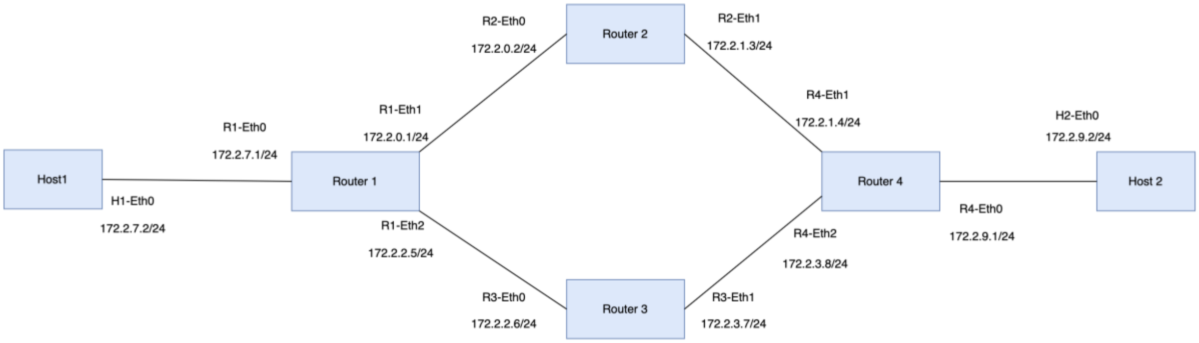
Part A

A1. Create my own topology as below.

(a) MyTopo.py file created

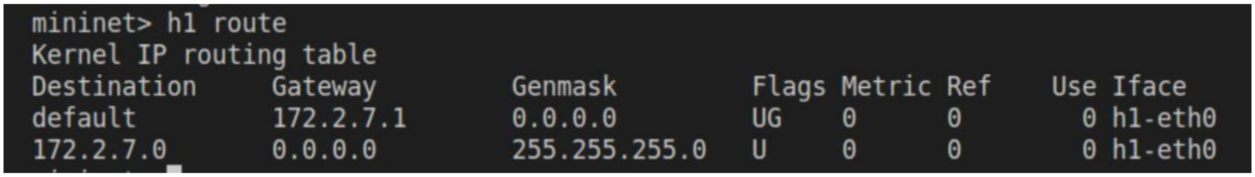
(b) The network topology figure



A2. Create static routes

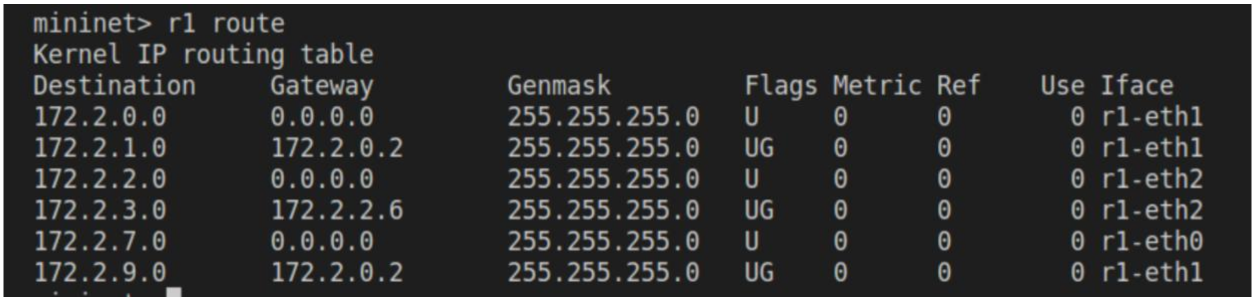
(a) Routing tables at all nodes

H1 routing table



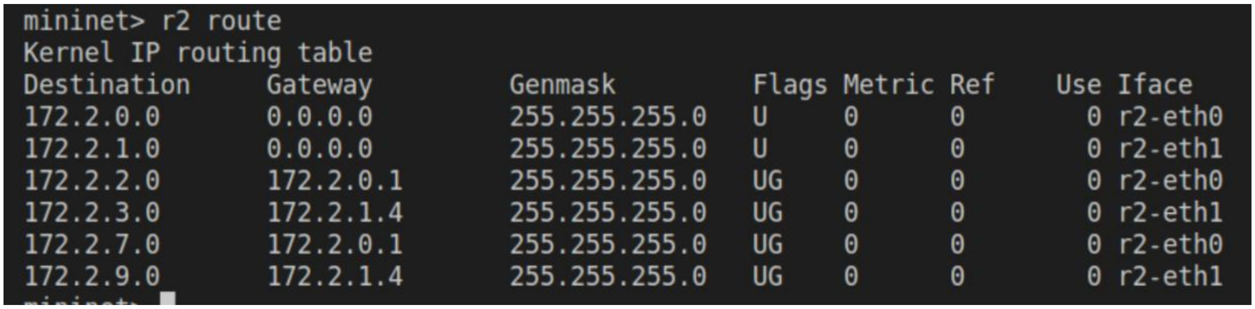
```
mininet> h1 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         172.2.7.1       0.0.0.0         UG    0      0        0 h1-eth0
172.2.7.0       0.0.0.0         255.255.255.0   U     0      0        0 h1-eth0
```

R1 routing table



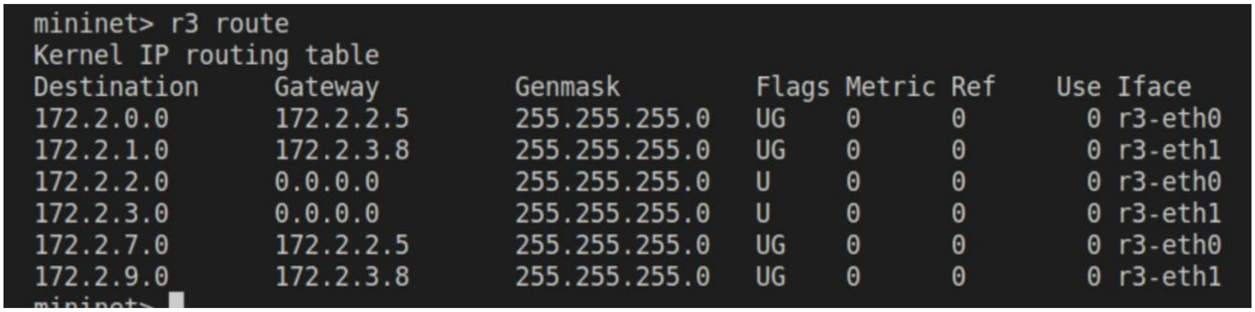
```
mininet> r1 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
172.2.0.0       0.0.0.0         255.255.255.0   U     0      0        0 r1-eth1
172.2.1.0       172.2.0.2       255.255.255.0   UG    0      0        0 r1-eth1
172.2.2.0       0.0.0.0         255.255.255.0   U     0      0        0 r1-eth2
172.2.3.0       172.2.2.6       255.255.255.0   UG    0      0        0 r1-eth2
172.2.7.0       0.0.0.0         255.255.255.0   U     0      0        0 r1-eth0
172.2.9.0       172.2.0.2       255.255.255.0   UG    0      0        0 r1-eth1
```

R2 routing table



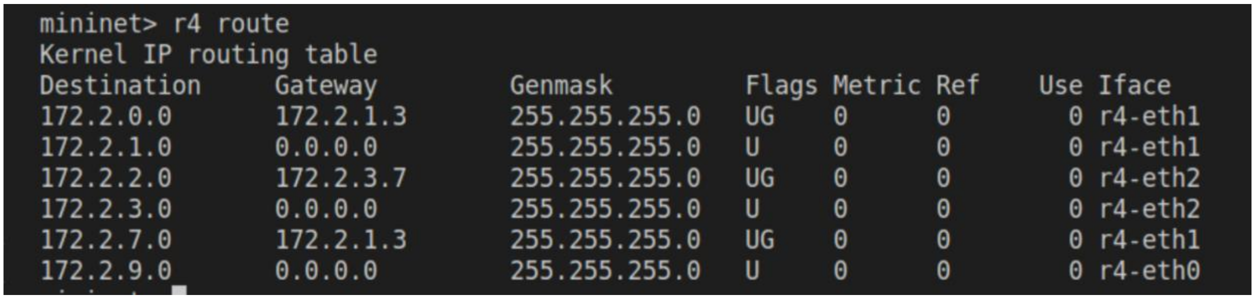
```
mininet> r2 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
172.2.0.0       0.0.0.0         255.255.255.0   U     0      0        0 r2-eth0
172.2.1.0       0.0.0.0         255.255.255.0   U     0      0        0 r2-eth1
172.2.2.0       172.2.0.1       255.255.255.0   UG    0      0        0 r2-eth0
172.2.3.0       172.2.1.4       255.255.255.0   UG    0      0        0 r2-eth1
172.2.7.0       172.2.0.1       255.255.255.0   UG    0      0        0 r2-eth0
172.2.9.0       172.2.1.4       255.255.255.0   UG    0      0        0 r2-eth1
```

R3 routing table



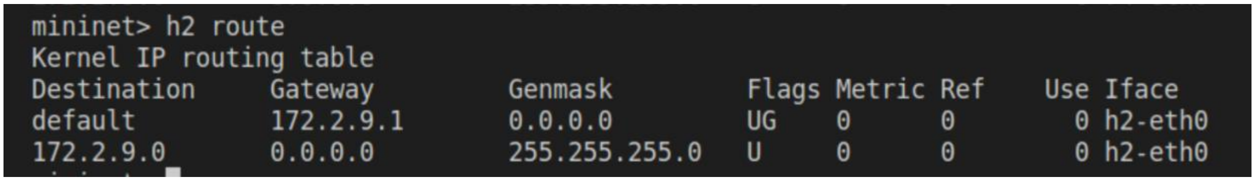
```
mininet> r3 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
172.2.0.0       172.2.2.5       255.255.255.0   UG    0      0        0 r3-eth0
172.2.1.0       172.2.3.8       255.255.255.0   UG    0      0        0 r3-eth1
172.2.2.0       0.0.0.0         255.255.255.0   U     0      0        0 r3-eth0
172.2.3.0       0.0.0.0         255.255.255.0   U     0      0        0 r3-eth1
172.2.7.0       172.2.2.5       255.255.255.0   UG    0      0        0 r3-eth0
172.2.9.0       172.2.3.8       255.255.255.0   UG    0      0        0 r3-eth1
```

R4 routing table



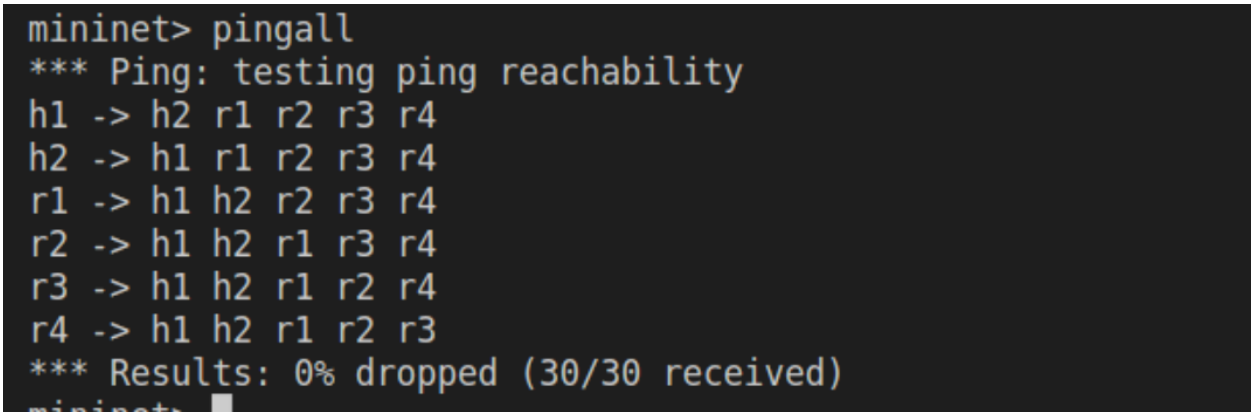
```
mininet> r4 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
172.2.0.0       172.2.1.3       255.255.255.0   UG    0      0        0 r4-eth1
172.2.1.0       0.0.0.0         255.255.255.0   U     0      0        0 r4-eth1
172.2.2.0       172.2.3.7       255.255.255.0   UG    0      0        0 r4-eth2
172.2.3.0       0.0.0.0         255.255.255.0   U     0      0        0 r4-eth2
172.2.7.0       172.2.1.3       255.255.255.0   UG    0      0        0 r4-eth1
172.2.9.0       0.0.0.0         255.255.255.0   U     0      0        0 r4-eth0
```

H2 routing table



```
mininet> h2 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         172.2.9.1       0.0.0.0         UG    0      0        0 h2-eth0
172.2.9.0       0.0.0.0         255.255.255.0   U     0      0        0 h2-eth0
```

Pingall

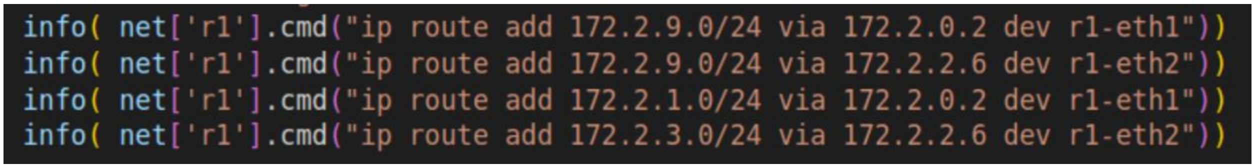


```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 r1 r2 r3 r4
h2 -> h1 r1 r2 r3 r4
r1 -> h1 h2 r2 r3 r4
r2 -> h1 h2 r1 r3 r4
r3 -> h1 h2 r1 r2 r4
r4 -> h1 h2 r1 r2 r3
*** Results: 0% dropped (30/30 received)
mininet>
```

To add routes statically I used the ‘ip route add’ command for those routes not connected directly by the *addLink* function. Doing this I was able to get where the next hop would be and would arrive at which interface i.e direct the traffic from 172.2.9.0/24 subnet to the next hop i.e 172.2.0.2 which would direct it to the eth1 interface of R1. Similarly, this is done for all the static routes as shown below.
The hosts h1 and h2 have a default route through r1 and r4 respectively.

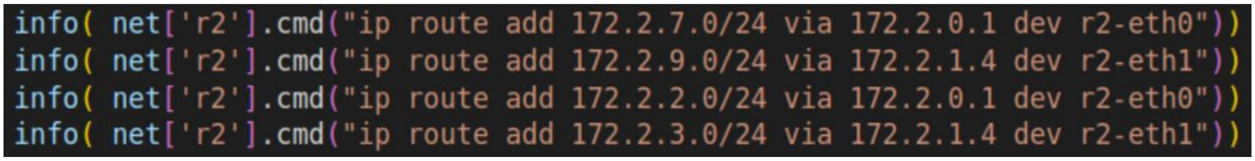
The routing tables are configured in the code as follows:

For router R1:



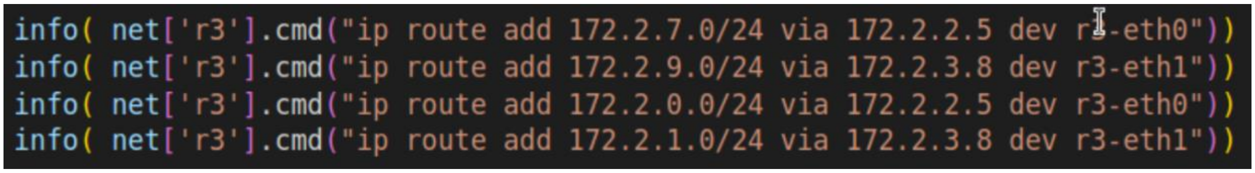
```
info( net['r1'].cmd("ip route add 172.2.9.0/24 via 172.2.0.2 dev r1-eth1"))
info( net['r1'].cmd("ip route add 172.2.9.0/24 via 172.2.2.6 dev r1-eth2"))
info( net['r1'].cmd("ip route add 172.2.1.0/24 via 172.2.0.2 dev r1-eth1"))
info( net['r1'].cmd("ip route add 172.2.3.0/24 via 172.2.2.6 dev r1-eth2"))
```

For router R2:



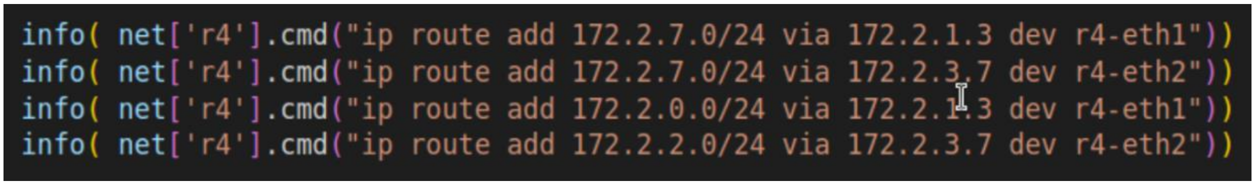
```
info( net['r2'].cmd("ip route add 172.2.7.0/24 via 172.2.0.1 dev r2-eth0"))
info( net['r2'].cmd("ip route add 172.2.9.0/24 via 172.2.1.4 dev r2-eth1"))
info( net['r2'].cmd("ip route add 172.2.2.0/24 via 172.2.0.1 dev r2-eth0"))
info( net['r2'].cmd("ip route add 172.2.3.0/24 via 172.2.1.4 dev r2-eth1"))
```

For router R3:



```
info( net['r3'].cmd("ip route add 172.2.7.0/24 via 172.2.2.5 dev r3-eth0"))
info( net['r3'].cmd("ip route add 172.2.9.0/24 via 172.2.3.8 dev r3-eth1"))
info( net['r3'].cmd("ip route add 172.2.0.0/24 via 172.2.2.5 dev r3-eth0"))
info( net['r3'].cmd("ip route add 172.2.1.0/24 via 172.2.3.8 dev r3-eth1"))
```

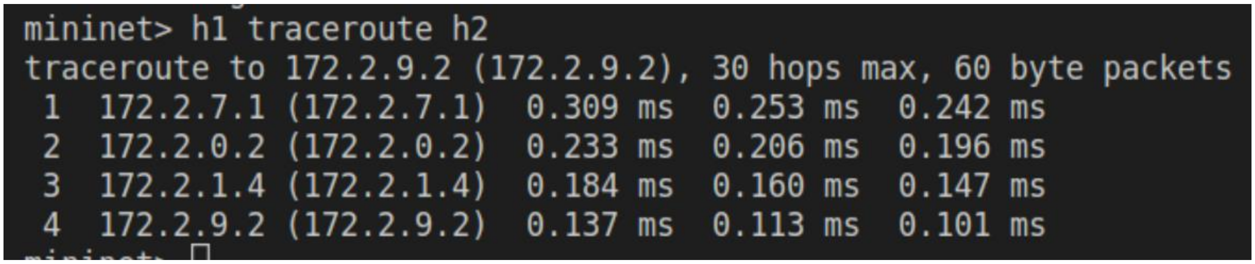
For router R4:



```
info( net['r4'].cmd("ip route add 172.2.7.0/24 via 172.2.1.3 dev r4-eth1"))
info( net['r4'].cmd("ip route add 172.2.7.0/24 via 172.2.3.7 dev r4-eth2"))
info( net['r4'].cmd("ip route add 172.2.0.0/24 via 172.2.1.3 dev r4-eth1"))
info( net['r4'].cmd("ip route add 172.2.2.0/24 via 172.2.3.7 dev r4-eth2"))
```

(b) provide the trace route output that gives the path between nodes H1 & H2.

H1 traceroute h2



```
mininet> h1 traceroute h2
traceroute to 172.2.9.2 (172.2.9.2), 30 hops max, 60 byte packets
 1  172.2.7.1 (172.2.7.1)  0.309 ms  0.253 ms  0.242 ms
 2  172.2.0.2 (172.2.0.2)  0.233 ms  0.206 ms  0.196 ms
 3  172.2.1.4 (172.2.1.4)  0.184 ms  0.160 ms  0.147 ms
 4  172.2.9.2 (172.2.9.2)  0.137 ms  0.113 ms  0.101 ms
mininet> 
```