NAME: Debadrita Roy

CLASS: BCSE-III

GROUP: A1

ASSIGNMENT NUMBER: 7

PROBLEM STATEMENT: Network, Transport and Application layer protocols

Implement any two protocols using TCP/UDP Socket as suitable.

1. BOOTP 2. FTP 3. DHCP 4. BGP 5. RIP

DEADLINE: 9th November, 2021

DATE OF SUBMISSION: 16th November, 2021

DESIGN

I have implemented FTP (File Transfer Protocol) and DHCP (Dynamic Host Configuration Protocol) using TCP socket and UDP socket respectively. The programs are written in Python 3 and I have used the socket library available for network programming in python.

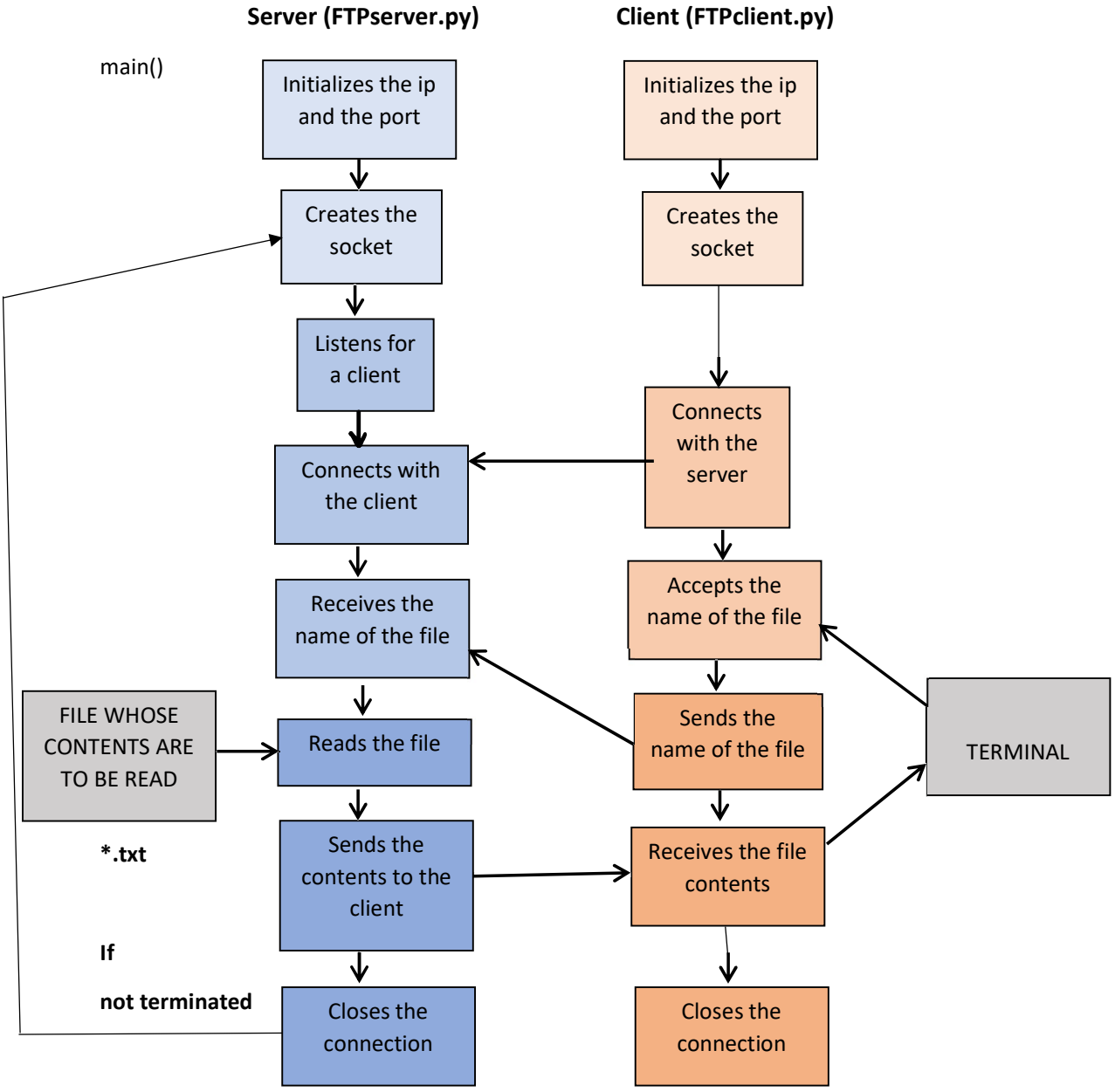
FTP: The **File Transfer Protocol** (**FTP**) is a standard communication protocol used for the transfer of computer files from a server to a client on a computer network. FTP is built on a client-server model architecture using separate control and data connections between the client and the server.

It is implemented with the help of TCP Socket. The server (**FTPserver.py**) initializes the ip and port, creates the socket and listens for clients. The client (**FTPclient.py**) sends a request for connecting with the server. The server accepts it then receives the name of the file needed by the client. The server sends the contents of the file to the client. The connection is then closed.

DHCP: The **Dynamic Host Configuration Protocol** (**DHCP**) is a network management protocol used on Internet Protocol (IP) networks for automatically assigning IP addresses and other communication parameters to devices connected to the network using a client-server architecture.

It is implemented with the help of UDP Socket. The server (**DHCPserver.py**) initializes the host, port, creates the socket and creates a list to store the addresses assigned to the clients in the network. The client (**DHCPclient.py**) sends a request to the server for getting an address, the server receives the request and assigns an address to the client. It then sends the address to the client, who accepts it. If the server receives an acceptance request from the client, it adds the address to the ARP Address Cache and sends an acknowledgement to the client.

Procedural Diagram (FTP):



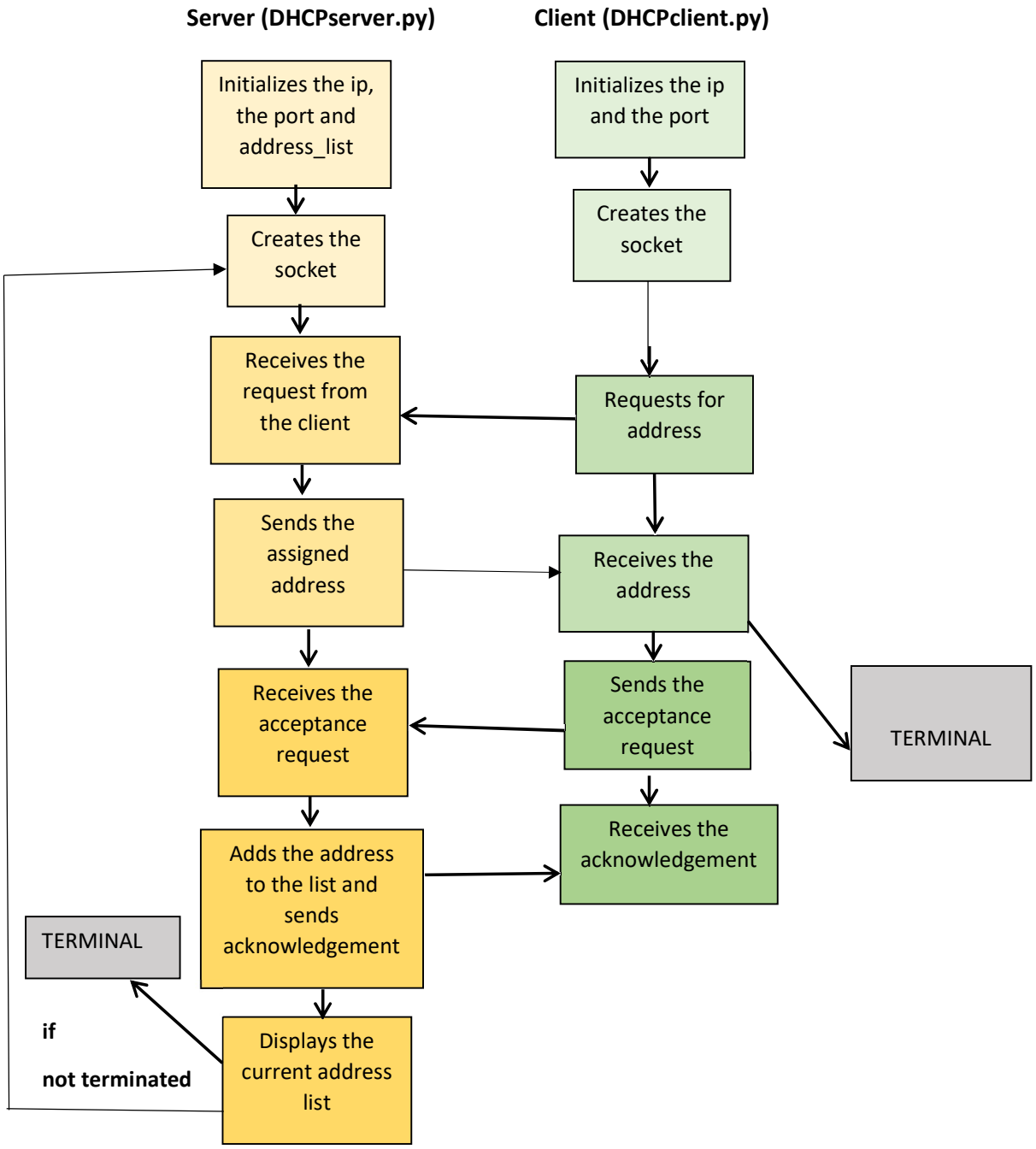
Input Format:

The name of the file to be transferred is given as user input on the client terminal. On the server terminal, we need to provide an input to determine whether the server will terminate or not.

Output Format:

The contents of the file transferred from the server to the client are displayed on the client terminal. Various stages of the program(s) are also displayed on the terminal(s) to keep track of the execution.

Procedural Diagram(DHCP):



Input Format:

No input taken

Output Format:

The address assigned to the client is displayed on the client terminal. The current values in the address list are displayed in the server terminal after a new address is added. Various stages of the program(s) are also displayed on the terminal(s) to keep track of the execution.

IMPLEMENTATION

FTPserver.py

main()

Method Description: The main() function sets up the connection between the server and the client and accomplishes the file transfer from the server to the client. The user is asked whether they want the server to terminate or not.

Code Snippet:

```
if __name__=='__main__':
    TCP_IP = '0.0.0.0'
    TCP_PORT = 2004
    BUFFER_SIZE = 1024
    while True:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        s.bind((TCP_IP, TCP_PORT))
        print('Listening for a client')
        s.listen(5)
        (conn, (ip, port)) = s.accept()
        print('Connected with a client')
        filename=conn.recv(BUFFER_SIZE).decode('utf-8')
        print('Client is requesting for file ',filename)
        f=open(filename,'r')
        data=f.read()
        conn.send(data.encode('utf-8'))
        s.close()
        ch=input('Do you wish to terminate the server process? Enter Y if so, otherwise press
any other key: ')
        if ch=='y' or ch=='Y':
            break
```

FTPclient.py

main()

Method Description: The main() function accepts the name of the file needed from the user, sends the name to the server and after receiving the contents of the file from the server, displays them on the terminal.

Code Snippet:

```
if __name__=='__main__':
    host = socket.gethostname()
    port = 2004  # port for FTP server
    BUFFER_SIZE = 1024
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((host, port))
    name=input('Enter the name of the file needed:')
```

```
    s.send(name.encode('utf-8'))
    data=s.recv(BUFFER_SIZE).decode('utf-8')
    print('The file received:')
    print(data)
    s.close()
```

DHCPserver.py

main()

Method Description: The main() function contains the code for implementing DHCP on the server side. The server assigns address to the client after generating a random IPv4 address and ensuring that it is not in the address list already. The user is asked whether they want the server to terminate or not.

Code Snippet:

```
if __name__=='__main__':
    UDP_IP=socket.gethostname()
    UDP_PORT=2014
    BUFFER_SIZE=1024
    address_list=[]
    i=1
    while True:
        s=socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
        s.bind((UDP_IP,UDP_PORT))
        print('Listening for a client')
        data,addr=s.recvfrom(BUFFER_SIZE)
        print('Received request from a client')
        add=''
        changed=True
        while changed:
            add = '192.168.0.' + str(int(random.random() * 200))
            changed=False
            for ad in address_list:
                if ad[1]==add:     # if generated address is already in the list
                    add = '192.168.0.' + str(int(random.random() * 200))  # generate new address
                    changed=True  # address has been changed, so have to check the list again

        s.sendto(str(add).encode('utf-8'),addr)
        acc,addr2=s.recvfrom(BUFFER_SIZE)
        if acc.decode('utf-8')=='Address accepted':
            address_list.append(('Client '+str(i),add))
            i=i+1
            s.sendto('Address added to the ARP Cache'.encode('utf-8'),addr)
        print('ARP Cache Contents currently:')
        for ad in address_list:
            print(ad)
        ch = input('Do you wish to terminate the server process? Enter Y if so, otherwise press
any other key: ')
```

```
if ch == 'y' or ch == 'Y':
    break
```

DHCPclient.py

main()

Method Description: The main() function sends a request to the server to get an address. After receiving the address from the server, it sends an acceptance message to the server and displays the assigned address on the terminal. It also receives and displays the acknowledgement sent by the server.

Code Snippet:

```
if __name__=='__main__':
    host=socket.gethostname()
    port=2014  # for the server
    BUFFER_SIZE=1024
    s=socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
    s.sendto('Request for IP Address'.encode('utf-8'),(host,port))
    data,addr=s.recvfrom(BUFFER_SIZE)
    print('Have been assigned address ',data.decode('utf-8'))
    s.sendto('Address accepted'.encode('utf-8'),(host,port))
    ack,addr2=s.recvfrom(BUFFER_SIZE)
    print(ack.decode('utf-8'))
```

TEST CASES

Sample Test 1: To check the working of the FTP Server and client

CONTENTS OF test.txt

Debadrita Roy BCSE-III A1 001910501025

CONTENTS OF testfile.txt

hello..this is a file used to check the working of the ftp protocol
if this can be seen in the client terminal, then it is obviously working efficiently

SERVER TERMINAL

```
(venv) C:\Users\USER19\PycharmProjects\python_assignments\NetworkLab>python FTPserver.py
Listening for a client
Connected with a client
Client is requesting for file  test.txt
Do you wish to terminate the server process? Enter Y if so, otherwise press any other key: h
Listening for a client
Connected with a client
Client is requesting for file  testfile.txt
Do you wish to terminate the server process? Enter Y if so, otherwise press any other key: y
```

CLIENT 1 TERMINAL

```
(venv) C:\Users\USER19\PycharmProjects\python_assignments\NetworkLab>python FTPclient.py
Enter the name of the file needed:test.txt
The file received:
Debadrita Roy BCSE-III A1 001910501025
```

CLIENT 2 TERMINAL

```
(venv) C:\Users\USER19\PycharmProjects\python_assignments\NetworkLab>python FTPclient.py
Enter the name of the file needed:testfile.txt
The file received:
hello..this is a file used to check the working of the ftp protocol
if this can be seen in the client terminal, then it is obviously working efficiently
```

Sample Test 2: To check whether the addresses are appended correctly to the list in DHCP

SERVER TERMINAL

```
C:\Users\USER19\PycharmProjects\python_assignments\venv\Scripts\python.exe C:/Users/USER19/Pycha
Listening for a client
Received request from a client
ARP Cache Contents currently:
('Client 1', '192.168.0.46')
Do you wish to terminate the server process? Enter Y if so, otherwise press any other key: n
Listening for a client
Received request from a client
ARP Cache Contents currently:
('Client 1', '192.168.0.46')
('Client 2', '192.168.0.65')
Do you wish to terminate the server process? Enter Y if so, otherwise press any other key: y
```

CLIENT TERMINAL

```
C:\Users\USER19\PycharmProjects\python_assig
Have been assigned address  192.168.0.65
Address added to the ARP Cache
```

Sample Test 3: To check the working of the DHCP server and client

SERVER TERMINAL

```
C:\Users\USER19\PycharmProjects\python_assignments\venv\Scripts\python.exe C:/Users/USER19/Pycl
Listening for a client
Received request from a client
ARP Cache Contents currently:
('Client 1', '192.168.0.65')
Do you wish to terminate the server process? Enter Y if so, otherwise press any other key: n
Listening for a client
Received request from a client
ARP Cache Contents currently:
('Client 1', '192.168.0.65')
('Client 2', '192.168.0.86')
Do you wish to terminate the server process? Enter Y if so, otherwise press any other key: y
```

CLIENT 1 TERMINAL

```
C:\Users\USER19\PycharmProjects\python_assi
Have been assigned address  192.168.0.65
Address added to the ARP Cache
```

CLIENT 2 TERMINAL

```
C:\Users\USER19\PycharmProjects\python_assignments\venv\Scripts\python.exe
Have been assigned address  192.168.0.86
Address added to the ARP Cache
```

RESULTS

The file was successfully transferred using FTP. The communication between the FTP server and the client was done using TCP socket. The DHCP server successfully assigned address to the DHCP client and the passing of messages between the server and the client was done properly using UDP socket.

ANALYSIS

The TCP Socket is used for FTP as TCP is a reliable, lossless, connection-oriented protocol which provides good flow control mechanism. Thus, TCP allows for reliable, sequential transfer and the order of bits in the file is not changed or lost. TCP provides error-checking and guarantees delivery of data. UDP does not have any flow control mechanism nor does it provide guaranteed delivery. With UDP, the packets arrive in a continuous stream or they are dropped.

The UDP Socket is used for DHCP as UDP is connectionless and has low overhead. When the client sends the request, it does not have an IP address yet and TCP requires both the ends to have unique IP addresses as it is a connection-oriented protocol. By the time the client receives an IP Address, the work of DHCP is already complete. Also, the data in a DHCP message is too small for it to be worth the overhead in TCP. UDP is faster than TCP and as there is no real need for flow control or error checking, UDP is better-suited for DHCP.

If we need accuracy and reliability, then TCP is used, whereas UDP is used when speed is more important and reliability is not much of a factor as compared to speed.

COMMENTS

The assignment enabled us to use TCP and UDP Sockets to implement two protocols which we have read about in the theory class. It also allowed us to see the differences between TCP and UDP and in which cases they should be used. We also got the chance to observe how the protocols—FTP and DHCP act in a simulated environment, which enabled us to get an idea of how they work in the real-world. Overall, the design and implementation were easy.