Part C

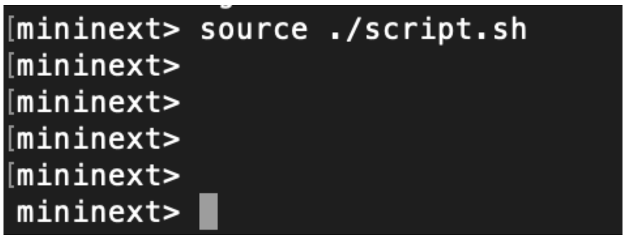
C1. Run your protocol and get the time taken for the protocol to find the shortest path between H1 and H2.

Instruction to run :

All codes in PART C folder should be placed in a directory for the “sudo python start.py” to run properly. After running start.py, give the below commands to set the configuration (which was already explained in Part A)

```
                h1 echo 1 > /proc/sys/net/ipv4/ip_forward
                h2 echo 1 > /proc/sys/net/ipv4/ip_forward
                r1 echo 1 > /proc/sys/net/ipv4/ip_forward
                r2 echo 1 > /proc/sys/net/ipv4/ip_forward
                r3 echo 1 > /proc/sys/net/ipv4/ip_forward
                r4 echo 1 > /proc/sys/net/ipv4/ip_forward

                r1 ip addr add 172.0.2.1/24 dev r1-eth1
                r1 ip addr add 172.0.6.2/24 dev r1-eth2
                r2 ip addr add 172.0.3.1/24 dev r2-eth1
                r4 ip addr add 172.0.3.2/24 dev r4-eth1
                r4 ip addr add 172.0.5.1/24 dev r4-eth2
                r3 ip addr add 172.0.5.2/24 dev r3-eth1

                h1 ip route add default via 172.0.1.2 dev h1-eth0
                r1 ip route add 172.0.3.0 via 172.0.2.2 dev r1-eth1
                r1 ip route add 172.0.4.0/24 via 172.0.2.2 dev r1-eth1
                r2 ip route add 172.0.4.0/24 via 172.0.3.2 dev r2-eth1

                r1 iptables -t nat -A POSTROUTING -o R1-eth1 -j
MASQUERADE
                r1 iptables -A FORWARD -i R1-eth1 -o R1-eth0 -m state
--state RELATED,ESTABLISHED -j ACCEPT
                r1 iptables -A FORWARD -i R1-eth0 -o R1-eth1 -j ACCEPT

                r2 iptables -t nat -A POSTROUTING -o R2-eth1 -j
MASQUERADE
                r2 iptables -A FORWARD -i R2-eth1 -o R2-eth0 -m state
--state RELATED,ESTABLISHED -j ACCEPT
                r2 iptables -A FORWARD -i R2-eth0 -o R2-eth1 -j ACCEPT

                r4 iptables -t nat -A POSTROUTING -o R4-eth0 -j
MASQUERADE
                r4 iptables -A FORWARD -i R4-eth0 -o R4-eth1 -m state
--state RELATED,ESTABLISHED -j ACCEPT
                r4 iptables -A FORWARD -i R4-eth1 -o R4-eth0 -j ACCEPT
```

Now run “source ./script.sh” which will fire server and client for h1, h2, r1, r2, r3, r4 which will use the bellman ford algorithm to change the routing tables of h1, r1, r2, r3, r4, h2.



```
[mininext> source ./script.sh
[mininext>
[mininext>
[mininext>
[mininext>
 mininext>
```

rtable<nodename>.csv ———-> routing tables

Once program is run, these files will be modified.

Algorithm used - Bellman ford

Initially cost to every node other than the neighbors for all routers will be marked as max.

In my code, each router has a server and client to it. Router's client will be sending it's data, while the router's server will receive data from other clients.

I tried to print the server, client logs for all routers. Each node tries updating its neighbours whenever there is a change in their routes.

Every router's client keeps continuously checking if there is a change on any node & if it sees a change, the information is forwarded to its neighbours by connecting to adjacent router's server.

If distance information of any node is less than the existing value, it will update it's distance information.

Thus, updates propagate from one router to the other router & will finally reach a stable state and log won't be printed anymore, indicating that the rip-lite protocol has converged.

C3. If one of the links has a negative weight, explain how you handle this situation.

The bellman ford algorithm used in my implementation above can detect negative cycles and thus can handle negative weighted edges. Once after running the algorithm, we get a result & if the outputs stay same after running the dynamic programming loop for the second time, it implies that there is a negative cycle, else it is fine. So, there won't be a problem even if links have negative weights.

In practical, there won't be negative weights between two routers. Still, its better if we make the negative edge weights to zero and store it in the routing table and cross verify our data with that of the other neighbours to see if they have same data.