Adhesh Reghu Kumar
COE18B001

Computer Networking Practice
Lab - 9

Simulation of Distance Vector Routing Algorithm (using sockets)

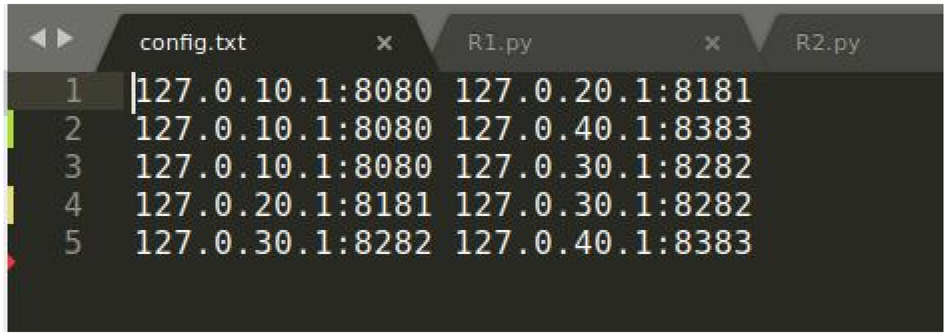
Logic Used:

In this lab exercise we are supposed to simulate the Distance Vector Routing (DVR) algorithm using socket programming.
For this, we have to set up four nodes in a network and check the link cost based on the latency (RTT) of sending a message to one another . Using the latency as the cost, we will implement the DVR algorithm.

The entire program can divided into the following steps:

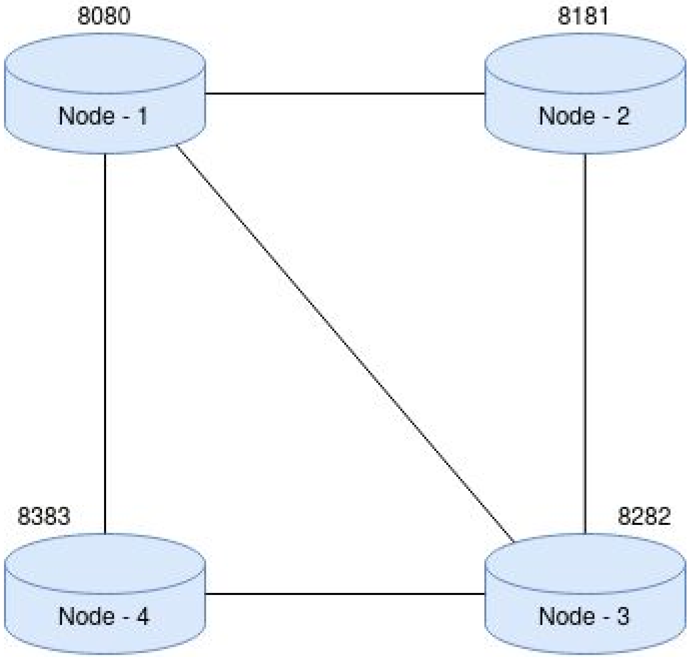
Step 1: Setup the four nodes in the network -

To do this, we need information about the topology of the network, i.e, the connection details of each node such as the neighbors, ip address of the node and the port it is hosted on.
For this exercise, since the four nodes are hosted within the same network, each can be uniquely identified by their corresponding *PORT NUMBERS*. The topology information is stored in a "config.txt" file. Let us look at a sample config file.



```
127.0.10.1:8080 127.0.20.1:8181
127.0.10.1:8080 127.0.40.1:8383
127.0.10.1:8080 127.0.30.1:8282
127.0.20.1:8181 127.0.30.1:8282
127.0.30.1:8282 127.0.40.1:8383
```

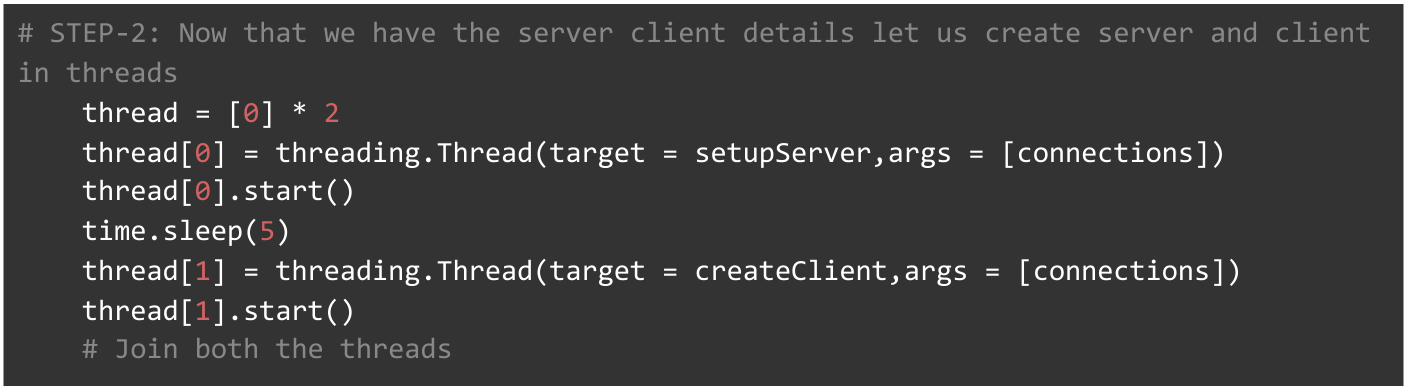
The network topology corresponding to the given configuration file is:



Step 2: Setup servers and clients to emulate the node connections.

Each node will be a TCP server to which the neighboring nodes (as TCP clients) will connect to using separate client sockets. Each node should also be clients (TCP clients) so that the neighboring nodes can also know of the connections. This can be achieved using just server-server connection but since a TCP model doesn't permit such connections, we use client connections instead.

The setup of the servers and clients must happen parallely, hence the functions required to set them up are run in threads. Let us take a look at the corresponding code.



```
# STEP-2: Now that we have the server client details let us create server and client
in threads
    thread = [0] * 2
    thread[0] = threading.Thread(target = setupServer,args = [connections])
    thread[0].start()
    time.sleep(5)
    thread[1] = threading.Thread(target = createClient,args = [connections])
    thread[1].start()
    # Join both the threads
    thread[0].join()
    thread[1].join()

    # Sleep for 2 seconds to ensure the topology is constructed for all nodes
    time.sleep(2)
```

The setupServer() and createClient() functions contain standard TCP implementations of setting up servers and clients along binding addresses and other operations.

Step 3: From each node let us ping (send and recv message) to each of its connections and find the latency in terms of RTT.

Once we have all the nodes setup, we can send a standard message and time the response duration to compute the latency in terms of RTT. We also have to set up threads to manage incoming pings for each node so that there is no blocking in response.

```
# STEP-3: Now let us find the RTT of nodes connected to current node
# Setup all the clients in separate threads to respond to any incoming pings
ping_threads = [0] * num_connections
for i in range(0,num_connections):
    ping_threads[i] = threading.Thread(target = listenToPing, args = [client[i]])
    ping_threads[i].start()

print("[NETWORK TOPOLOGY] Pinging all connected nodes ...")
# Make the server ping all connections
for item in client_sockets:
    conn = item[1]
    start = time.time()
    conn.send(PING_MSG.encode(FORMAT))
    ret_msg = conn.recv(1024)
    end = time.time()
    latencies[ALL_CONN.index(int(item[0][1]))] = (end - start) * FACTOR

# Join all ping threads
for i in range(0,num_connections):
    ping_threads[i].join()
```

Step 4: Initialise the DVR table

Based on the RTT obtained in step 3, we create the initial DVR table. It contains the three columns:

- Destination
- Cost (latency)
- Next hop

```
# STEP-4: Init the routing table
print("\n[DVR] Initial Routing Table is:")
print("%-20s %-25s %-20s" %("Destination","Cost (Latency)","Next Hop"))
for indx in range(0,4):
    rt[indx] = [str('R'+str(indx+1)),latencies[indx],str('R'+str(indx+1))]
    print("%-20s %-25s %-20s" %(rt[indx][0],rt[indx][1],rt[indx][2]))
```

Step 5: Update the DVR Table

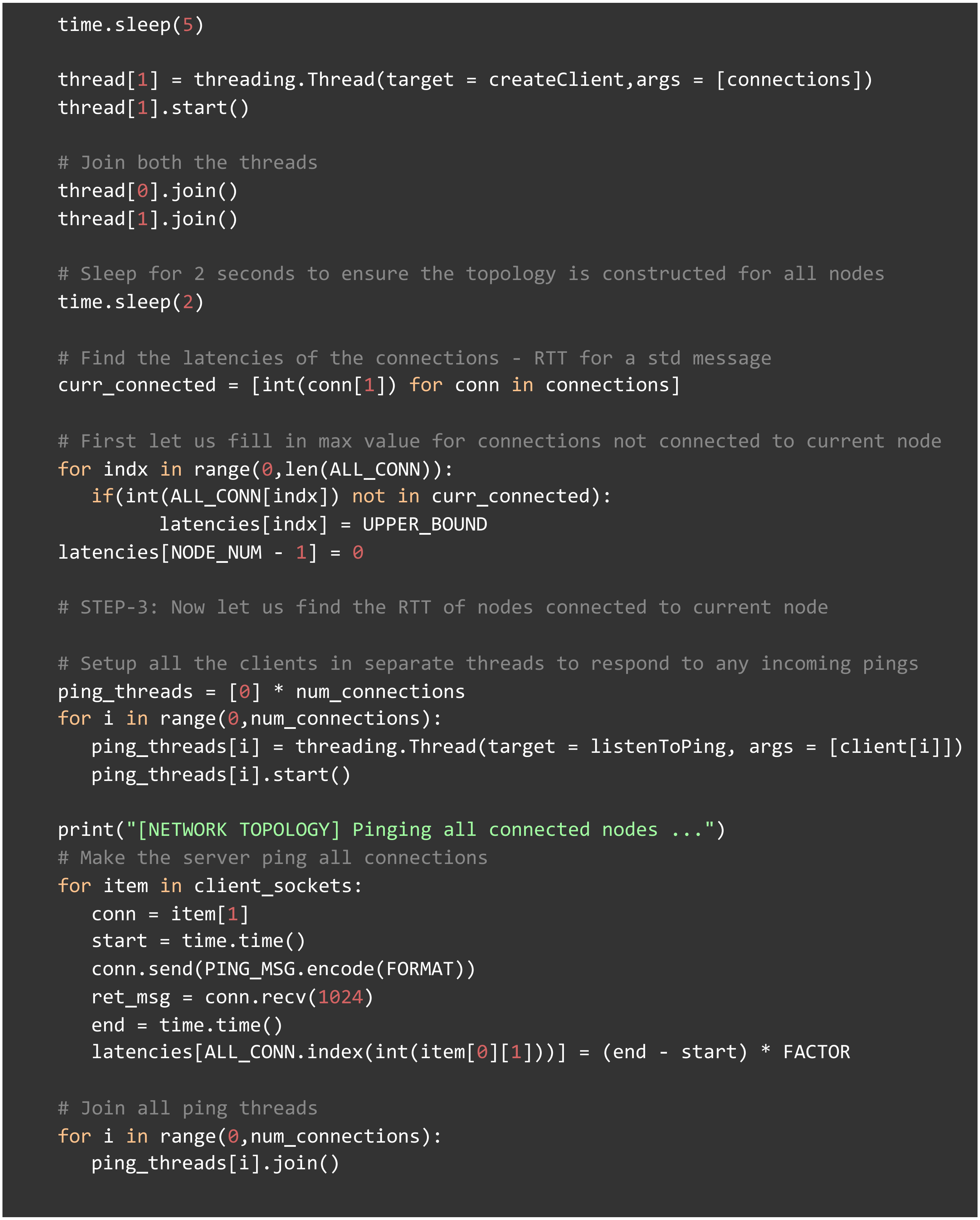
This is the main part of the entire program. In this step, each node shares its latency information with each of its connections. Once the routing information is shared, DVR algorithm determines the minimum path to reach the destination node from a given source node via the shared node. It employs Bellman's shortest path algorithm for this purpose.
Based on the algorithm, the update happens to the current routing table and the next hop and cost might change. The algorithm has to run for **3 iterations** (in a 4 node setup) to find the shortest path from source to destination. Within the 3 iterations, each node would have explored all possible paths and found the shortest path from current node to every other node.

Step 6: Find the routing path from each node to every other node

Once the DVR table is updated, we would like to print the path that node takes to every other node. To do this we would also need next_hop details from other nodes as well. So in each node we set up a thread to respond with information in the node’s routing table if requested. In this manner, if a node wants to know the next_hop after an intermediate node, it can request that information from the intermediate node. In this manner the routing information is printed.

Code: (For a node - R1)

```
import socket
import threading
import time
import sys

# Define constant parameters
ALL_CONN = [8080,8181,8282,8383]

SERVER_PORT = 8080
IP_ADDR = "127.0.10.1"
ADDR = (IP_ADDR,SERVER_PORT)

CLIENT_ADDR = list(IP_ADDR)
CLIENT_ADDR[-1] = str(int(CLIENT_ADDR[-1]) + 1)
CLIENT_ADDR = "".join(CLIENT_ADDR)

CONFIG_PATH = "config.txt"
NODE_NUM = 1

PING_MSG = "abcdef"
PACKET_SIZE = 1024
FORMAT = "utf-8"
FACTOR = 10e3
UPPER_BOUND = 10e7

# define global variables
server = socket.socket()
client_sockets = []
client = [socket.socket()]*4
client_addrs = []

# Initialize global router table
rt = [['nil',-1,'nil']] * 4
rt[NODE_NUM-1] = [str('R'+str(NODE_NUM)),0,str('R'+str(NODE_NUM))]
latencies = [0.0] * 4

# getTopology() - gets the connection details of the nodes in the network
def getTopology():

    # Open file
    file = open(CONFIG_PATH,"r")
    connections = []

    # read the topology details line by line
    line = file.readline()
    while line:

       # Get list of words in the line
       words = line.strip().split(" ")

       # Get ip and port details
       ip_1,port_1 = words[0].split(":")
       ip_2,port_2 = words[1].split(":")

       # Update connection details
       if(ip_1 == IP_ADDR):
             connections.append([ip_2,port_2])
       elif(ip_2 == IP_ADDR):
             connections.append([ip_1,port_1])

       line = file.readline()

    return connections

# Define function to setup server
def setupServer(connections):
    global server
    global client_sockets
    server = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    server.bind(ADDR)
    server.listen()
    print(f"[LISTENING Server is listening on {IP_ADDR}]")
    time.sleep(5)
    for i in range(0,len(connections)):
       client_conn,cli_addr = server.accept()
       client_sockets.append([cli_addr,client_conn])
       print(f"[NEW CONNECTION] {cli_addr} connected.")

# Define the function to create client that connects with all nodes specified in the
topology
def createClient(connections):
    global client
    global CLIENT_ADDR
    i = 0
    for conn in connections:
       addr = (conn[0],int(conn[1]))
       client[i] = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
       client[i].bind((CLIENT_ADDR,SERVER_PORT))
       client[i].connect(addr)
       CLIENT_ADDR = list(CLIENT_ADDR)
       CLIENT_ADDR[-1] = str(int(CLIENT_ADDR[-1]) + 1)
       CLIENT_ADDR = "".join(CLIENT_ADDR)
       i = i + 1

# Let us define th listenToPing() function that responds to incoming pings
def listenToPing(conn):
    msg = conn.recv(1024)
    conn.send(msg)

# Runner thread to exchange latency contribution of current node to all requesting
nodes
def exchangeLatency(conn, lat_str):
    msg = conn.recv(1024).decode(FORMAT)
    if(msg == "EXCHG"):
       conn.send(lat_str.encode(FORMAT))

# function to update the RT based on latency costs from neighbors using Bellman Ford
def updateRT(index,lat_str):
    latency = lat_str.strip().split(",")
    latency = list(map(float,latency))
    cost_x = rt[index][1]

    for i in range(0,4):
       updated_cost = cost_x + latency[i]
       if(rt[i][1] > updated_cost):                               # update based on
min cost
             rt[i][1] = updated_cost
             rt[i][2] = str("R"+str(index+1))

# Given the current hop and destination find the next hop by calling the appropriate
server
def getNextHop(curr_hop,dest,conn):
    # First send request to node
    request_msg = str(dest)
    # time.sleep(2)
    conn.send(request_msg.encode(FORMAT))
    # Get next hop from node
    next_hop = conn.recv(1024).decode(FORMAT)
    next_hop = next_hop.strip().split(",")
    return next_hop

# runner function to handle next hop requests
def nextHop(conn):
    # global client_addrs
    # global client_sockets
    while(1):
       req_msg = conn.recv(1024).decode(FORMAT)
       dest = int(req_msg)

       # Get next hop
       next_hop = rt[dest][2]
       # print("sada",next_hop)
       if(int(next_hop[1]) != dest+1):
             next_conn = 
client_sockets[client_addrs.index(int(ALL_CONN[int(rt[dest][2][-1]) - 1]))][1]
             next_conn.send(str(dest).encode(FORMAT))
             next_hop = next_hop + ","  + next_conn.recv(1024).decode(FORMAT)

       conn.send(next_hop.encode(FORMAT))


def main():

    # STEP-1: First let us obtain the topology details from the config.txt file
    connections = []
    connections = getTopology()

    num_connections = len(connections)

    print("[NETWORK TOPOLOGY] Number of connections =",len(connections))
    for conn in connections:
       print("[NETWORK TOPOLOGY] ",IP_ADDR," --> ",conn[0],":",conn[1],sep ="")

    # STEP-2: Now that we have the server client details let us create server and 
client in threads
    thread = [0] * 2
    thread[0] = threading.Thread(target = setupServer,args = [connections])
    thread[0].start()
```



```
    time.sleep(5)

    thread[1] = threading.Thread(target = createClient,args = [connections])
    thread[1].start()

    # Join both the threads
    thread[0].join()
    thread[1].join()

    # Sleep for 2 seconds to ensure the topology is constructed for all nodes
    time.sleep(2)

    # Find the latencies of the connections - RTT for a std message
    curr_connected = [int(conn[1]) for conn in connections]

    # First let us fill in max value for connections not connected to current node
    for indx in range(0,len(ALL_CONN)):
       if(int(ALL_CONN[indx]) not in curr_connected):
             latencies[indx] = UPPER_BOUND
    latencies[NODE_NUM - 1] = 0

    # STEP-3: Now let us find the RTT of nodes connected to current node

    # Setup all the clients in separate threads to respond to any incoming pings
    ping_threads = [0] * num_connections
    for i in range(0,num_connections):
       ping_threads[i] = threading.Thread(target = listenToPing, args = [client[i]])
       ping_threads[i].start()

    print("[NETWORK TOPOLOGY] Pinging all connected nodes ...")
    # Make the server ping all connections
    for item in client_sockets:
       conn = item[1]
       start = time.time()
       conn.send(PING_MSG.encode(FORMAT))
       ret_msg = conn.recv(1024)
       end = time.time()
       latencies[ALL_CONN.index(int(item[0][1]))] = (end - start) * FACTOR

    # Join all ping threads
    for i in range(0,num_connections):
       ping_threads[i].join()
    print("[NETWORK TOPOLOGY] Latencies:",latencies)

    # STEP-4: Init the routing table
    print("\n[DVR] Initial Routing Table is:")
    print("%-20s %-25s %-20s" %("Destination","Cost (Latency)","Next Hop"))
    for indx in range(0,4):
       rt[indx] = [str('R'+str(indx+1)),latencies[indx],str('R'+str(indx+1))]
       print("%-20s %-25s %-20s" %(rt[indx][0],rt[indx][1],rt[indx][2]))

    # STEP-5: Update routing table - For 3 iterations
    for loop in range(0,3):

       print("\n****************** ITERATION -",loop+1,":
************************")

       # First let us setup the string to be passed from R1 (comma separated
latencies)
       latency_str = ",".join([str(lat[1]) for lat in rt])

       # Iterate over all nodes and request if connected
       print("\n[DVR] Exchanging Routing Information ...")
       for indx in range(0,4):
             if indx == NODE_NUM-1:
                   continue
             elif ALL_CONN[indx] not in curr_connected:
                   print("[DVR]",rt[NODE_NUM-1][0],"is not connected
to",rt[indx][0])

       # Setup threads to exchange the latency contributions of current code to
requesting clients
       latency_threads = [0] * num_connections
       for i in range(0,num_connections):
             latency_threads[i] = threading.Thread(target = exchangeLatency, args =
[client[i],latency_str])
             latency_threads[i].start()

       request_msg = "EXCHG"
       received_lat_str = ["0,0,0,0"]*4
       i = 0
       for item in client_sockets:
             conn = item[1]
             conn.send(request_msg.encode(FORMAT))
             received_lat_str[ALL_CONN.index(int(item[0][1]))] =
conn.recv(1024).decode(FORMAT)

       for i in range(0,num_connections):
             latency_threads[i].join()

       print("[DVR] Received routing information is:")
       print(received_lat_str)

       # Update the router table based on the received latencies - Bellman Ford will
used here
       for indx in range(0,4):
             if(received_lat_str[indx] != "0,0,0,0"):
                   updateRT(indx,received_lat_str[indx])

       print("\n[DVR] Routing Table after iteration -",loop+1,"is: ")
       print("%-20s %-25s %-20s" %("Destination","Cost (Latency)","Next Hop"))
       for indx in range(0,4):
             print("%-20s %-25s %-20s" %(rt[indx][0],rt[indx][1],rt[indx][2]))

    # Print the route for each current src - destination pair
    global client_addrs
    client_addrs = [int(item[0][1]) for item in client_sockets]

    # First setup the server thatll respond to requests from from any connection if
any (regarding next hops)
    hop_threads = [0] * num_connections
    for i in range(0,num_connections):
       hop_threads[i] = threading.Thread(target = nextHop, args = [client[i]])
       hop_threads[i].start()

    # Iterate over each destination and find the route by requesting appropriate
clients for the next hop
    hop_list = [rt[NODE_NUM-1][0]]
    print("\n[DVR] Printing routing information")
    for i in range(0,4):
       if i != NODE_NUM - 1:
             dest = rt[i][0]
             next_hop = rt[i][2]
             hop_list.append(next_hop)
             while(dest not in hop_list):
                   conn =
client_sockets[client_addrs.index(ALL_CONN[int(rt[i][2][-1]) - 1])][1]
                   next_hop = getNextHop(int(next_hop[-1])-1,i,conn)
```

```
                  hop_list.extend(next_hop)
            print(*hop_list, sep=' -> ')
            hop_list = [rt[NODE_NUM-1][0]]

    # Sleep 5 seconds and then close all hop_threads
    time.sleep(5)

if __name__ == '__main__':
    main()
```

Some constant parameters used are

```
# Define constant parameters
ALL_CONN = [8080,8181,8282,8383]

SERVER_PORT = 8080
IP_ADDR = "127.0.10.1"
ADDR = (IP_ADDR,SERVER_PORT)

CLIENT_ADDR = list(IP_ADDR)
CLIENT_ADDR[-1] = str(int(CLIENT_ADDR[-1]) + 1)
CLIENT_ADDR = "".join(CLIENT_ADDR)

CONFIG_PATH = "config.txt"
NODE_NUM = 1

PING_MSG = "abcdef"
PACKET_SIZE = 1024
FORMAT = "utf-8"
FACTOR = 10e3
UPPER_BOUND = 10e7
```

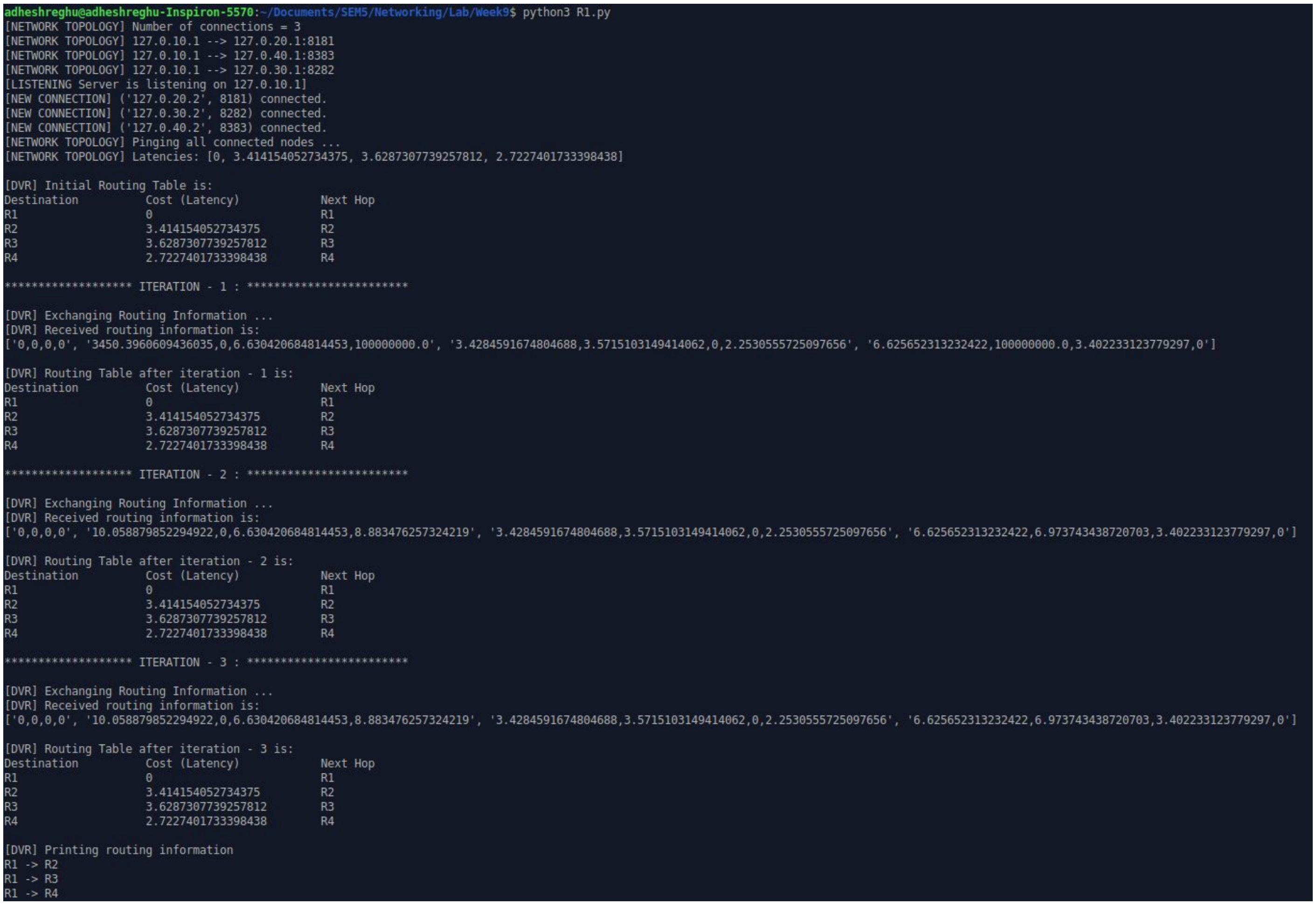
The SERVER_PORT address and IP_ADDR changes for each node. The remaining parameters remain the same.

Note:

- To make the latency (RTT) a bit more significant, it has been multiplied with the parameter FACTOR.
- A node that is unreachable from a given node has a cost of UPPER_BOUND which is used to represent the infinite cost.

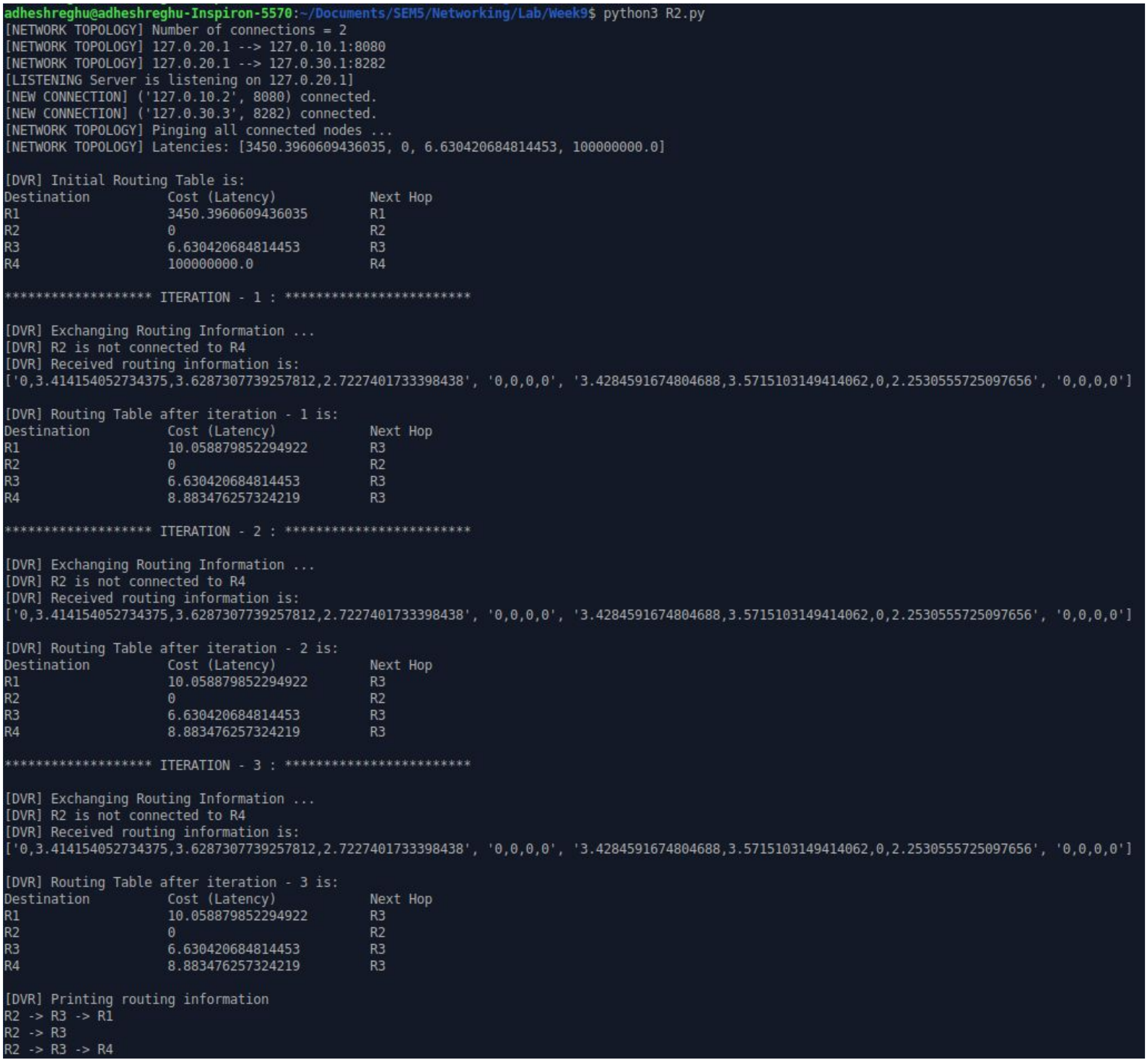
Output:

Node R1:



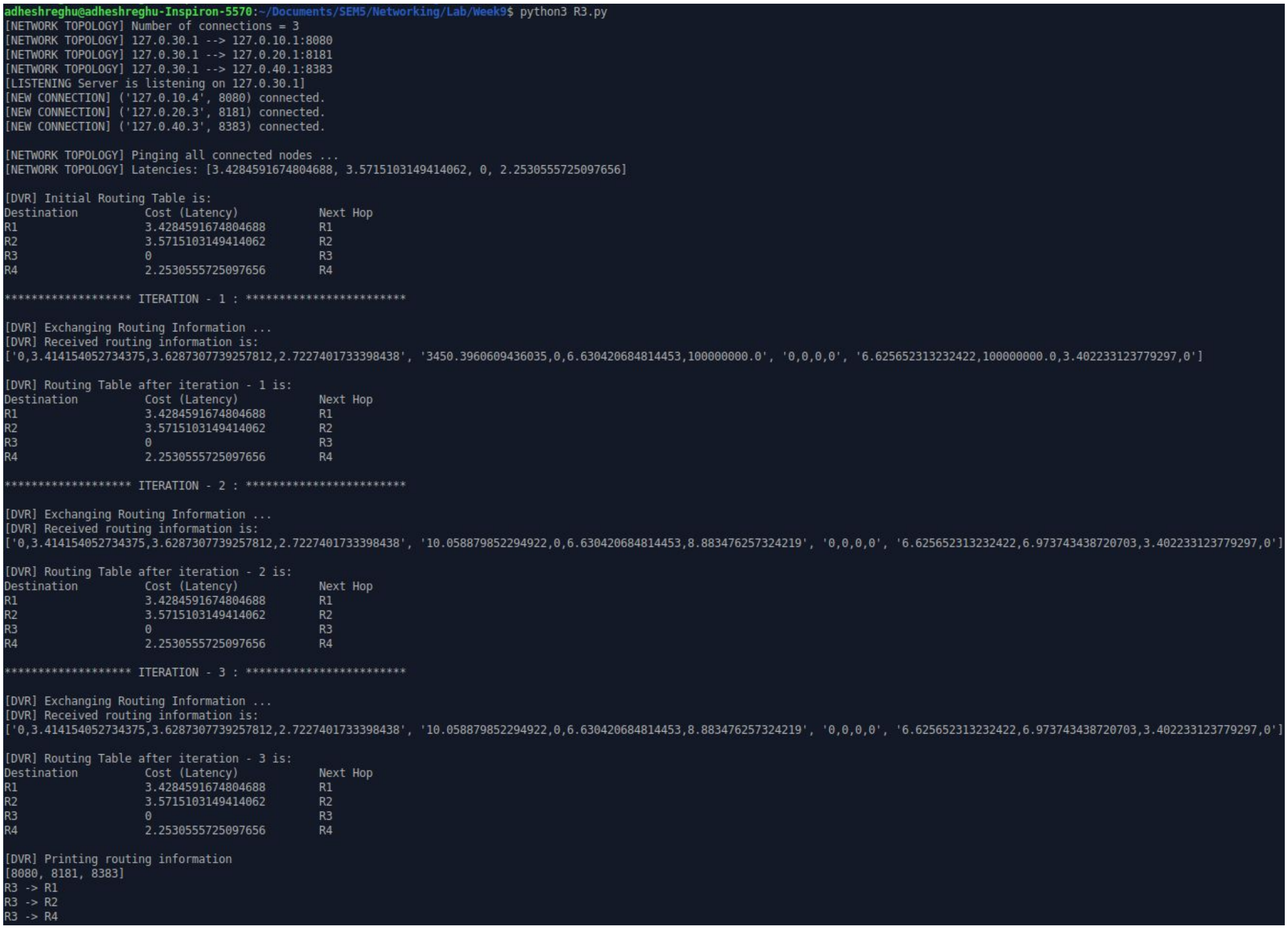
```
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/Networking/Lab/Week9$ python3 R1.py
[NETWORK TOPOLOGY] Number of connections = 3
[NETWORK TOPOLOGY] 127.0.10.1 --> 127.0.20.1:8181
[NETWORK TOPOLOGY] 127.0.10.1 --> 127.0.40.1:8383
[NETWORK TOPOLOGY] 127.0.10.1 --> 127.0.30.1:8282
[LISTENING Server is listening on 127.0.10.1]
[NEW CONNECTION] ('127.0.20.2', 8181) connected.
[NEW CONNECTION] ('127.0.30.2', 8282) connected.
[NEW CONNECTION] ('127.0.40.2', 8383) connected.
[NETWORK TOPOLOGY] Pinging all connected nodes ...
[NETWORK TOPOLOGY] Latencies: [0, 3.414154052734375, 3.6287307739257812, 2.7227401733398438]

[DVR] Initial Routing Table is:
Destination          Cost (Latency)            Next Hop
R1                   0                         R1
R2                   3.414154052734375         R2
R3                   3.6287307739257812        R3
R4                   2.7227401733398438        R4

******************* ITERATION - 1 : ************************

[DVR] Exchanging Routing Information ...
[DVR] Received routing information is:
['0,0,0,0', '3450.3960609436035,0,6.630420684814453,100000000.0', '3.4284591674804688,3.5715103149414062,0,2.2530555725097656', '6.625652313232422,100000000.0,3.402233123779297,0']

[DVR] Routing Table after iteration - 1 is:
Destination          Cost (Latency)            Next Hop
R1                   0                         R1
R2                   3.414154052734375         R2
R3                   3.6287307739257812        R3
R4                   2.7227401733398438        R4

******************* ITERATION - 2 : ************************

[DVR] Exchanging Routing Information ...
[DVR] Received routing information is:
['0,0,0,0', '10.058879852294922,0,6.630420684814453,8.883476257324219', '3.4284591674804688,3.5715103149414062,0,2.2530555725097656', '6.625652313232422,6.973743438720703,3.402233123779297,0']

[DVR] Routing Table after iteration - 2 is:
Destination          Cost (Latency)            Next Hop
R1                   0                         R1
R2                   3.414154052734375         R2
R3                   3.6287307739257812        R3
R4                   2.7227401733398438        R4

******************* ITERATION - 3 : ************************

[DVR] Exchanging Routing Information ...
[DVR] Received routing information is:
['0,0,0,0', '10.058879852294922,0,6.630420684814453,8.883476257324219', '3.4284591674804688,3.5715103149414062,0,2.2530555725097656', '6.625652313232422,6.973743438720703,3.402233123779297,0']

[DVR] Routing Table after iteration - 3 is:
Destination          Cost (Latency)            Next Hop
R1                   0                         R1
R2                   3.414154052734375         R2
R3                   3.6287307739257812        R3
R4                   2.7227401733398438        R4

[DVR] Printing routing information
R1 -> R2
R1 -> R3
R1 -> R4
```

Node R2:



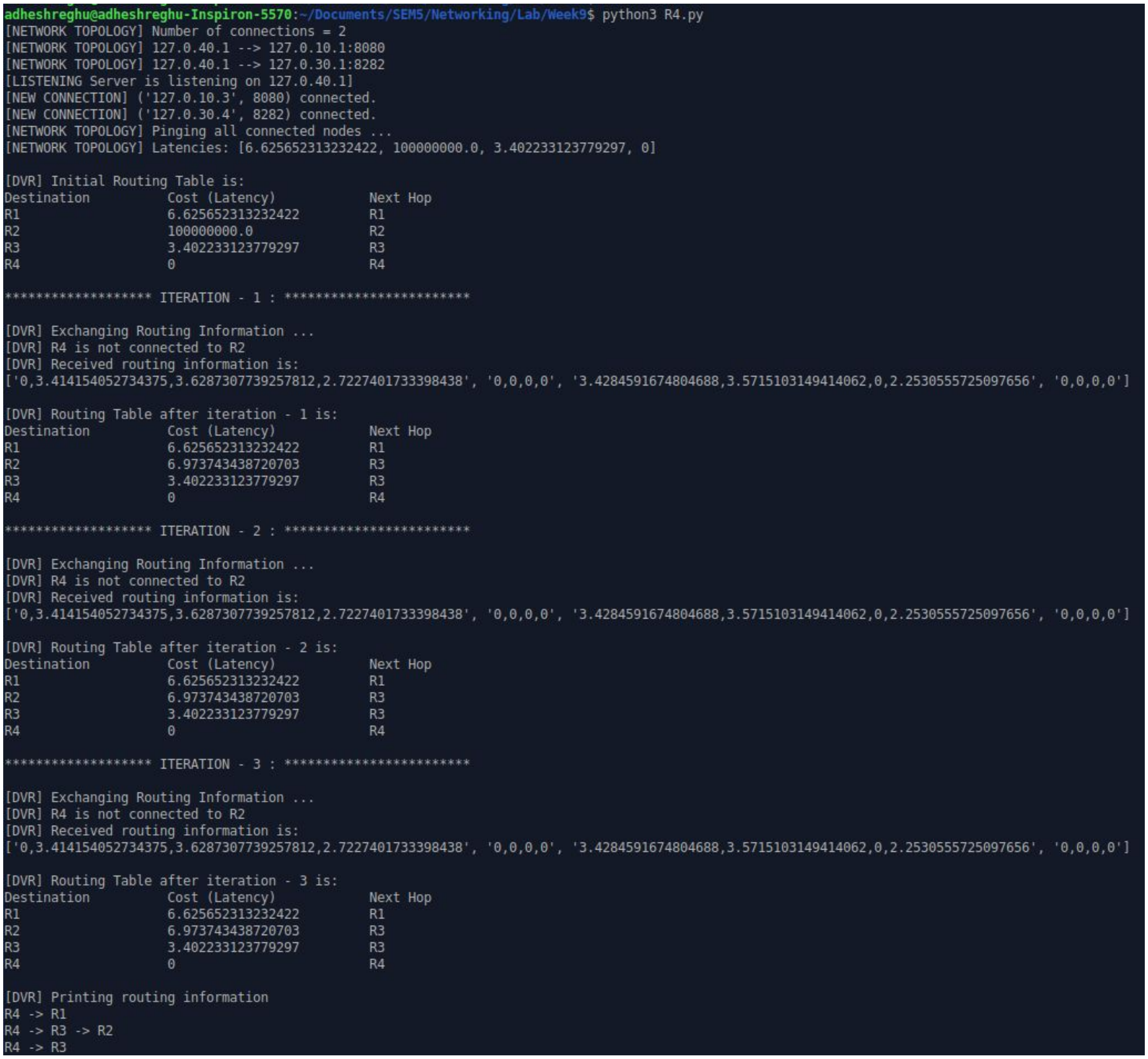
```
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/Networking/Lab/Week9$ python3 R2.py
[NETWORK TOPOLOGY] Number of connections = 2
[NETWORK TOPOLOGY] 127.0.20.1 --> 127.0.10.1:8080
[NETWORK TOPOLOGY] 127.0.20.1 --> 127.0.30.1:8282
[LISTENING Server is listening on 127.0.20.1]
[NEW CONNECTION] ('127.0.10.2', 8080) connected.
[NEW CONNECTION] ('127.0.30.3', 8282) connected.
[NETWORK TOPOLOGY] Pinging all connected nodes ...
[NETWORK TOPOLOGY] Latencies: [3450.3960609436035, 0, 6.630420684814453, 100000000.0]

[DVR] Initial Routing Table is:
Destination          Cost (Latency)            Next Hop
R1                   3450.3960609436035        R1
R2                   0                         R2
R3                   6.630420684814453         R3
R4                   100000000.0               R4

******************* ITERATION - 1 : ************************

[DVR] Exchanging Routing Information ...
[DVR] R2 is not connected to R4
[DVR] Received routing information is:
['0,3.414154052734375,3.6287307739257812,2.7227401733398438', '0,0,0,0', '3.4284591674804688,3.5715103149414062,0,2.2530555725097656', '0,0,0,0']

[DVR] Routing Table after iteration - 1 is:
Destination          Cost (Latency)            Next Hop
R1                   10.058879852294922        R3
R2                   0                         R2
R3                   6.630420684814453         R3
R4                   8.883476257324219         R3

******************* ITERATION - 2 : ************************

[DVR] Exchanging Routing Information ...
[DVR] R2 is not connected to R4
[DVR] Received routing information is:
['0,3.414154052734375,3.6287307739257812,2.7227401733398438', '0,0,0,0', '3.4284591674804688,3.5715103149414062,0,2.2530555725097656', '0,0,0,0']

[DVR] Routing Table after iteration - 2 is:
Destination          Cost (Latency)            Next Hop
R1                   10.058879852294922        R3
R2                   0                         R2
R3                   6.630420684814453         R3
R4                   8.883476257324219         R3

******************* ITERATION - 3 : ************************

[DVR] Exchanging Routing Information ...
[DVR] R2 is not connected to R4
[DVR] Received routing information is:
['0,3.414154052734375,3.6287307739257812,2.7227401733398438', '0,0,0,0', '3.4284591674804688,3.5715103149414062,0,2.2530555725097656', '0,0,0,0']

[DVR] Routing Table after iteration - 3 is:
Destination          Cost (Latency)            Next Hop
R1                   10.058879852294922        R3
R2                   0                         R2
R3                   6.630420684814453         R3
R4                   8.883476257324219         R3

[DVR] Printing routing information
R2 -> R3 -> R1
R2 -> R3
R2 -> R3 -> R4
```

Node R3:



```
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/Networking/Lab/Week9$ python3 R3.py
[NETWORK TOPOLOGY] Number of connections = 3
[NETWORK TOPOLOGY] 127.0.30.1 --> 127.0.10.1:8080
[NETWORK TOPOLOGY] 127.0.30.1 --> 127.0.20.1:8181
[NETWORK TOPOLOGY] 127.0.30.1 --> 127.0.40.1:8383
[LISTENING Server is listening on 127.0.30.1]
[NEW CONNECTION] ('127.0.10.4', 8080) connected.
[NEW CONNECTION] ('127.0.20.3', 8181) connected.
[NEW CONNECTION] ('127.0.40.3', 8383) connected.

[NETWORK TOPOLOGY] Pinging all connected nodes ...
[NETWORK TOPOLOGY] Latencies: [3.4284591674804688, 3.5715103149414062, 0, 2.2530555725097656]

[DVR] Initial Routing Table is:
Destination          Cost (Latency)            Next Hop
R1                   3.4284591674804688        R1
R2                   3.5715103149414062        R2
R3                   0                         R3
R4                   2.2530555725097656        R4

******************* ITERATION - 1 : ************************

[DVR] Exchanging Routing Information ...
[DVR] Received routing information is:
['0,3.414154052734375,3.6287307739257812,2.7227401733398438', '3450.3960609436035,0,6.630420684814453,100000000.0', '0,0,0,0', '6.625652313232422,100000000.0,3.402233123779297,0']

[DVR] Routing Table after iteration - 1 is:
Destination          Cost (Latency)            Next Hop
R1                   3.4284591674804688        R1
R2                   3.5715103149414062        R2
R3                   0                         R3
R4                   2.2530555725097656        R4

******************* ITERATION - 2 : ************************

[DVR] Exchanging Routing Information ...
[DVR] Received routing information is:
['0,3.414154052734375,3.6287307739257812,2.7227401733398438', '10.058879852294922,0,6.630420684814453,8.883476257324219', '0,0,0,0', '6.625652313232422,6.973743438720703,3.402233123779297,0']

[DVR] Routing Table after iteration - 2 is:
Destination          Cost (Latency)            Next Hop
R1                   3.4284591674804688        R1
R2                   3.5715103149414062        R2
R3                   0                         R3
R4                   2.2530555725097656        R4

******************* ITERATION - 3 : ************************

[DVR] Exchanging Routing Information ...
[DVR] Received routing information is:
['0,3.414154052734375,3.6287307739257812,2.7227401733398438', '10.058879852294922,0,6.630420684814453,8.883476257324219', '0,0,0,0', '6.625652313232422,6.973743438720703,3.402233123779297,0']

[DVR] Routing Table after iteration - 3 is:
Destination          Cost (Latency)            Next Hop
R1                   3.4284591674804688        R1
R2                   3.5715103149414062        R2
R3                   0                         R3
R4                   2.2530555725097656        R4

[DVR] Printing routing information
[8080, 8181, 8383]
R3 -> R1
R3 -> R2
R3 -> R4
```

Node R4:



```
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/Networking/Lab/Week9$ python3 R4.py
[NETWORK TOPOLOGY] Number of connections = 2
[NETWORK TOPOLOGY] 127.0.40.1 --> 127.0.10.1:8080
[NETWORK TOPOLOGY] 127.0.40.1 --> 127.0.30.1:8282
[LISTENING Server is listening on 127.0.40.1]
[NEW CONNECTION] ('127.0.10.3', 8080) connected.
[NEW CONNECTION] ('127.0.30.4', 8282) connected.
[NETWORK TOPOLOGY] Pinging all connected nodes ...
[NETWORK TOPOLOGY] Latencies: [6.625652313232422, 100000000.0, 3.402233123779297, 0]

[DVR] Initial Routing Table is:
Destination          Cost (Latency)           Next Hop
R1                   6.625652313232422        R1
R2                   100000000.0              R2
R3                   3.402233123779297        R3
R4                   0                        R4

****************** ITERATION - 1 : ***********************

[DVR] Exchanging Routing Information ...
[DVR] R4 is not connected to R2
[DVR] Received routing information is:
['0,3.414154052734375,3.6287307739257812,2.7227401733398438', '0,0,0,0', '3.4284591674804688,3.5715103149414062,0,2.2530555725097656', '0,0,0,0']

[DVR] Routing Table after iteration - 1 is:
Destination          Cost (Latency)           Next Hop
R1                   6.625652313232422        R1
R2                   6.973743438720703        R3
R3                   3.402233123779297        R3
R4                   0                        R4

****************** ITERATION - 2 : ***********************

[DVR] Exchanging Routing Information ...
[DVR] R4 is not connected to R2
[DVR] Received routing information is:
['0,3.414154052734375,3.6287307739257812,2.7227401733398438', '0,0,0,0', '3.4284591674804688,3.5715103149414062,0,2.2530555725097656', '0,0,0,0']

[DVR] Routing Table after iteration - 2 is:
Destination          Cost (Latency)           Next Hop
R1                   6.625652313232422        R1
R2                   6.973743438720703        R3
R3                   3.402233123779297        R3
R4                   0                        R4

****************** ITERATION - 3 : ***********************

[DVR] Exchanging Routing Information ...
[DVR] R4 is not connected to R2
[DVR] Received routing information is:
['0,3.414154052734375,3.6287307739257812,2.7227401733398438', '0,0,0,0', '3.4284591674804688,3.5715103149414062,0,2.2530555725097656', '0,0,0,0']

[DVR] Routing Table after iteration - 3 is:
Destination          Cost (Latency)           Next Hop
R1                   6.625652313232422        R1
R2                   6.973743438720703        R3
R3                   3.402233123779297        R3
R4                   0                        R4

[DVR] Printing routing information
R4 -> R1
R4 -> R3 -> R2
R4 -> R3
```

(Refer to the output folder for clearer images)