

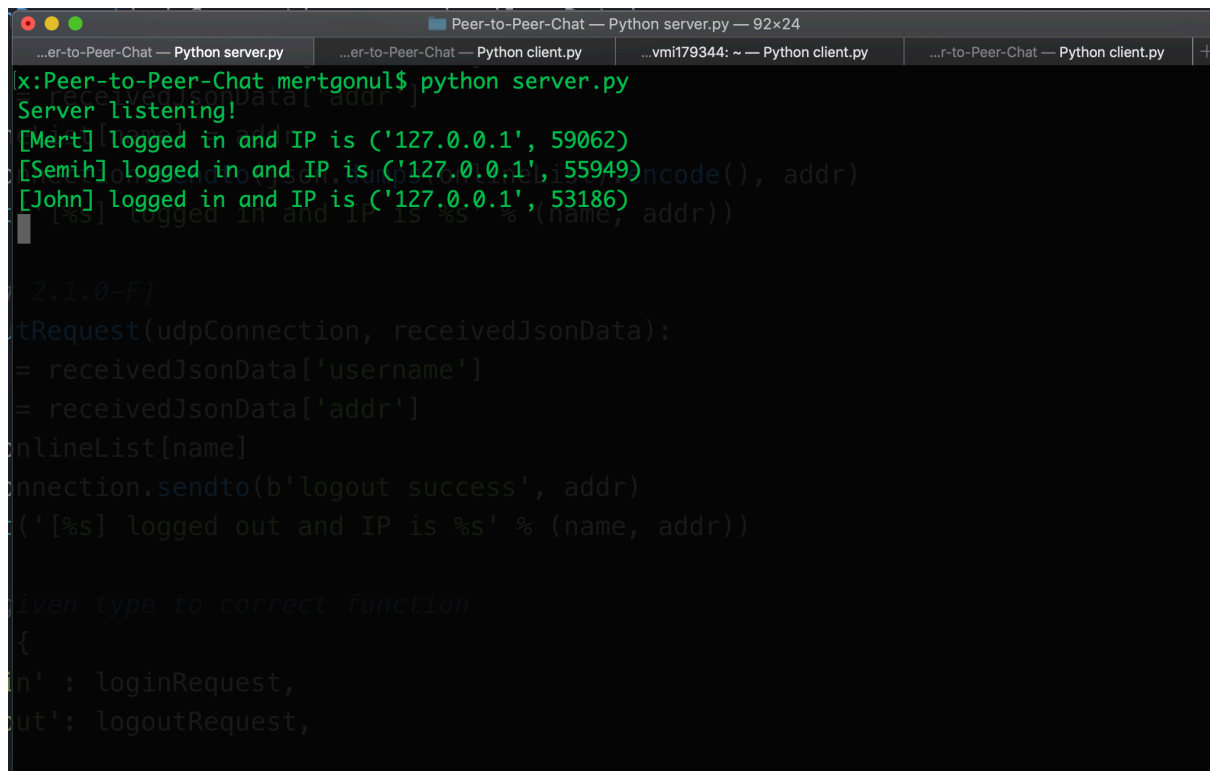
Peer to Peer Chat

Server.py

The server stands in the middle, creating an "Online user" list by collecting "username" and "IP" information for clients. There is only **one mission** of server ; **create online user list and deliver it!**

This process is carried out via UDP with 5000 port.

Customers also transmit their "username" and "IP" information to the **server** via UDP 5000 port.

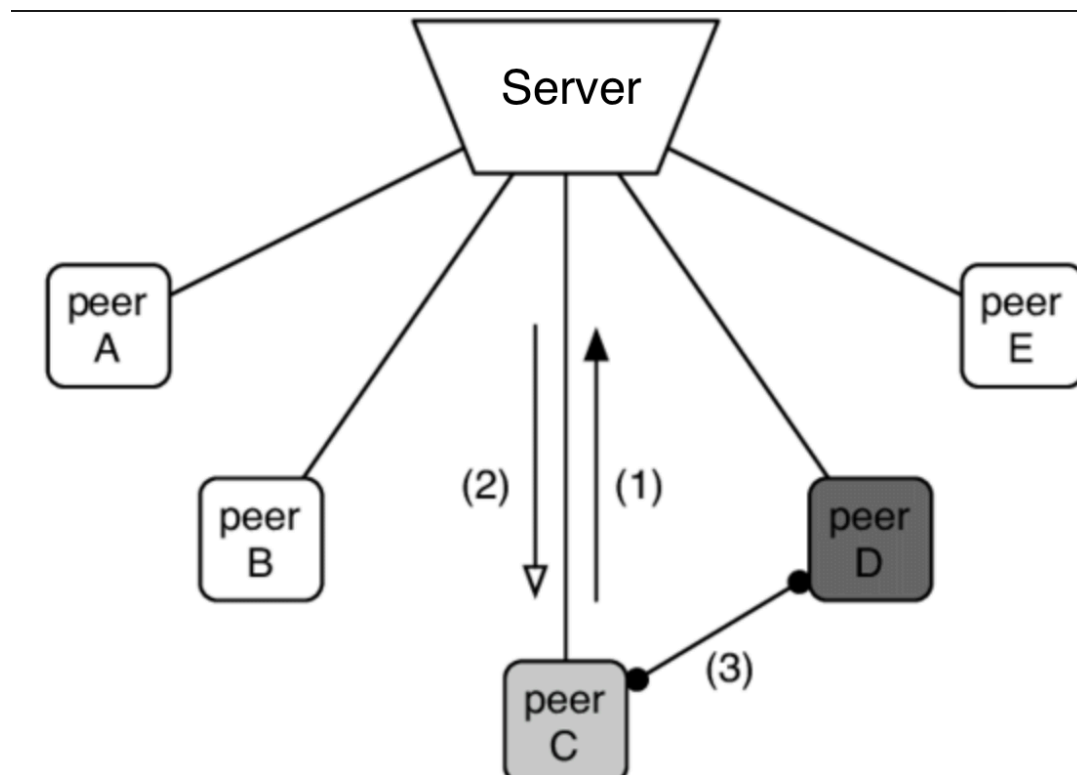


```
Peer-to-Peer-Chat — Python server.py — 92x24
...er-to-Peer-Chat — Python server.py  ...er-to-Peer-Chat — Python client.py  ...vml179344: ~ — Python client.py  ...f-to-Peer-Chat — Python client.py
x:Peer-to-Peer-Chat mertgonul$ python server.py
Server listening!
[Mert] logged in and IP is ('127.0.0.1', 59062)
[Semih] logged in and IP is ('127.0.0.1', 55949)
[John] logged in and IP is ('127.0.0.1', 53186)
[ ] logged in and IP is %s %s (name, addr)

[2,1,0-F]
handleRequest(udpConnection, receivedJsonData):
    name = receivedJsonData['username']
    addr = receivedJsonData['addr']
    onlineList[name]
    connection.sendto(b'logout success', addr)
    print('[ ] logged out and IP is %s' % (name, addr))

given type to correct function
{
    'in' : loginRequest,
    'out': logoutRequest,
```

Heres activity list of connected users to the server.



1 - 'Im here and add me to online user list' request

2 - 'I added you into 'user list' response

3- P2P communication between peers by TCP (client.py)

Client.py

The client.py file displays the list of 'online users' that it receives from the server and enables communication with other people in this list.

```
x:Peer-to-Peer-Chat mertgonul$ python client.py
Enter your username: John
onlineList[name]
connection.sendto(b'logout success', addr)
-----[ONLINE USERS]-----
| Mert : 127.0.0.1
| Semih : 127.0.0.1
| John : 127.0.0.1
-----
n' : loginRequest,
-----[COMMANDS]-----
- Enter [1] to connect to peer
- Enter [2] to wait P2P connection
- Enter [3] to logout
- Enter [4] to refresh online user list
- /back command is closing the active chat session and enabling the menu
**** Your messages automatically saved into [messages.txt] file
connection = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
connection.bind(server_addr)
('Server listening!')
Enter choice:
```

The protocol between server and client is UDP.

The protocol between “client” and “client” is TCP.

The server can never access the message between two people because the messages are exchanged between two people.

```
66 def waitForPeer(udpConnection, clientUsername):
67     tcpConnection.bind(('localhost', 5002))
68     tcpConnection.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
69     print("\nEnter [1] to connect to peer\nEnter [2] to wait P2P connection\nEnter [3] to logout\nEnter [4] to refresh online user list\n/back command is closing the active chat session and enabling the menu\n**** Your messages automatically saved into [messages.txt] file")
70     while True:
71         choice = input("Enter choice: ")
72         if choice == '1':
73             tcpConnection.connect(('localhost', 5002))
74             tcpConnection.send(clientUsername.encode())
75             tcpConnection.recv(1024)
76             tcpConnection.close()
77         elif choice == '2':
78             tcpConnection.bind(('localhost', 5002))
79             tcpConnection.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
80             tcpConnection.listen(1)
81             while True:
82                 (conn, addr) = tcpConnection.accept()
83                 conn.send(clientUsername.encode())
84                 conn.recv(1024)
85                 conn.close()
86         elif choice == '3':
87             tcpConnection.close()
88         elif choice == '4':
89             tcpConnection.close()
90             tcpConnection.bind(('localhost', 5002))
91             tcpConnection.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
92             tcpConnection.listen(1)
93             while True:
94                 (conn, addr) = tcpConnection.accept()
95                 conn.send(clientUsername.encode())
96                 conn.recv(1024)
97                 conn.close()
98         else:
99             break
100
101 # Receive message from peer
102 def readMessage(s, clientUsername):
103     while True:
104         max: hi josh!
105         josh: hi max!
106         josh: lets chat!!!
107         max: sure!
108         max: [RECEIVED MESSAGE] = s.recv(1024).decode()
109         if not receivedMessage: break
```

Example conversation between two peer.