Real Time Chat Application using Socket Programming

Team Socket:
Ramakrishnan R - 21BLC1013
Kowshik S B - 21BLC1067
Rohith S - 21BLC1456

Project description

Implementation of Real Time Chat application using Python under Socket programming methodology which supports File transfer and various features.

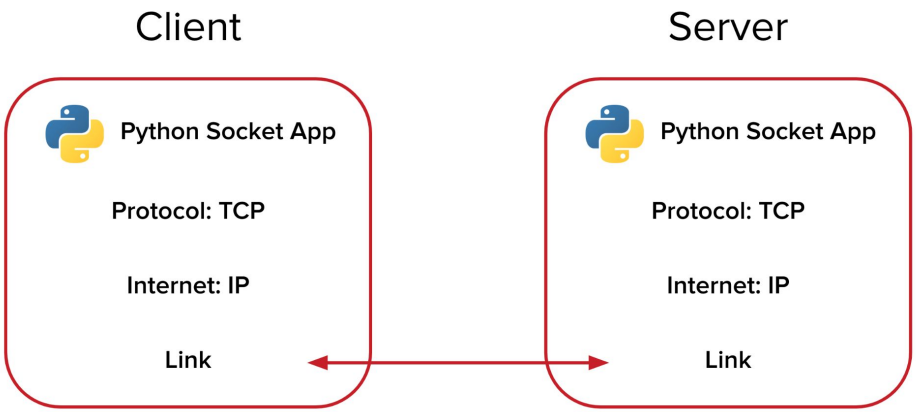
Software components & Modules Involved

Software - Python IDLE

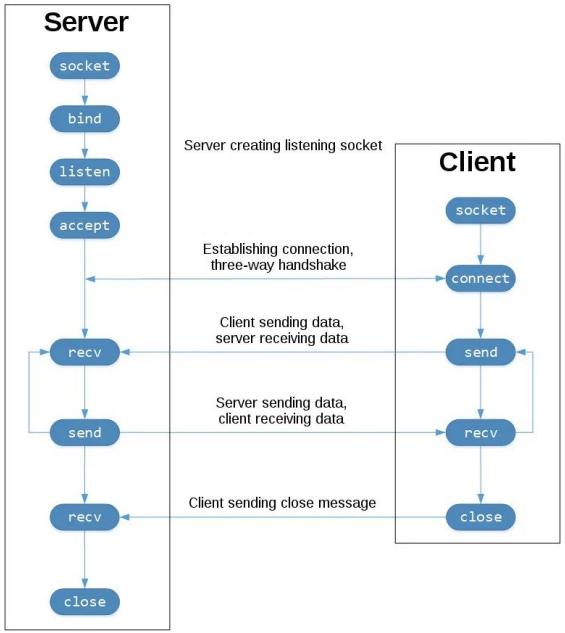
Modules -

1. Socket
2. Tkinter (FOR GUI)

Block Diagram



Algorithm Involved



We have developed a basic connection between server and client which involves transfer of text messages between Client and server.

Server-side:

1. Import the `socket` module.
2. Create a socket object using `socket.socket()`.
3. Bind the socket to a specific address and port using `socket.bind()`.
4. Listen for incoming connections using `socket.listen()`.
5. Accept a client connection using `socket.accept()`.
6. Receive and process data from the client using the accepted socket object.
7. Send a response back to the client, if necessary.
8. Close the connection using `socket.close()`.

Client-side:

1. Import the `socket` module.
2. Create a socket object using `socket.socket()`.
3. Connect to the server using `socket.connect()`.
4. Send data to the server using the connected socket object.
5. Receive and process data from the server.
6. Close the connection using `socket.close()`.

Progress of the work so far

Client side code:

client.py | server.py

C: > Networks > client.py > ...

```python
import socket
client=socket.socket(socket.AF_INET, socket.SOCK_STREAM)

client.connect(("localhost",9999))

done = False

while not done:
    client.send(input ("message from client to server: " ).encode('utf-8'))
    msg = client.recv(1024).decode('utf-8')
    if msg == "exit":
        done = True
    else:
        print("reply from server : ",msg)
client.close()
```

Server side code

client.py | server.py

C: > Networks > server.py > ...

```python
import socket

server=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(("localhost",9999))

server.listen()
client, addr = server.accept()

done=False
while not done:
    msg=client.recv(1024).decode('utf-8')
    if msg == 'exit':
        done =True
    else:
        print("msg from client : ",msg)
    client.send(input("message from server to client: ").encode('utf-8'))
client.close()
server.close()
```

Output

```
PS C:\Networks>  & 'C:\Users\Ram\AppData\Local\Programs\Python\Python311\python.exe' 'c:\Users\Ram\.vscode\extensions\ms-python.python-2023.8.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '57281' '--' 'C:\Networks\client.py'
message from client to server: HELLO
reply from server :  "DA 1 NETWORKS - TEAM SOCKET"
message from client to server: HI KOWSHIK, THIS IS RAM
reply from server :  TEST - 1
message from client to server: exit
PS C:\Networks>
```

Our goals right now

We are planning to make this chat application support multiple clients who can connect to the server, and every client is given an associated thread for managing client requests individually which would support file transfer feature and many more.

THANK YOU