PART A

A1.

(a) MyTopo.py file has been submitted along with the folder (inside Part A folder). Below is the result after running the script.

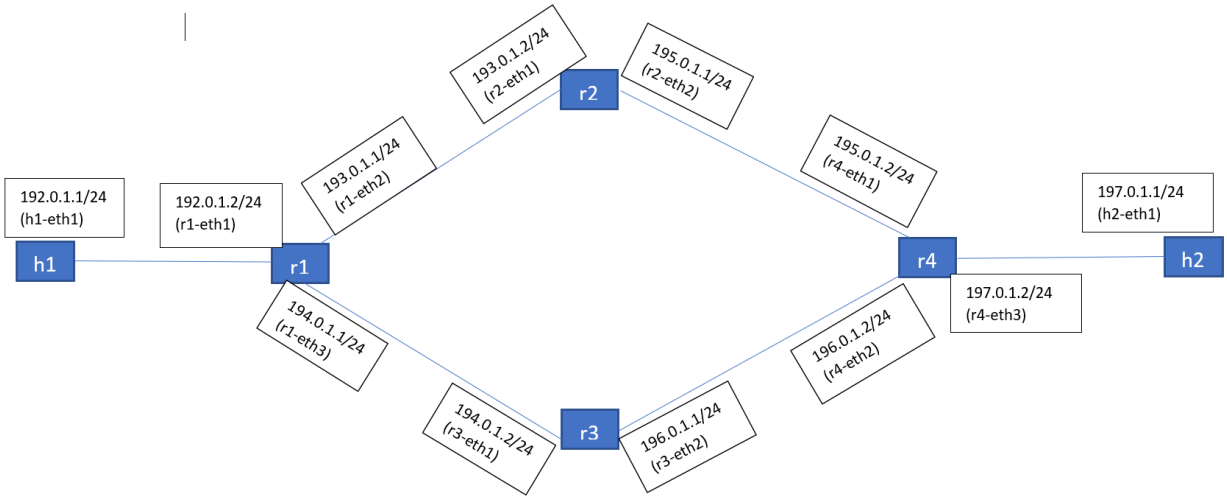
```
mininet@mininet-vm:/media/sf_CSE_534_FCN/Goel-Dinker-HW3/Part A$ sudo python MyTopo.py
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 r1 r2 r3 r4
*** Adding switches:

*** Adding links:
(h1, r1) (h2, r4) (r1, r2) (r1, r3) (r2, r4) (r3, r4)
*** Configuring hosts
h1 h2 r1 r2 r3 r4
*** Starting controller
c0
*** Starting 0 switches

*** Waiting for switches to connect

*** Starting CLI:
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>
```

(b) Below is the network topology figure including the IP addresses:
Nodes: h1, h2, r1, r2, r3, r4 (in blue boxes)
IP and subnet information (in white boxes)



A2.

(a) Routing tables at each node (screenshots):

For H1:

```
mininet> h1 route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.0.1.2       0.0.0.0         UG    0      0        0 h1-eth1
192.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 h1-eth1
```

For H2:

```
mininet> h2 route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         197.0.1.2       0.0.0.0         UG    0      0        0 h2-eth1
197.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 h2-eth1
```

For R1:

```
mininet> r1 route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         193.0.1.2       0.0.0.0         UG    0      0        0 r1-eth2
192.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 r1-eth1
193.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 r1-eth2
194.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 r1-eth3
196.0.1.0       194.0.1.2       255.255.255.0   UG    0      0        0 r1-eth3
```

For R2:

```
mininet> r2 route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         195.0.1.2       0.0.0.0         UG    0      0        0 r2-eth2
192.0.1.0       193.0.1.1       255.255.255.0   UG    0      0        0 r2-eth1
193.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 r2-eth1
194.0.1.0       193.0.1.1       255.255.255.0   UG    0      0        0 r2-eth1
195.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 r2-eth2
```

For R3:

```
mininet> r3 route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         196.0.1.2       0.0.0.0         UG    0      0        0 r3-eth2
192.0.1.0       194.0.1.1       255.255.255.0   UG    0      0        0 r3-eth1
193.0.1.0       194.0.1.1       255.255.255.0   UG    0      0        0 r3-eth1
194.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 r3-eth1
196.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 r3-eth2
mininet>
```

For R4:

```
mininet> r4 route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         195.0.1.1       0.0.0.0         UG    0      0        0 r4-eth1
194.0.1.0       196.0.1.1       255.255.255.0   UG    0      0        0 r4-eth2
195.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 r4-eth1
196.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 r4-eth2
197.0.1.0       0.0.0.0         255.255.255.0   U     0      0        0 r4-eth3
mininet>
```

Explanation to configure them correctly:

- First, we need to enable IP forwarding to 1 for all nodes. By default, this value is set to 0. For this, I have used below command in LinuxRouter class:
  self.cmd('sysctl net.ipv4.ip_forward=1')

- Next, I have given each interface an IP address while creating links in the NetworkTopo class (such as h1-eth1, h2-eth1, r1-eth1).
- Then, I have set a static routing path for each node. Below is the command that I used:
  net.get("r1").cmd("ip route add 196.0.1.0/24 via 194.0.1.2 dev r1-eth3")

  This command adds the destination subnet and the next gateway along with the interface through which the packet will proceed further. In the above example, the destination is 196.0.1.0/24, gateway is 194.0.1.2 and the interface is r1-eth3. This will be followed for each node. Below screenshot shows all the other routes.

```
def run():
    "Test linux router"
    topo = NetworkTopo()
    net = Mininet( topo=topo, waitConnected=True )
    net.start()

    net.get("h1").cmd("ip route add default via 192.0.1.2 dev h1-eth1")
    net.get("h2").cmd("ip route add default via 197.0.1.2 dev h2-eth1")

    net.get("r1").cmd("ip route add 196.0.1.0/24 via 194.0.1.2 dev r1-eth3")
    net.get("r1").cmd("ip route add default via 193.0.1.2 dev r1-eth2")

    net.get("r2").cmd("ip route add 192.0.1.0/24 via 193.0.1.1 dev r2-eth1")
    net.get("r2").cmd("ip route add 194.0.1.0/24 via 193.0.1.1 dev r2-eth1")
    net.get("r2").cmd("ip route add default via 195.0.1.2 dev r2-eth2")

    net.get("r3").cmd("ip route add 192.0.1.0/24 via 194.0.1.1 dev r3-eth1")
    net.get("r3").cmd("ip route add 193.0.1.0/24 via 194.0.1.1 dev r3-eth1")
    net.get("r3").cmd("ip route add default via 196.0.1.2 dev r3-eth2")

    net.get("r4").cmd("ip route add 194.0.1.0/24 via 196.0.1.1 dev r4-eth2")
    net.get("r4").cmd("ip route add default via 195.0.1.1 dev r4-eth1")
```

(b) Traceroute output between h1 and h2.

```
mininet> h1 traceroute h2
traceroute to 197.0.1.1 (197.0.1.1), 30 hops max, 60 byte packets
 1  192.0.1.2 (192.0.1.2)  0.034 ms  0.008 ms  0.041 ms
 2  193.0.1.2 (193.0.1.2)  0.017 ms  0.009 ms  0.009 ms
 3  195.0.1.2 (195.0.1.2)  0.018 ms  0.011 ms  0.011 ms
 4  197.0.1.1 (197.0.1.1)  0.019 ms  0.012 ms  0.015 ms
mininet>
```

Here, the path followed is: h1-> r1-> r2 -> r4 -> h2.