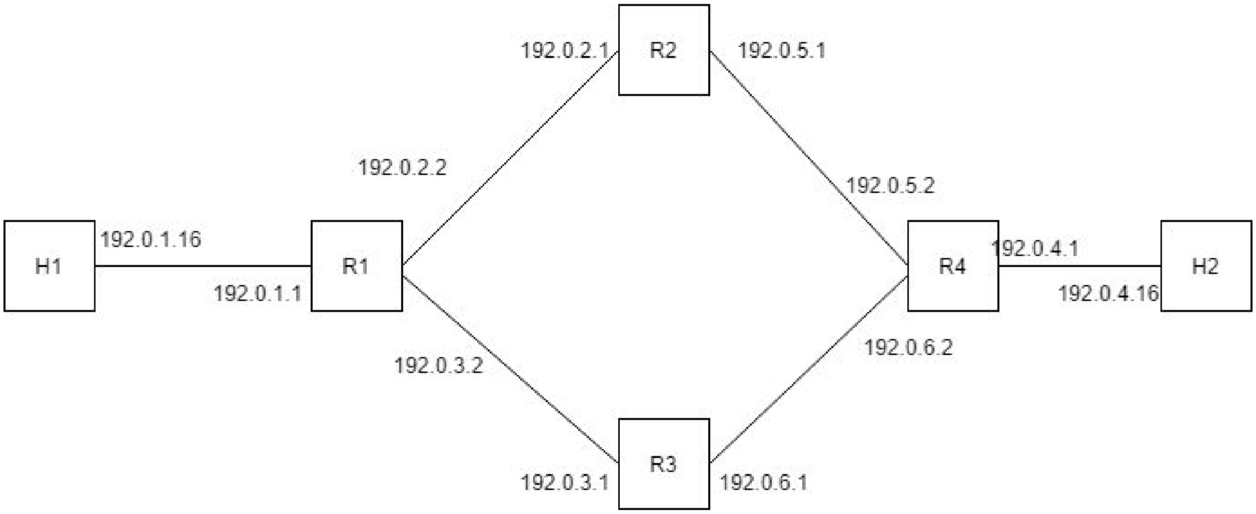
Part A

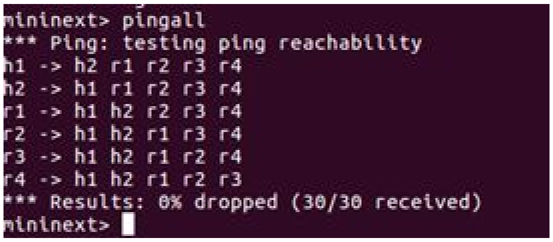
A1 b):



Subnet 1: 192.0.1.0/24
Subnet 2: 192.0.2.0/24
Subnet 3: 192.0.3.0/24
Subnet 4: 192.0.4.0/24
Subnet 5 :192.0.5.0/24
Subnet 6: 192.0.6.0/24

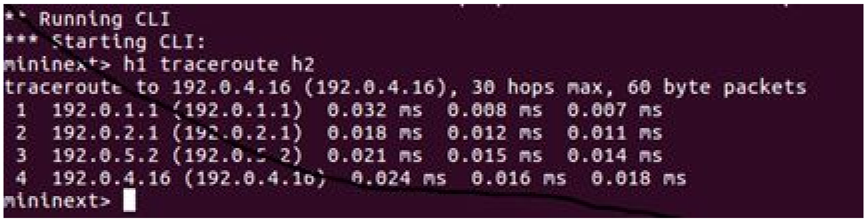
A2 a):

ping successful:



```
mininext> pingall
*** Ping: testing ping reachability
h1 -> h2 r1 r2 r3 r4
h2 -> h1 r1 r2 r3 r4
r1 -> h1 h2 r2 r3 r4
r2 -> h1 h2 r1 r3 r4
r3 -> h1 h2 r1 r2 r4
r4 -> h1 h2 r1 r2 r3
*** Results: 0% dropped (30/30 received)
mininext> █
```

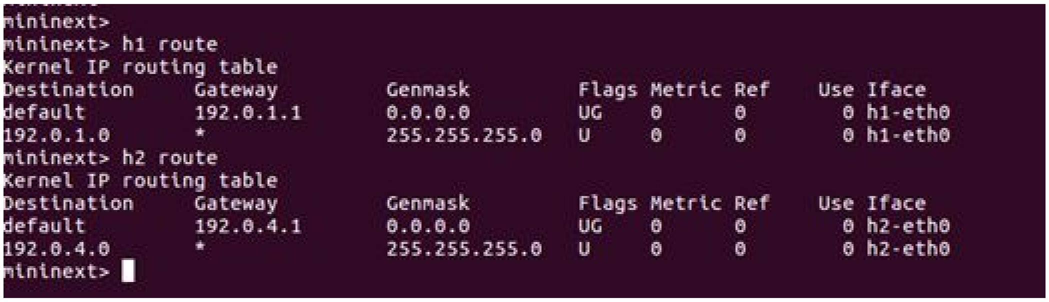
Traceroute output:



```
** Running CLI
*** Starting CLI:
mininext> h1 traceroute h2
traceroute to 192.0.4.16 (192.0.4.16), 30 hops max, 60 byte packets
 1  192.0.1.1 (192.0.1.1)  0.032 ms  0.008 ms  0.007 ms
 2  192.0.2.1 (192.0.2.1)  0.018 ms  0.012 ms  0.011 ms
 3  192.0.5.2 (192.0.5.2)  0.021 ms  0.015 ms  0.014 ms
 4  192.0.4.16 (192.0.4.16)  0.024 ms  0.016 ms  0.018 ms
mininext> █
```

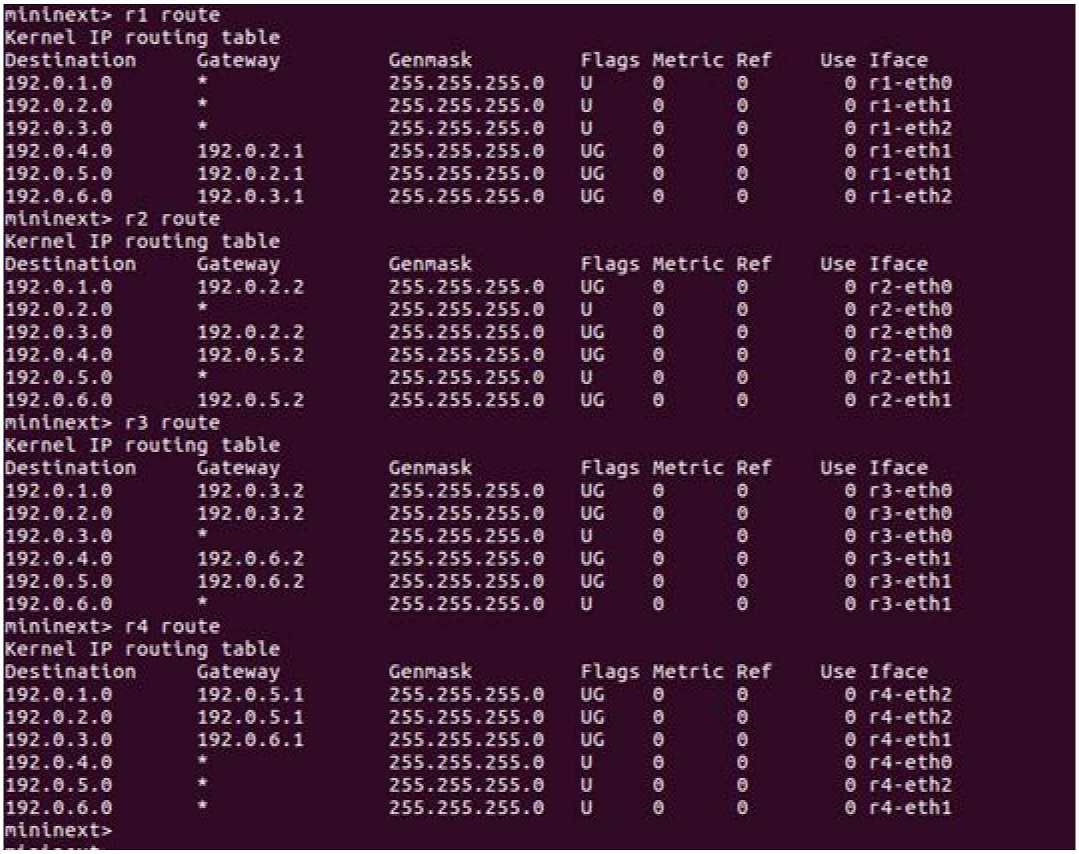
Routing tables :

H1 and H2 :



```
mininext>
mininext> h1 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         192.0.1.1       0.0.0.0         UG    0      0        0 h1-eth0
192.0.1.0       *               255.255.255.0   U     0      0        0 h1-eth0
mininext> h2 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         192.0.4.1       0.0.0.0         UG    0      0        0 h2-eth0
192.0.4.0       *               255.255.255.0   U     0      0        0 h2-eth0
mininext> █
```

R1 to R4 routing tables :



```
mininext> r1 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.0.1.0       *               255.255.255.0   U     0      0        0 r1-eth0
192.0.2.0       *               255.255.255.0   U     0      0        0 r1-eth1
192.0.3.0       *               255.255.255.0   U     0      0        0 r1-eth2
192.0.4.0       192.0.2.1       255.255.255.0   UG    0      0        0 r1-eth1
192.0.5.0       192.0.2.1       255.255.255.0   UG    0      0        0 r1-eth1
192.0.6.0       192.0.3.1       255.255.255.0   UG    0      0        0 r1-eth2
mininext> r2 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.0.1.0       192.0.2.2       255.255.255.0   UG    0      0        0 r2-eth0
192.0.2.0       *               255.255.255.0   U     0      0        0 r2-eth0
192.0.3.0       192.0.2.2       255.255.255.0   UG    0      0        0 r2-eth0
192.0.4.0       192.0.5.2       255.255.255.0   UG    0      0        0 r2-eth1
192.0.5.0       *               255.255.255.0   U     0      0        0 r2-eth1
192.0.6.0       192.0.5.2       255.255.255.0   UG    0      0        0 r2-eth1
mininext> r3 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.0.1.0       192.0.3.2       255.255.255.0   UG    0      0        0 r3-eth0
192.0.2.0       192.0.3.2       255.255.255.0   UG    0      0        0 r3-eth0
192.0.3.0       *               255.255.255.0   U     0      0        0 r3-eth0
192.0.4.0       192.0.6.2       255.255.255.0   UG    0      0        0 r3-eth1
192.0.5.0       192.0.6.2       255.255.255.0   UG    0      0        0 r3-eth1
192.0.6.0       *               255.255.255.0   U     0      0        0 r3-eth1
mininext> r4 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.0.1.0       192.0.5.1       255.255.255.0   UG    0      0        0 r4-eth2
192.0.2.0       192.0.5.1       255.255.255.0   UG    0      0        0 r4-eth2
192.0.3.0       192.0.6.1       255.255.255.0   UG    0      0        0 r4-eth1
192.0.4.0       *               255.255.255.0   U     0      0        0 r4-eth0
192.0.5.0       *               255.255.255.0   U     0      0        0 r4-eth2
192.0.6.0       *               255.255.255.0   U     0      0        0 r4-eth1
mininext>
```

Steps to set up static routes:

Step 1: Enable ip forwarding on each router r1,r2,r3,r4. The commands are written in start.py

Step 2:Set ip to each interface eth1 and eth2 of each router using command ifconfig.

e.g. ifconfig r1-eth1 192.0.2.2/24

Step 3: Add static routes on each router using ip route add “subnet” via the first hop of the directly connected subnet through which we have to take that path.

Step 4:Add default gateways to hosts h1 and h2 using command : route add default gw 192.0.1.1

Commands in start.py:

```
net.get("r1").cmd("sysctl net.ipv4.ip_forward=1")
    net.get("r2").cmd("sysctl net.ipv4.ip_forward=1")
    net.get("r3").cmd("sysctl net.ipv4.ip_forward=1")
    net.get("r4").cmd("sysctl net.ipv4.ip_forward=1")


    net.get("r1").cmd("ifconfig r1-eth1 192.0.2.2/24")



    net.get("r2").cmd("ip route add 192.0.1.0/24 via 192.0.2.2")
    net.get("r2").cmd("ifconfig r2-eth1 192.0.5.1/24")
    net.get("r2").cmd("ip route add 192.0.4.0/24 via 192.0.5.2")


    net.get("r4").cmd("ifconfig r4-eth2 192.0.5.2/24")
    net.get("r4").cmd("ip route add 192.0.2.0/24 via 192.0.5.1")
    net.get("r4").cmd("ip route add 192.0.1.0/24 via 192.0.5.1")


    net.get("r1").cmd("ip route add 192.0.5.0/24 via 192.0.2.1")
    net.get("r1").cmd("ip route add 192.0.4.0/24 via 192.0.2.1")

    net.get("h1").cmd("route add default gw 192.0.1.1")
    net.get("h2").cmd("route add default gw 192.0.4.1")

    net.get("r1").cmd("ifconfig r1-eth2 192.0.3.2/24")
    net.get("r3").cmd("ip route add 192.0.1.0/24 via 192.0.3.2")
    net.get("r1").cmd("ip route add 192.0.6.0/24 via 192.0.3.1")
    net.get("r3").cmd("ifconfig r3-eth1 192.0.6.1/24")
    net.get("r4").cmd("ifconfig r4-eth1 192.0.6.2/24")
    net.get("r3").cmd("ip route add 192.0.4.0/24 via 192.0.6.2")
    net.get("r1").cmd("ip route add 192.0.3.0/24 via 192.0.6.1")

    net.get("r2").cmd("ip route add 192.0.3.0/24 via 192.0.2.2")

    net.get("r2").cmd("ip route add 192.0.6.0/24 via 192.0.5.2")

    net.get("r3").cmd("ip route add 192.0.2.0/24 via 192.0.3.2")
    net.get("r3").cmd("ip route add 192.0.5.0/24 via 192.0.6.2")
    net.get("r4").cmd("ip route add 192.0.3.0/24 via 192.0.6.1")
```

Part B:

B1 a) & b)

Step 1: copy daemon.conf, zebra.conf and ripd.conf on each node from sample code

Step 2 : edited daemon.conf in for each node. Zebra and ripd to yes

```
zebra=yes
bgpd=no
ospfd=no
ospf6d=no
ripd=yes
ripngd=no
isisd=no
```

Step 3: edited all zebra.conf and ripd.conf for hostname and password

```
e.g. hostname r1
   password quagga
```

Step 4 : start quagga services by following command

```
e.g. r1 service quagga restart
```

Step 5: Give following permissions to configurations file

```
e.g. r1  chown quagga:quaggavty /etc/quagga/*.conf
     r1 chmod 640 /etc/quagga/*.conf
```

Step 6:Now we have to login to **zebra** through telnet using following command and provide following commands to edit zebra file’s configurations

r1 telnet localhost 2601

```
provide password
enable
provide password
config term // configure terminal
interface r1-eth0 //  set interface ip
ip address 192.0.1.1/24
no shutdown //  set that interface up
Exit //  exit from this interface configuration
interface r1-eth1
ip address 192.0.1.1/24
no shutdown
exit
interface r1-eth2
ip address 192.0.1.1/24
no shutdown
Exit
```

```
Write // writing these configurations to config files
```

Step 7:Now we have to login to **ripd** through telnet using following command and provide following commands to edit zebra file’s configurations
R1 telnet localhost 2601

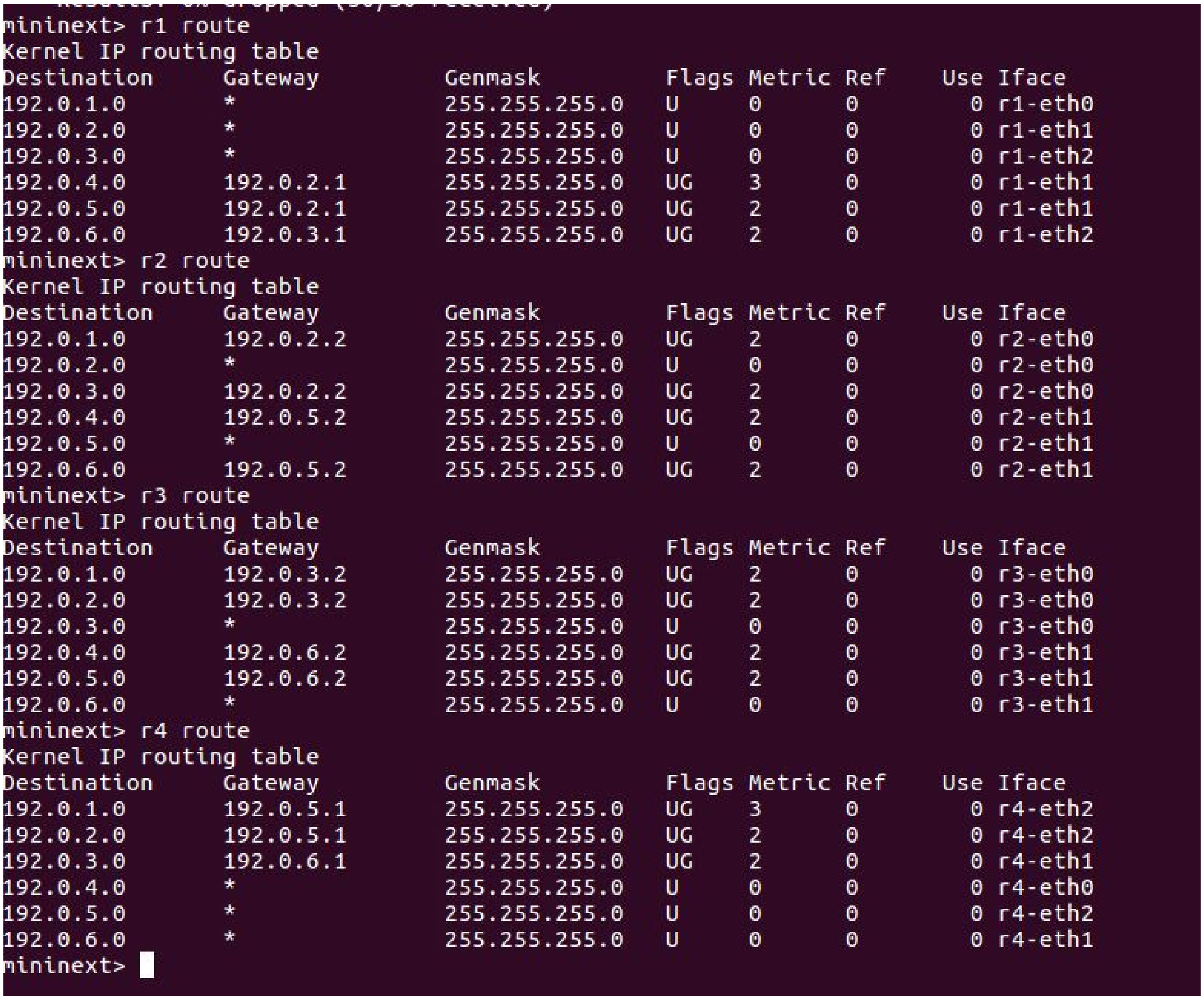
```
Provide password
Enable
Config term
Router rip //RIP protocol we are setting up
Version 2
network 192.0.1.0/24 //setting the subnet networks which are connected to this router
Network 192.0.2.0/24
Network 192.0.3.0/24
Passive interface eth1 //  declaring the passive interface for this router as eth0 will be it’s primary interface
Passive interface eth2
Exit
Write // writing these configurations to config files
```

Step 8: Do above steps for router r2 r3 r4 and for part b2 h1 and h2.
I simply done this edit for single router and then edited conf files for all other routers accordingly.
When the h1 and h2 conf files are not edited I have used default gateway to make them communincate to entire network.
But in part B2 I have configured daemons for both of these hosts also.

B2.

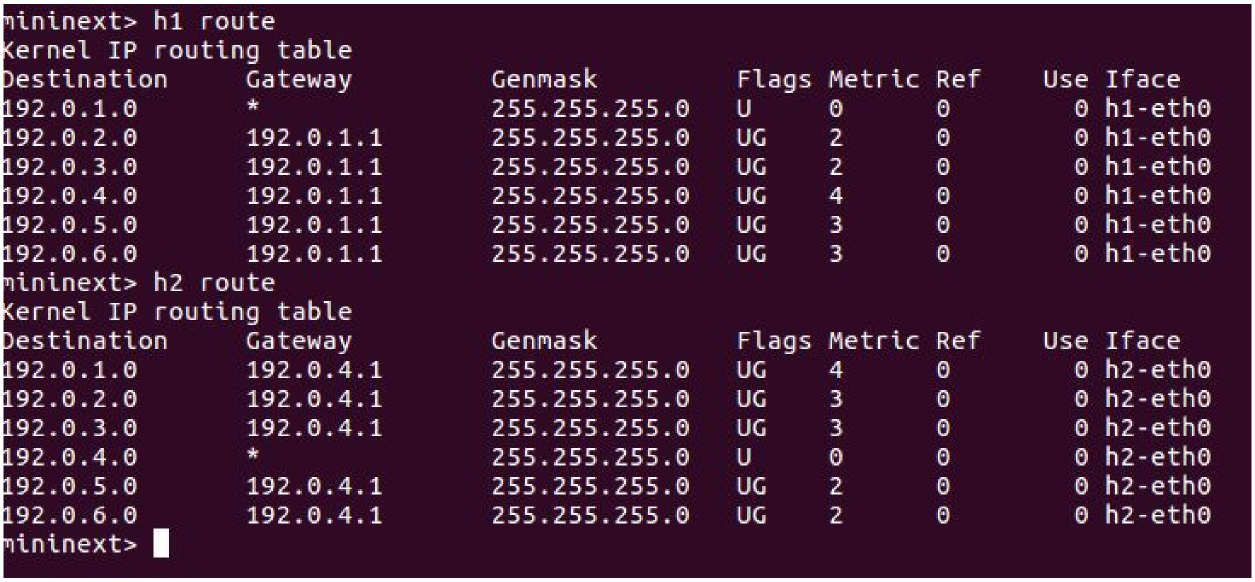
a)

The routing tables are after stting u ripd and zebra config for all hosts and routers
Kernel routing tables:
R1,r2,r3,r4 Kernel routing tables



```
*** Results: 0% dropped (30/30 received)
mininext> r1 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.0.1.0       *               255.255.255.0   U     0      0        0 r1-eth0
192.0.2.0       *               255.255.255.0   U     0      0        0 r1-eth1
192.0.3.0       *               255.255.255.0   U     0      0        0 r1-eth2
192.0.4.0       192.0.2.1       255.255.255.0   UG    3      0        0 r1-eth1
192.0.5.0       192.0.2.1       255.255.255.0   UG    2      0        0 r1-eth1
192.0.6.0       192.0.3.1       255.255.255.0   UG    2      0        0 r1-eth2
mininext> r2 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.0.1.0       192.0.2.2       255.255.255.0   UG    2      0        0 r2-eth0
192.0.2.0       *               255.255.255.0   U     0      0        0 r2-eth0
192.0.3.0       192.0.2.2       255.255.255.0   UG    2      0        0 r2-eth0
192.0.4.0       192.0.5.2       255.255.255.0   UG    2      0        0 r2-eth1
192.0.5.0       *               255.255.255.0   U     0      0        0 r2-eth1
192.0.6.0       192.0.5.2       255.255.255.0   UG    2      0        0 r2-eth1
mininext> r3 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.0.1.0       192.0.3.2       255.255.255.0   UG    2      0        0 r3-eth0
192.0.2.0       192.0.3.2       255.255.255.0   UG    2      0        0 r3-eth0
192.0.3.0       *               255.255.255.0   U     0      0        0 r3-eth0
192.0.4.0       192.0.6.2       255.255.255.0   UG    2      0        0 r3-eth1
192.0.5.0       192.0.6.2       255.255.255.0   UG    2      0        0 r3-eth1
192.0.6.0       *               255.255.255.0   U     0      0        0 r3-eth1
mininext> r4 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.0.1.0       192.0.5.1       255.255.255.0   UG    3      0        0 r4-eth2
192.0.2.0       192.0.5.1       255.255.255.0   UG    2      0        0 r4-eth2
192.0.3.0       192.0.6.1       255.255.255.0   UG    2      0        0 r4-eth1
192.0.4.0       *               255.255.255.0   U     0      0        0 r4-eth0
192.0.5.0       *               255.255.255.0   U     0      0        0 r4-eth2
192.0.6.0       *               255.255.255.0   U     0      0        0 r4-eth1
mininext>
```

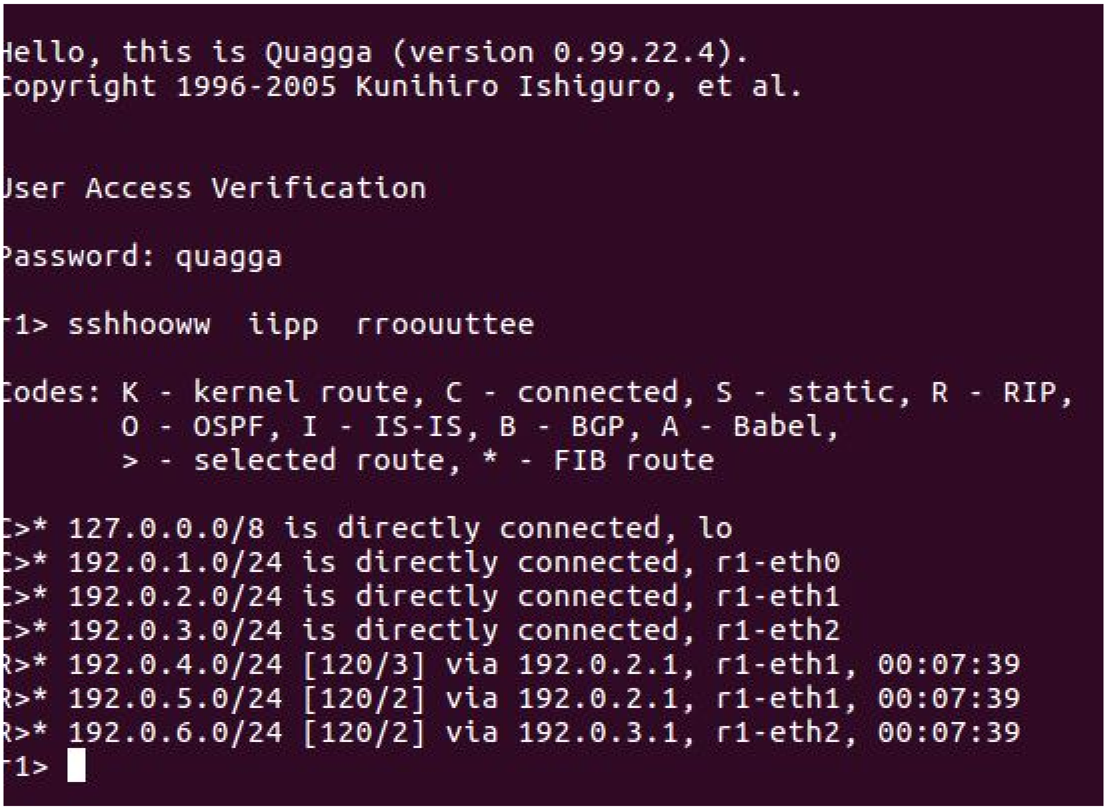
H1,h2 routing tables:



```
mininext> h1 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.0.1.0       *               255.255.255.0   U     0      0        0 h1-eth0
192.0.2.0       192.0.1.1       255.255.255.0   UG    2      0        0 h1-eth0
192.0.3.0       192.0.1.1       255.255.255.0   UG    2      0        0 h1-eth0
192.0.4.0       192.0.1.1       255.255.255.0   UG    4      0        0 h1-eth0
192.0.5.0       192.0.1.1       255.255.255.0   UG    3      0        0 h1-eth0
192.0.6.0       192.0.1.1       255.255.255.0   UG    3      0        0 h1-eth0
mininext> h2 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.0.1.0       192.0.4.1       255.255.255.0   UG    4      0        0 h2-eth0
192.0.2.0       192.0.4.1       255.255.255.0   UG    3      0        0 h2-eth0
192.0.3.0       192.0.4.1       255.255.255.0   UG    3      0        0 h2-eth0
192.0.4.0       *               255.255.255.0   U     0      0        0 h2-eth0
192.0.5.0       192.0.4.1       255.255.255.0   UG    2      0        0 h2-eth0
192.0.6.0       192.0.4.1       255.255.255.0   UG    2      0        0 h2-eth0
mininext>
```

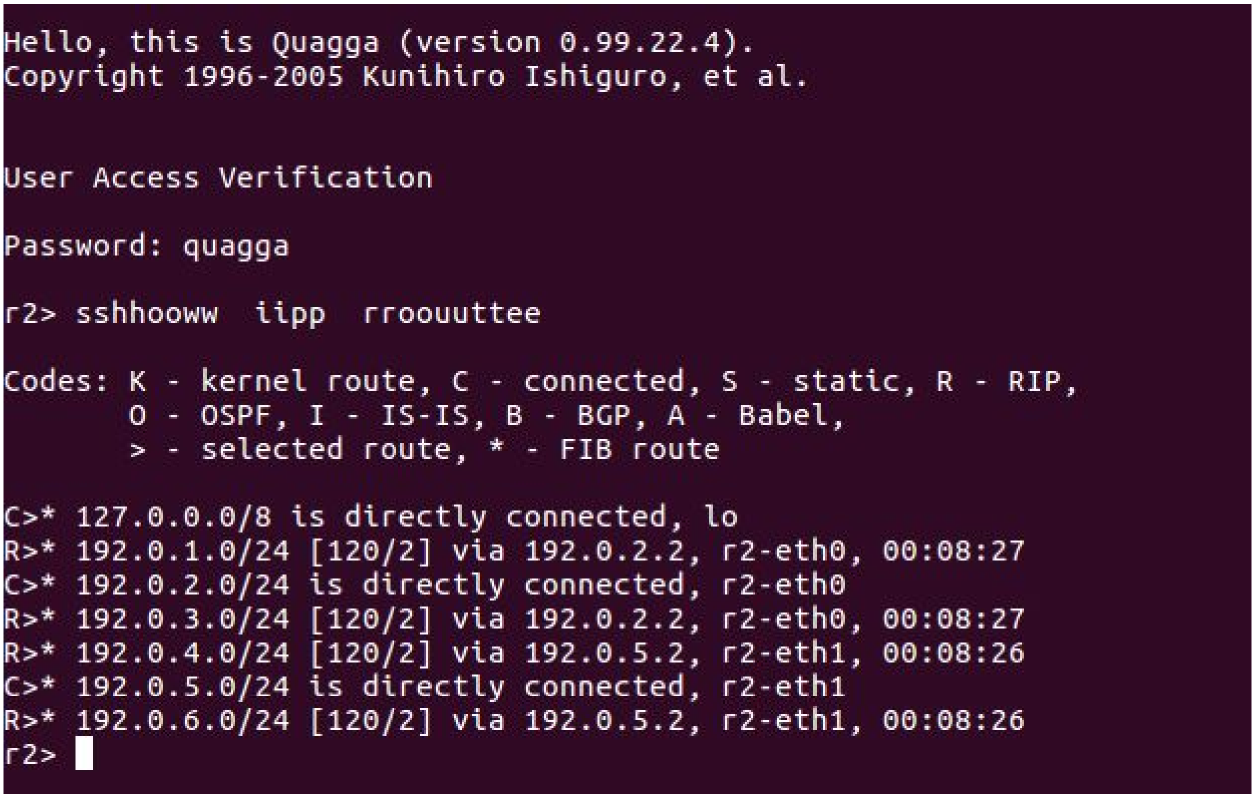
2. Quagga routing tables

R1 :



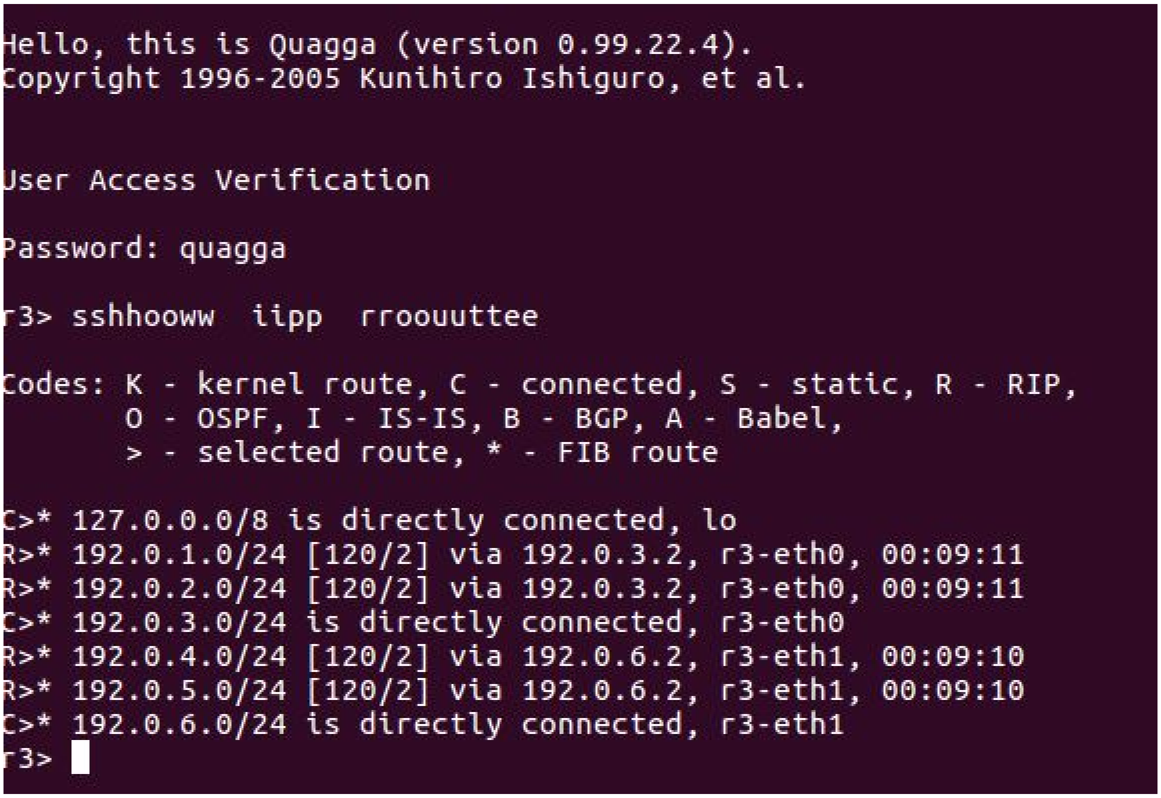
```
Hello, this is Quagga (version 0.99.22.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.


User Access Verification

Password: quagga

r1> sshhooww  iipp  rroouuttee

Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, A - Babel,
       > - selected route, * - FIB route

C>* 127.0.0.0/8 is directly connected, lo
C>* 192.0.1.0/24 is directly connected, r1-eth0
C>* 192.0.2.0/24 is directly connected, r1-eth1
C>* 192.0.3.0/24 is directly connected, r1-eth2
R>* 192.0.4.0/24 [120/3] via 192.0.2.1, r1-eth1, 00:07:39
R>* 192.0.5.0/24 [120/2] via 192.0.2.1, r1-eth1, 00:07:39
R>* 192.0.6.0/24 [120/2] via 192.0.3.1, r1-eth2, 00:07:39
r1> █
```

R2:



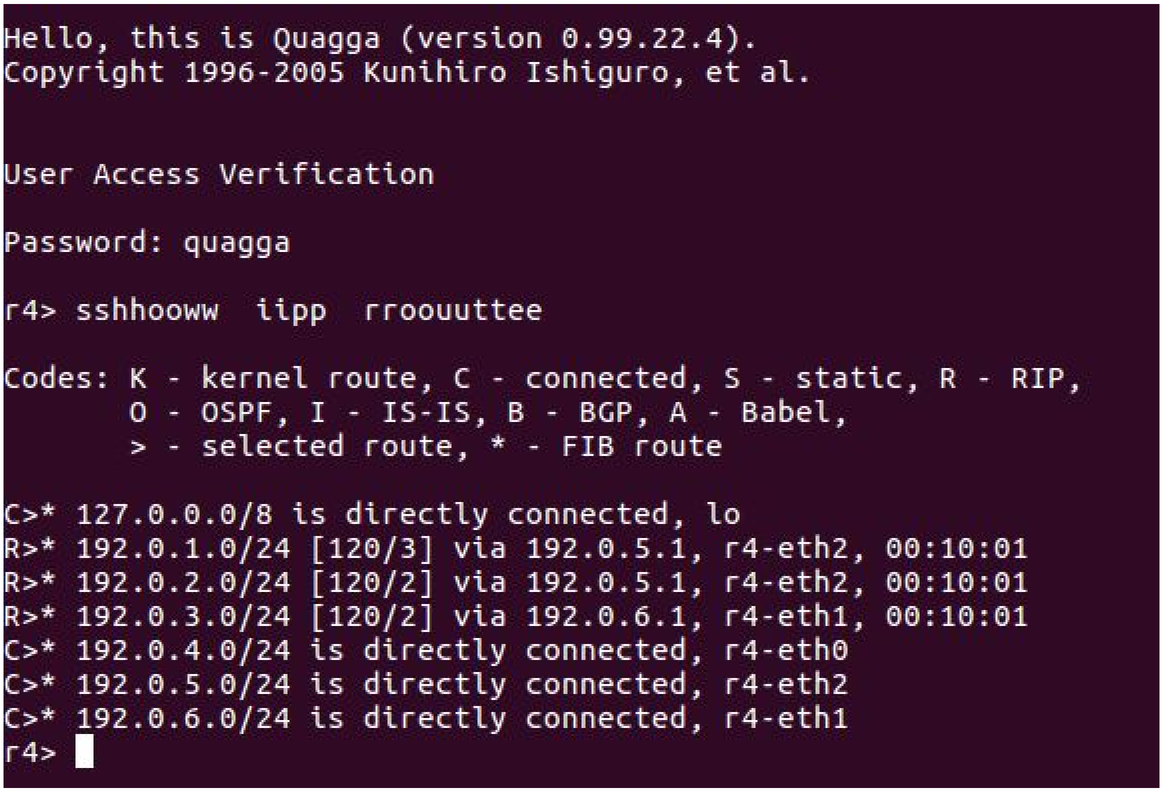
```
Hello, this is Quagga (version 0.99.22.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.


User Access Verification

Password: quagga

r2> sshhooww  iipp  rroouuttee

Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, A - Babel,
       > - selected route, * - FIB route

C>* 127.0.0.0/8 is directly connected, lo
R>* 192.0.1.0/24 [120/2] via 192.0.2.2, r2-eth0, 00:08:27
C>* 192.0.2.0/24 is directly connected, r2-eth0
R>* 192.0.3.0/24 [120/2] via 192.0.2.2, r2-eth0, 00:08:27
R>* 192.0.4.0/24 [120/2] via 192.0.5.2, r2-eth1, 00:08:26
C>* 192.0.5.0/24 is directly connected, r2-eth1
R>* 192.0.6.0/24 [120/2] via 192.0.5.2, r2-eth1, 00:08:26
r2>
```

R3:



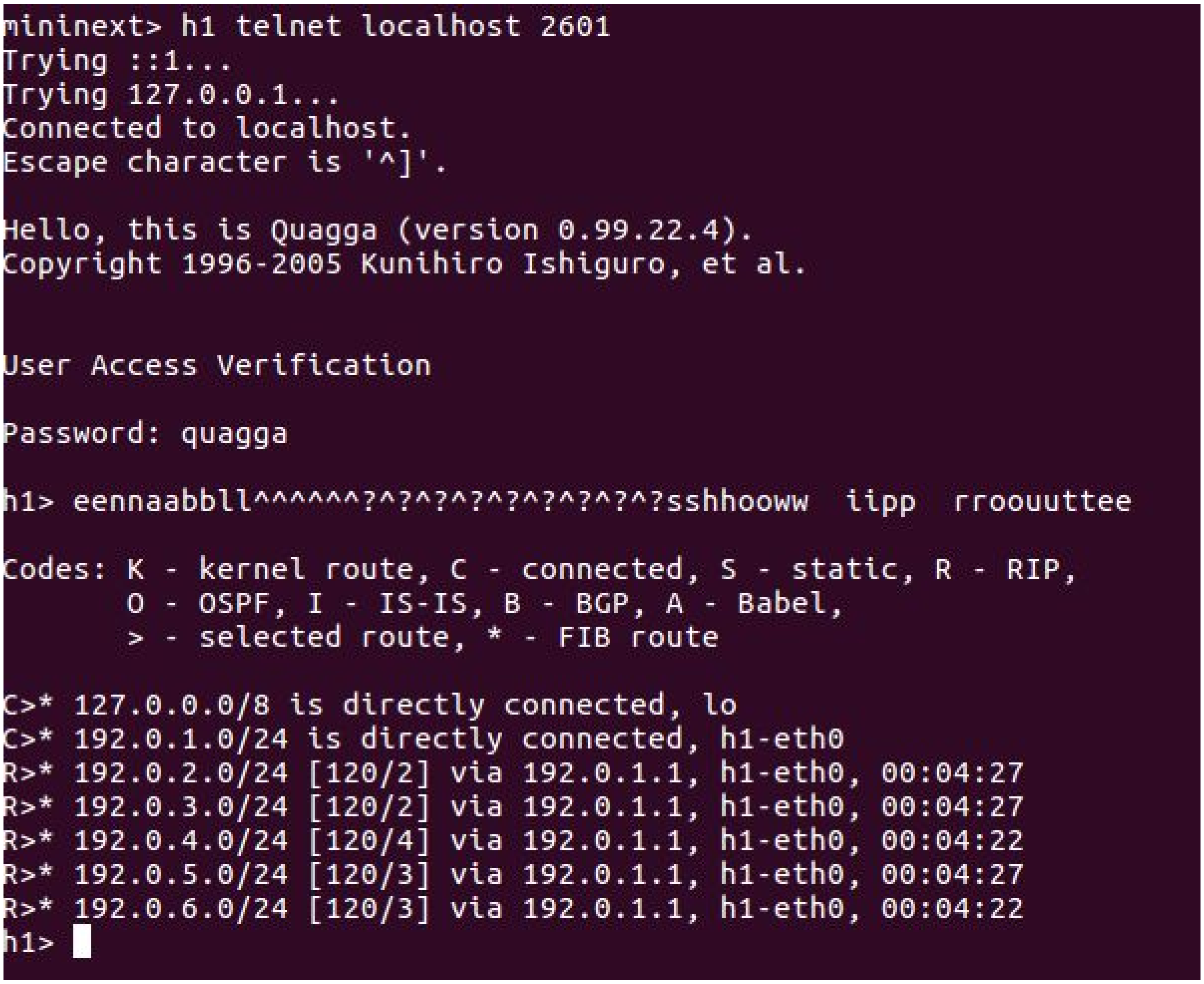
```
Hello, this is Quagga (version 0.99.22.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.


User Access Verification

Password: quagga

r3> sshhooww  iipp  rroouuttee

Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, A - Babel,
       > - selected route, * - FIB route

C>* 127.0.0.0/8 is directly connected, lo
R>* 192.0.1.0/24 [120/2] via 192.0.3.2, r3-eth0, 00:09:11
R>* 192.0.2.0/24 [120/2] via 192.0.3.2, r3-eth0, 00:09:11
C>* 192.0.3.0/24 is directly connected, r3-eth0
R>* 192.0.4.0/24 [120/2] via 192.0.6.2, r3-eth1, 00:09:10
R>* 192.0.5.0/24 [120/2] via 192.0.6.2, r3-eth1, 00:09:10
C>* 192.0.6.0/24 is directly connected, r3-eth1
r3>
```

R4:



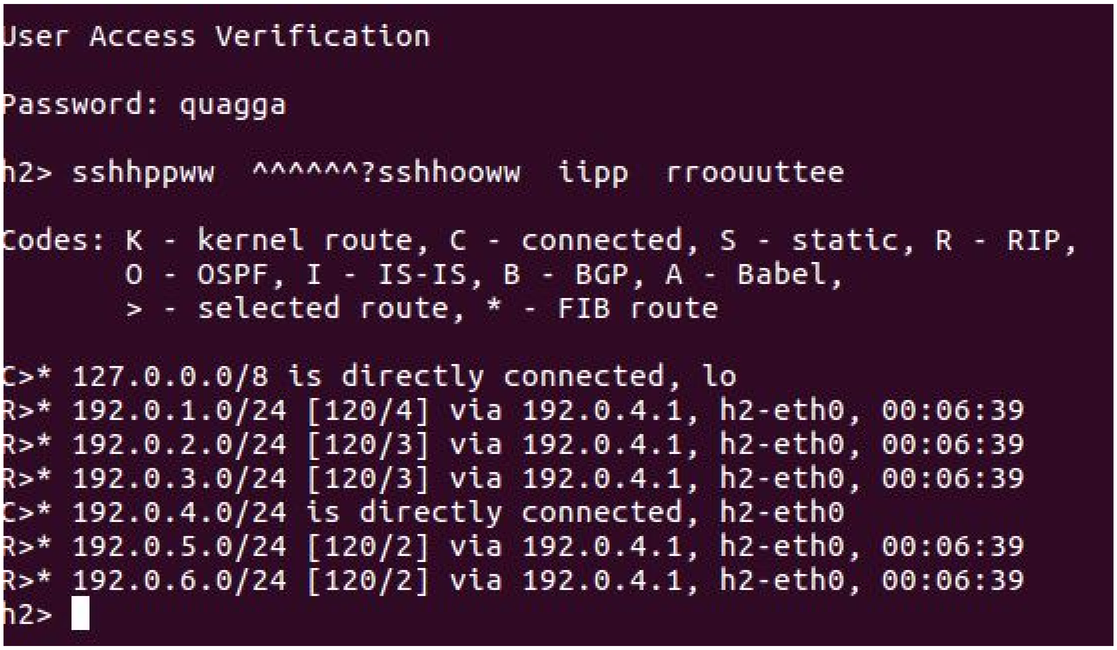
```
Hello, this is Quagga (version 0.99.22.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.


User Access Verification

Password: quagga

r4> sshhooww  iipp  rroouuttee

Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, A - Babel,
       > - selected route, * - FIB route

C>* 127.0.0.0/8 is directly connected, lo
R>* 192.0.1.0/24 [120/3] via 192.0.5.1, r4-eth2, 00:10:01
R>* 192.0.2.0/24 [120/2] via 192.0.5.1, r4-eth2, 00:10:01
R>* 192.0.3.0/24 [120/2] via 192.0.6.1, r4-eth1, 00:10:01
C>* 192.0.4.0/24 is directly connected, r4-eth0
C>* 192.0.5.0/24 is directly connected, r4-eth2
C>* 192.0.6.0/24 is directly connected, r4-eth1
r4> 
```

H1:



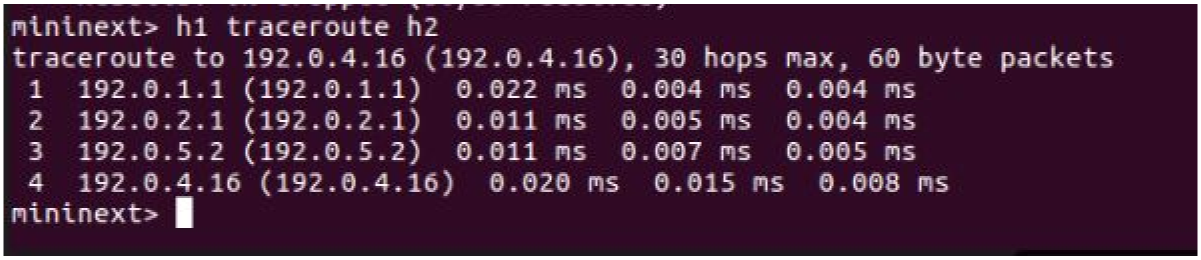
```
mininext> h1 telnet localhost 2601
Trying ::1...
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Hello, this is Quagga (version 0.99.22.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.


User Access Verification

Password: quagga

h1> eennaabbll^^^^^^?^?^?^?^?^?^?^?^?sshhooww  iipp  rroouuttee

Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, A - Babel,
       > - selected route, * - FIB route

C>* 127.0.0.0/8 is directly connected, lo
C>* 192.0.1.0/24 is directly connected, h1-eth0
R>* 192.0.2.0/24 [120/2] via 192.0.1.1, h1-eth0, 00:04:27
R>* 192.0.3.0/24 [120/2] via 192.0.1.1, h1-eth0, 00:04:27
R>* 192.0.4.0/24 [120/4] via 192.0.1.1, h1-eth0, 00:04:22
R>* 192.0.5.0/24 [120/3] via 192.0.1.1, h1-eth0, 00:04:27
R>* 192.0.6.0/24 [120/3] via 192.0.1.1, h1-eth0, 00:04:22
h1> 
```

H2:



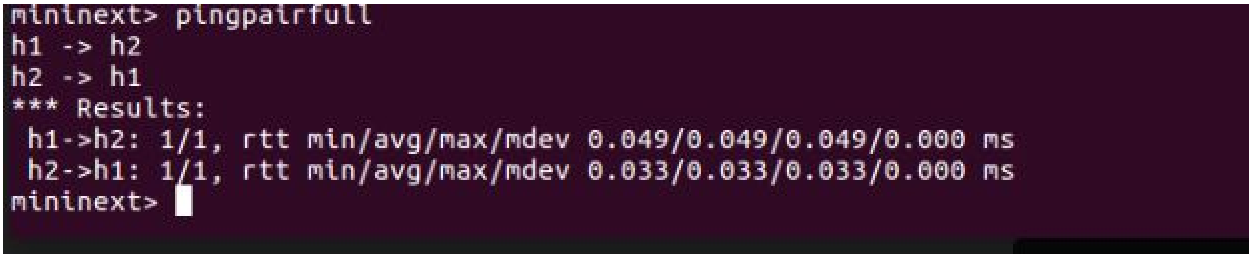
```
User Access Verification

Password: quagga

h2> sshhppww  ^^^^^^?sshhooww  iipp  rroouuttee

Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, A - Babel,
       > - selected route, * - FIB route

C>* 127.0.0.0/8 is directly connected, lo
R>* 192.0.1.0/24 [120/4] via 192.0.4.1, h2-eth0, 00:06:39
R>* 192.0.2.0/24 [120/3] via 192.0.4.1, h2-eth0, 00:06:39
R>* 192.0.3.0/24 [120/3] via 192.0.4.1, h2-eth0, 00:06:39
C>* 192.0.4.0/24 is directly connected, h2-eth0
R>* 192.0.5.0/24 [120/2] via 192.0.4.1, h2-eth0, 00:06:39
R>* 192.0.6.0/24 [120/2] via 192.0.4.1, h2-eth0, 00:06:39
h2>
```

b)H1 traceroute h2



```
mininext> h1 traceroute h2
traceroute to 192.0.4.16 (192.0.4.16), 30 hops max, 60 byte packets
 1  192.0.1.1 (192.0.1.1)  0.022 ms  0.004 ms  0.004 ms
 2  192.0.2.1 (192.0.2.1)  0.011 ms  0.005 ms  0.004 ms
 3  192.0.5.2 (192.0.5.2)  0.011 ms  0.007 ms  0.005 ms
 4  192.0.4.16 (192.0.4.16)  0.020 ms  0.015 ms  0.008 ms
mininext>
```

c)



```
mininext> pingpairfull
h1 -> h2
h2 -> h1
*** Results:
 h1->h2: 1/1, rtt min/avg/max/mdev 0.049/0.049/0.049/0.000 ms
 h2->h1: 1/1, rtt min/avg/max/mdev 0.033/0.033/0.033/0.000 ms
mininext>
```

The average ping time is 0.049 ms

d)

I have written following logic to calculate convergence time, just after setting up ipforwading. Because after that the network will start converging

I have written a loop in start.py to calculate convergence time.

after the pingall command in the loop, output is 0% loss

I am breaking the loop and taking time difference when I started the loop and ended.

My convergence time is : 11.03 seconds

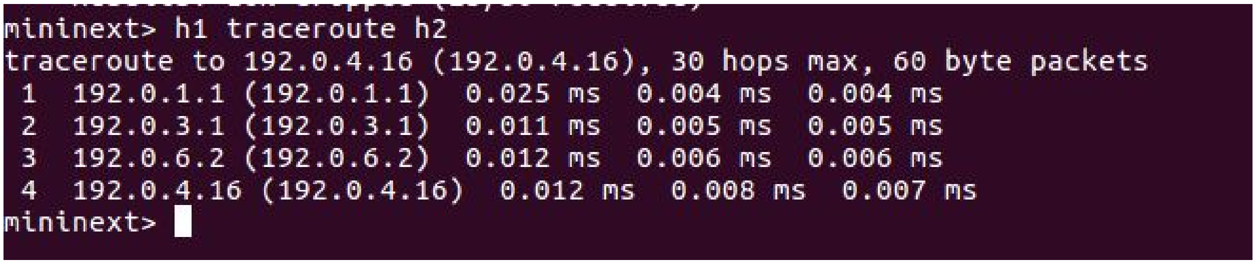
B3:

a)

To get the link r1 and r2 down I used following command in mininext command prompt:
 link r1 r2 down

b)I have checked the convergence time manually. First I thought of writing bash file but it was not working on mininext command prompt. So I checked the time the link got down and used ping command to check connectivity again and again.
My convergence time is around 32.29 seconds

c)H1 traceroute h2 after link down



```
mininext> h1 traceroute h2
traceroute to 192.0.4.16 (192.0.4.16), 30 hops max, 60 byte packets
 1  192.0.1.1 (192.0.1.1)  0.025 ms  0.004 ms  0.004 ms
 2  192.0.3.1 (192.0.3.1)  0.011 ms  0.005 ms  0.005 ms
 3  192.0.6.2 (192.0.6.2)  0.012 ms  0.006 ms  0.006 ms
 4  192.0.4.16 (192.0.4.16)  0.012 ms  0.008 ms  0.007 ms
mininext> 
```

Part C:

C1

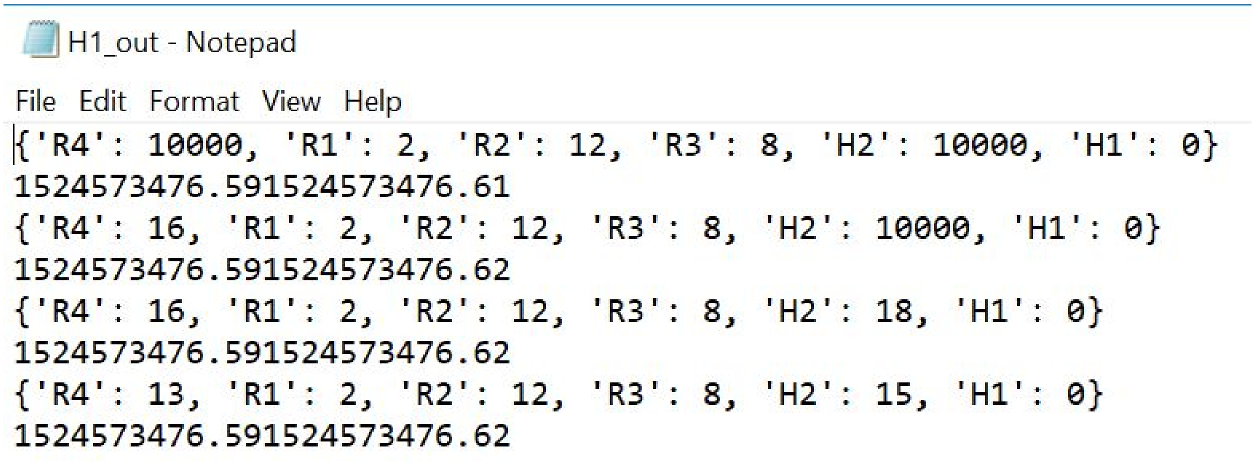
a)The server_class.py is my routing protocol file.
I have used server client configuration to share the information between the neighbors
Node_info.txt store ip and port information of each node
Nb.txt store information about the neighbors of the nodes
R1_nb.txt stores distance weights of it’s neighbors. Similarly for others also
The server keep of updating the weights to it’s neighbor if they receive info from neighbor of low edge weights. The network will converge and nodes will stop sending info if they reach to shortest path distances by applying bellman ford algorithm.

b) I am writing the start time when I made first request on each host in output file of each node.
At every upadate of routing table when I am writing update in the file I am adding time in

the file also for that update.
To calsulate convergence time, I am taking the least value of start time among all the nodes and max value of update time among all the nodes.
The difference between these two time measure is network’s actual time of convergence.

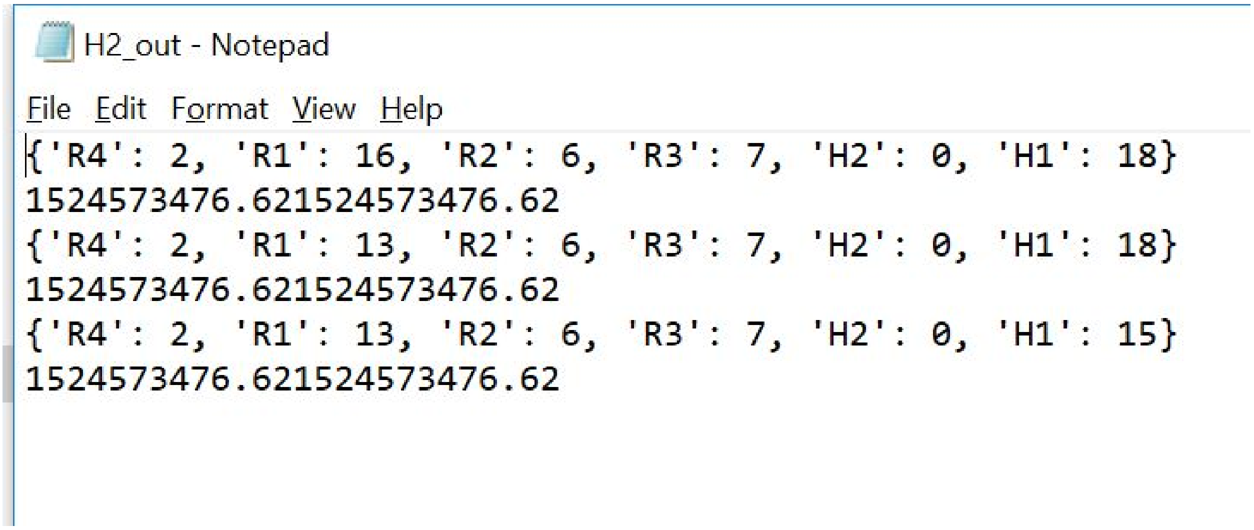
0.62-0.59=0.01ms

c)
H1 routing table:



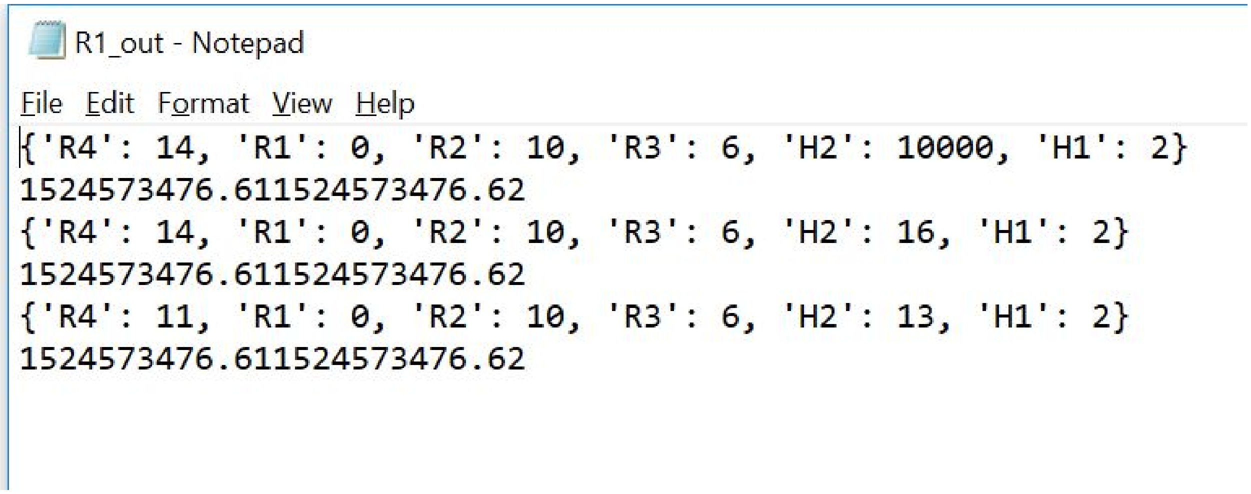
```
{'R4': 10000, 'R1': 2, 'R2': 12, 'R3': 8, 'H2': 10000, 'H1': 0}
1524573476.591524573476.61
{'R4': 16, 'R1': 2, 'R2': 12, 'R3': 8, 'H2': 10000, 'H1': 0}
1524573476.591524573476.62
{'R4': 16, 'R1': 2, 'R2': 12, 'R3': 8, 'H2': 18, 'H1': 0}
1524573476.591524573476.62
{'R4': 13, 'R1': 2, 'R2': 12, 'R3': 8, 'H2': 15, 'H1': 0}
1524573476.591524573476.62
```

H2 routing table:



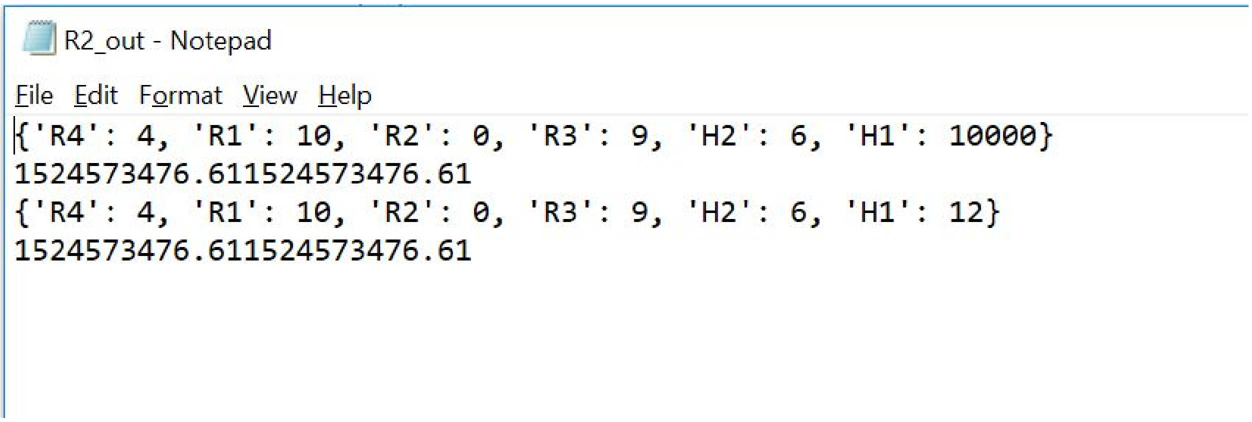
```
{'R4': 2, 'R1': 16, 'R2': 6, 'R3': 7, 'H2': 0, 'H1': 18}
1524573476.621524573476.62
{'R4': 2, 'R1': 13, 'R2': 6, 'R3': 7, 'H2': 0, 'H1': 18}
1524573476.621524573476.62
{'R4': 2, 'R1': 13, 'R2': 6, 'R3': 7, 'H2': 0, 'H1': 15}
1524573476.621524573476.62
```

R1 routing table:



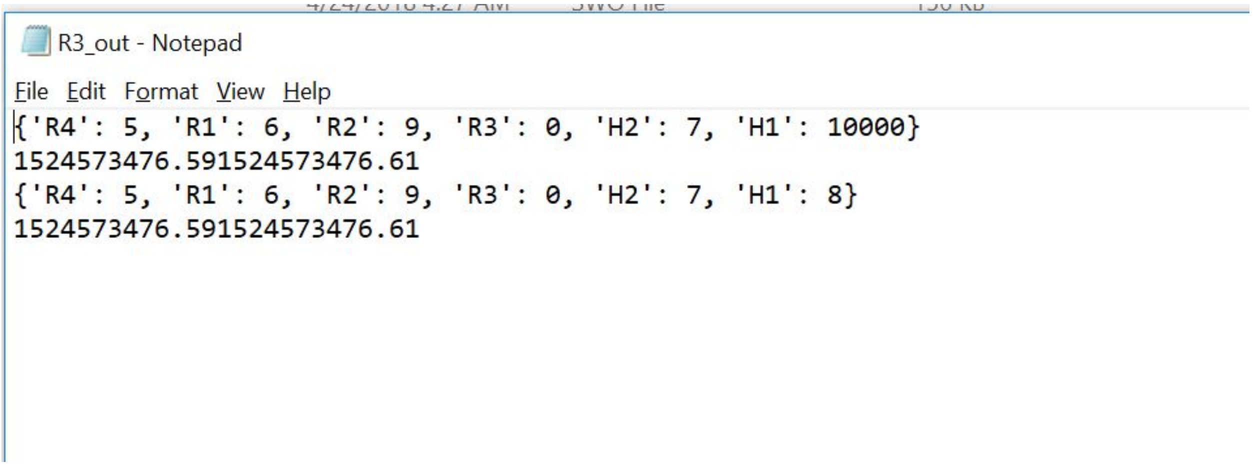
```
{'R4': 14, 'R1': 0, 'R2': 10, 'R3': 6, 'H2': 10000, 'H1': 2}
1524573476.611524573476.62
{'R4': 14, 'R1': 0, 'R2': 10, 'R3': 6, 'H2': 16, 'H1': 2}
1524573476.611524573476.62
{'R4': 11, 'R1': 0, 'R2': 10, 'R3': 6, 'H2': 13, 'H1': 2}
1524573476.611524573476.62
```

R2 routing table:



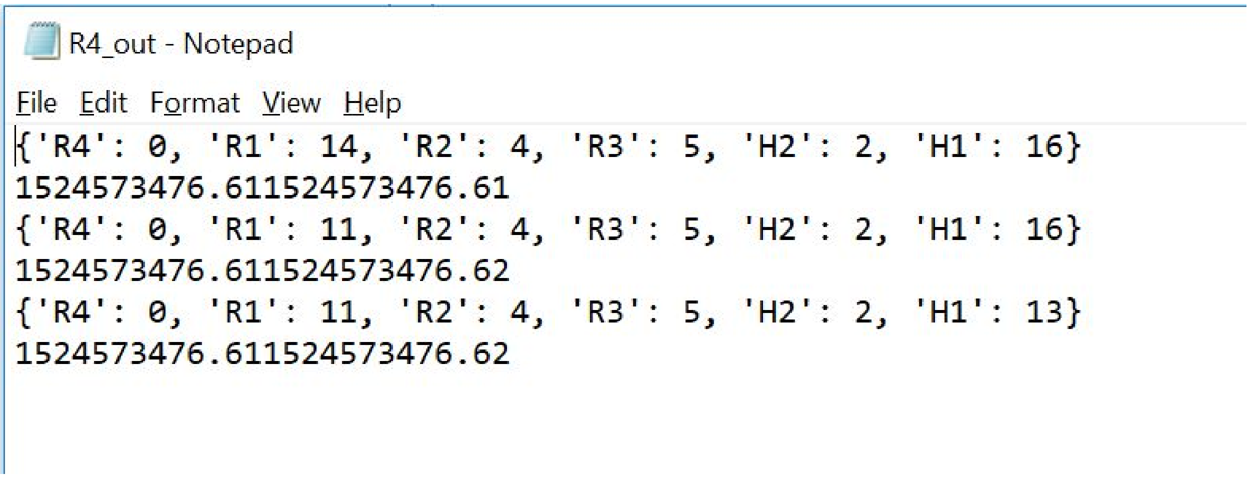
```
{'R4': 4, 'R1': 10, 'R2': 0, 'R3': 9, 'H2': 6, 'H1': 10000}
1524573476.611524573476.61
{'R4': 4, 'R1': 10, 'R2': 0, 'R3': 9, 'H2': 6, 'H1': 12}
1524573476.611524573476.61
```

R3 routing table:



```
{'R4': 5, 'R1': 6, 'R2': 9, 'R3': 0, 'H2': 7, 'H1': 10000}
1524573476.591524573476.61
{'R4': 5, 'R1': 6, 'R2': 9, 'R3': 0, 'H2': 7, 'H1': 8}
1524573476.591524573476.61
```

R4 routing table



```
{'R4': 0, 'R1': 14, 'R2': 4, 'R3': 5, 'H2': 2, 'H1': 16}
1524573476.611524573476.61
{'R4': 0, 'R1': 11, 'R2': 4, 'R3': 5, 'H2': 2, 'H1': 16}
1524573476.611524573476.62
{'R4': 0, 'R1': 11, 'R2': 4, 'R3': 5, 'H2': 2, 'H1': 13}
1524573476.611524573476.62
```

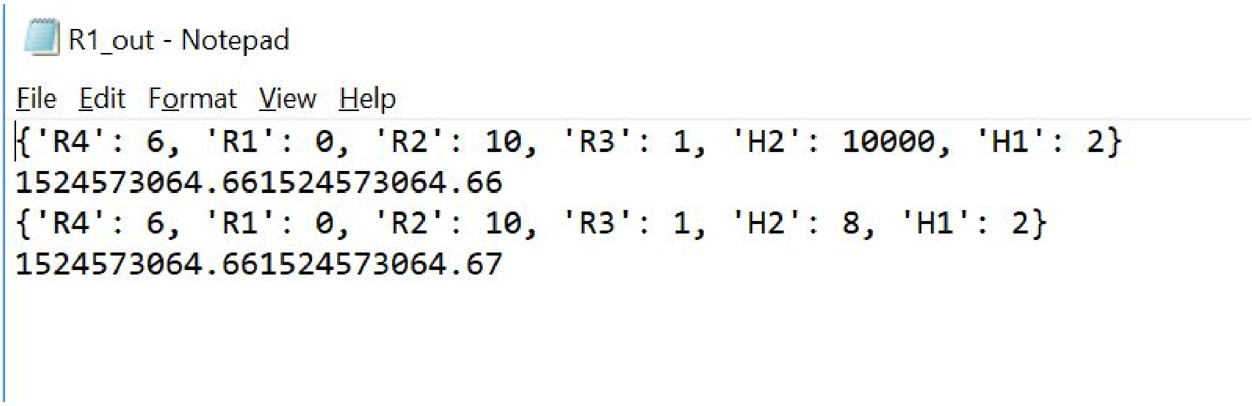
C2:

a) The convergence time calculation is same as above

0.68-0.66=0.02ms

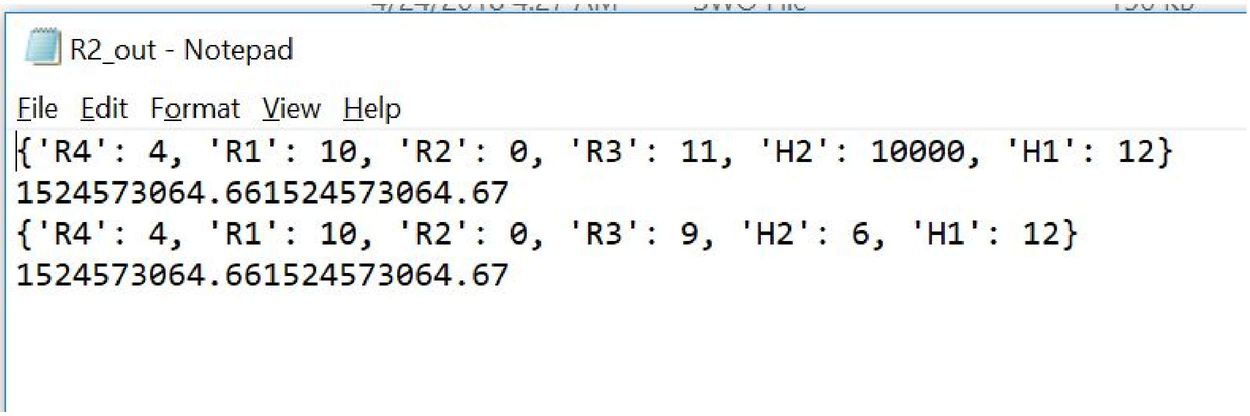
b)

R1 routing table:



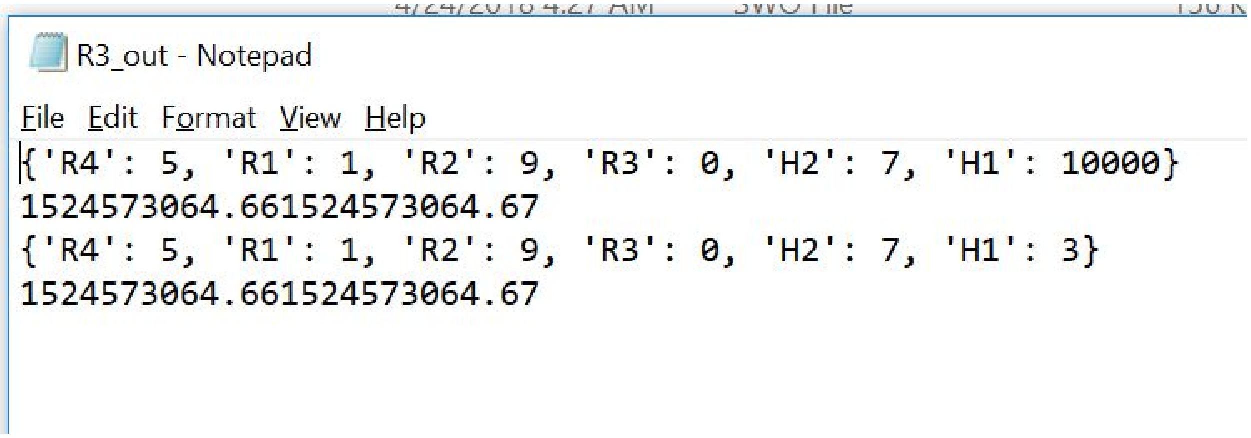
```
{'R4': 6, 'R1': 0, 'R2': 10, 'R3': 1, 'H2': 10000, 'H1': 2}
1524573064.661524573064.66
{'R4': 6, 'R1': 0, 'R2': 10, 'R3': 1, 'H2': 8, 'H1': 2}
1524573064.661524573064.67
```

R2 routing table:



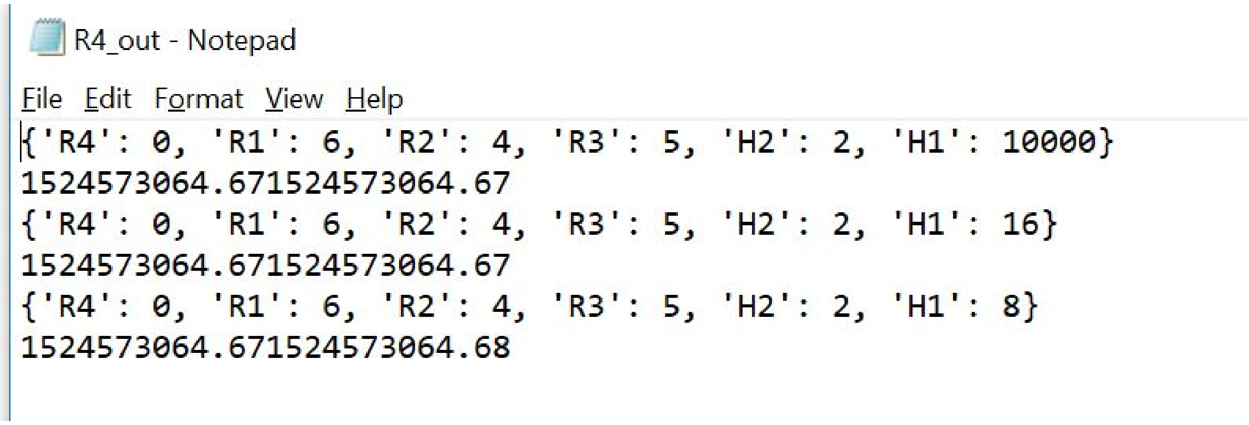
```
{'R4': 4, 'R1': 10, 'R2': 0, 'R3': 11, 'H2': 10000, 'H1': 12}
1524573064.661524573064.67
{'R4': 4, 'R1': 10, 'R2': 0, 'R3': 9, 'H2': 6, 'H1': 12}
1524573064.661524573064.67
```

R3 routing table:



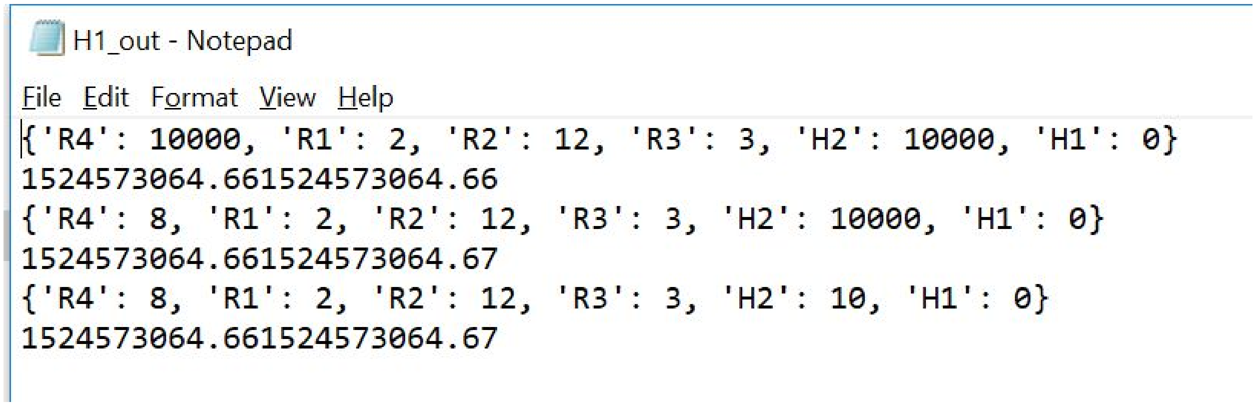
```
{'R4': 5, 'R1': 1, 'R2': 9, 'R3': 0, 'H2': 7, 'H1': 10000}
1524573064.661524573064.67
{'R4': 5, 'R1': 1, 'R2': 9, 'R3': 0, 'H2': 7, 'H1': 3}
1524573064.661524573064.67
```

R4 routing table:



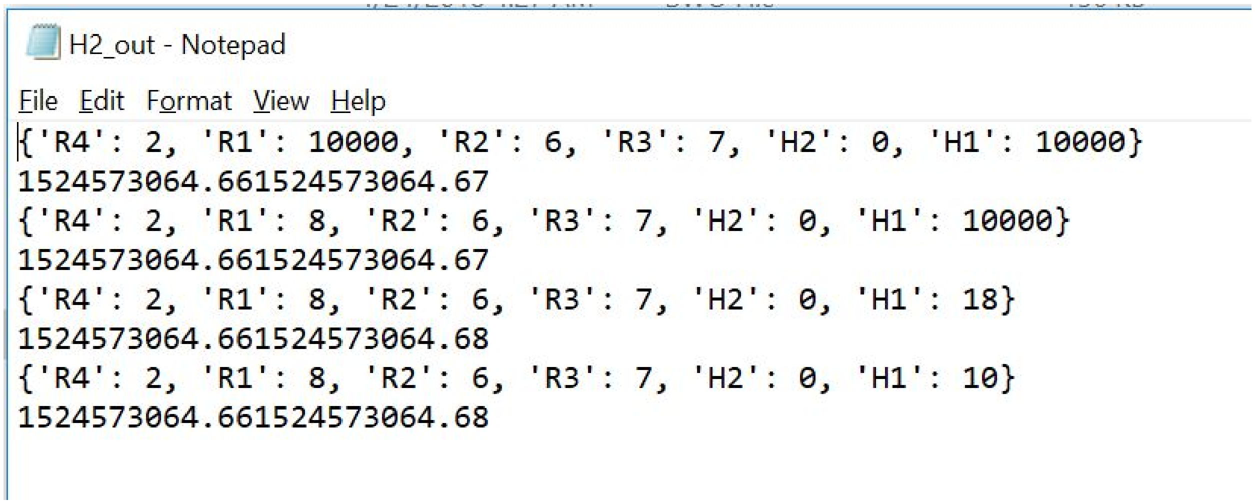
```
{'R4': 0, 'R1': 6, 'R2': 4, 'R3': 5, 'H2': 2, 'H1': 10000}
1524573064.671524573064.67
{'R4': 0, 'R1': 6, 'R2': 4, 'R3': 5, 'H2': 2, 'H1': 16}
1524573064.671524573064.67
{'R4': 0, 'R1': 6, 'R2': 4, 'R3': 5, 'H2': 2, 'H1': 8}
1524573064.671524573064.68
```

H1 routing table:



```
{'R4': 10000, 'R1': 2, 'R2': 12, 'R3': 3, 'H2': 10000, 'H1': 0}
1524573064.661524573064.66
{'R4': 8, 'R1': 2, 'R2': 12, 'R3': 3, 'H2': 10000, 'H1': 0}
1524573064.661524573064.67
{'R4': 8, 'R1': 2, 'R2': 12, 'R3': 3, 'H2': 10, 'H1': 0}
1524573064.661524573064.67
```

H2 routing table:



```
{'R4': 2, 'R1': 10000, 'R2': 6, 'R3': 7, 'H2': 0, 'H1': 10000}
1524573064.661524573064.67
{'R4': 2, 'R1': 8, 'R2': 6, 'R3': 7, 'H2': 0, 'H1': 10000}
1524573064.661524573064.67
{'R4': 2, 'R1': 8, 'R2': 6, 'R3': 7, 'H2': 0, 'H1': 18}
1524573064.661524573064.68
{'R4': 2, 'R1': 8, 'R2': 6, 'R3': 7, 'H2': 0, 'H1': 10}
1524573064.661524573064.68
```

C3

The bellman ford algorithm that we have implemented cannot handle negative weight cycle. It can be used to check if there is negative weight edge in the network. If the negative edge is present in the network then just stop updating the distance routing table.

We use bellman ford algorithm because we cannot use dijkstra as it needs the whole picture of whole graph i.e network topology.

That is it is really hard to handle negative weight edge in network graph but we can surely detect it.

Refrences: https://en.wikipedia.org/wiki/Bellman%E2%80%93Ford_algorithm
https://networkengineering.stackexchange.com/questions/34829/how-does-the-bellman-ford-algorithm-apply-to-real-networks