1. PROJECT SUMMARY

In that project, we have two different application one of them is client as named peer and the other is server as named registery. Registery application is used for registering a peer account and providing to login by using that account. Also the peer can search the other peer which it wants to know whether online or offline. After that searching, registery sends an IP address to the peer if the searched peer is online. Then the peer which have search can send a chad request message if it want. If the other user accepts the request message, they start the chatting. Sometimes, the searched peer can communicate another peer. In such cases, peer application sends a busy message to the peer which have sent request message.

2. OUR SOLUTION APPROACH

2.1 REGİSTERY APPLICATION

We have used TCP connection. When the connection which is send by the peer is accepted, the registery application opens a new thread for that peer. After that all operations are generated in that that thread.

If we look those operations,

- **Register**
  The registery is kept client informations in a array list. That array list is placed in interface as static object. In the array list ClientInfo objects are hold. Each of those objects have a name, a password and IP address for the peer.
  When a register message is came, firstly array list is researched with name which by sending register message. If the name is not found in array list, a new ClientInfo object is created and moved into the array list.
- **Login**

  When the login message is sent to the registery, firstly array list which kept the ClientInfo Object is researched. If the any object in array list is matched the sending name and password, IP address of that object which is null is changed with the IP address that is in sending message and LOGIN OK message is sent back.

- **Leave**

  The peer is run on a thread so registery firstly goes to the ClientInfo object which is logged in by the peer and the IP address of that ClientInfo object is changed as null. After that, the thread which is used by the peer is closed.

- **Search**

  In the array list the ClientInfo object which of name object is equal to the sending name that by search message is researched. If it is found the IP address of that peer is sent back.

2.2 PEER APPLICATION

We have used TCP protocol for communicating with registery and other peers. All operations which are related with communication of registery are generated in main class. When the LOGIN OK message is sent by the registery, peer application opens a new thread for listening to CHAT REQUEST message. If CHAT REQUEST message is taken by the thread, a different thread is opened for controlling the communication of the peers. In that thread which is newly opened, a window which informs about a CHAT REQUEST message is displayed to the user. If the user accepts that message by using the window, the thread sends back a YES message and the chatting starts. If not accepted by the user, the thread sends back NO message and it is closed.

If we look the other side which sends CHAT REQUEST message, the thread is created while sending the message and waits YES or NO message to understand whether the request is accepted or not. If YES message is came back, the communication is started. If it is NO message the user is informed about that it is not accepted by opening a window and the thread is closed.

If the user who is sent the CHAT REQUEST message is communicating with another user, the peer application sent back a BUSY message to user who sent the CHAT REQUEST message without informing.

3. ENCOUNTERED PROBLEMS AND SOLUTIONS

- **Problem 1:** In section of chatting when the connection is set up between the peers, the connection which is between peer and registry is became disconnect.
  Solution 1: We have solved that problem by opening a new thread for chatting.
- **Problem 2:** When required, how the peer act as a registery by listening CHAT REQUEST message.
  Solution 2: We have solved that problem by opening a two different thread.
- **Problem 3:** After solution of problem 2, we get a new problem which is “address already in use” because of using the same port by server thread and client thread.
  Solution 3: When the server thread opens a connection, we have protected opening a client connection.

4. UNRESOLVED ISSUES

We have set timeout on UDP socket. It is required that socket is closed when it takes any HELLO message in 200 seconds but UDP socket is not closed when more than one peer is logged in because of using only one socket for all connection by UDP.

5. USAGE EXPLANATION

When the peer application is opened the user is guided by application. Firstly the user creates an account by register command. If the user wants to chat initially he/she requires to login with the created account. After that the user is researched with his/her name by using search command and then chatting is started by using chat request command.

- Direction for commands
    - Register
      The user enters 1 for register command and then must enter a name and password. These should not include “&” character.
    - Login
      The user enters 2 for login command and then must enter registered name and password.
    - Leave
      The user enters 3 for leave command.
    - Search
      The user enters 4 for search command and then must enter a name.
    - Chat
      The user enters 5 for chat request command. Before chat request command is run, search command should be worked.

After CHAT REQUEST message is came, a dialog window is opened. The user accepts or not by clicking one of “yes” or “no” buttons in that window. The peer application also informs about whether the chat request is accepted or not accepted by using a dialog window.