COMPUTER NETWORKS LAB

NAME:	ARITRA DATTA
ROLL NO:	002010501054
CLASS:	BCSE – III
GROUP:	A2
ASSIGNMENT:	8
DEADLINE:	11th November 2022

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

1. FTP

2. DNS

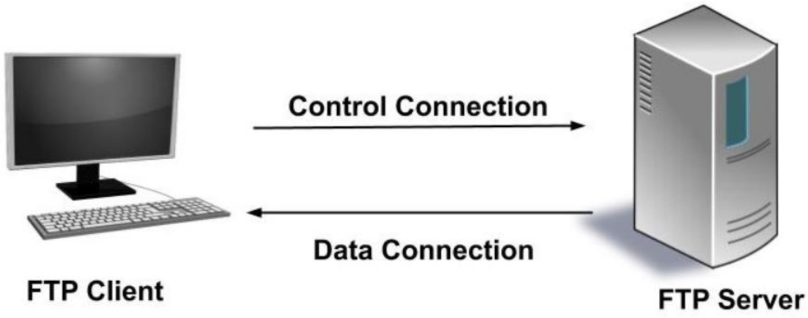
3. Telnet

Date of Submission: 7th November, 2022

DESIGN

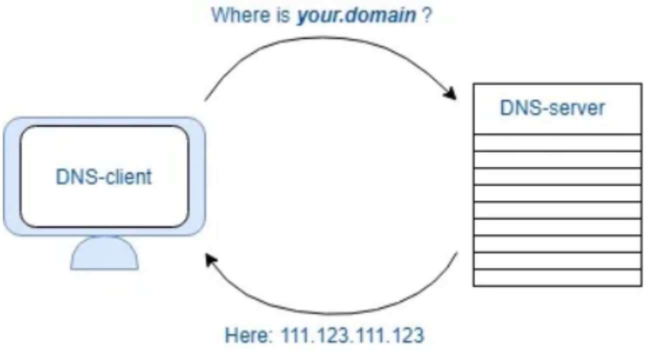
FTP (File Transfer Protocol)

FTP is implemented using TCP sockets here and **socket.SOCK_STREAM** has been used accordingly. Whenever a client connects to the FTP Server, it requests a file by sending the filename. The FTP server reads the file and transfers the contents of the file to the client. The Client also has the options to Upload a file to the server, and to List the files with the server.



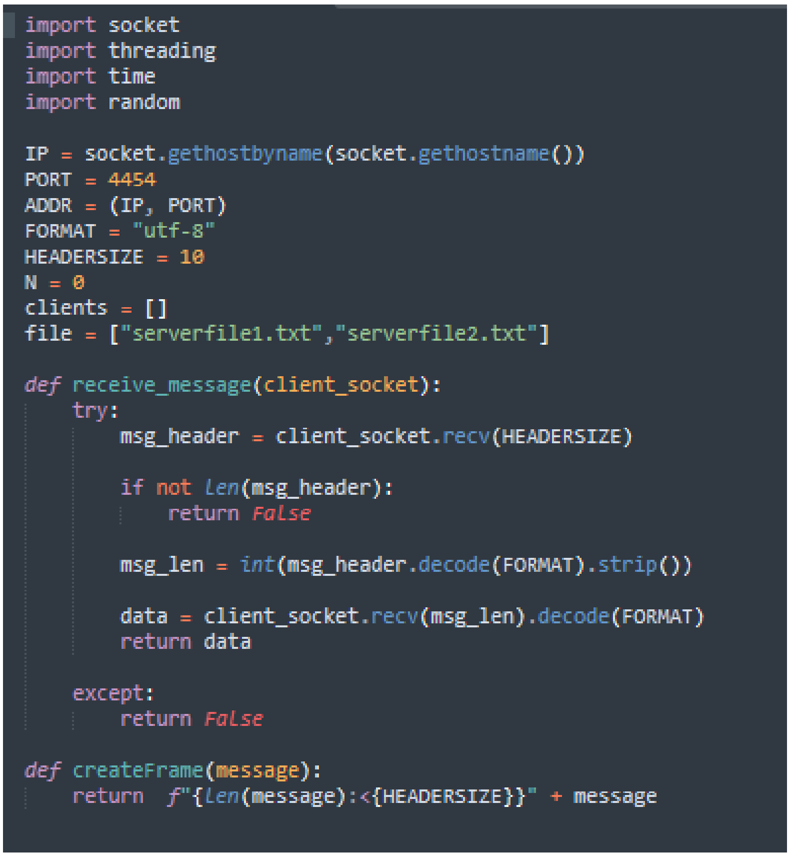
DNS (Domain Name System)

DNS is implemented using TCP sockets here and **socket.SOCK_STREAM** has been used accordingly. DNS is a hostname for IP address translation service. The client sends a domain name to the DNS server, which then responds with the corresponding IP address of the server accordingly.

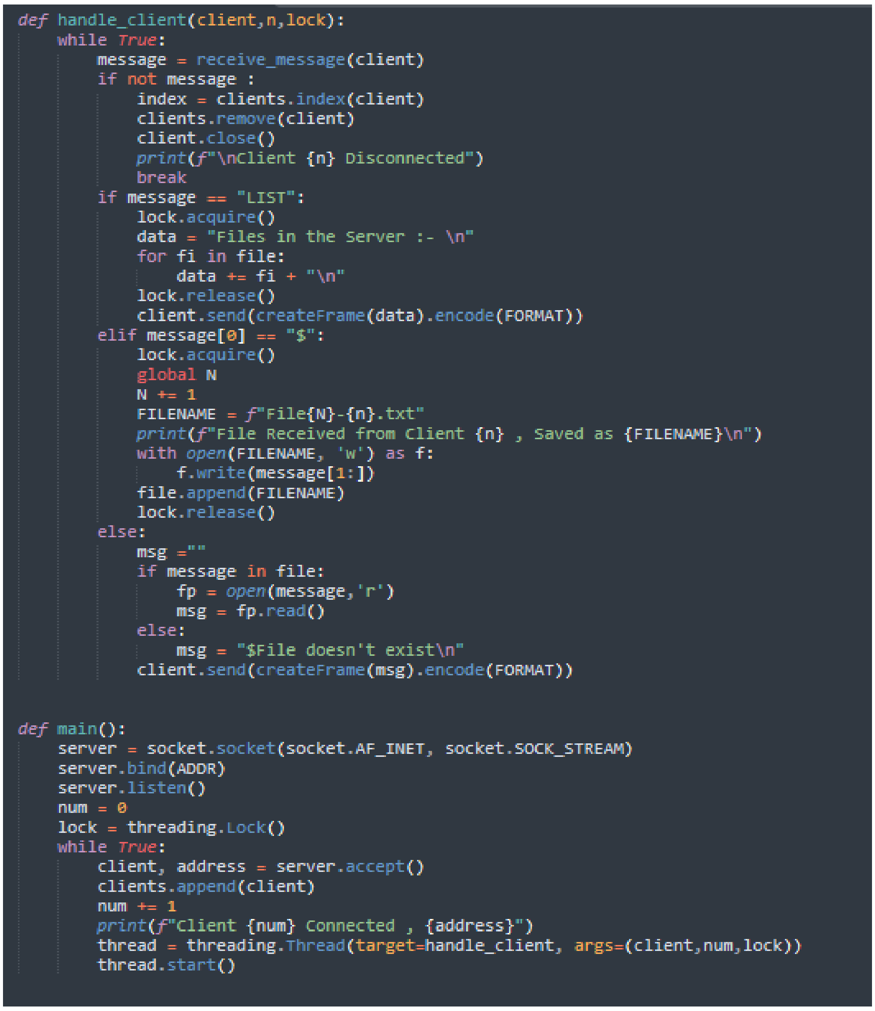


CODE

FTP_Server.py

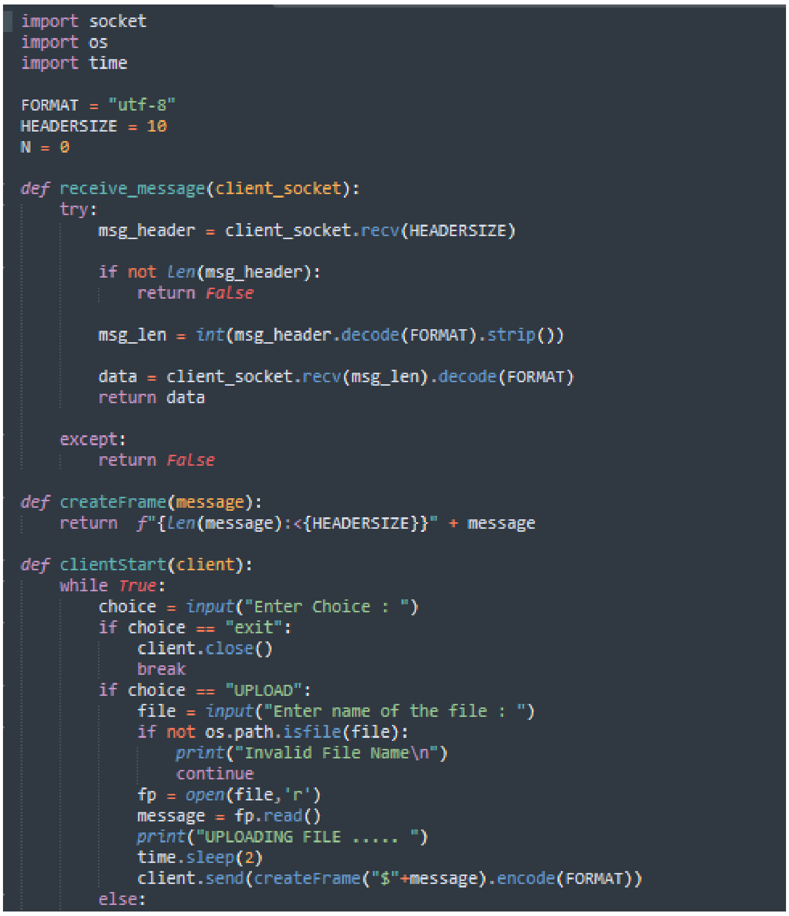


```python
import socket
import threading
import time
import random

IP = socket.gethostbyname(socket.gethostname())
PORT = 4454
ADDR = (IP, PORT)
FORMAT = "utf-8"
HEADERSIZE = 10
N = 0
clients = []
file = ["serverfile1.txt","serverfile2.txt"]

def receive_message(client_socket):
    try:
        msg_header = client_socket.recv(HEADERSIZE)

        if not len(msg_header):
            return False

        msg_len = int(msg_header.decode(FORMAT).strip())

        data = client_socket.recv(msg_len).decode(FORMAT)
        return data

    except:
        return False

def createFrame(message):
    return  f"{len(message):<{HEADERSIZE}}" + message
```

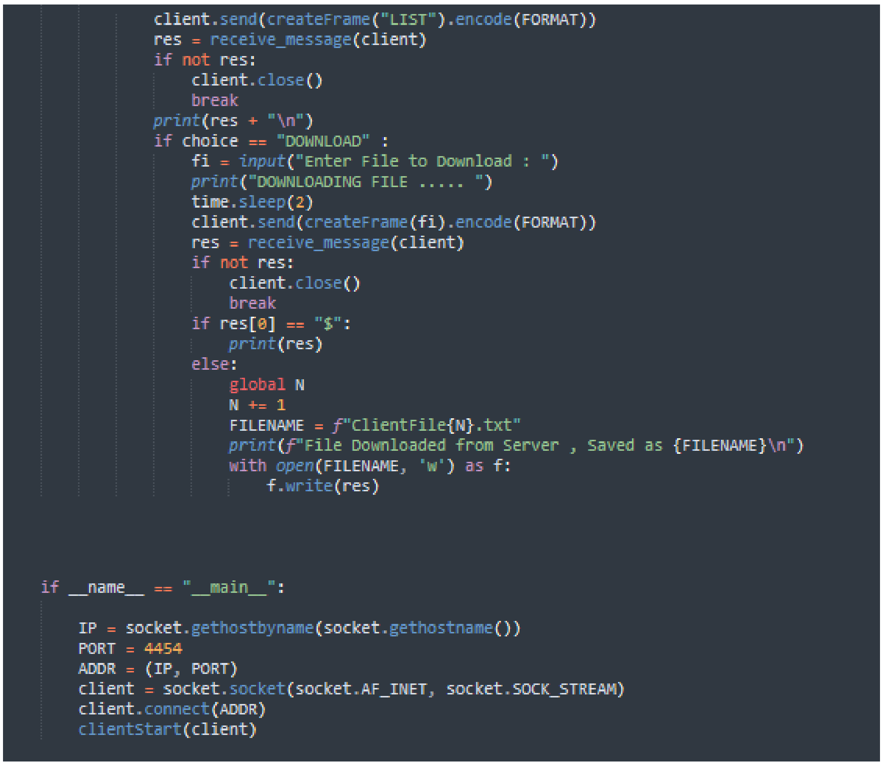


```python
def handle_client(client,n,lock):
    while True:
        message = receive_message(client)
        if not message :
            index = clients.index(client)
            clients.remove(client)
            client.close()
            print(f"\nClient {n} Disconnected")
            break
        if message == "LIST":
            lock.acquire()
            data = "Files in the Server :- \n"
            for fi in file:
                data += fi + "\n"
            lock.release()
            client.send(createFrame(data).encode(FORMAT))
        elif message[0] == "$":
            lock.acquire()
            global N
            N += 1
            FILENAME  = f"File{N}-{n}.txt"
            print(f"File Received from Client {n} , Saved as {FILENAME}\n")
            with open(FILENAME, 'w') as f:
                f.write(message[1:])
            file.append(FILENAME)
            lock.release()
        else:
            msg =""
            if message in file:
                fp = open(message,'r')
                msg = fp.read()
            else:
                msg = "$File doesn't exist\n"
            client.send(createFrame(msg).encode(FORMAT))


def main():
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind(ADDR)
    server.listen()
    num = 0
    lock = threading.Lock()
    while True:
        client, address = server.accept()
        clients.append(client)
        num += 1
        print(f"Client {num} Connected , {address}")
        thread = threading.Thread(target=handle_client, args=(client,num,lock))
        thread.start()
```

FTP_Client.py

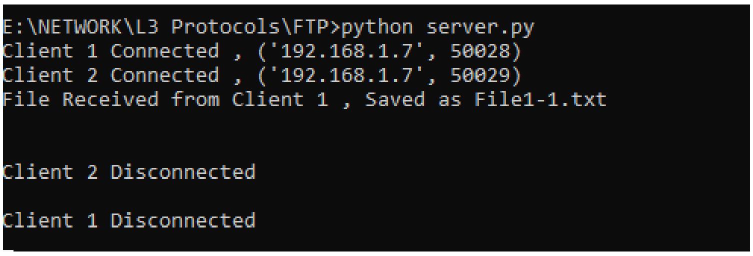


```python
import socket
import os
import time

FORMAT = "utf-8"
HEADERSIZE = 10
N = 0

def receive_message(client_socket):
    try:
        msg_header = client_socket.recv(HEADERSIZE)

        if not len(msg_header):
            return False

        msg_len = int(msg_header.decode(FORMAT).strip())

        data = client_socket.recv(msg_len).decode(FORMAT)
        return data

    except:
        return False

def createFrame(message):
    return  f"{len(message):<{HEADERSIZE}}" + message

def clientStart(client):
    while True:
        choice = input("Enter Choice : ")
        if choice == "exit":
            client.close()
            break
        if choice == "UPLOAD":
            file = input("Enter name of the file : ")
            if not os.path.isfile(file):
                print("Invalid File Name\n")
                continue
            fp = open(file,'r')
            message = fp.read()
            print("UPLOADING FILE ..... ")
            time.sleep(2)
            client.send(createFrame("$"+message).encode(FORMAT))
        else:
```



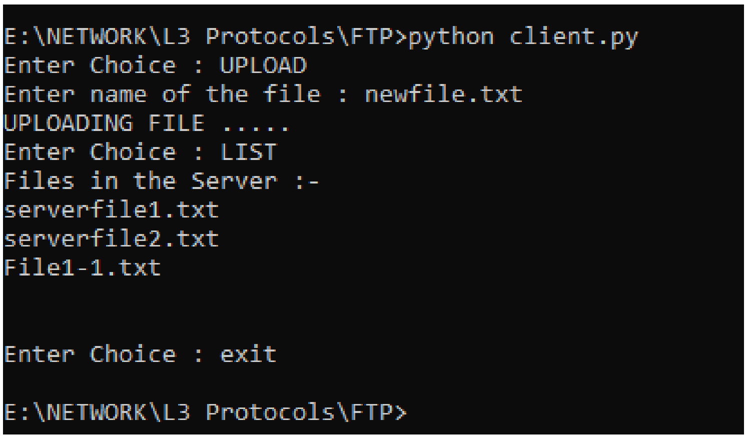
```python
                client.send(createFrame("LIST").encode(FORMAT))
                res = receive_message(client)
                if not res:
                    client.close()
                    break
                print(res + "\n")
                if choice == "DOWNLOAD" :
                    fi = input("Enter File to Download : ")
                    print("DOWNLOADING FILE ..... ")
                    time.sleep(2)
                    client.send(createFrame(fi).encode(FORMAT))
                    res = receive_message(client)
                    if not res:
                        client.close()
                        break
                    if res[0] == "$":
                        print(res)
                    else:
                        global N
                        N += 1
                        FILENAME = f"ClientFile{N}.txt"
                        print(f"File Downloaded from Server , Saved as {FILENAME}\n")
                        with open(FILENAME, 'w') as f:
                            f.write(res)



if __name__ == "__main__":

    IP = socket.gethostbyname(socket.gethostname())
    PORT = 4454
    ADDR = (IP, PORT)
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect(ADDR)
    clientStart(client)
```

OUTPUT



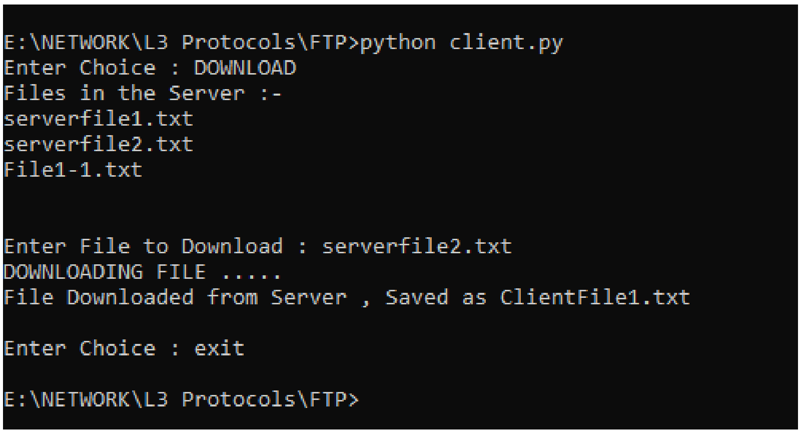
```
E:\NETWORK\L3 Protocols\FTP>python server.py
Client 1 Connected , ('192.168.1.7', 50028)
Client 2 Connected , ('192.168.1.7', 50029)
File Received from Client 1 , Saved as File1-1.txt


Client 2 Disconnected

Client 1 Disconnected
```

FTP SERVER



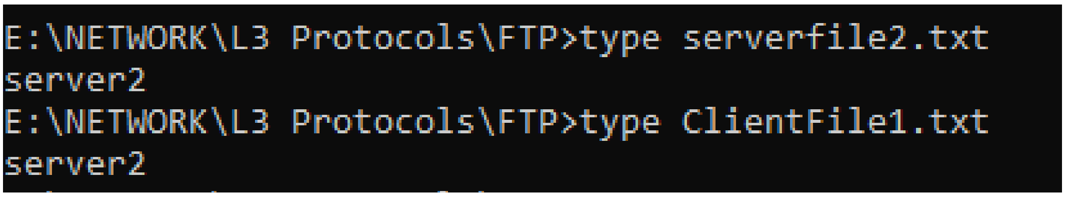
```
E:\NETWORK\L3 Protocols\FTP>python client.py
Enter Choice : UPLOAD
Enter name of the file : newfile.txt
UPLOADING FILE .....
Enter Choice : LIST
Files in the Server :-
serverfile1.txt
serverfile2.txt
File1-1.txt


Enter Choice : exit

E:\NETWORK\L3 Protocols\FTP>
```

Client 1



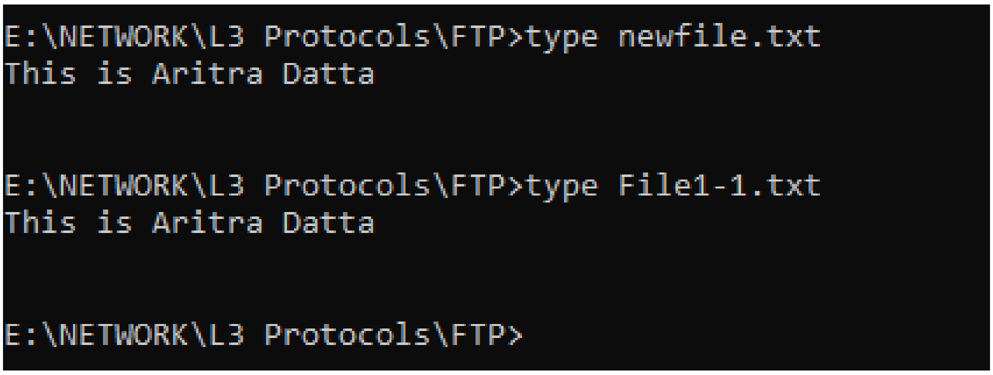
```
E:\NETWORK\L3 Protocols\FTP>python client.py
Enter Choice : DOWNLOAD
Files in the Server :-
serverfile1.txt
serverfile2.txt
File1-1.txt


Enter File to Download : serverfile2.txt
DOWNLOADING FILE .....
File Downloaded from Server , Saved as ClientFile1.txt

Enter Choice : exit

E:\NETWORK\L3 Protocols\FTP>
```

Client 2



```
E:\NETWORK\L3 Protocols\FTP>type serverfile2.txt
server2
E:\NETWORK\L3 Protocols\FTP>type ClientFile1.txt
server2
```

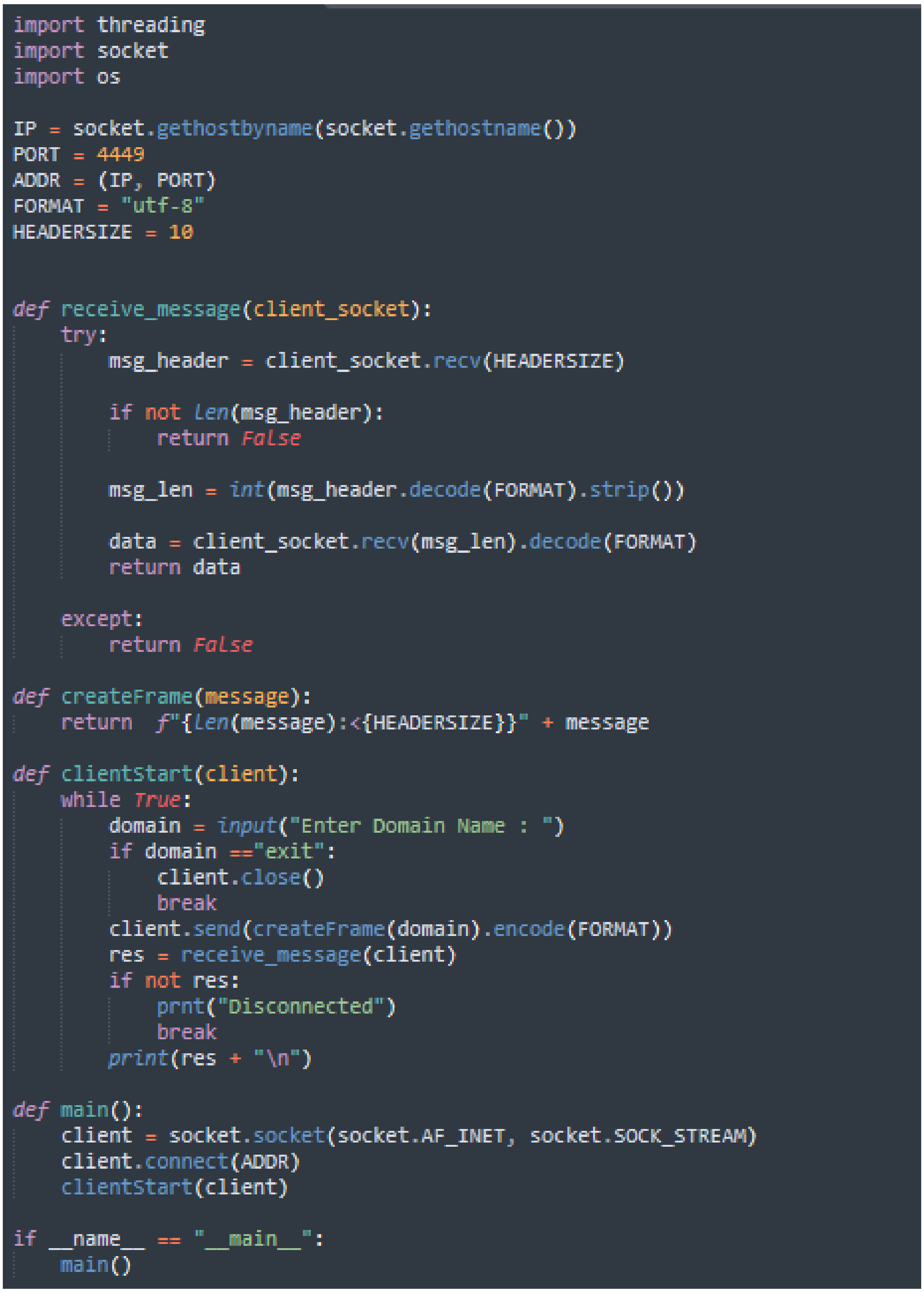
DOWNLOAD STATUS



```
E:\NETWORK\L3 Protocols\FTP>type newfile.txt
This is Aritra Datta


E:\NETWORK\L3 Protocols\FTP>type File1-1.txt
This is Aritra Datta


E:\NETWORK\L3 Protocols\FTP>
```

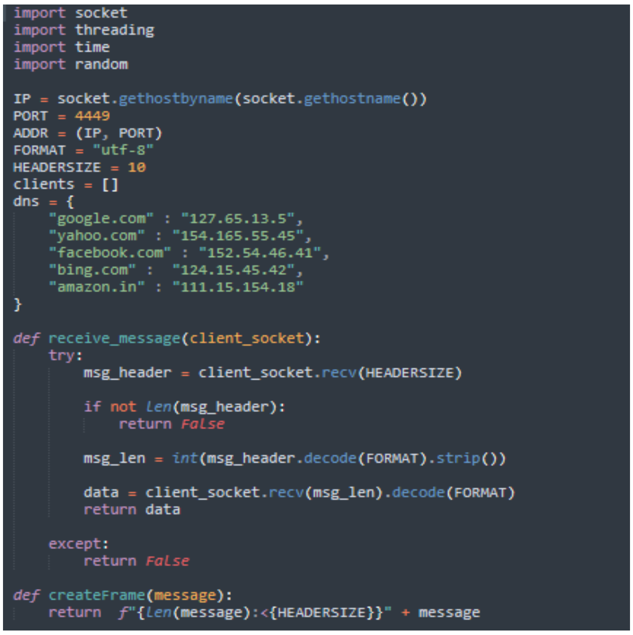
UPLOAD STATUS

DNS_Client.py

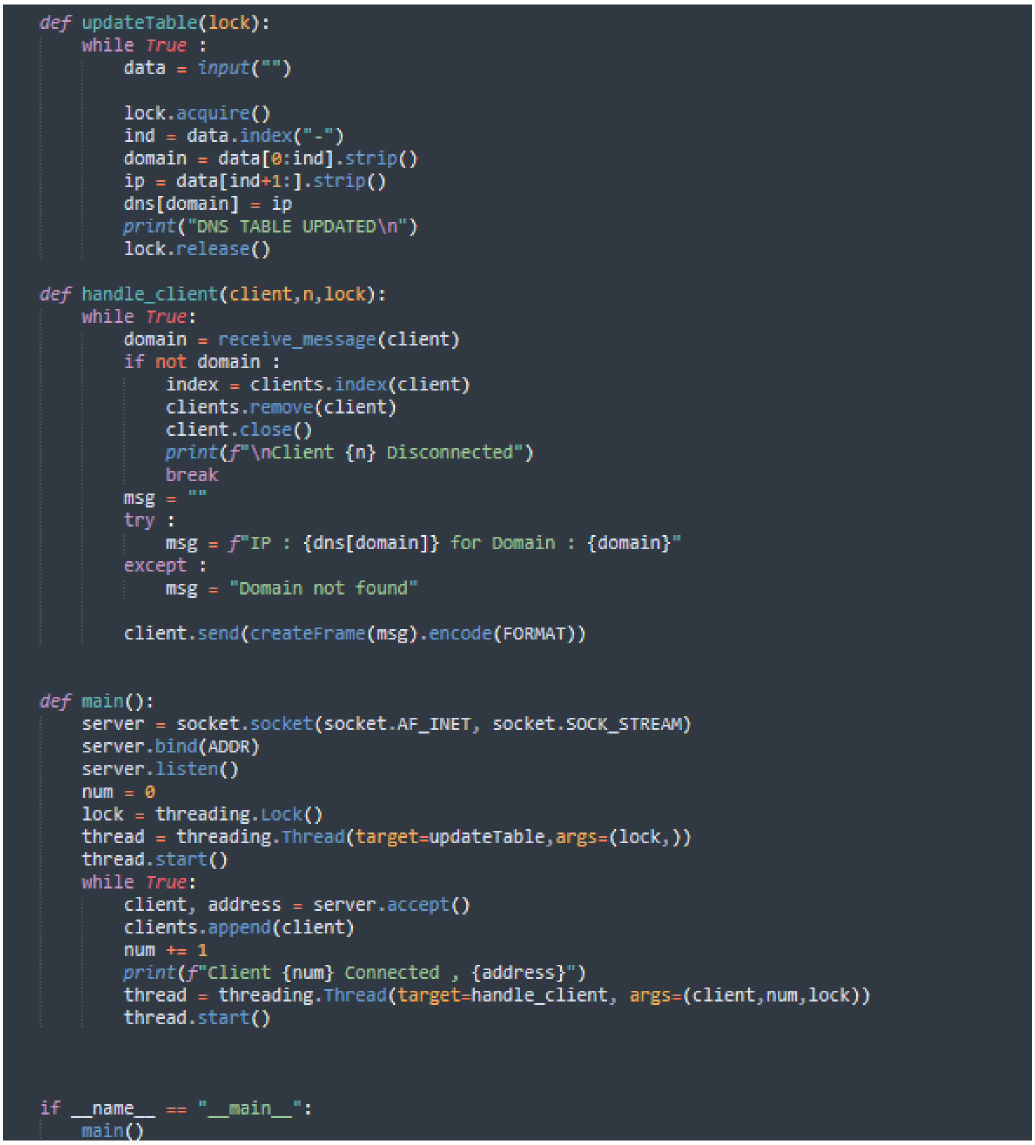


```
import threading
import socket
import os

IP = socket.gethostbyname(socket.gethostname())
PORT = 4449
ADDR = (IP, PORT)
FORMAT = "utf-8"
HEADERSIZE = 10


def receive_message(client_socket):
    try:
        msg_header = client_socket.recv(HEADERSIZE)

        if not len(msg_header):
            return False

        msg_len = int(msg_header.decode(FORMAT).strip())

        data = client_socket.recv(msg_len).decode(FORMAT)
        return data

    except:
        return False

def createFrame(message):
    return  f"{len(message):<{HEADERSIZE}}" + message

def clientStart(client):
    while True:
        domain = input("Enter Domain Name : ")
        if domain =="exit":
            client.close()
            break
        client.send(createFrame(domain).encode(FORMAT))
        res = receive_message(client)
        if not res:
            prnt("Disconnected")
            break
        print(res + "\n")

def main():
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect(ADDR)
    clientStart(client)

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

DNS_Server.py

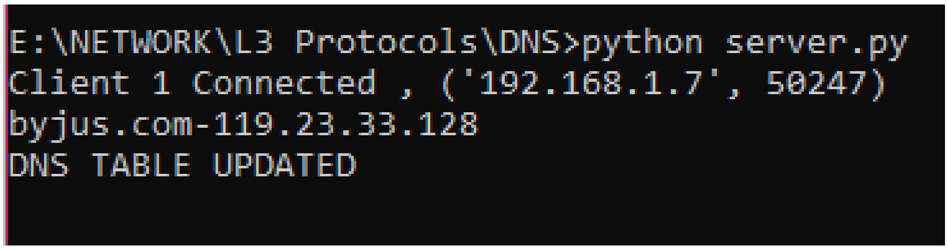


```
import socket
import threading
import time
import random

IP = socket.gethostbyname(socket.gethostname())
PORT = 4449
ADDR = (IP, PORT)
FORMAT = "utf-8"
HEADERSIZE = 10
clients = []
dns = {
    "google.com" : "127.65.13.5",
    "yahoo.com" : "154.165.55.45",
    "facebook.com" : "152.54.46.41",
    "bing.com" :  "124.15.45.42",
    "amazon.in" : "111.15.154.18"
}

def receive_message(client_socket):
    try:
        msg_header = client_socket.recv(HEADERSIZE)

        if not len(msg_header):
            return False

        msg_len = int(msg_header.decode(FORMAT).strip())

        data = client_socket.recv(msg_len).decode(FORMAT)
        return data

    except:
        return False

def createFrame(message):
    return  f"{len(message):<{HEADERSIZE}}" + message
```



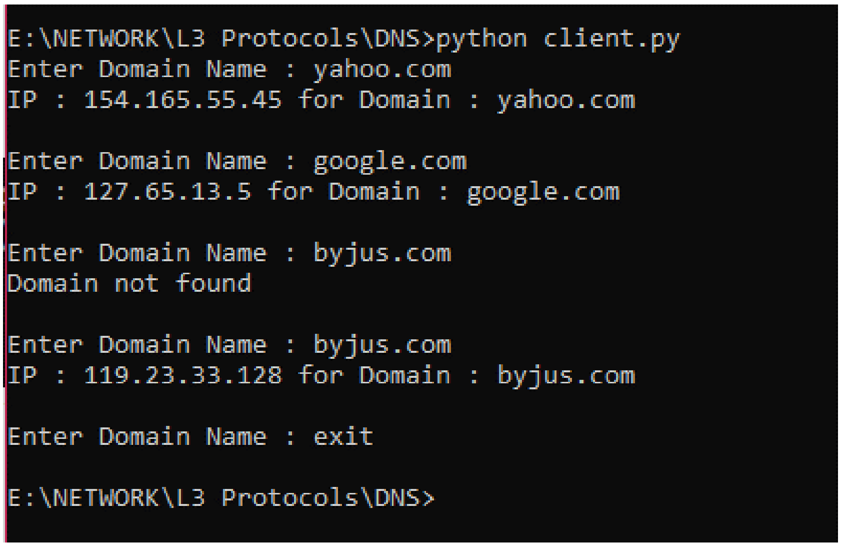
```
def updateTable(lock):
    while True :
        data = input("")

        lock.acquire()
        ind = data.index("-")
        domain = data[0:ind].strip()
        ip = data[ind+1:].strip()
        dns[domain] = ip
        print("DNS TABLE UPDATED\n")
        lock.release()

def handle_client(client,n,lock):
    while True:
        domain = receive_message(client)
        if not domain :
            index = clients.index(client)
            clients.remove(client)
            client.close()
            print(f"\nClient {n} Disconnected")
            break
        msg = ""
        try :
            msg = f"IP : {dns[domain]} for Domain : {domain}"
        except :
            msg = "Domain not found"

        client.send(createFrame(msg).encode(FORMAT))


def main():
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind(ADDR)
    server.listen()
    num = 0
    lock = threading.Lock()
    thread = threading.Thread(target=updateTable,args=(lock,))
    thread.start()
    while True:
        client, address = server.accept()
        clients.append(client)
        num += 1
        print(f"Client {num} Connected , {address}")
        thread = threading.Thread(target=handle_client, args=(client,num,lock))
        thread.start()


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

OUTPUT



```
E:\NETWORK\L3 Protocols\DNS>python server.py
Client 1 Connected , ('192.168.1.7', 50247)
byjus.com-119.23.33.128
DNS TABLE UPDATED
```

DNS SERVER



```
E:\NETWORK\L3 Protocols\DNS>python client.py
Enter Domain Name : yahoo.com
IP : 154.165.55.45 for Domain : yahoo.com

Enter Domain Name : google.com
IP : 127.65.13.5 for Domain : google.com

Enter Domain Name : byjus.com
Domain not found

Enter Domain Name : byjus.com
IP : 119.23.33.128 for Domain : byjus.com

Enter Domain Name : exit

E:\NETWORK\L3 Protocols\DNS>
```

Client