Part-C

C1).

A).

Routing protocol code is divided in between **server.py** and **client.py.** The code resides in the directory of Part-C.

Working of Code:

server.py: This file contains the code which acts as a server and opens up a listening port on **port number 60**. Once registering the port and connecting the socket, it listens for any connection and spawns a new thread when a client connects to it.

The data passed by the client to socket is the name of routing table stored as ".pkl" format. Once the server receives the routing table, it runs the Bellman Ford algorithm to find any latest path. After successful processing of the Bellman ford algorithm, it again dumps its own routing table in pkl format. So, individual pkl files of each nodes have always updated routing information.

To execute the code, run:

python server.py <configuration file for node>

where, <configuration file for node> stores the neighbor details of each node.

client.py: This file contains the code which acts as a client whose task is to connect to its neighbors and broadcast its own routing table to every neighbor after an interval of every 30 seconds.

To execute the code, run:

python server.py <configuration file for node>

where, <configuration file for node> stores the neighbor details of each node.

Configuration Files: All configuration files are stored in directory : **neighbours**

All routing tables are stored in directory named: **routing**

Working: I have first run the mininet with my custom topology and ripd running on network layer to establish TCP connection on each node. Then, I have installed xming on my windows desktop and ran xterm for every node. On every xterm screen, I ran one server instance and one client instance to help populating routing table information.

Attached are the screenshots of server interfaces running on every node:

Server Interface:

Attached are the screenshots of client interfaces running on every node:

Client Interface:

Node: h2

```
root@mininet-vm:~/hw3# python client.py h2
Wating for 30 sec. to broadcast table..
Broadcasting routing table to Client:  r4
received data: routing/h2
```

Node: r4

```
root@mininet-vm:~/hw3# python client.py r4
Wating for 30 sec. to broadcast table..
Broadcasting routing table to Client:  r2
received data: routing/r4
Broadcasting routing table to Client:  r3
received data: routing/r4
Wating for 30 sec. to broadcast table..
```

Node: r3

```
root@mininet-vm:~/hw3# python client.py r3
Wating for 30 sec. to broadcast table..
Broadcasting routing table to Client:  r1
received data: routing/r3
Wating for 30 sec. to broadcast table..
```

Node: r2

```
root@mininet-vm:~/hw3# python client.py r2
Wating for 30 sec. to broadcast table..
Broadcasting routing table to Client:  r1
received data: routing/r2
Wating for 30 sec. to broadcast table..
```

Node: r1

```
root@mininet-vm:~/hw3# python client.py r1
Wating for 30 sec. to broadcast table..
Broadcasting routing table to Client:  h1
received data: routing/r1
Wating for 30 sec. to broadcast table..
```

Node: h1

```
root@mininet-vm:~/hw3# python client.py h1
Wating for 30 sec. to broadcast table..
```

B). Total time taken for application layer protocol to find shortest path is **47 sec 23 ms.**

Every node in the topology broadcasts its routing table to its neighbor after every 30 seconds.

I have taken two timestamps to calculate this time:

1. Initial Timestamp when sever.py was instantiated on all active nodes.
2. Time taken by nodes to share the routing tables with its neighbors and finally propagating it to host h1.

   I have then taken a difference of both these timestamps to calculate the final time.

C). **Application Layer routing table at each node:**

Node: H1

Routing Table:

Updated Table:

Destination	Next Hop	Cost
h2	r1	10
r4	r1	8
r1	r1	2
r2	r1	9
r3	r1	8

Node: R1

Routing Table:

Updated Table:

Destination	Next Hop	Cost
h2	r2	8
r4	r2	6
h1	h1	2
r2	r3	7
r3	r3	6

Node: R2

Routing Table:

Updated Table:

Destination	Next Hop	Cost
h2	r4	-2
r4	r4	-4
r1	r1	10
r3	r4	1

Node: R3

Routing Table:

Updated Table:

Destination	Next Hop	Cost
h2	r4	7
r4	r4	5
r1	r1	6
r2	r4	1

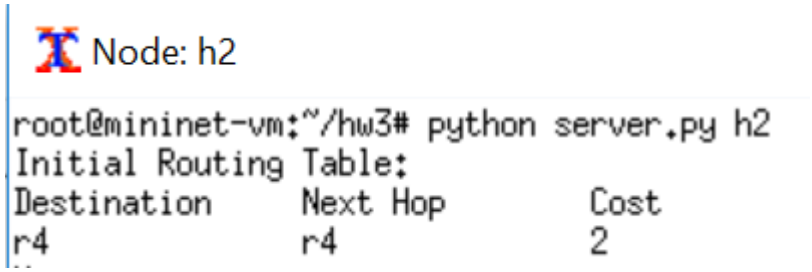
Node: R4

Routing Table:

```
Updated Table:
Destination     Next Hop        Cost
h2              h2              2
r2              r2              -4
r3              r3              5
```

Node: H2

Routing Table:



C2).

A) Time taken for protocol to converge when weight of the link R1-R3 changes from 6 to 1 is **31.56 seconds.**
I calculated the time by using stopwatch timer and the delay in updating the route by more than 30 seconds is justified by the fact that R1 is updating its routing time only after 30 seconds. Once the weight is increased of link R1-R3, it broadcasts its updated routing table to its neighbor and H1 then updates the shortest path.

B) Application layer routing table at each node (after link R1-R3 weight changes from 6 to 1)

Node: H1

Routing Table:

```
Updated Table:
Destination     Next Hop        Cost
h2              r1              10
r4              r1              8
r1              r1              2
r2              r1              4
r3              r1              3
```

Node: R1

Routing Table:

```
Updated Table:
Destination     Next Hop        Cost
h2              r3              8
r4              r3              6
h1              h1              2
r2              r3              2
r3              r3              1
```

Node: R2

Routing Table:

```
Updated Table:
Destination     Next Hop        Cost
h2              r4              -2
r4              r4              -4
r1              r1              10
r3              r4              1
```

Node: R3

Routing Table:

```
Updated Table:
Destination     Next Hop        Cost
h2              r4              7
r4              r4              5
r1              r1              1
r2              r4              1
```

Node: R4

Routing Table:

```
Updated Table:
Destination     Next Hop        Cost
h2              h2              2
r2              r2              -4
r3              r3              5
None
```

Node: H2

Routing Table:

```
Destination     Next Hop        Cost
r4              r4              2
None
```