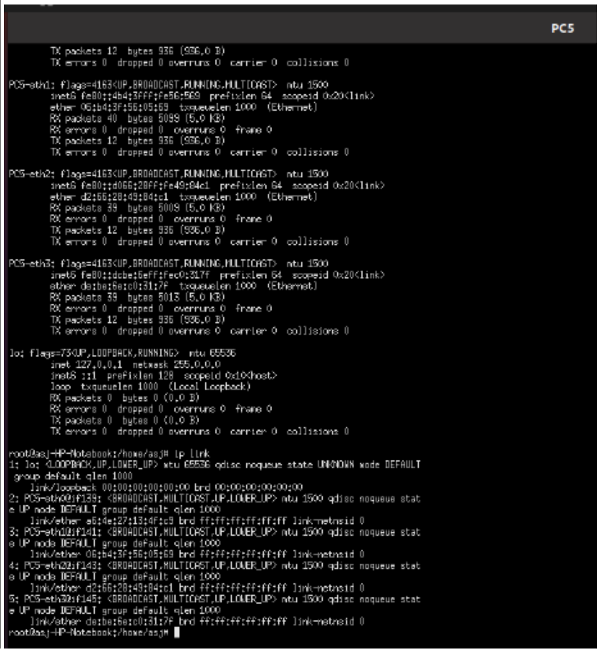
```
root@asj-HP-Notebook:/home/asj# sudo ifconfig -a
PC3-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.10.30.3  netmask 255.255.255.0  broadcast 10.10.30.2
        inet6 fe80::9865:edff:fea1:bb2d  prefixlen 64  scopeid 0x20<
        ether 9a:65:ed:a1:bb:2d  txqueuelen 1000  (Ethernet)
        RX packets 40  bytes 5099 (5.0 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

root@asj-HP-Notebook:/home/asj# ip link show PC3-eth0
2: PC3-eth0@if142: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
e UP mode DEFAULT group default qlen 1000
    link/ether 9a:65:ed:a1:bb:2d brd ff:ff:ff:ff:ff:ff link-netnsid
root@asj-HP-Notebook:/home/asj#
```



```
root@asj-HP-Notebook:/home/asj# sudo ifconfig -a
PC4-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.10.40.4  netmask 255.255.255.0  broadcast 10.10.40.255
        inet6 fe80::2468:feff:fe89:43ae  prefixlen 64  scopeid 0x20<link>
        ether 26:68:fe:89:43:ae  txqueuelen 1000  (Ethernet)
        RX packets 38  bytes 4923 (4.9 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

root@asj-HP-Notebook:/home/asj# ip link show PC4-eth0
2: PC4-eth0@if144: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue stat
e UP mode DEFAULT group default qlen 1000
    link/ether 26:68:fe:89:43:ae brd ff:ff:ff:ff:ff:ff link-netnsid 0
root@asj-HP-Notebook:/home/asj#
```



```
root@asj-HP-Notebook:/home/asj# sudo ifconfig -a
PC5-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.5  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::a4de:27ff:fe13:4fc9  prefixlen 64  scopeid 0x20<link>
        ether a6:de:27:13:4f:c9  txqueuelen 1000  (Ethernet)
        RX packets 40  bytes 5099 (5.0 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

PC5-eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::4b4:3fff:fe56:569  prefixlen 64  scopeid 0x20<link>
        ether 06:b4:3f:56:05:69  txqueuelen 1000  (Ethernet)
        RX packets 40  bytes 5099 (5.0 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

PC5-eth2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::d066:28ff:fe49:84c1  prefixlen 64  scopeid 0x20<link>
        ether d2:66:28:49:84:c1  txqueuelen 1000  (Ethernet)
        RX packets 39  bytes 5009 (5.0 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

PC5-eth3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::dcbe:6eff:fec0:317f  prefixlen 64  scopeid 0x20<link>
        ether de:be:6e:c0:31:7f  txqueuelen 1000  (Ethernet)
        RX packets 39  bytes 5013 (5.0 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

root@asj-HP-Notebook:/home/asj# ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
 group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: PC5-eth0@if139: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue stat
e UP mode DEFAULT group default qlen 1000
    link/ether a6:4e:27:13:4f:c9 brd ff:ff:ff:ff:ff:ff link-netnsid 0
3: PC5-eth1@if141: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue stat
e UP mode DEFAULT group default qlen 1000
    link/ether 06:b4:3f:56:05:69 brd ff:ff:ff:ff:ff:ff link-netnsid 0
```



```
        TX packets 12  bytes 936 (936.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

PC5-eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::4b4:3fff:fe56:569  prefixlen 64  scopeid 0x20<link>
        ether 06:b4:3f:56:05:69  txqueuelen 1000  (Ethernet)
        RX packets 40  bytes 5099 (5.0 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

PC5-eth2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::d066:28ff:fe49:84c1  prefixlen 64  scopeid 0x20<link>
        ether d2:66:28:49:84:c1  txqueuelen 1000  (Ethernet)
        RX packets 39  bytes 5009 (5.0 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

PC5-eth3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::dcbe:6eff:fec0:317f  prefixlen 64  scopeid 0x20<link>
        ether de:be:6e:c0:31:7f  txqueuelen 1000  (Ethernet)
        RX packets 39  bytes 5013 (5.0 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

root@asj-HP-Notebook:/home/asj# ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
 group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: PC5-eth0@if139: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue stat
e UP mode DEFAULT group default qlen 1000
    link/ether a6:4e:27:13:4f:c9 brd ff:ff:ff:ff:ff:ff link-netnsid 0
3: PC5-eth1@if141: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue stat
e UP mode DEFAULT group default qlen 1000
    link/ether 06:b4:3f:56:05:69 brd ff:ff:ff:ff:ff:ff link-netnsid 0
4: PC5-eth2@if143: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue stat
e UP mode DEFAULT group default qlen 1000
    link/ether d2:66:28:49:84:c1 brd ff:ff:ff:ff:ff:ff link-netnsid 0
5: PC5-eth3@if145: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue stat
e UP mode DEFAULT group default qlen 1000
    link/ether de:be:6e:c0:31:7f brd ff:ff:ff:ff:ff:ff link-netnsid 0
root@asj-HP-Notebook:/home/asj#
```

Q7) Write down the IPv4 addresses you will use for the interfaces according to this addressing scheme.

Solution:

h1-eth0:	10.0.0.1
h2-eth0:	10.0.0.2
h3-eth0:	10.0.0.3
h4-eth0:	10.0.0.4
h5-eth0:	10.0.0.5

Q8) Write down the IPv6 addresses you will use for the interfaces according to this addressing scheme.

Solution: h1-eth0: fe80::e81d:dcff:fe67
h2-eth0:fe80::5458:2aff:fe7c:3bf9
h3-eth0: fe80::9865:edff:fea1:bb2d
h4-eth0:fe80::2468:feff:fe89:43ae
h5-eth0: fe80::a44e:27ff:fe13:4fc9

Q9) Write down the MAC address of the interfaces.

Solution: h1(eth0): 06:49:78:7c:7a:69

h2(eth0): a6:4e:27:13:89:c9

h3(eth0): de:c8:02:5c:35:59

h4(eth0): d2:2f:00:ae:50:5e

h5(eth0): de:3c:5d:86:b0:e4

Q10)Compute the IPv6 link-local addresses for the interfaces below.

Solution:

h3-eth0–fe80::dcc8:02ff:fe5c:3559/64

H4-eth0–fe80::d22f:00ff:feae:505e/64

Q11) Explain what happens.

Solution: ping cannot find the destination host since there exists no host with the specified IP address in the same subnet. Therefore, only ARP packets are sent.

Q12) Describe the traffic you observe with Wireshark.

Solution: In IPv6, link-local addresses are assigned to interfaces by the OS (using SLAAC) without requiring any configuration, thus the ping command works with no problem. In Wireshark we observe echo requests and replies.

Q13) Describe and explain the differences (if any) between the output from the two ping commands.

Solution: IPv6 link-local addresses cannot traverse routers thus h4's lo link-local address is unreachable from h1. Only devices that are in the same LAN, like h4's eth0 link-local address, will be reachable.

Q14) Write down the routing table entries.

Solution: ip route: 10.0.0.0/8 dev h1-eth0 proto kernel scope link src 10.0.0.1 ip -6 route: fe80::/64 dev h1-eth0 proto kernel metric 256 pref medium

Q15) What are the routing table entries after assigning the new ip addresses?

Solution: ip route: 10.10.10.0/24 dev h1-eth0 proto kernel scope link src 10.10.10.1

ip -6 route: fe80::/64 dev h1-eth0 proto kernel metric 256 pref medium

Q16) For the IPv4 addresses assigned to h1-eth0, identify the host and subnetwork portions. Why is it important for the PC to know this information?

Solution:With the netmask /24, the sub network is 10.10.10 and the host is represented with the last octet.This information is useful to know if a destination is on the same LAN or not (whether to communicate directly or through the default gateway).

Q17) What do you observe on Wireshark? Explain.

Solution: ARP packets are sent, but still no ICMP since h4 is not configured.

Q18) Does it work? Why?

Solution: No it does not work, no route is available because there is no default gateway configured inh1.

Q19) Which command(s) you need to fix the problem? On which PC did you apply them?

Solution: A route needs to be added on h3, for example by configuring h4 as a default gateway.

Q20) Perform the following pings simultaneously: PC2 from PC1, PC3 from PC2, and PC1 from PC3. Comment on the round-trip times that you observe.

Solution: They are roughly the same in all cases.

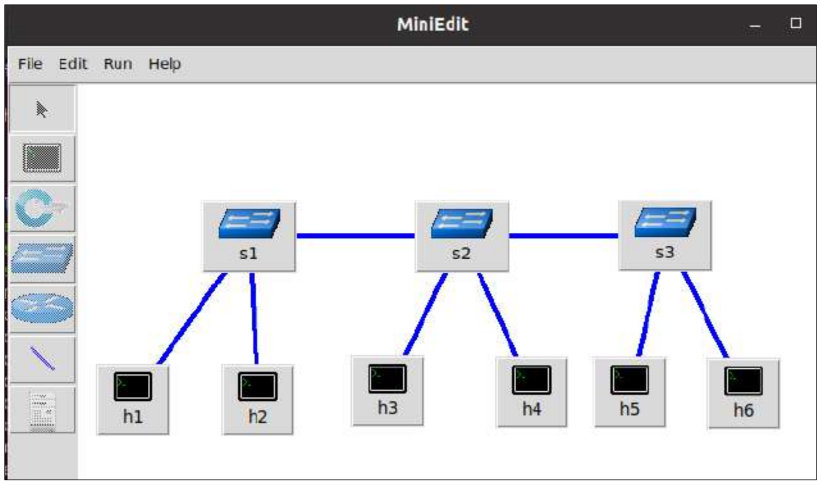
Q21) Do an iperf test between h1 and h2 (h1 run as server) using TCP and UDP. Report the performance (bandwidth values) you obtain with TCP and UDP. Are the port numbers for server and client changing if you repeat experiment several times?

Solution: The bandwidth with TCP is 39.5 Gbits/sec, and with UDP it is 1.05 Mbits/sec. The port number for server remains the same for TCP and UDP(5001) whereas for the client it changes every time you launch

VI. TP 2: L2 vs L3

i. Using a switch as a networking device

Custom Topology:



Q1) how many LANs are there in the figure?
Ans. The Above topology is only in one Lan Network as the Topology Doesn't have any other network it is connected to and hence only the One LAN Network which is present.

Q2) They might be different for every machine, but all the hosts should have 64 bits prefix starting with fe80?
Ans. H1- fe80::ace0:93ff:fe57:bd90

H2- fe80::845f:98ff:fe5d:1ad6

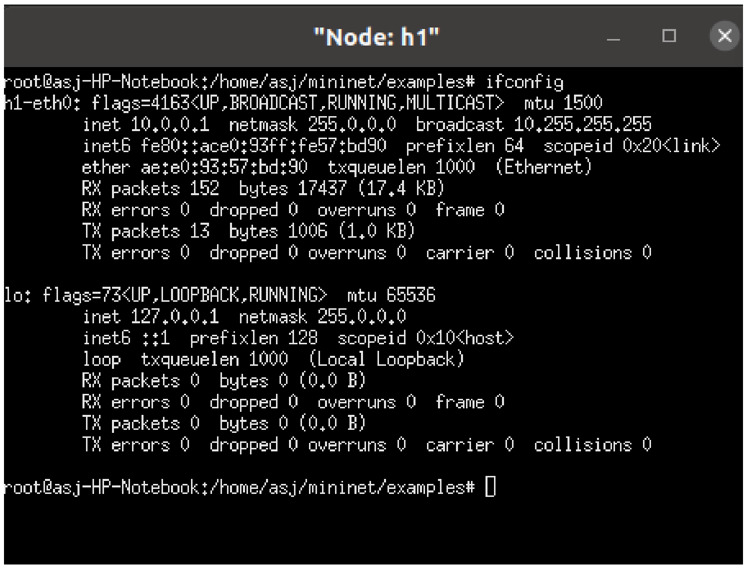
H3- fe80::ecb6:53ff:fe1e:fa48

H4- fe80::6c48:94ff:fecf:bfd6

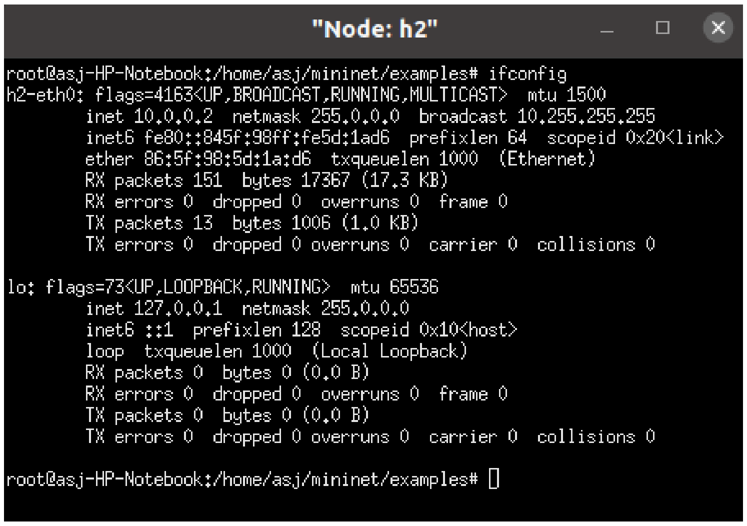
H5- fe80::18f1:12ff:fe75:48e4

H6- fe80::38f4:d0ff:fe89:97fd

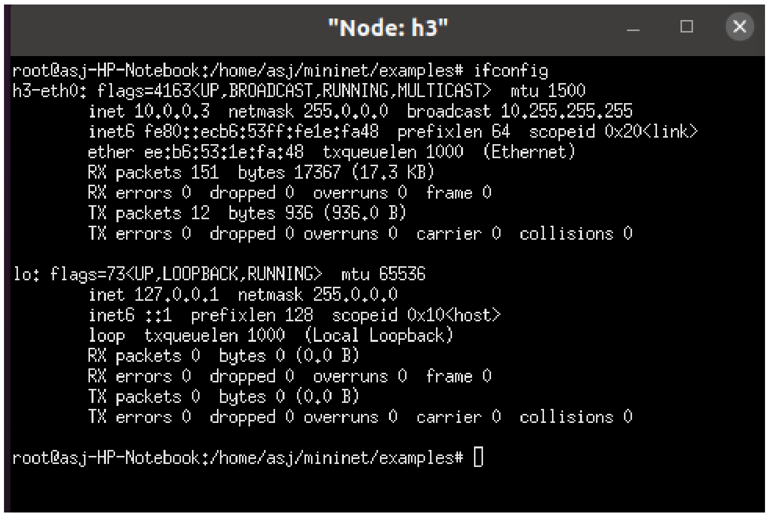
They might be different for every machine, but all the hosts should have 64 bits prefix starting with fe80. These are link-local addresses, i.e. addresses that can be used for communicating in the same LAN only.



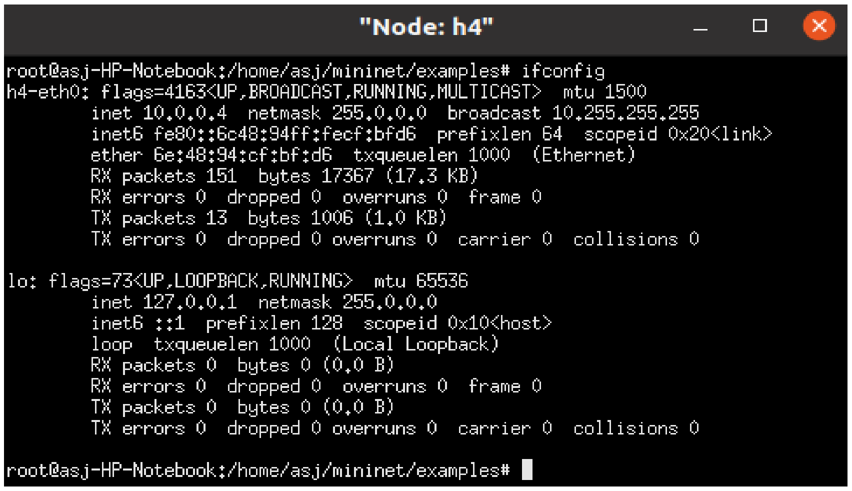
```
root@asj-HP-Notebook:/home/asj/mininet/examples# ifconfig
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.1  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::ace0:93ff:fe57:bd90  prefixlen 64  scopeid 0x20<link>
        ether ae:e0:93:57:bd:90  txqueuelen 1000  (Ethernet)
        RX packets 152  bytes 17437 (17.4 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

root@asj-HP-Notebook:/home/asj/mininet/examples# []
```



```
root@asj-HP-Notebook:/home/asj/mininet/examples# 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::845f:98ff:fe5d:1ad6  prefixlen 64  scopeid 0x20<link>
        ether 86:5f:98:5d:1a:d6  txqueuelen 1000  (Ethernet)
        RX packets 151  bytes 17367 (17.3 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

root@asj-HP-Notebook:/home/asj/mininet/examples# []
```

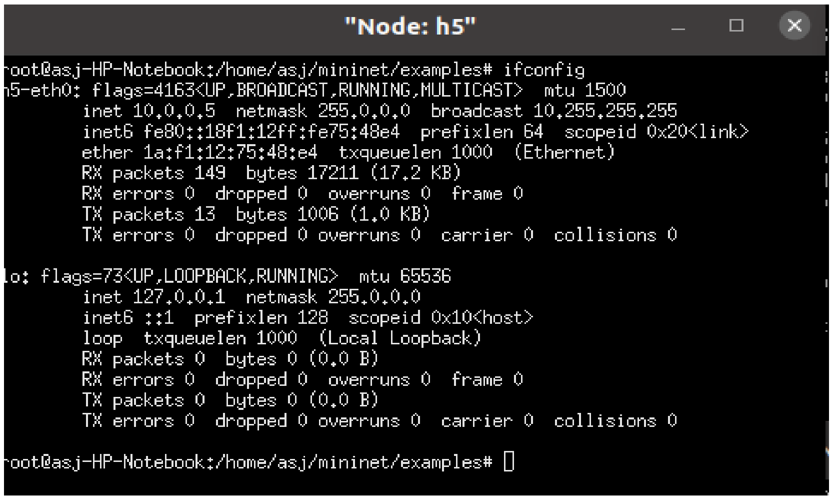


```
root@asj-HP-Notebook:/home/asj/mininet/examples# ifconfig
h3-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.3  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::ecb6:53ff:fe1e:fa48  prefixlen 64  scopeid 0x20<link>
        ether ee:b6:53:1e:fa:48  txqueuelen 1000  (Ethernet)
        RX packets 151  bytes 17367 (17.3 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

root@asj-HP-Notebook:/home/asj/mininet/examples# []
```



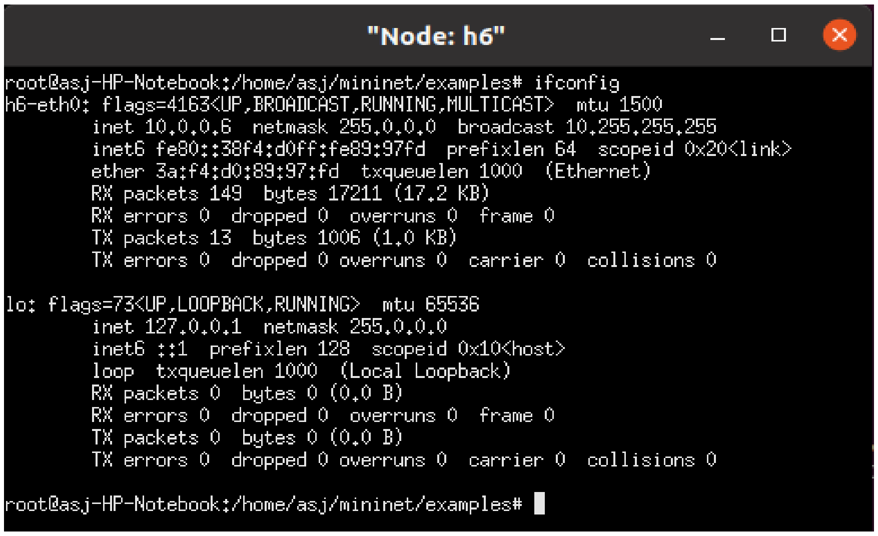
"Node: h4"

```
root@asj-HP-Notebook:/home/asj/mininet/examples# ifconfig
h4-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.4  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::6c48:94ff:fecf:bfd6  prefixlen 64  scopeid 0x20<link>
        ether 6e:48:94:cf:bf:d6  txqueuelen 1000  (Ethernet)
        RX packets 151  bytes 17367 (17.3 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

root@asj-HP-Notebook:/home/asj/mininet/examples# 
```



"Node: h5"

```
oot@asj-HP-Notebook:/home/asj/mininet/examples# ifconfig
5-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.5  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::18f1:12ff:fe75:48e4  prefixlen 64  scopeid 0x20<link>
        ether 1a:f1:12:75:48:e4  txqueuelen 1000  (Ethernet)
        RX packets 149  bytes 17211 (17.2 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

o: 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

oot@asj-HP-Notebook:/home/asj/mininet/examples# 
```

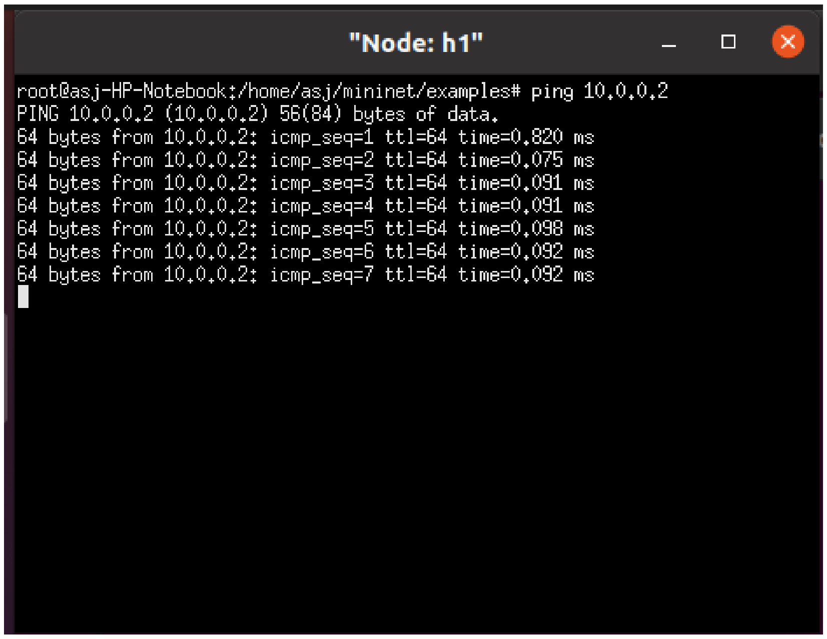


"Node: h6"

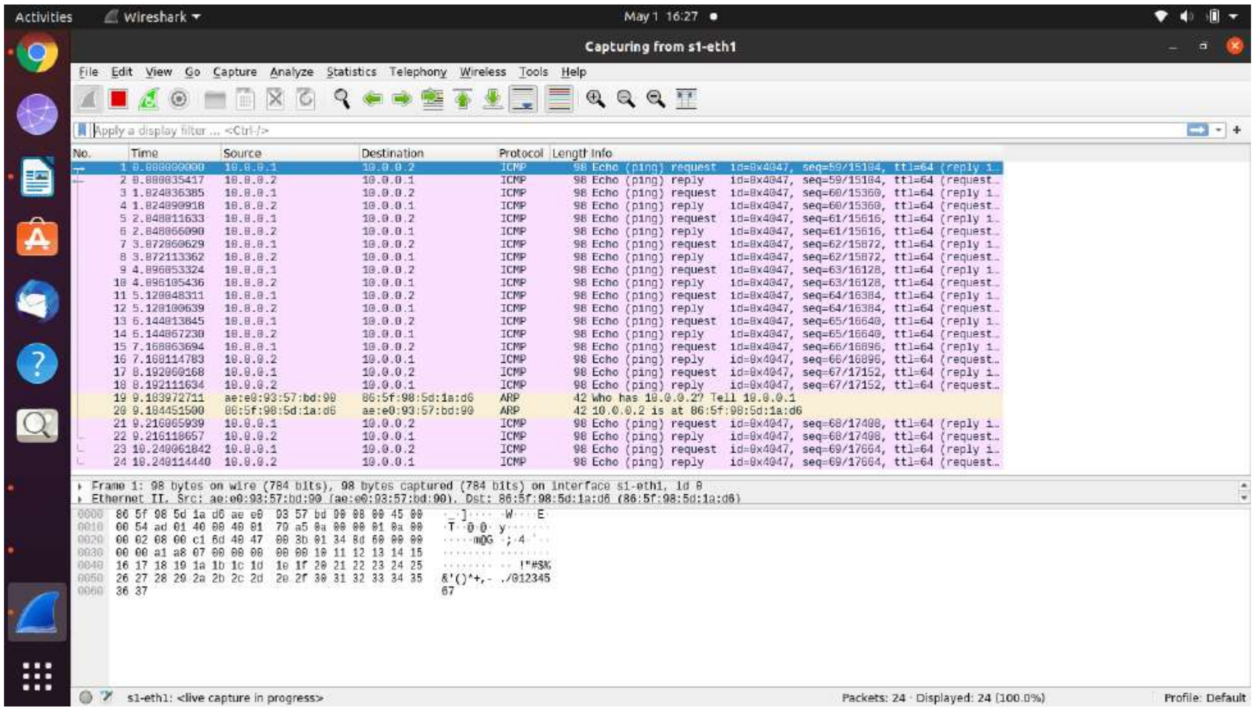
```
root@asj-HP-Notebook:/home/asj/mininet/examples# ifconfig
h6-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.6  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::38f4:d0ff:fe89:97fd  prefixlen 64  scopeid 0x20<link>
        ether 3a:f4:d0:89:97:fd  txqueuelen 1000  (Ethernet)
        RX packets 149  bytes 17211 (17.2 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

root@asj-HP-Notebook:/home/asj/mininet/examples# 
```

Q3) which hosts do not receive ping-reply? Explain

Ans. Yes the Ping requests to all the Hosts are successful. All hosts receive ping-reply. Since each host is using its link-local IPv6 address and all hosts are in the same LAN, it can send ping-request and receive ping-reply from other hosts.

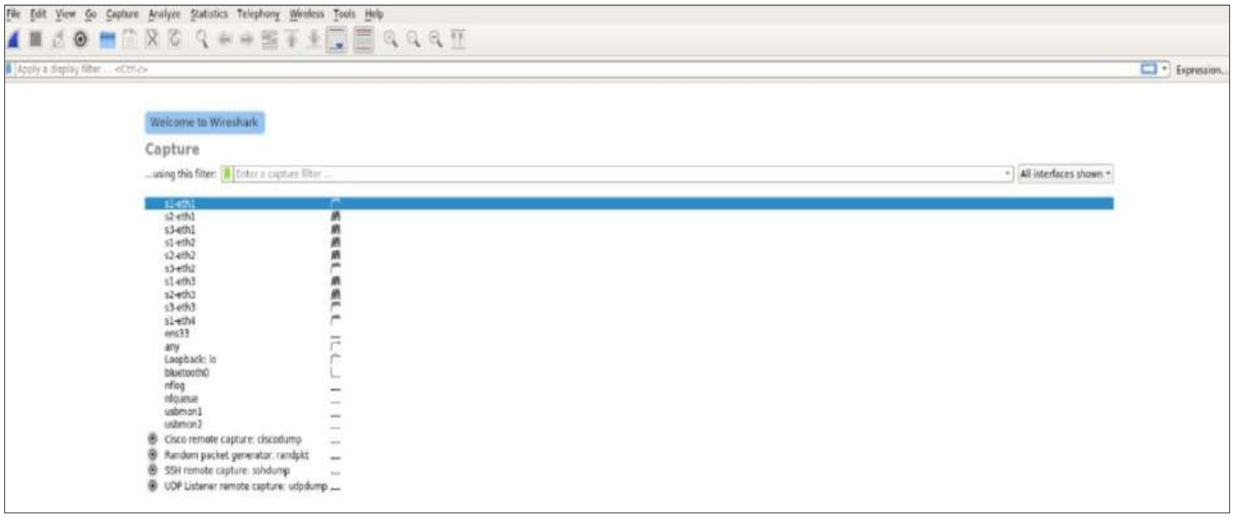


```
root@asj-HP-Notebook:/home/asj/mininet/examples# ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.820 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.075 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.091 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.091 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.098 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.092 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.092 ms
```


Activities
Wireshark
May 1 16:27
Capturing from s1-eth1
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

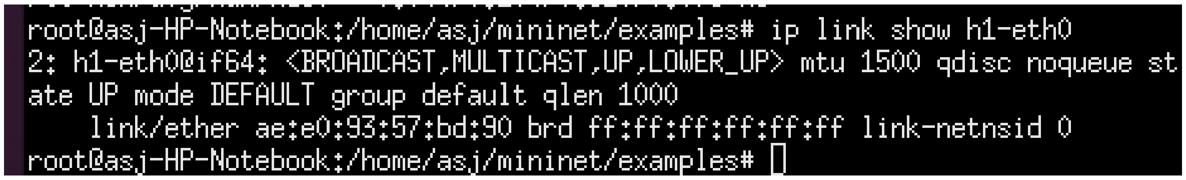
```
No.  Time          Source             Destination        Protocol Length Info
   1 0.000000000   10.0.0.1           10.0.0.2           ICMP     98 Echo (ping) request  id=0x4047, seq=59/15104, ttl=64 (reply i…
   2 0.000035417   10.0.0.2           10.0.0.1           ICMP     98 Echo (ping) reply    id=0x4047, seq=59/15104, ttl=64 (request…
   3 1.024036385   10.0.0.1           10.0.0.2           ICMP     98 Echo (ping) request  id=0x4047, seq=60/15360, ttl=64 (reply i…
   4 1.024090918   10.0.0.2           10.0.0.1           ICMP     98 Echo (ping) reply    id=0x4047, seq=60/15360, ttl=64 (request…
   5 2.048011633   10.0.0.1           10.0.0.2           ICMP     98 Echo (ping) request  id=0x4047, seq=61/15616, ttl=64 (reply i…
   6 2.048066090   10.0.0.2           10.0.0.1           ICMP     98 Echo (ping) reply    id=0x4047, seq=61/15616, ttl=64 (request…
   7 3.072060629   10.0.0.1           10.0.0.2           ICMP     98 Echo (ping) request  id=0x4047, seq=62/15872, ttl=64 (reply i…
   8 3.072113362   10.0.0.2           10.0.0.1           ICMP     98 Echo (ping) reply    id=0x4047, seq=62/15872, ttl=64 (request…
   9 4.096053324   10.0.0.1           10.0.0.2           ICMP     98 Echo (ping) request  id=0x4047, seq=63/16128, ttl=64 (reply i…
  10 4.096105436   10.0.0.2           10.0.0.1           ICMP     98 Echo (ping) reply    id=0x4047, seq=63/16128, ttl=64 (request…
  11 5.120048311   10.0.0.1           10.0.0.2           ICMP     98 Echo (ping) request  id=0x4047, seq=64/16384, ttl=64 (reply i…
  12 5.120100639   10.0.0.2           10.0.0.1           ICMP     98 Echo (ping) reply    id=0x4047, seq=64/16384, ttl=64 (request…
  13 6.144013845   10.0.0.1           10.0.0.2           ICMP     98 Echo (ping) request  id=0x4047, seq=65/16640, ttl=64 (reply i…
  14 6.144067230   10.0.0.2           10.0.0.1           ICMP     98 Echo (ping) reply    id=0x4047, seq=65/16640, ttl=64 (request…
  15 7.168063694   10.0.0.1           10.0.0.2           ICMP     98 Echo (ping) request  id=0x4047, seq=66/16896, ttl=64 (reply i…
  16 7.168114783   10.0.0.2           10.0.0.1           ICMP     98 Echo (ping) reply    id=0x4047, seq=66/16896, ttl=64 (request…
  17 8.192060168   10.0.0.1           10.0.0.2           ICMP     98 Echo (ping) request  id=0x4047, seq=67/17152, ttl=64 (reply i…
  18 8.192111634   10.0.0.2           10.0.0.1           ICMP     98 Echo (ping) reply    id=0x4047, seq=67/17152, ttl=64 (request…
  19 9.183972711   ae:e0:93:57:bd:90  86:5f:98:5d:1a:d6  ARP      42 Who has 10.0.0.2? Tell 10.0.0.1
  20 9.184451500   86:5f:98:5d:1a:d6  ae:e0:93:57:bd:90  ARP      42 10.0.0.2 is at 86:5f:98:5d:1a:d6
  21 9.216065939   10.0.0.1           10.0.0.2           ICMP     98 Echo (ping) request  id=0x4047, seq=68/17408, ttl=64 (reply i…
  22 9.216118657   10.0.0.2           10.0.0.1           ICMP     98 Echo (ping) reply    id=0x4047, seq=68/17408, ttl=64 (request…
  23 10.240061842  10.0.0.1           10.0.0.2           ICMP     98 Echo (ping) request  id=0x4047, seq=69/17664, ttl=64 (reply i…
  24 10.240114440  10.0.0.2           10.0.0.1           ICMP     98 Echo (ping) reply    id=0x4047, seq=69/17664, ttl=64 (request…

▸ Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface s1-eth1, id 0
▸ Ethernet II, Src: ae:e0:93:57:bd:90 (ae:e0:93:57:bd:90), Dst: 86:5f:98:5d:1a:d6 (86:5f:98:5d:1a:d6)

0000  86 5f 98 5d 1a d6 ae e0  93 57 bd 90 08 00 45 00   ·_·]···· ·W····E·
0010  00 54 ad 01 40 00 40 01  79 a5 0a 00 00 01 0a 00   ·T··@·@· y·······
0020  00 02 08 00 c1 6d 40 47  00 3b 01 34 8d 60 00 00   ·····m@G ·;·4·`··
0030  00 00 a1 a8 07 00 00 00  00 00 10 11 12 13 14 15   ········ ········
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ········ ·· !"#$%
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35   &'()*+,- ./012345
0060  36 37                                              67

s1-eth1: <live capture in progress>        Packets: 24 · Displayed: 24 (100.0%)        Profile: Default
```

Q4) Describe the different types of packets observed on h1, h2, h3 and h4?
Ans. The Different types of packets observed are as follows where On h1 and h2 we are able to see ICMP echo-request, ICMP echo-reply, ARP requests and ARP replies (if any exists). In h3 h4 h5 h6 we only observed ARP request packets (broadcast), if any exists.


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
Welcome to Wireshark
Capture
...using this filter:
Enter a capture filter ...
All interfaces shown
s1-eth1
s2-eth1
s3-eth1
s1-eth2
s2-eth2
s3-eth2
s1-eth3
s2-eth3
s3-eth3
s1-eth4
ens33
any
Loopback: lo
bluetooth0
nflog
nfqueue
usbmon1
usbmon2
Cisco remote capture: ciscodump
Random packet generator: randpkt
SSH remote capture: sshdump
UDP Listener remote capture: udpdump

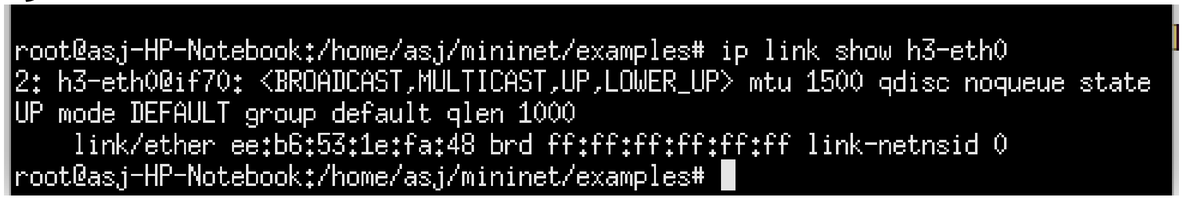
Q5) Compare the packets sent by h1 to the ones received by h3, specifically at source/destination MAC addresses. Explain the similarities and differences, if any.
Ans. Traffic is the same, same ethernet header, same source/destination MAC-addresses. The Ethernet bridge does not affect any source/destination MAC-address, it is transparent to the MAC and IP layers.

H1



```
root@asj-HP-Notebook:/home/asj/mininet/examples# ip link show h1-eth0
2: h1-eth0@if64: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue st
ate UP mode DEFAULT group default qlen 1000
    link/ether ae:e0:93:57:bd:90 brd ff:ff:ff:ff:ff:ff link-netnsid 0
root@asj-HP-Notebook:/home/asj/mininet/examples# 
```

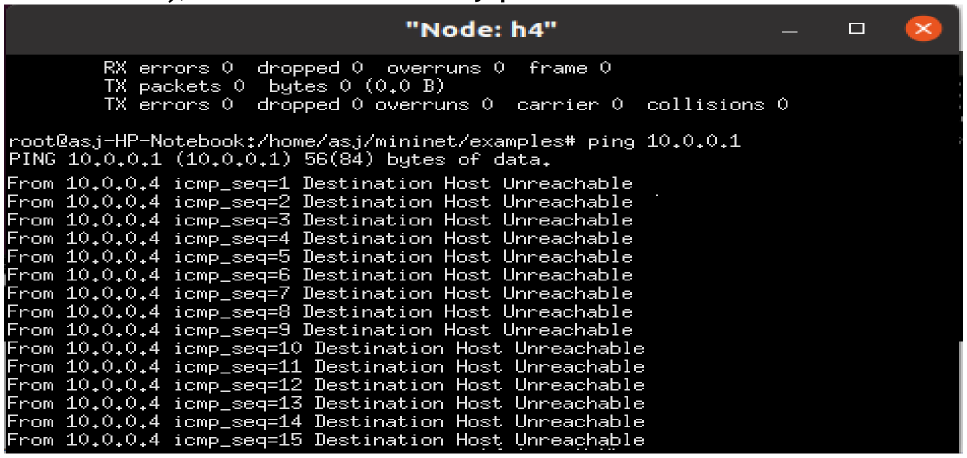
H3



```
root@asj-HP-Notebook:/home/asj/mininet/examples# ip link show h3-eth0
2: h3-eth0@if70: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UP mode DEFAULT group default qlen 1000
    link/ether ee:b6:53:1e:fa:48 brd ff:ff:ff:ff:ff:ff link-netnsid 0
root@asj-HP-Notebook:/home/asj/mininet/examples# 
```

Q6) Ping from h4 to h1 using IPv4. Observe the traffic captured and explain your findings

Ans. When we do ping 10.0.0.1 from the H4 terminal, we see that there are no packets. H4 will use its network mask on h1's IP address and check if they are in the same subnet. As they are not in the same subnet, h4 will attempt to contact its default gateway to send the packet, and if no default gateway is configured (which is this case), it will not send any packet at all.



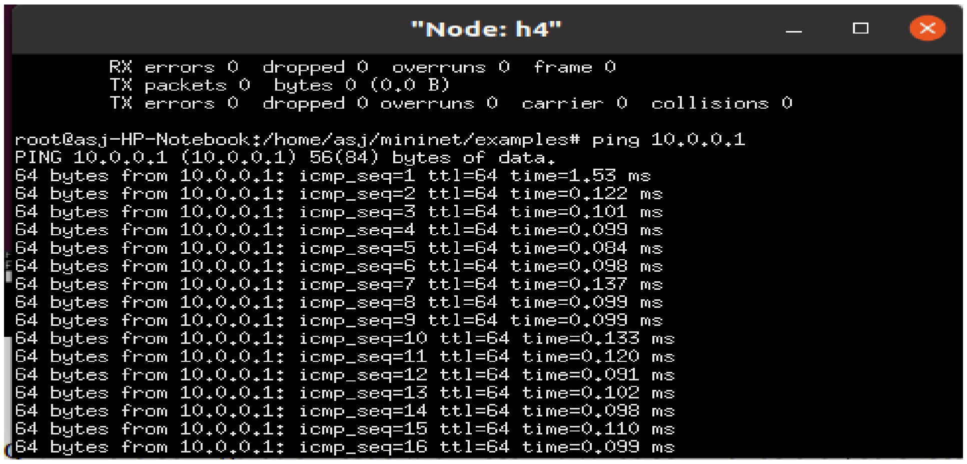
```
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@asj-HP-Notebook:/home/asj/mininet/examples# ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
From 10.0.0.4 icmp_seq=1 Destination Host Unreachable
From 10.0.0.4 icmp_seq=2 Destination Host Unreachable
From 10.0.0.4 icmp_seq=3 Destination Host Unreachable
From 10.0.0.4 icmp_seq=4 Destination Host Unreachable
From 10.0.0.4 icmp_seq=5 Destination Host Unreachable
From 10.0.0.4 icmp_seq=6 Destination Host Unreachable
From 10.0.0.4 icmp_seq=7 Destination Host Unreachable
From 10.0.0.4 icmp_seq=8 Destination Host Unreachable
From 10.0.0.4 icmp_seq=9 Destination Host Unreachable
From 10.0.0.4 icmp_seq=10 Destination Host Unreachable
From 10.0.0.4 icmp_seq=11 Destination Host Unreachable
From 10.0.0.4 icmp_seq=12 Destination Host Unreachable
From 10.0.0.4 icmp_seq=13 Destination Host Unreachable
From 10.0.0.4 icmp_seq=14 Destination Host Unreachable
From 10.0.0.4 icmp_seq=15 Destination Host Unreachable
```

Observation: Destination Host unreachable

Q7) Fix the configuration issue with host h4. What commands did you execute?

Ans. We see that h4 is not in the same subnet as the other hosts and hence there arises a problem in communication.We would have to change the subnet mask of h4.The commands executed are as follows:

1) ip addr flush dev h4-eth0
2) ip addr add 10.0.0.4/24 dev h4-eth0



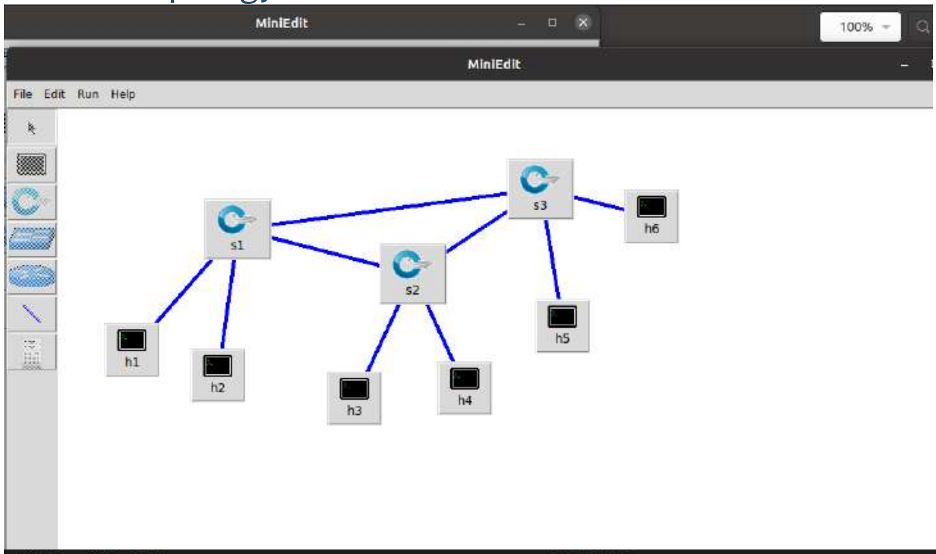
```
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@asj-HP-Notebook:/home/asj/mininet/examples# ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=1.53 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.122 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.101 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.099 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=0.084 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=64 time=0.098 ms
64 bytes from 10.0.0.1: icmp_seq=7 ttl=64 time=0.137 ms
64 bytes from 10.0.0.1: icmp_seq=8 ttl=64 time=0.099 ms
64 bytes from 10.0.0.1: icmp_seq=9 ttl=64 time=0.099 ms
64 bytes from 10.0.0.1: icmp_seq=10 ttl=64 time=0.133 ms
64 bytes from 10.0.0.1: icmp_seq=11 ttl=64 time=0.120 ms
64 bytes from 10.0.0.1: icmp_seq=12 ttl=64 time=0.091 ms
64 bytes from 10.0.0.1: icmp_seq=13 ttl=64 time=0.102 ms
64 bytes from 10.0.0.1: icmp_seq=14 ttl=64 time=0.098 ms
64 bytes from 10.0.0.1: icmp_seq=15 ttl=64 time=0.110 ms
64 bytes from 10.0.0.1: icmp_seq=16 ttl=64 time=0.099 ms
```

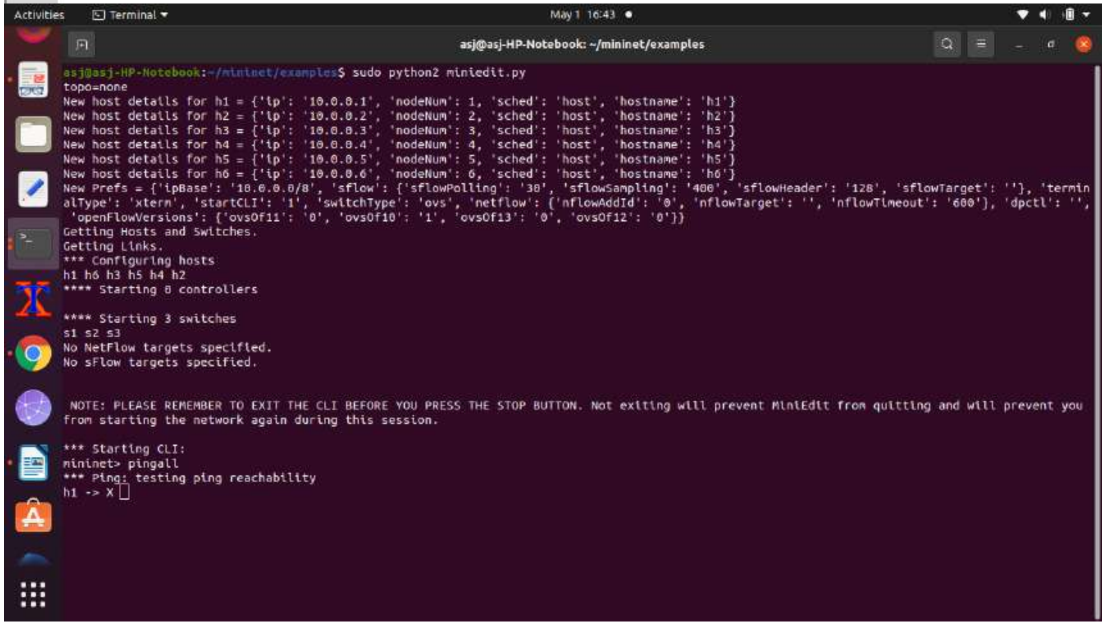
Observation: Ping command executed successfully

Q8) what is the benefit of IPv6 over IPv4? Explain your answer based on your findings in previous questions.
Ans. IPV6 hosts usually self-allocate a link-local address automatically. This allows alls hosts that are in the same LAN to communicate using IPv6 without any configuration. In contrast, for the linux distribution we are using, no such thing happens with IPv4 (in contrast, link-local IPv4 addresses may be automatically allocated by Windows systems).

ii. Configure a switch to handle loops

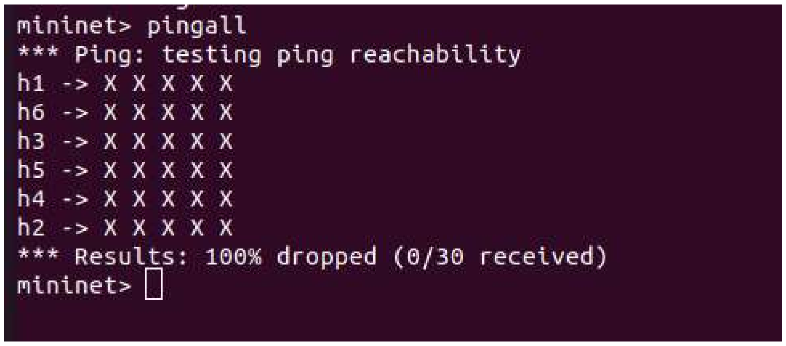
Custom Topology:





```
asj@asj-HP-Notebook:~/mininet/examples$ sudo python2 miniedit.py
topo=none
New host details for h1 = {'ip': '10.0.0.1', 'nodeNum': 1, 'sched': 'host', 'hostname': 'h1'}
New host details for h2 = {'ip': '10.0.0.2', 'nodeNum': 2, 'sched': 'host', 'hostname': 'h2'}
New host details for h3 = {'ip': '10.0.0.3', 'nodeNum': 3, 'sched': 'host', 'hostname': 'h3'}
New host details for h4 = {'ip': '10.0.0.4', 'nodeNum': 4, 'sched': 'host', 'hostname': 'h4'}
New host details for h5 = {'ip': '10.0.0.5', 'nodeNum': 5, 'sched': 'host', 'hostname': 'h5'}
New host details for h6 = {'ip': '10.0.0.6', 'nodeNum': 6, 'sched': 'host', 'hostname': 'h6'}
New Prefs = {'ipBase': '10.0.0.0/8', 'sflow': {'sflowPolling': '30', 'sflowSampling': '400', 'sflowHeader': '128', 'sflowTarget': ''}, 'termin
alType': 'xterm', 'startCLI': '1', 'switchType': 'ovs', 'netflow': {'nflowAddId': '0', 'nflowTarget': '', 'nflowTimeout': '600'}, 'dpctl': '',
 'openFlowVersions': {'ovsOf11': '0', 'ovsOf10': '1', 'ovsOf13': '0', 'ovsOf12': '0'}}
Getting Hosts and Switches.
Getting Links.
*** Configuring hosts
h1 h6 h3 h5 h4 h2
**** Starting 0 controllers

**** Starting 3 switches
s1 s2 s3
No NetFlow targets specified.
No sFlow targets specified.


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

*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> X
```

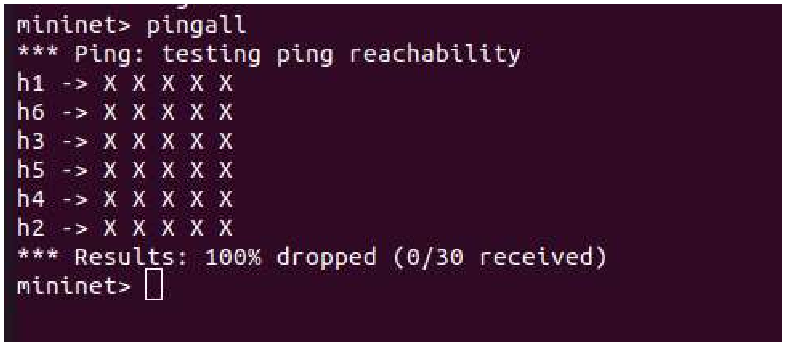
Q9) What is the percentage of dropped packets? Why does it happen?



```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X X X X
h6 -> X X X X X
h3 -> X X X X X
h5 -> X X X X X
h4 -> X X X X X
h2 -> X X X X X
*** Results: 100% dropped (0/30 received)
mininet> 
```

All the packets are dropped. There are two reasons:
1) The broadcasting of packets does not work due to the loop in the network; and
2) Since the switches learn and update their forwarding table based on the response packets, the existence of a loop in the network creates inconsistency in the forwarding tables. For example, h2 wants to send a packet to h4 for the first time. The forwarding table on s1 does not have any information about this MAC address of h4 in its forwarding table. It sends a broadcast packet to the LAN, searching for the owner of this MAC address (h4). The packet to h4 traverses through s3 and also through s2 (and then 4 s3). H4 responds to both messages and send them back from their own path and in return the forwarding tables of the switches are updated. The switch s1 receives both responses and has to update its forwarding table with two different values. This may cause inconsistency in the LAN

Q10) What is the percentage of dropped packets when using IPv6 address? Why does it happen?
Ans. 100%. Similarly to IPv4 addressing, using IPv6 addresses has the same issues in the network with loop: the broadcast issue and the inconsistency in the forwarding tables



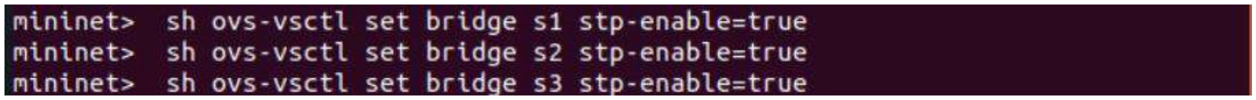
```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X X X X
h6 -> X X X X X
h3 -> X X X X X
h5 -> X X X X X
h4 -> X X X X X
h2 -> X X X X X
*** Results: 100% dropped (0/30 received)
mininet> 
```

Q11) What does the spanning tree protocol achieve?

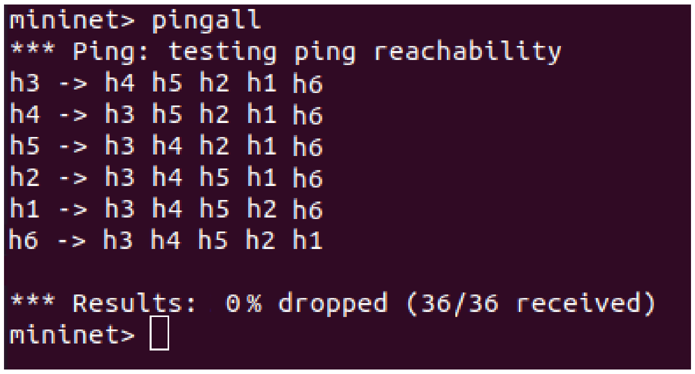
Ans. It breaks the loops in the LAN by forcing the active topology to be a tree; here it disables one of the switch interfaces.

Q12) Write down the path of the packets for each pair of hosts?

Ans.



```
mininet>  sh ovs-vsctl set bridge s1 stp-enable=true
mininet>  sh ovs-vsctl set bridge s2 stp-enable=true
mininet>  sh ovs-vsctl set bridge s3 stp-enable=true
```



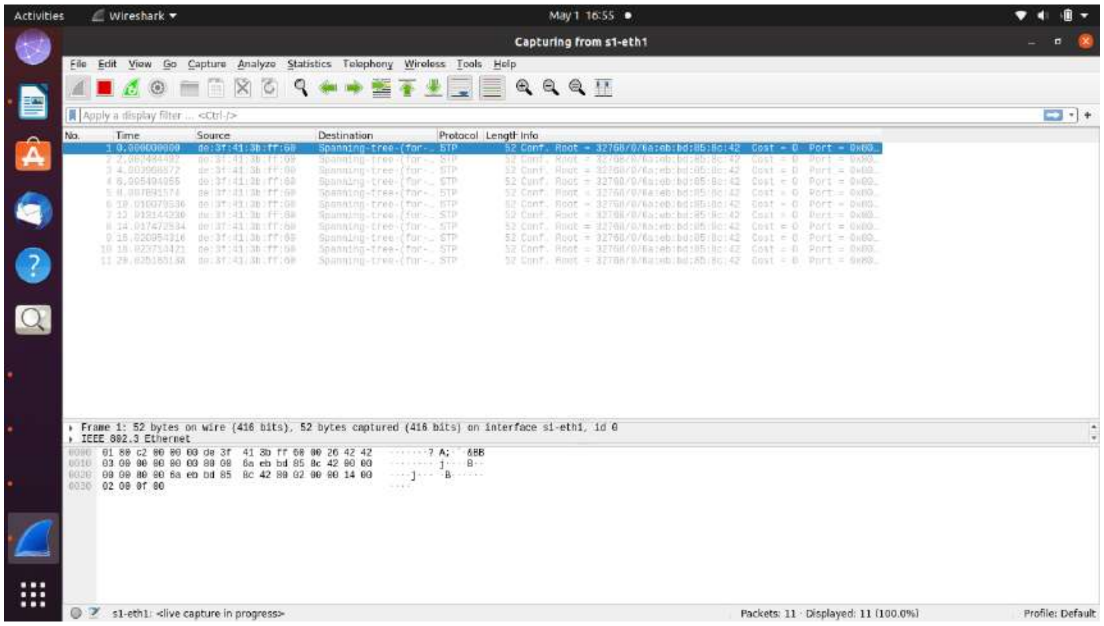
```
mininet> pingall
*** Ping: testing ping reachability
h3 -> h4 h5 h2 h1 h6
h4 -> h3 h5 h2 h1 h6
h5 -> h3 h4 h2 h1 h6
h2 -> h3 h4 h5 h1 h6
h1 -> h3 h4 h5 h2 h6
h6 -> h3 h4 h5 h2 h1

*** Results:  0% dropped (36/36 received)
mininet> 
```

Q13) Are the hosts following the shortest path to send their packets to the destinations? Explain

Ans. No. The shortest path between h1 and h3 is: h1 → s1 → s2 → h3 (might be different for each person; but at least one of the pairs should not follow the shortest path). However, the link between the switches s1 and s2 is disabled due to STP and h1 to h3 does not follow the shortest path.

Q14) Write down the status (enabled or disabled) of each link between the switches. How did STP react when the link broke down? Explain.

Ans. The link that was disabled by STP is now enabled. It takes some time for STP to detect a broken link; as soon as discovery of no connectivity in the network, STP updates the forwarding tables of the switches again.

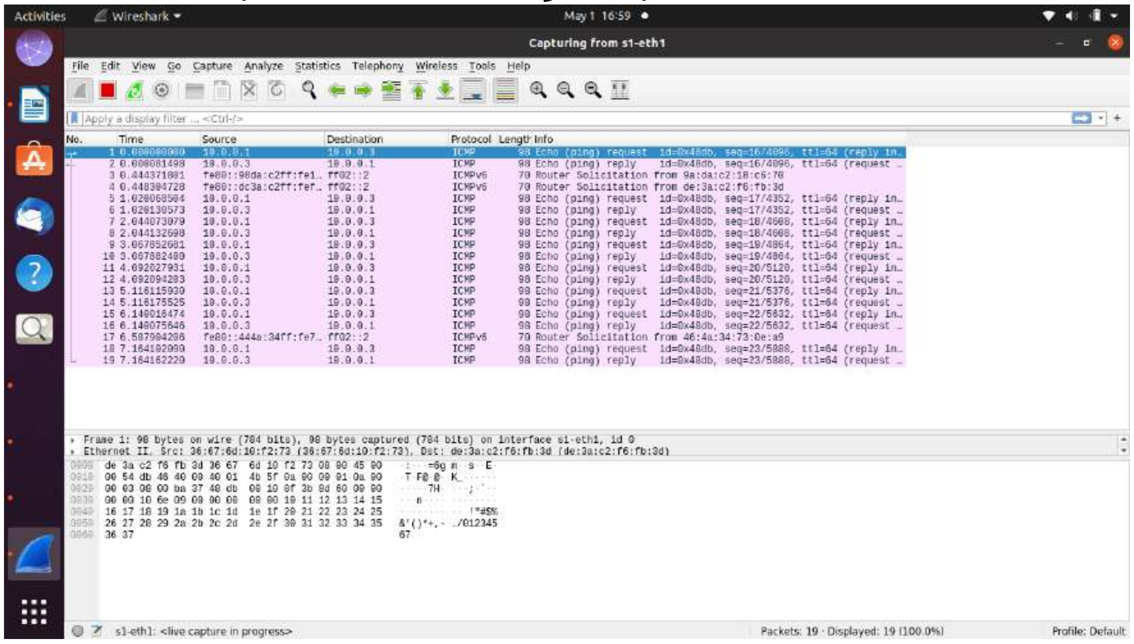


```
No.  Time          Source              Destination            Protocol Length Info
   1 0.000000000   de:3f:41:3b:ff:60   Spanning-tree-(for-…   STP      52 Conf. Root = 32768/0/6a:eb:bd:85:8c:42  Cost = 0  Port = 0x80…
   2 2.002484492   de:3f:41:3b:ff:60   Spanning-tree-(for-…   STP      52 Conf. Root = 32768/0/6a:eb:bd:85:8c:42  Cost = 0  Port = 0x80…
   3 4.003908872   de:3f:41:3b:ff:60   Spanning-tree-(for-…   STP      52 Conf. Root = 32768/0/6a:eb:bd:85:8c:42  Cost = 0  Port = 0x80…
   4 6.005494055   de:3f:41:3b:ff:60   Spanning-tree-(for-…   STP      52 Conf. Root = 32768/0/6a:eb:bd:85:8c:42  Cost = 0  Port = 0x80…
   5 8.007891574   de:3f:41:3b:ff:60   Spanning-tree-(for-…   STP      52 Conf. Root = 32768/0/6a:eb:bd:85:8c:42  Cost = 0  Port = 0x80…
   6 10.010079536  de:3f:41:3b:ff:60   Spanning-tree-(for-…   STP      52 Conf. Root = 32768/0/6a:eb:bd:85:8c:42  Cost = 0  Port = 0x80…
   7 12.012144230  de:3f:41:3b:ff:60   Spanning-tree-(for-…   STP      52 Conf. Root = 32768/0/6a:eb:bd:85:8c:42  Cost = 0  Port = 0x80…
   8 14.017472534  de:3f:41:3b:ff:60   Spanning-tree-(for-…   STP      52 Conf. Root = 32768/0/6a:eb:bd:85:8c:42  Cost = 0  Port = 0x80…
   9 16.020954316  de:3f:41:3b:ff:60   Spanning-tree-(for-…   STP      52 Conf. Root = 32768/0/6a:eb:bd:85:8c:42  Cost = 0  Port = 0x80…
  10 18.023714421  de:3f:41:3b:ff:60   Spanning-tree-(for-…   STP      52 Conf. Root = 32768/0/6a:eb:bd:85:8c:42  Cost = 0  Port = 0x80…
  11 20.025165138  de:3f:41:3b:ff:60   Spanning-tree-(for-…   STP      52 Conf. Root = 32768/0/6a:eb:bd:85:8c:42  Cost = 0  Port = 0x80…

▸ Frame 1: 52 bytes on wire (416 bits), 52 bytes captured (416 bits) on interface s1-eth1, id 0
▸ IEEE 802.3 Ethernet
0000  01 80 c2 00 00 00 de 3f  41 3b ff 60 00 26 42 42   ·······? A;·`·&BB
0010  03 00 00 00 00 00 80 00  6a eb bd 85 8c 42 00 00   ········j····B··
0020  00 00 80 00 6a eb bd 85  8c 42 80 02 00 00 14 00   ····j····B······
0030  02 00 0f 00                                        ····

s1-eth1: <live capture in progress>          Packets: 11 · Displayed: 11 (100.0%)          Profile: Default
```

Q15) Write down again the path of the packets for each pair of Q12.

Ans.

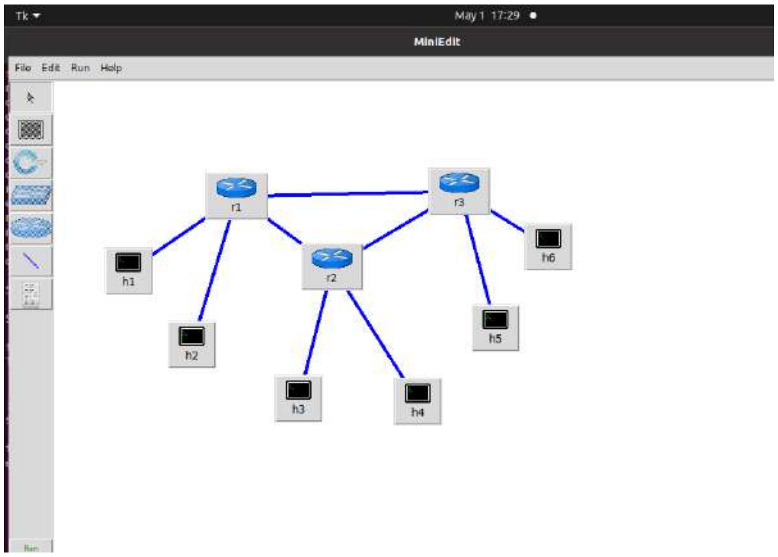
- From h1 to h3 : h1 → s1 → s2 → h3
- From h3 to h4 : h3 → s2 → s3 → h4
- From h2 to h4 : h2 → s1 → s2 → s3 → h4



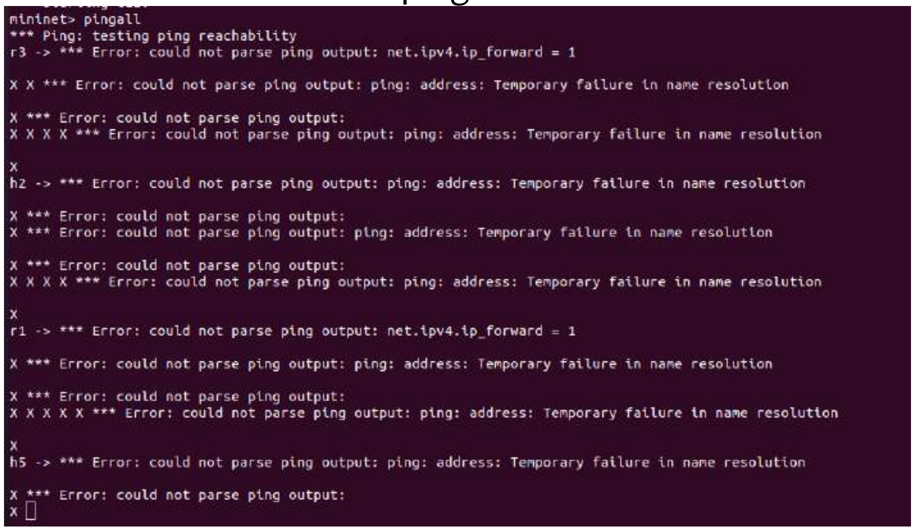
```
No.  Time         Source                 Destination  Protocol Length Info
   1 0.000000000  10.0.0.1               10.0.0.3     ICMP     98 Echo (ping) request  id=0x48db, seq=16/4096, ttl=64 (reply in…
   2 0.000081498  10.0.0.3               10.0.0.1     ICMP     98 Echo (ping) reply    id=0x48db, seq=16/4096, ttl=64 (request …
   3 0.444371081  fe80::98da:c2ff:fe1…   ff02::2      ICMPv6   70 Router Solicitation from 9a:da:c2:18:c6:70
   4 0.448304728  fe80::dc3a:c2ff:fef…   ff02::2      ICMPv6   70 Router Solicitation from de:3a:c2:f6:fb:3d
   5 1.020068504  10.0.0.1               10.0.0.3     ICMP     98 Echo (ping) request  id=0x48db, seq=17/4352, ttl=64 (reply in…
   6 1.020130573  10.0.0.3               10.0.0.1     ICMP     98 Echo (ping) reply    id=0x48db, seq=17/4352, ttl=64 (request …
   7 2.044073079  10.0.0.1               10.0.0.3     ICMP     98 Echo (ping) request  id=0x48db, seq=18/4608, ttl=64 (reply in…
   8 2.044132698  10.0.0.3               10.0.0.1     ICMP     98 Echo (ping) reply    id=0x48db, seq=18/4608, ttl=64 (request …
   9 3.067852681  10.0.0.1               10.0.0.3     ICMP     98 Echo (ping) request  id=0x48db, seq=19/4864, ttl=64 (reply in…
  10 3.067882480  10.0.0.3               10.0.0.1     ICMP     98 Echo (ping) reply    id=0x48db, seq=19/4864, ttl=64 (request …
  11 4.092027931  10.0.0.1               10.0.0.3     ICMP     98 Echo (ping) request  id=0x48db, seq=20/5120, ttl=64 (reply in…
  12 4.092094283  10.0.0.3               10.0.0.1     ICMP     98 Echo (ping) reply    id=0x48db, seq=20/5120, ttl=64 (request …
  13 5.116115930  10.0.0.1               10.0.0.3     ICMP     98 Echo (ping) request  id=0x48db, seq=21/5376, ttl=64 (reply in…
  14 5.116175525  10.0.0.3               10.0.0.1     ICMP     98 Echo (ping) reply    id=0x48db, seq=21/5376, ttl=64 (request …
  15 6.140016474  10.0.0.1               10.0.0.3     ICMP     98 Echo (ping) request  id=0x48db, seq=22/5632, ttl=64 (reply in…
  16 6.140075646  10.0.0.3               10.0.0.1     ICMP     98 Echo (ping) reply    id=0x48db, seq=22/5632, ttl=64 (request …
  17 6.587904206  fe80::444a:34ff:fe7…   ff02::2      ICMPv6   70 Router Solicitation from 46:4a:34:73:0e:a9
  18 7.164102090  10.0.0.1               10.0.0.3     ICMP     98 Echo (ping) request  id=0x48db, seq=23/5888, ttl=64 (reply in…
  19 7.164162220  10.0.0.3               10.0.0.1     ICMP     98 Echo (ping) reply    id=0x48db, seq=23/5888, ttl=64 (request …

▸ Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface s1-eth1, id 0
▸ Ethernet II, Src: 36:67:6d:10:f2:73 (36:67:6d:10:f2:73), Dst: de:3a:c2:f6:fb:3d (de:3a:c2:f6:fb:3d)
0000  de 3a c2 f6 fb 3d 36 67  6d 10 f2 73 08 00 45 00   ·:···=6g m··s··E·
0010  00 54 db 46 40 00 40 01  4b 5f 0a 00 00 01 0a 00   ·T·F@·@· K_······
0020  00 03 08 00 ba 37 48 db  00 10 0f 3b 8d 60 00 00   ·····7H· ···;·`··
0030  00 00 10 6e 09 00 00 00  00 00 10 11 12 13 14 15   ···n···· ········
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ········ ·· !"#$%
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35   &'()*+,- ./012345
0060  36 37                                              67

s1-eth1: <live capture in progress>          Packets: 19 · Displayed: 19 (100.0%)          Profile: Default
```

iii. Using a router as a networking device

Custom Topology:



Q16) Ping all hosts from each other using IPv4 and IPv6. Which hosts are unable to reach one another?
Ans. Neither of hosts can ping each other



```
mininet> pingall
*** Ping: testing ping reachability
r3 -> *** Error: could not parse ping output: net.ipv4.ip_forward = 1

X X *** Error: could not parse ping output: ping: address: Temporary failure in name resolution

X *** Error: could not parse ping output:
X X X X *** Error: could not parse ping output: ping: address: Temporary failure in name resolution

X
h2 -> *** Error: could not parse ping output: ping: address: Temporary failure in name resolution

X *** Error: could not parse ping output:
X *** Error: could not parse ping output: ping: address: Temporary failure in name resolution

X *** Error: could not parse ping output:
X X X X *** Error: could not parse ping output: ping: address: Temporary failure in name resolution

X
r1 -> *** Error: could not parse ping output: net.ipv4.ip_forward = 1

X *** Error: could not parse ping output: ping: address: Temporary failure in name resolution

X *** Error: could not parse ping output:
X X X X X *** Error: could not parse ping output: ping: address: Temporary failure in name resolution

X
h5 -> *** Error: could not parse ping output: ping: address: Temporary failure in name resolution

X *** Error: could not parse ping output:
X ▯
```

Q17) What are the interfaces & respective IP addresses (v4 and v6) of the router r1?
Ans. The interfaces for the respective IP addresses are as follows:
r1-eth0 → IPv4: 10.0.1.1;
r1-eth1 → IPv4: 10.0.2.1;
r1-eth2 → IPv4: 10.0.5.1;
r1-eth3 → IPv4: 10.0.7.1;

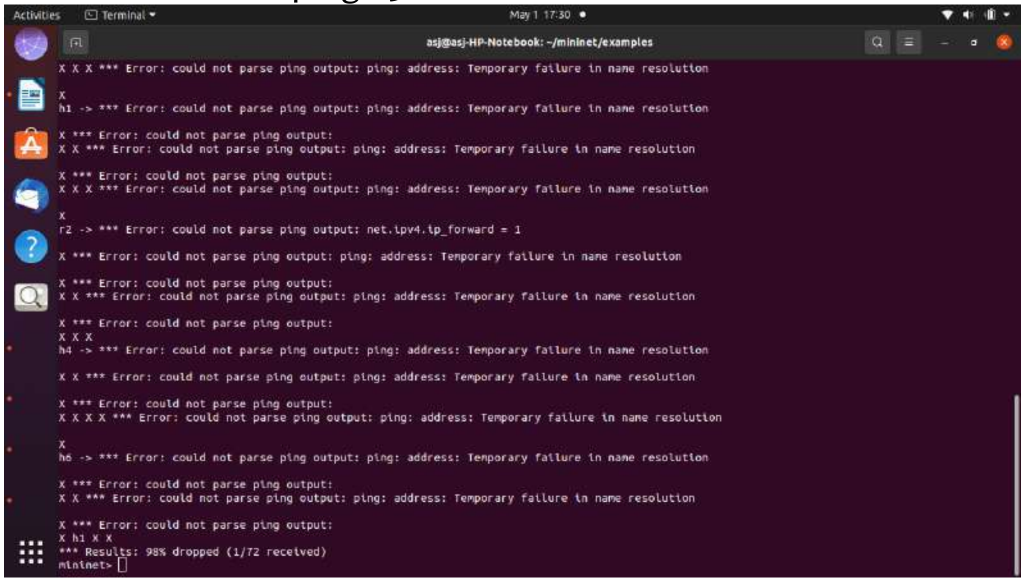
Q18) Can you spot any misconfiguration in the file? (Hint: Take a look at the configurations of h2, r1 and r2?
Ans . We noticed that ip forwarding is disabled at r1; the default gateway of h2 is wrong: it is set to 10.5.0.100 for IPv4 and 2020::10:2 for IPv6, but these IP addresses do not exist in our network; and routing rules are not set in r2.

Q19/ What is the command you used at r1?
Ans. We enable on r1 IPv4 forwarding:
# echo 1 > /proc/sys/net/ipv4/ip forward
and IPv6 forwarding:
# echo 1 > /proc/sys/net/ipv6/conf/all/forwarding

Q20)Which hosts are still unable to ping each other?
Ans. Only h1 can ping h4 and vice versa.

Q21) What are the commands you used in h2 to achieve this?
Ans. For IPv4: ip route add default via 10.0.2.1
For IPv6: ip -6 route add default via 2020::2:1

Q22) Try again to ping hosts from each other with IPv4 and IPv6 addresses. Is there any change in ping results?
Ans. Hosts cannot ping h3 and vice versa.



```
X X X *** Error: could not parse ping output: ping: address: Temporary failure in name resolution

X
h1 -> *** Error: could not parse ping output: ping: address: Temporary failure in name resolution

X *** Error: could not parse ping output:
X X *** Error: could not parse ping output: ping: address: Temporary failure in name resolution

X *** Error: could not parse ping output:
X X X *** Error: could not parse ping output: ping: address: Temporary failure in name resolution

X
r2 -> *** Error: could not parse ping output: net.ipv4.ip_forward = 1

X *** Error: could not parse ping output: ping: address: Temporary failure in name resolution

X *** Error: could not parse ping output:
X X *** Error: could not parse ping output: ping: address: Temporary failure in name resolution

X *** Error: could not parse ping output:
X X X
h4 -> *** Error: could not parse ping output: ping: address: Temporary failure in name resolution

X X *** Error: could not parse ping output: ping: address: Temporary failure in name resolution

X *** Error: could not parse ping output:
X X X X *** Error: could not parse ping output: ping: address: Temporary failure in name resolution

X
h6 -> *** Error: could not parse ping output: ping: address: Temporary failure in name resolution

X *** Error: could not parse ping output:
X X *** Error: could not parse ping output: ping: address: Temporary failure in name resolution

X *** Error: could not parse ping output:
X h1 X X
*** Results: 98% dropped (1/72 received)
mininet> 
```

Q23) Now, check the routing table of r2 for IPv4 addresses. What is the problem with it?
Ans. The routing table is not set properly on r2.

Q24) Solve the issue in r2. What commands do you use?
Ans.

```
 ip route add 10.0.1.0/24 via 10.0.5.1
 ip route add 10.0.2.0/24 via 10.0.5.1
 ip route add 10.0.4.0/24 via 10.0.6.2
 ip route add 10.0.7.0/24 via 10.0.6.2
```

Q25) Check the routing table for IPv6 addresses. Is there any problem with it? Write down the commands to solve it.
Ans. The routing table for IPv6 addresses is not set properly on r2.

```
 ip -6 route add 2020::1:0/112 via 2020::5:1
ip -6 route add 2020::2:0/112 via 2020::5:1
ip -6 route add 2020::4:0/112 via 2020::6:2
 ip -6 route add 2020::7:0/112 via 2020::6:2
```

Q26) How do they differ in subnet mask and IP address assignment (IPv4 and IPv6)?
Ans.
In a network with switches only, all hosts receive link-local IPv6 addresses automatically therefore can communicate with each other without configuration. With IPv4, all hosts should be configured with the same IPv4 subnet mask and prefix (except if IPv4 auto-configuration is happening). In contrast, when using routers, each of its interface is connected to a separate IPv4/v6 network, i.e., subnet prefixes are different, and hosts can take whatever IPv4 and IPv6 addresses within the valid range. In this case, the hosts still receive link-local IPv6 addresses, but they cannot use them to communicate with another host in another subnet.

Q27) How do they differ in creating routing/forwarding tables?
Ans.
The switches set the forwarding tables by learning. In a network without loops, no configuration is needed to connect the hosts. If the network has loops of switches, then by enabling STP in all of switches the forwarding tables are created . However, in a network of routers, the routing tables must be set manually (this can be done automatic by running protocols like OSPF (you will do it in the next labs).

Q28) How do they differ in performance/efficiency in a network with loops?
Ans. When there is a loop in the network of switches, one (or several) link is disabled to break the loop. The side-effects of this action is increasing the load on the other active links in the network and paths are not the shortest. However, in the network of routers, all links are used and the traffic is balanced. Moreover, all paths can be the shortest, e.g., if you run OSPF