PES University, Bengaluru

VI Sem

Session: January – May, 2020

COMPUTER NETWORK PROJECT UE17EC351

"MULTIMEDIA FILE SHARING SYSTEM USING CLIENT-SERVER ARCHITECTURE"

Submitted by

6th B

PRATHIMA CHOWDARY(PES1201700776)

V SAISRI(PES1201701763)

PES
UNIVERSITY

PROBLEM STATEMENT:

TO CREATE A CLIENT-SERVER CONNECTION ESTABLISHMENT WHICH CAN BE USED TO SHARE MULTIMEDIA FILES BETWEEN MULTIPLE CLIENTS

INTRODUCTION:

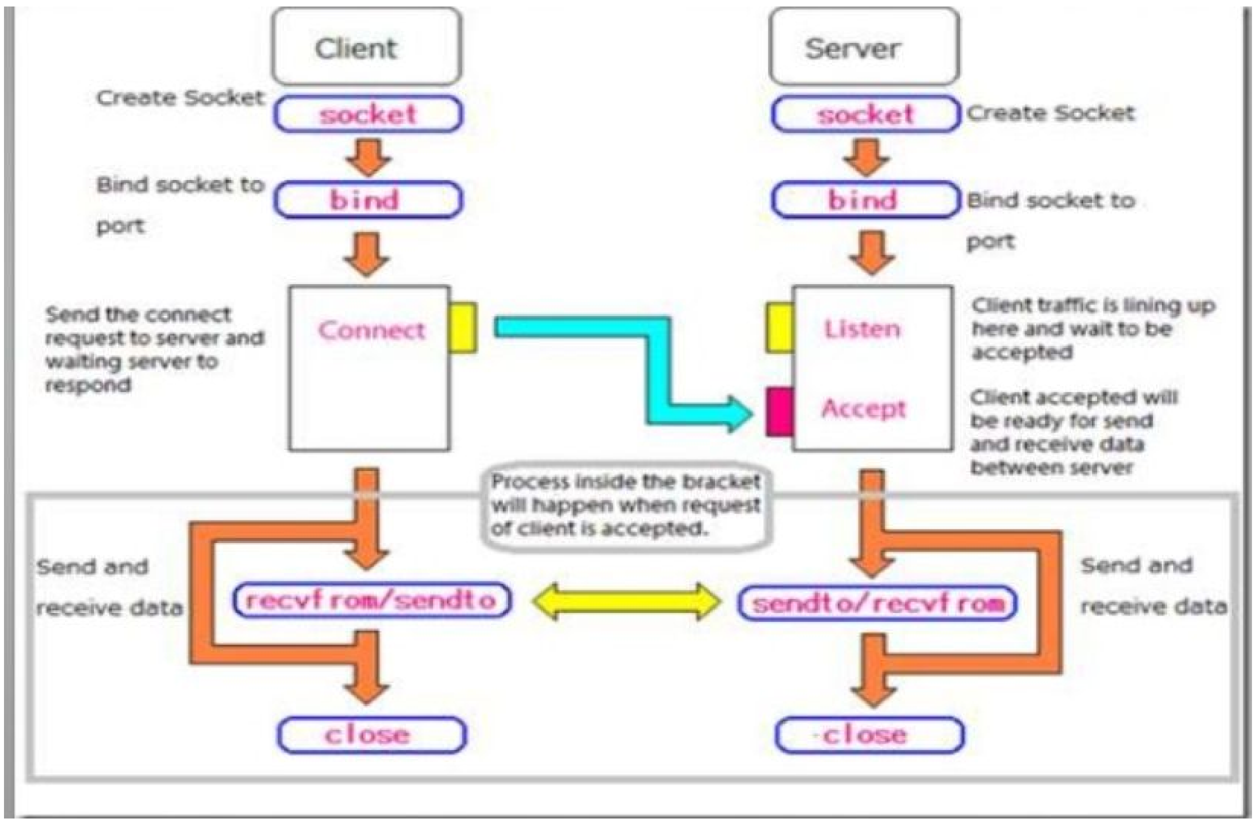
Chatroom is a feature generally used in various applications which allows multiple users to communicate and share data with each other while within a gaming application or any other website. This feature allows users to send/receive messages, transfer files such as audio, video etc., among users.

Our project implements this feature of exchange of messages and transfer of files among users in a chatroom implemented with Tkinter GUI.

Firstly, a chat server is required and hence created, that responds to the requests sent by the clients wanting to communicate. This was done using socket programming (TCP sockets were used) and the concept of multithreading. Clients are served by accepting new connections, broadcasting messages and handling clients.

Next important thing is a client. Client is provided with a GUI for communication. A client would wish to send/receive messages or transfer files or both. These functions are implemented using Tkinter modules.

BLOCK DIAGRAM:


Client
Server
Create Socket
socket
socket
Create Socket
Bind socket to port
bind
bind
Bind socket to port
Send the connect request to server and waiting server to respond
Connect
Listen
Accept
Client traffic is lining up here and wait to be accepted
Client accepted will be ready for send and receive data between server
Process inside the bracket will happen when request of client is accepted.
Send and receive data
recvfrom/sendto
sendto/recvfrom
Send and receive data
close
close

CONFIGURATION:

Modules Developed:

- Accept incoming connections
- Handle individual clients
- Message broadcasting
- Message transfer
- File transfer

Network Concepts used:

- TCP Socket programming

New concepts Learnt:

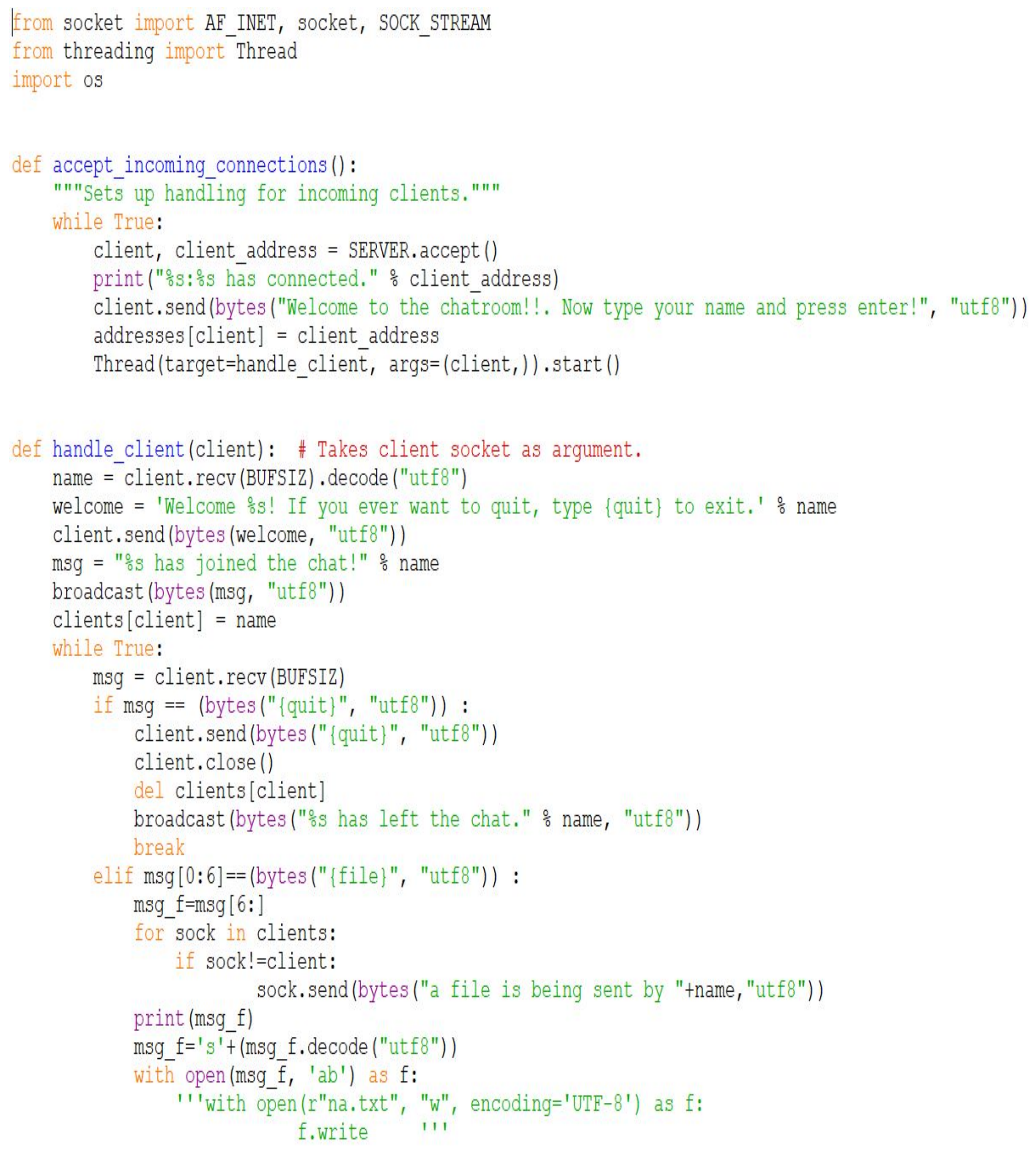
- Multithreading programming
- Tkinter GUI usage and application

Execution :

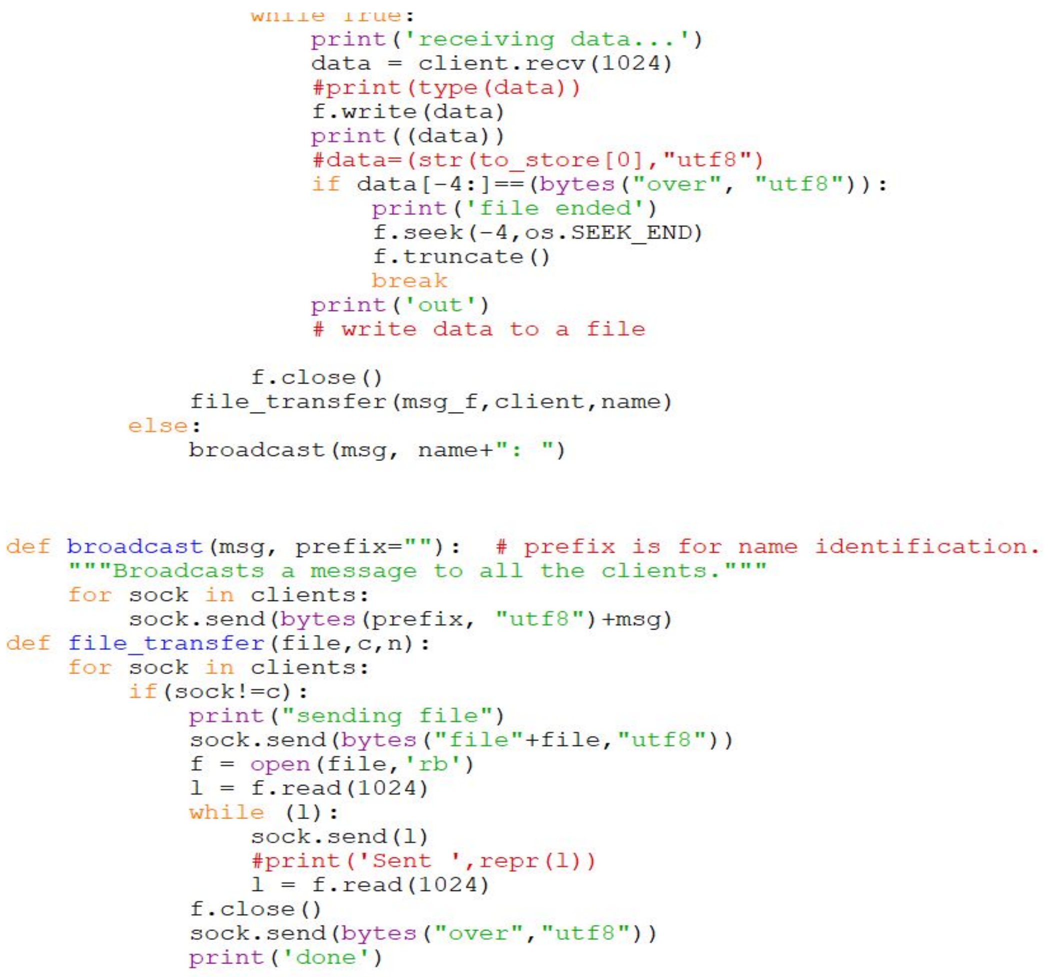
By changing the host address to the required address,we will be able to share files of any device

CODES:

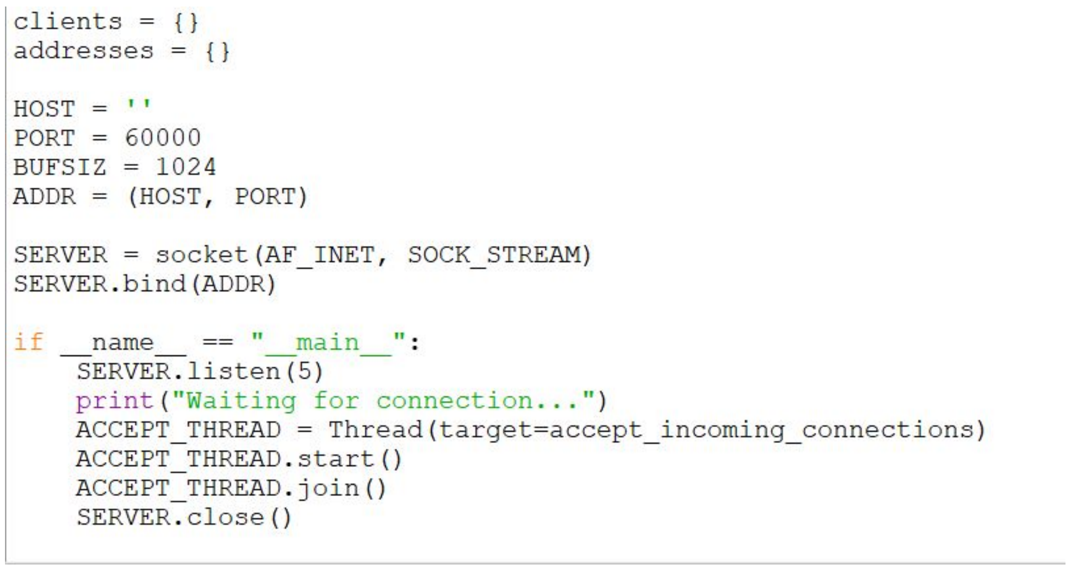
SERVER CODE:



```python
from socket import AF_INET, socket, SOCK_STREAM
from threading import Thread
import os


def accept_incoming_connections():
    """Sets up handling for incoming clients."""
    while True:
        client, client_address = SERVER.accept()
        print("%s:%s has connected." % client_address)
        client.send(bytes("Welcome to the chatroom!!. Now type your name and press enter!", "utf8"))
        addresses[client] = client_address
        Thread(target=handle_client, args=(client,)).start()


def handle_client(client):  # Takes client socket as argument.
    name = client.recv(BUFSIZ).decode("utf8")
    welcome = 'Welcome %s! If you ever want to quit, type {quit} to exit.' % name
    client.send(bytes(welcome, "utf8"))
    msg = "%s has joined the chat!" % name
    broadcast(bytes(msg, "utf8"))
    clients[client] = name
    while True:
        msg = client.recv(BUFSIZ)
        if msg == (bytes("{quit}", "utf8")) :
            client.send(bytes("{quit}", "utf8"))
            client.close()
            del clients[client]
            broadcast(bytes("%s has left the chat." % name, "utf8"))
            break
        elif msg[0:6]==(bytes("{file}", "utf8")) :
            msg_f=msg[6:]
            for sock in clients:
                if sock!=client:
                        sock.send(bytes("a file is being sent by "+name,"utf8"))
            print(msg_f)
            msg_f='s'+(msg_f.decode("utf8"))
            with open(msg_f, 'ab') as f:
                '''with open(r"na.txt", "w", encoding='UTF-8') as f:
                        f.write     '''
```

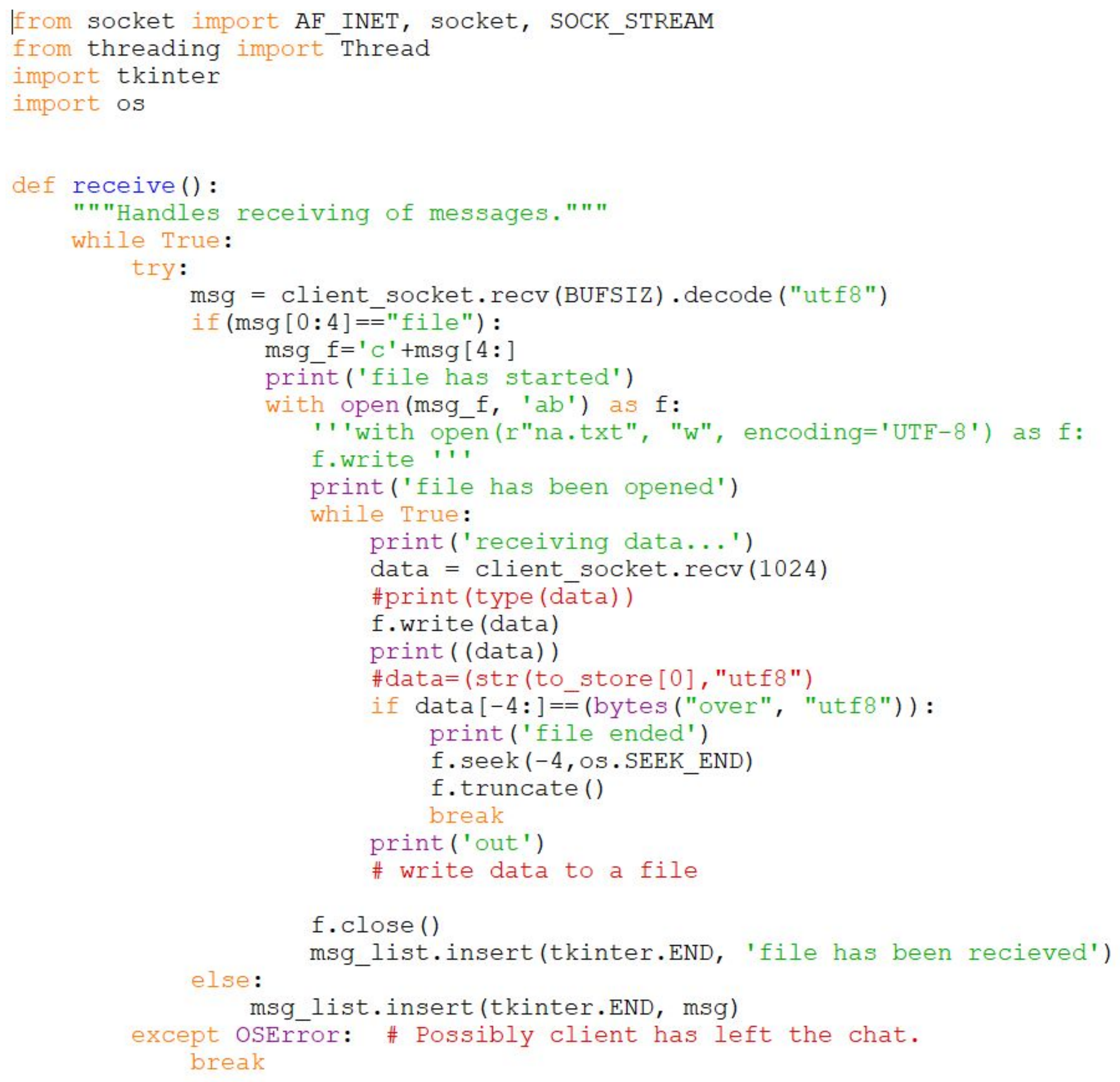


```
                while True:
                    print('receiving data...')
                    data = client.recv(1024)
                    #print(type(data))
                    f.write(data)
                    print((data))
                    #data=(str(to_store[0],"utf8")
                    if data[-4:]==(bytes("over", "utf8")):
                        print('file ended')
                        f.seek(-4,os.SEEK_END)
                        f.truncate()
                        break
                    print('out')
                    # write data to a file

                f.close()
            file_transfer(msg_f,client,name)
        else:
            broadcast(msg, name+": ")


def broadcast(msg, prefix=""):  # prefix is for name identification.
    """Broadcasts a message to all the clients."""
    for sock in clients:
        sock.send(bytes(prefix, "utf8")+msg)
def file_transfer(file,c,n):
    for sock in clients:
        if(sock!=c):
            print("sending file")
            sock.send(bytes("file"+file,"utf8"))
            f = open(file,'rb')
            l = f.read(1024)
            while (l):
                sock.send(l)
                #print('Sent ',repr(l))
                l = f.read(1024)
            f.close()
            sock.send(bytes("over","utf8"))
            print('done')
```

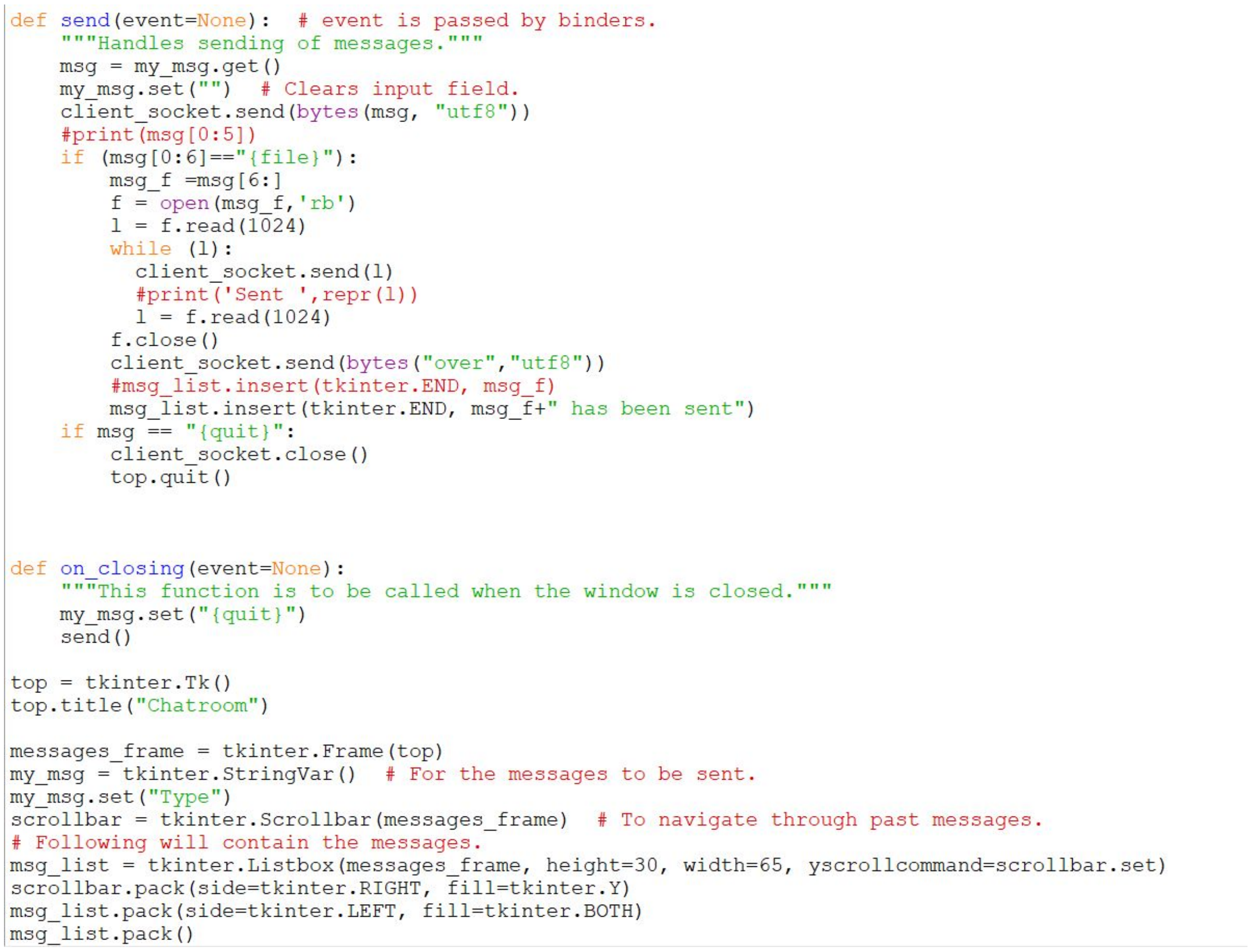


```
clients = {}
addresses = {}

HOST = ''
PORT = 60000
BUFSIZ = 1024
ADDR = (HOST, PORT)

SERVER = socket(AF_INET, SOCK_STREAM)
SERVER.bind(ADDR)

if __name__ == "__main__":
    SERVER.listen(5)
    print("Waiting for connection...")
    ACCEPT_THREAD = Thread(target=accept_incoming_connections)
    ACCEPT_THREAD.start()
    ACCEPT_THREAD.join()
    SERVER.close()
```

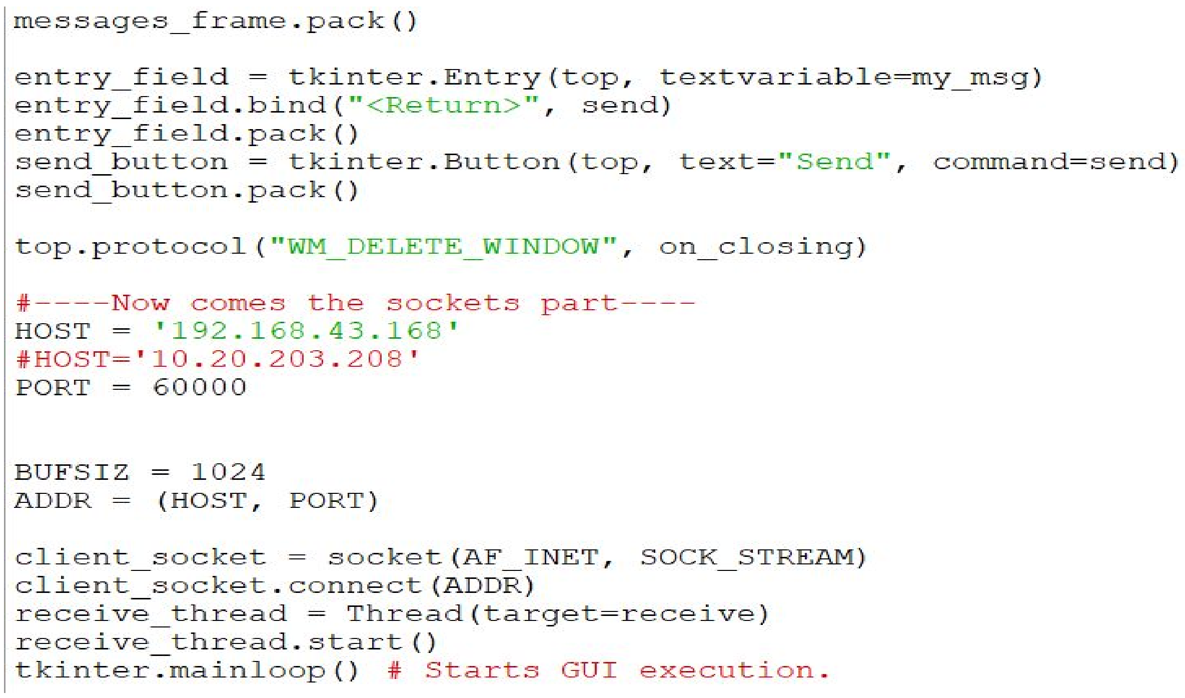
CLIENT CODE:



```
from socket import AF_INET, socket, SOCK_STREAM
from threading import Thread
import tkinter
import os


def receive():
    """Handles receiving of messages."""
    while True:
        try:
            msg = client_socket.recv(BUFSIZ).decode("utf8")
            if(msg[0:4]=="file"):
                 msg_f='c'+msg[4:]
                 print('file has started')
                 with open(msg_f, 'ab') as f:
                    '''with open(r"na.txt", "w", encoding='UTF-8') as f:
                    f.write '''
                    print('file has been opened')
                    while True:
                        print('receiving data...')
                        data = client_socket.recv(1024)
                        #print(type(data))
                        f.write(data)
                        print((data))
                        #data=(str(to_store[0],"utf8")
                        if data[-4:]==(bytes("over", "utf8")):
                            print('file ended')
                            f.seek(-4,os.SEEK_END)
                            f.truncate()
                            break
                        print('out')
                        # write data to a file

                    f.close()
                    msg_list.insert(tkinter.END, 'file has been recieved')
            else:
                msg_list.insert(tkinter.END, msg)
        except OSError:  # Possibly client has left the chat.
            break
```

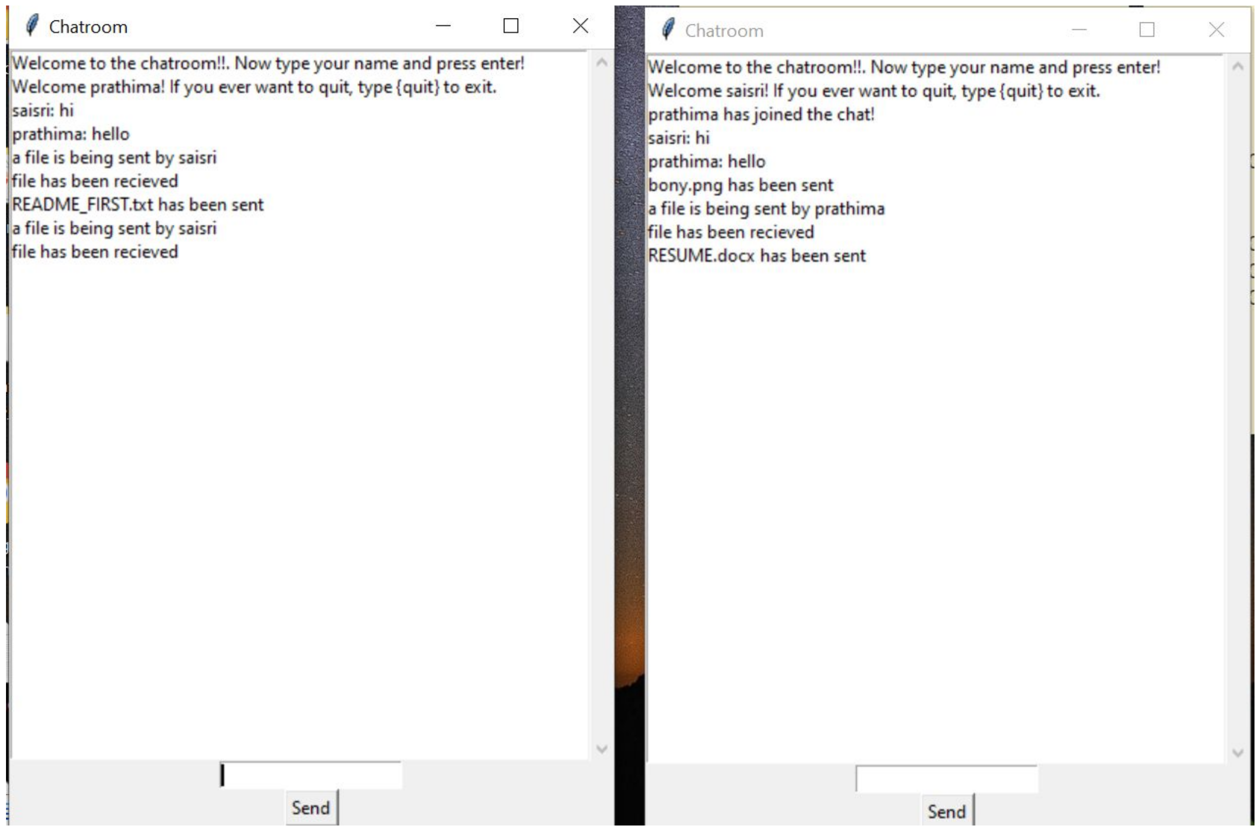


```
def send(event=None):  # event is passed by binders.
    """Handles sending of messages."""
    msg = my_msg.get()
    my_msg.set("")  # Clears input field.
    client_socket.send(bytes(msg, "utf8"))
    #print(msg[0:5])
    if (msg[0:6]=="{file}"):
        msg_f =msg[6:]
        f = open(msg_f,'rb')
        l = f.read(1024)
        while (l):
          client_socket.send(l)
          #print('Sent ',repr(l))
          l = f.read(1024)
        f.close()
        client_socket.send(bytes("over","utf8"))
        #msg_list.insert(tkinter.END, msg_f)
        msg_list.insert(tkinter.END, msg_f+" has been sent")
    if msg == "{quit}":
        client_socket.close()
        top.quit()



def on_closing(event=None):
    """This function is to be called when the window is closed."""
    my_msg.set("{quit}")
    send()

top = tkinter.Tk()
top.title("Chatroom")

messages_frame = tkinter.Frame(top)
my_msg = tkinter.StringVar()  # For the messages to be sent.
my_msg.set("Type")
scrollbar = tkinter.Scrollbar(messages_frame)  # To navigate through past messages.
# Following will contain the messages.
msg_list = tkinter.Listbox(messages_frame, height=30, width=65, yscrollcommand=scrollbar.set)
scrollbar.pack(side=tkinter.RIGHT, fill=tkinter.Y)
msg_list.pack(side=tkinter.LEFT, fill=tkinter.BOTH)
msg_list.pack()
```



```
messages_frame.pack()

entry_field = tkinter.Entry(top, textvariable=my_msg)
entry_field.bind("<Return>", send)
entry_field.pack()
send_button = tkinter.Button(top, text="Send", command=send)
send_button.pack()

top.protocol("WM_DELETE_WINDOW", on_closing)

#----Now comes the sockets part----
HOST = '192.168.43.168'
#HOST='10.20.203.208'
PORT = 60000


BUFSIZ = 1024
ADDR = (HOST, PORT)

client_socket = socket(AF_INET, SOCK_STREAM)
client_socket.connect(ADDR)
receive_thread = Thread(target=receive)
receive_thread.start()
tkinter.mainloop()  # Starts GUI execution.
```

RESULTS:



```
Welcome to the chatroom!!. Now type your name and press enter!
Welcome prathima! If you ever want to quit, type {quit} to exit.
saisri: hi
prathima: hello
a file is being sent by saisri
file has been recieved
README_FIRST.txt has been sent
a file is being sent by saisri
file has been recieved
```

```
Welcome to the chatroom!!. Now type your name and press enter!
Welcome saisri! If you ever want to quit, type {quit} to exit.
prathima has joined the chat!
saisri: hi
prathima: hello
bony.png has been sent
a file is being sent by prathima
file has been recieved
RESUME.docx has been sent
```

CONCLUSION:

By running these codes we are able to share multimedia files between multi-user clients

Future Scope:

I can further extend this application in the future effectively by adding this service.

- Extending this application by providing Authorisation service.
- Creating Database and maintaining users.
- Increasing the effectiveness of the application by providing Voice Chat and video calling.
- Extending it to Web Support.

PES
UNIVERSITY

THANK

YOU