Project - 3 : Implementation of Routing Protocol

Course Name: Computer Networks

Course Number: CSE 4344/5344

Group Member Names:

Ravi Prakasha - 1002026832

Venkata Subrahmanya Abhinav Rallapalli - 1002078961

Instructor: Dr. Sajib Datta

Code Execution Steps :

1) Download the zip folder from the canvas.

2) Extract the folder.

3) Install visual studio code and requirements.txt file

   a) Steps to install visual studio code is given in

      https://medium.com/nerd-for-tech/install-visual-studio-code-fe3908c5cf15

   b) Run the command pip3 install -r requirements.txt to install all the required packages.

4) After installation click on File in visual studio code and click on Open Folder and select the extracted folder.

5) Click on Terminal and have two terminal opened on the same folder.

6) Then type the command python3 final_with_tst.py 5000 confir.config to run the first file in first terminal.

7) In the second Terminal run the command as python3 check_update.py

8) Please input the data from the terminal for the first file i.e. in first terminal

   a) Enter the value for n the 'n' updates . Ex - 2

   b) Enter the ip_address to bind for the socket. Ex - 127.0.0.1

9) Please input the data from the terminal for the second file i.e. in second terminal

   a) Enter the source ip_address in the network for which cost to be updated. Ex - 127.0.0.1

   b) Enter the destination ip_address in the network for which cost to be updated. Ex - 127.0.0.6

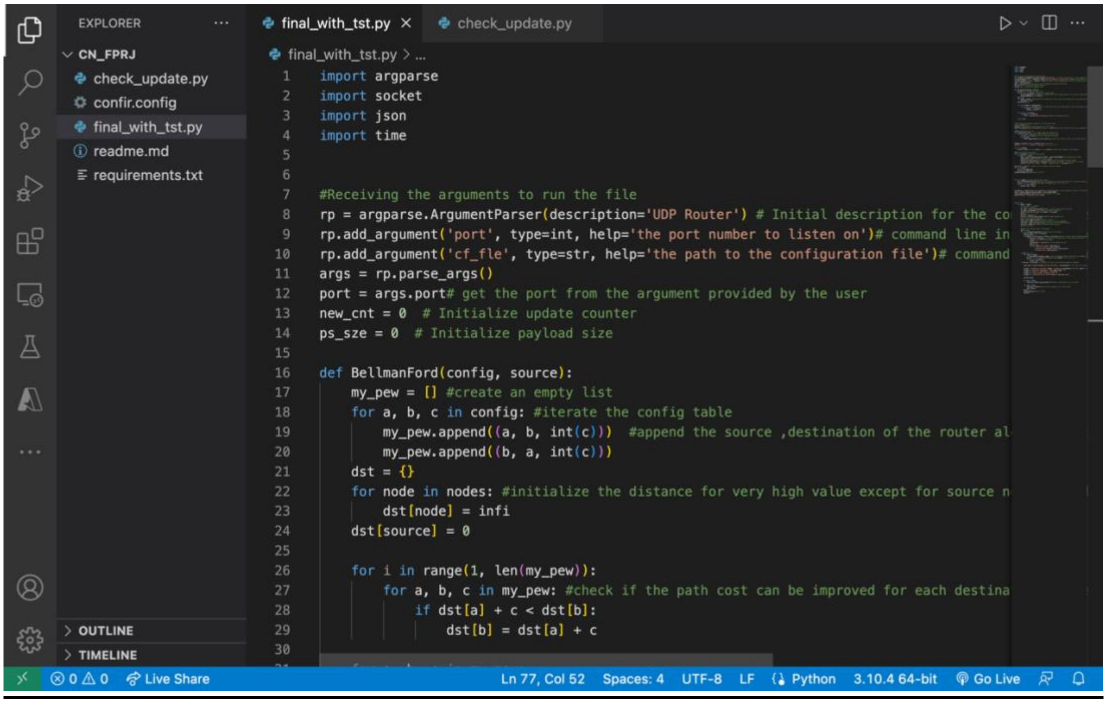
   c) Enter the cost to be updated . Ex - 2

   d) Enter the ip address to which you need to send data . Ex -127.0.0.1

10) The code runs until n updates and stops printing the final message.

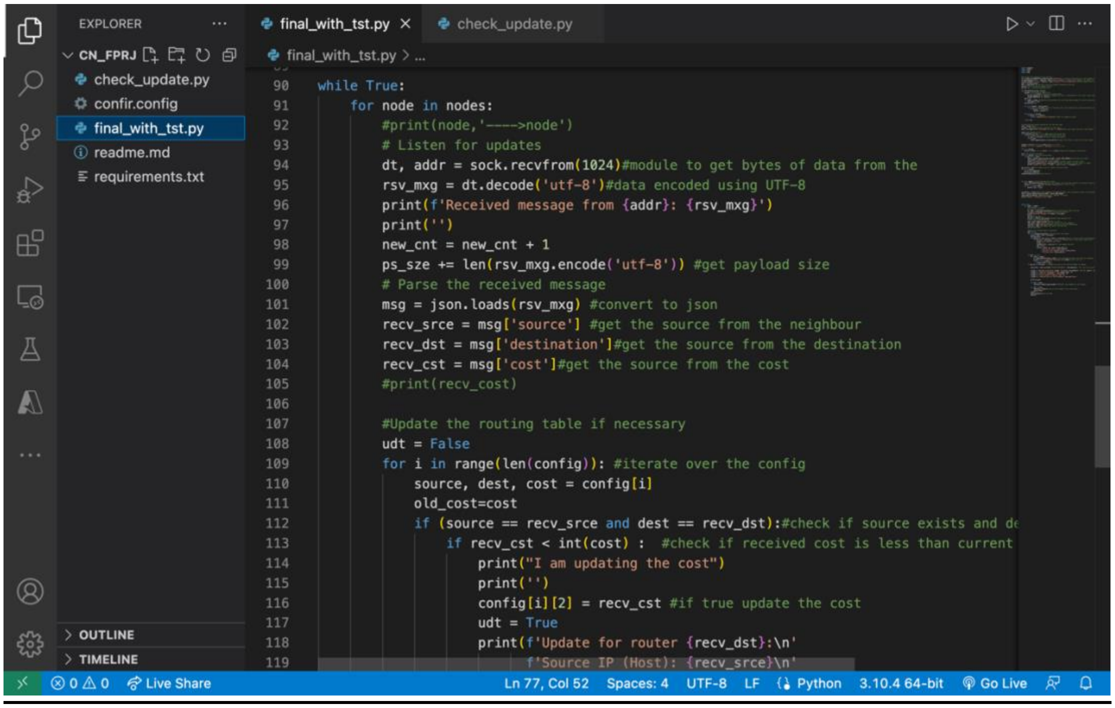
Test Case 2: Based on the given test case, when I add the last two digits of my team member's UTA ID number to my own number (1002026832, 1002078961), the resulting total is 93, which is an odd number.Furthermore, according to the test case, since our number is odd, it shows that there is a link failure between B and D in the network topology. As a result, we need to update the network topology by running the Bellman Ford algorithm again, which will calculate the shortest path between routers and update the cost accordingly.

Code Screenshots:



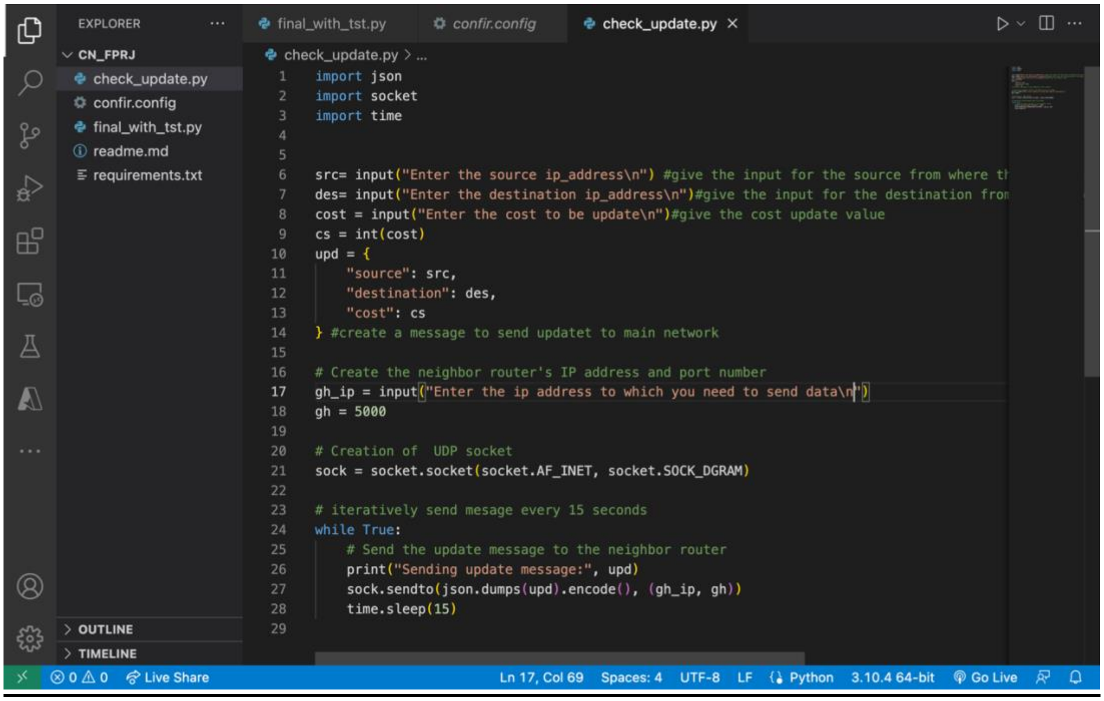
```python
import argparse
import socket
import json
import time


#Receiving the arguments to run the file
rp = argparse.ArgumentParser(description='UDP Router') # Initial description for the co
rp.add_argument('port', type=int, help='the port number to listen on')# command line in
rp.add_argument('cf_fle', type=str, help='the path to the configuration file')# command
args = rp.parse_args()
port = args.port# get the port from the argument provided by the user
new_cnt = 0  # Initialize update counter
ps_sze = 0  # Initialize payload size

def BellmanFord(config, source):
    my_pew = [] #create an empty list
    for a, b, c in config: #iterate the config table
        my_pew.append((a, b, int(c)))  #append the source ,destination of the router al
        my_pew.append((b, a, int(c)))
    dst = {}
    for node in nodes: #initialize the distance for very high value except for source n
        dst[node] = infi
    dst[source] = 0

    for i in range(1, len(my_pew)):
        for a, b, c in my_pew: #check if the path cost can be improved for each destina
            if dst[a] + c < dst[b]:
                dst[b] = dst[a] + c

```

Fig 1 : Screenshot of the main program showing the code to read network topology and Bellman Ford implementation



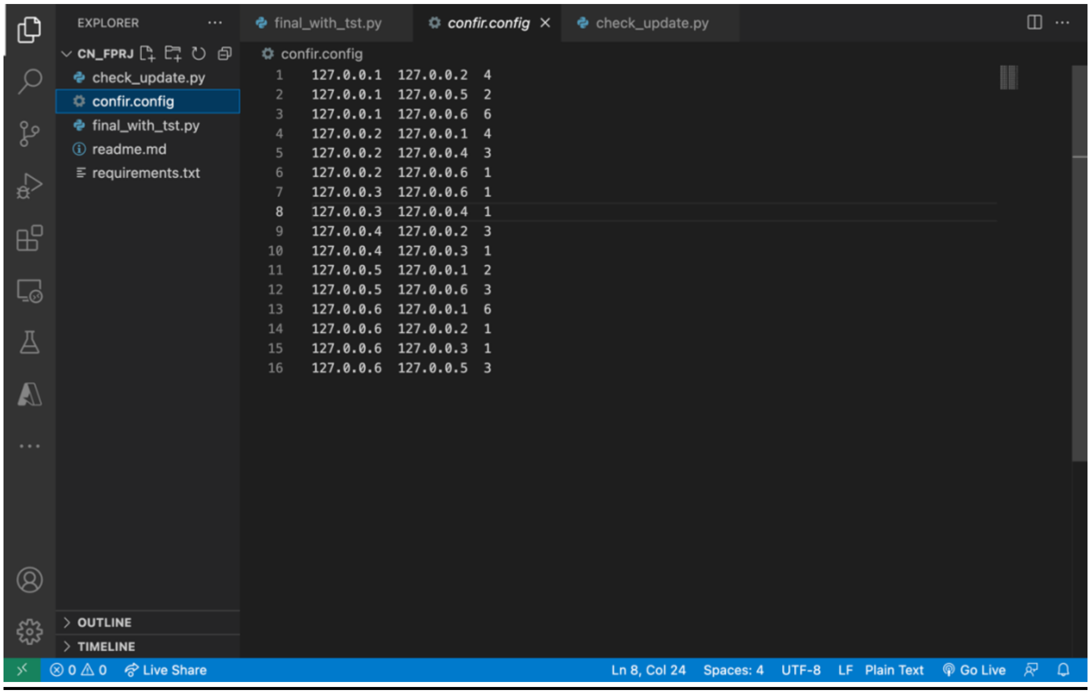
```python
while True:
    for node in nodes:
        #print(node,'---->node')
        # Listen for updates
        dt, addr = sock.recvfrom(1024)#module to get bytes of data from the
        rsv_mxg = dt.decode('utf-8')#data encoded using UTF-8
        print(f'Received message from {addr}: {rsv_mxg}')
        print('')
        new_cnt = new_cnt + 1
        ps_sze += len(rsv_mxg.encode('utf-8')) #get payload size
        # Parse the received message
        msg = json.loads(rsv_mxg) #convert to json
        recv_srce = msg['source'] #get the source from the neighbour
        recv_dst = msg['destination']#get the source from the destination
        recv_cst = msg['cost']#get the source from the cost
        #print(recv_cost)

        #Update the routing table if necessary
        udt = False
        for i in range(len(config)): #iterate over the config
            source, dest, cost = config[i]
            old_cost=cost
            if (source == recv_srce and dest == recv_dst):#check if source exists and de
                if recv_cst < int(cost) :  #check if received cost is less than current
                    print("I am updating the cost")
                    print('')
                    config[i][2] = recv_cst #if true update the cost
                    udt = True
                    print(f'Update for router {recv_dst}:\n'
                          f'Source IP (Host): {recv_srce}\n'
```

Fig 2 : Screenshot of the main program showing the code to receive updates from neighbor



```python
import json
import socket
import time


src= input("Enter the source ip_address\n") #give the input for the source from where th
des= input("Enter the destination ip_address\n")#give the input for the destination fro
cost = input("Enter the cost to be update\n")#give the cost update value
cs = int(cost)
upd = {
    "source": src,
    "destination": des,
    "cost": cs
} #create a message to send updatet to main network

# Create the neighbor router's IP address and port number
gh_ip = input("Enter the ip address to which you need to send data\n")
gh = 5000

# Creation of  UDP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

# iteratively send mesage every 15 seconds
while True:
    # Send the update message to the neighbor router
    print("Sending update message:", upd)
    sock.sendto(json.dumps(upd).encode(), (gh_ip, gh))
    time.sleep(15)

```

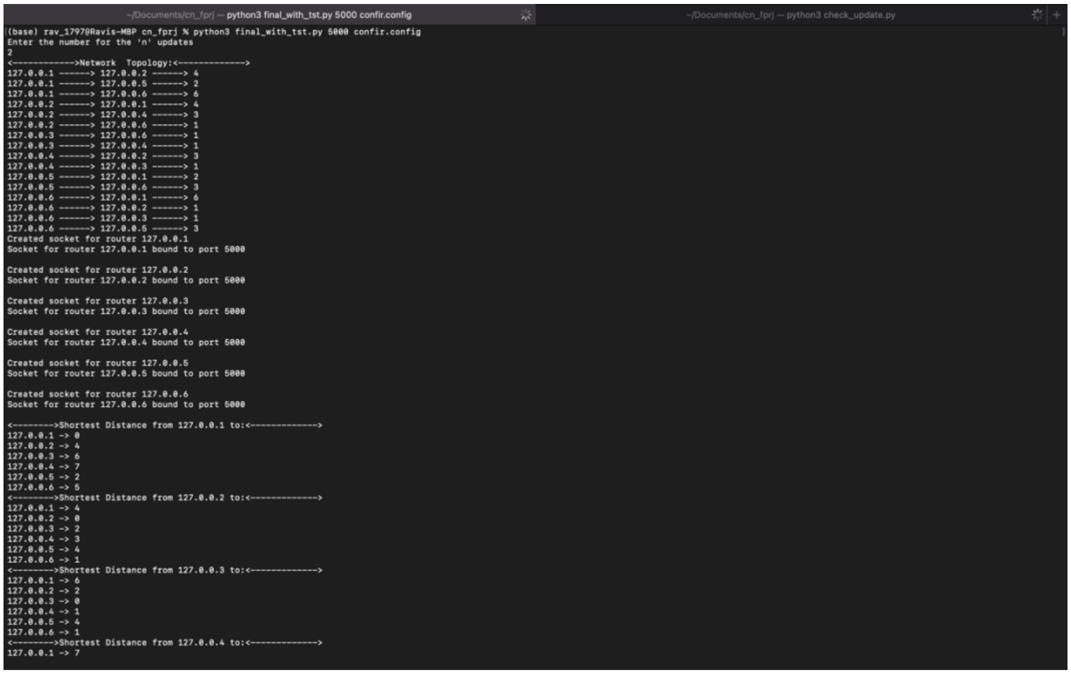
Fig 3 : Screenshot of the program to send updated cost to main program



```
127.0.0.1  127.0.0.2  4
127.0.0.1  127.0.0.5  2
127.0.0.1  127.0.0.6  6
127.0.0.2  127.0.0.1  4
127.0.0.2  127.0.0.4  3
127.0.0.2  127.0.0.6  1
127.0.0.3  127.0.0.6  1
127.0.0.3  127.0.0.4  1
127.0.0.4  127.0.0.2  3
127.0.0.4  127.0.0.3  1
127.0.0.5  127.0.0.1  2
127.0.0.5  127.0.0.6  3
127.0.0.6  127.0.0.1  6
127.0.0.6  127.0.0.2  1
127.0.0.6  127.0.0.3  1
127.0.0.6  127.0.0.5  3
```

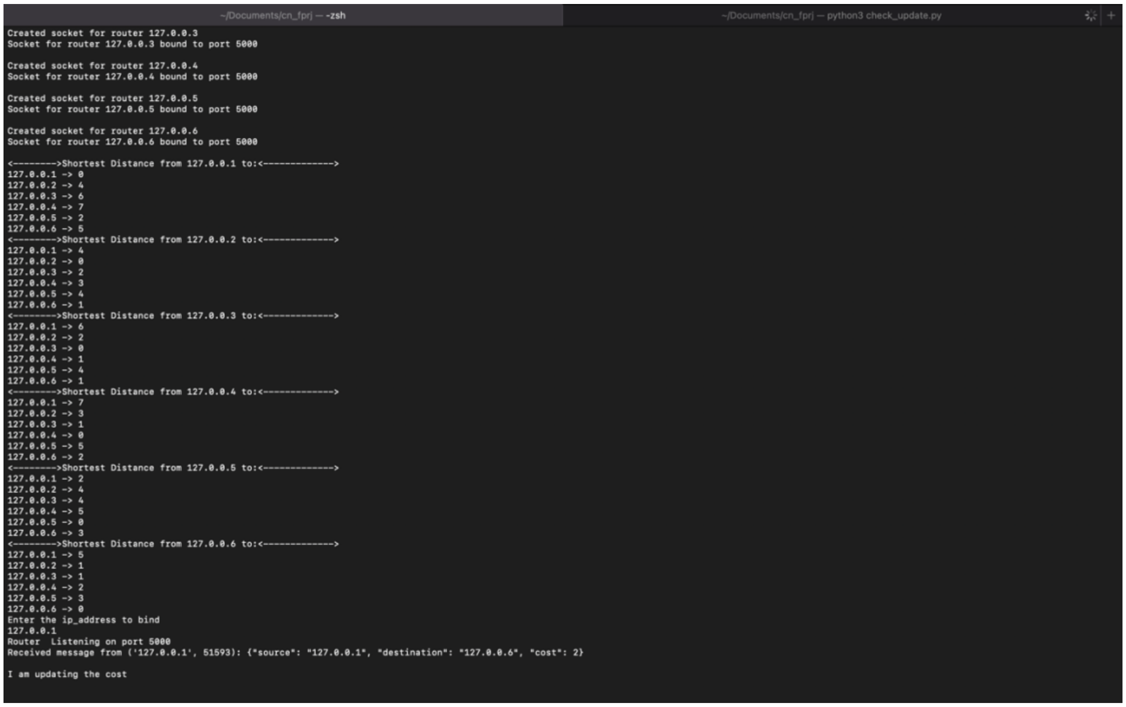
Fig 4: Screenshot of the network topology in the configuration file

Code Output Screenshots:


~/Documents/cn_fprj — python3 final_with_tst.py 5000 confir.config
~/Documents/cn_fprj — python3 check_update.py

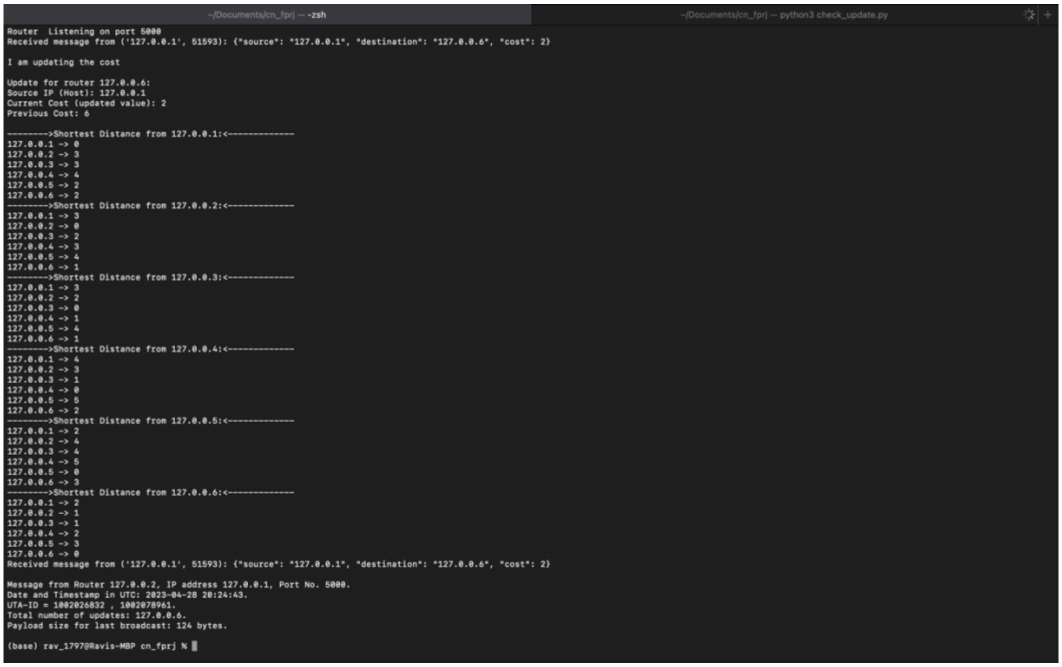
```
(base) rav_1797@Ravis-MBP cn_fprj % python3 final_with_tst.py 5000 confir.config
Enter the number for the 'n' updates
2
<------------>Network  Topology:<------------->
127.0.0.1 ------> 127.0.0.2 ------> 4
127.0.0.1 ------> 127.0.0.5 ------> 2
127.0.0.1 ------> 127.0.0.6 ------> 6
127.0.0.2 ------> 127.0.0.1 ------> 4
127.0.0.2 ------> 127.0.0.4 ------> 3
127.0.0.2 ------> 127.0.0.6 ------> 1
127.0.0.3 ------> 127.0.0.6 ------> 1
127.0.0.3 ------> 127.0.0.4 ------> 1
127.0.0.4 ------> 127.0.0.2 ------> 3
127.0.0.4 ------> 127.0.0.3 ------> 1
127.0.0.5 ------> 127.0.0.1 ------> 2
127.0.0.5 ------> 127.0.0.6 ------> 3
127.0.0.6 ------> 127.0.0.1 ------> 6
127.0.0.6 ------> 127.0.0.2 ------> 1
127.0.0.6 ------> 127.0.0.3 ------> 1
127.0.0.6 ------> 127.0.0.5 ------> 3
Created socket for router 127.0.0.1
Socket for router 127.0.0.1 bound to port 5000

Created socket for router 127.0.0.2
Socket for router 127.0.0.2 bound to port 5000

Created socket for router 127.0.0.3
Socket for router 127.0.0.3 bound to port 5000

Created socket for router 127.0.0.4
Socket for router 127.0.0.4 bound to port 5000

Created socket for router 127.0.0.5
Socket for router 127.0.0.5 bound to port 5000

Created socket for router 127.0.0.6
Socket for router 127.0.0.6 bound to port 5000

<-------->Shortest Distance from 127.0.0.1 to:<------------->
127.0.0.1 -> 0
127.0.0.2 -> 4
127.0.0.3 -> 6
127.0.0.4 -> 7
127.0.0.5 -> 2
127.0.0.6 -> 5
<-------->Shortest Distance from 127.0.0.2 to:<------------->
127.0.0.1 -> 4
127.0.0.2 -> 0
127.0.0.3 -> 2
127.0.0.4 -> 3
127.0.0.5 -> 4
127.0.0.6 -> 1
<-------->Shortest Distance from 127.0.0.3 to:<------------->
127.0.0.1 -> 6
127.0.0.2 -> 2
127.0.0.3 -> 0
127.0.0.4 -> 1
127.0.0.5 -> 4
127.0.0.6 -> 1
<-------->Shortest Distance from 127.0.0.4 to:<------------->
127.0.0.1 -> 7
```

Fig 5: Screenshot of the output network topology


~/Documents/cn_fprj — -zsh
~/Documents/cn_fprj — python3 check_update.py

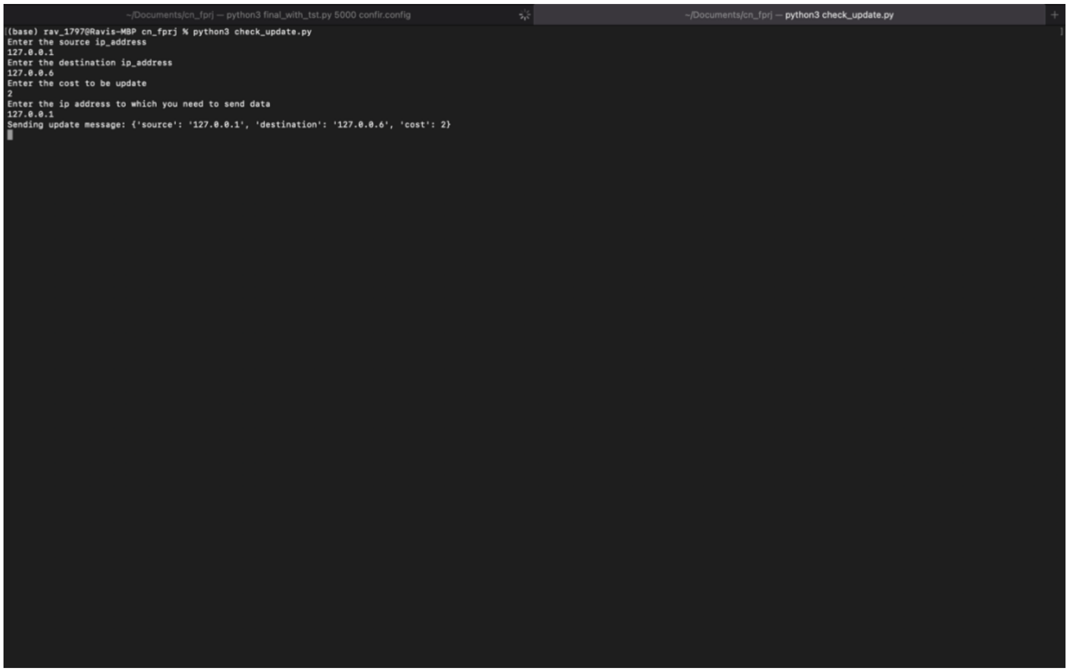
```
Created socket for router 127.0.0.3
Socket for router 127.0.0.3 bound to port 5000

Created socket for router 127.0.0.4
Socket for router 127.0.0.4 bound to port 5000

Created socket for router 127.0.0.5
Socket for router 127.0.0.5 bound to port 5000

Created socket for router 127.0.0.6
Socket for router 127.0.0.6 bound to port 5000

<-------->Shortest Distance from 127.0.0.1 to:<------------->
127.0.0.1 -> 0
127.0.0.2 -> 4
127.0.0.3 -> 6
127.0.0.4 -> 7
127.0.0.5 -> 2
127.0.0.6 -> 5
<-------->Shortest Distance from 127.0.0.2 to:<------------->
127.0.0.1 -> 4
127.0.0.2 -> 0
127.0.0.3 -> 2
127.0.0.4 -> 3
127.0.0.5 -> 4
127.0.0.6 -> 1
<-------->Shortest Distance from 127.0.0.3 to:<------------->
127.0.0.1 -> 6
127.0.0.2 -> 2
127.0.0.3 -> 0
127.0.0.4 -> 1
127.0.0.5 -> 4
127.0.0.6 -> 1
<-------->Shortest Distance from 127.0.0.4 to:<------------->
127.0.0.1 -> 7
127.0.0.2 -> 3
127.0.0.3 -> 1
127.0.0.4 -> 0
127.0.0.5 -> 5
127.0.0.6 -> 2
<-------->Shortest Distance from 127.0.0.5 to:<------------->
127.0.0.1 -> 2
127.0.0.2 -> 4
127.0.0.3 -> 4
127.0.0.4 -> 5
127.0.0.5 -> 0
127.0.0.6 -> 3
<-------->Shortest Distance from 127.0.0.6 to:<------------->
127.0.0.1 -> 5
127.0.0.2 -> 1
127.0.0.3 -> 1
127.0.0.4 -> 2
127.0.0.5 -> 3
127.0.0.6 -> 0
Enter the ip_address to bind
127.0.0.1
Router  Listening on port 5000
Received message from ('127.0.0.1', 51593): {"source": "127.0.0.1", "destination": "127.0.0.6", "cost": 2}

I am updating the cost
```

Fig 6: Screenshot showing the output of Bellman Ford Algorithm over the network topology


~/Documents/cn_fprj — -zsh
~/Documents/cn_fprj — python3 check_update.py

```
Router  Listening on port 5000
Received message from ('127.0.0.1', 51593): {"source": "127.0.0.1", "destination": "127.0.0.6", "cost": 2}

I am updating the cost

Update for router 127.0.0.6:
Source IP (Host): 127.0.0.1
Current Cost (updated value): 2
Previous Cost: 6

--------->Shortest Distance from 127.0.0.1:<-------------
127.0.0.1 -> 0
127.0.0.2 -> 3
127.0.0.3 -> 3
127.0.0.4 -> 4
127.0.0.5 -> 2
127.0.0.6 -> 2
--------->Shortest Distance from 127.0.0.2:<-------------
127.0.0.1 -> 3
127.0.0.2 -> 0
127.0.0.3 -> 2
127.0.0.4 -> 3
127.0.0.5 -> 4
127.0.0.6 -> 1
--------->Shortest Distance from 127.0.0.3:<-------------
127.0.0.1 -> 3
127.0.0.2 -> 2
127.0.0.3 -> 0
127.0.0.4 -> 1
127.0.0.5 -> 4
127.0.0.6 -> 1
--------->Shortest Distance from 127.0.0.4:<-------------
127.0.0.1 -> 4
127.0.0.2 -> 3
127.0.0.3 -> 1
127.0.0.4 -> 0
127.0.0.5 -> 5
127.0.0.6 -> 2
--------->Shortest Distance from 127.0.0.5:<-------------
127.0.0.1 -> 2
127.0.0.2 -> 4
127.0.0.3 -> 4
127.0.0.4 -> 5
127.0.0.5 -> 0
127.0.0.6 -> 3
--------->Shortest Distance from 127.0.0.6:<-------------
127.0.0.1 -> 2
127.0.0.2 -> 1
127.0.0.3 -> 1
127.0.0.4 -> 2
127.0.0.5 -> 3
127.0.0.6 -> 0
Received message from ('127.0.0.1', 51593): {"source": "127.0.0.1", "destination": "127.0.0.6", "cost": 2}

Message from Router 127.0.0.2, IP address 127.0.0.1, Port No. 5000.
Date and Timestamp in UTC: 2023-04-28 20:24:43.
UTA-ID = 1002026832 , 1002078961.
Total number of updates: 127.0.0.6.
Payload size for last broadcast: 124 bytes.

(base) rav_1797@Ravis-MBP cn_fprj % 
```

Fig 7: Screenshot showing the final output along with test cases


~/Documents/cn_fprj — python3 final_with_tst.py 5000 confir.config
~/Documents/cn_fprj — python3 check_update.py

```
(base) rav_1797@Ravis-MBP cn_fprj % python3 check_update.py
Enter the source ip_address
127.0.0.1
Enter the destination ip_address
127.0.0.6
Enter the cost to be update
2
Enter the ip address to which you need to send data
127.0.0.1
Sending update message: {'source': '127.0.0.1', 'destination': '127.0.0.6', 'cost': 2}
```

Fig 8: Screenshot of the user inputs for the program to send updated cost to main program

REFERENCES:

1. https://en.wikipedia.org/wiki/Bellman%E2%80%93Ford_algorithm
2. https://www.w3schools.com/python/python_dictionaries.asp
3. https://www.w3schools.com/python/python_lists.asp
4. https://www.geeksforgeeks.org/socket-programming-python/