Project
CS 3800 Computer networks

Grade: 15 points.

- This is a group project. Each group can have up to 4 members.

<u>Due Date:</u> **12/08 11:59PM on Blackboard**

PROJECT DESCRIPTION

1. [15pts] Design a multi-person <u>client-server</u> based chat program application using TCP sockets

In this project, you are required to write a simple multi-user chat system in JAVA. The system should use a message passing system over sockets to allow multiple clients to chat with each other.

You need to write a centralized chat server and a chat client using **<u>TCP based sockets</u>**. There should be a central server that clients use to connect into the system. From that point, the user should be able to chat with all other clients signed into the chat server.

Chat Client
You need to write a client that contacts your server, and lets the server know when the user connects and disconnects from the system. More specifically, your client needs to achieve the following requirements:

1. The client should contact the server using TCP sockets. Assume the hostname and the port number of the server are known to the client.
2. Your client will first prompt a welcome message that asks the user to enter a username using the keyboard. This username will then be sent to the server. Then, your server, after receiving the username from your client, will send an acknowledgment message.
3. Your client, after receiving the acknowledgment message will prompt a message to ask the user to enter the text message to be sent to other clients. Each time the user hit 'enter', the client sends the message line to the server. The server distributes the message to the other clients
4. From this point on, the client will send all message entered by the user to the server, and receive and display all messages received from the chat group via the server. The client should display the name of the user that sends the messages to the server and the message itself (along with the time stamp). In order to display the messages, you may need to use a Java graphics packages such as Swing.
5. To exit the client, a user should input a line with '.' only. Before the client can exit, the client should send sign off message to the server, leave the chat group, and wait for the confirmation message from the server.

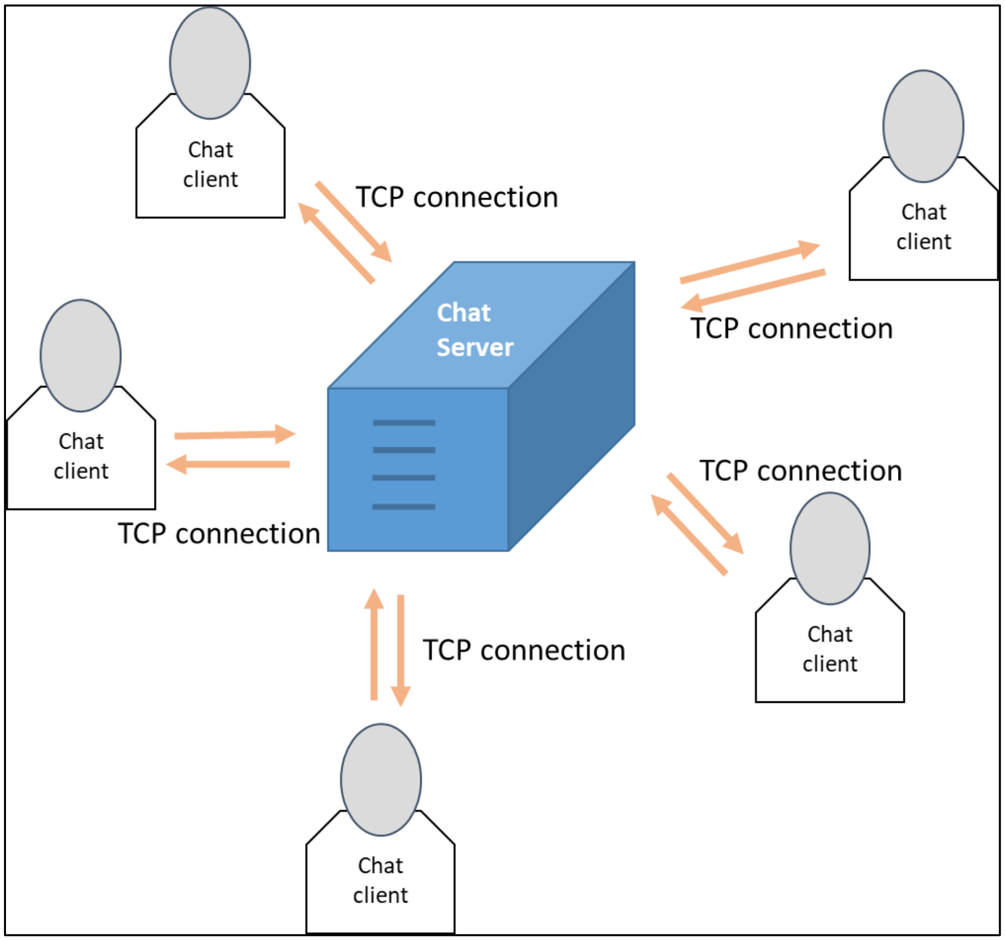

Figure 1 Client Server Chat Program

Chat Server
1.The server must keep track of all users in your chat system. It should operate on a known port. You should choose a port and make sure that your client will connect to the right port.
2.The server should listen on a well-known port and accept user's sign in request. The server should send back a welcome message when a user first joins the chat group, and the server should also inform everybody who is already in the group that a new user just joined the chat group. The server keeps the list of users that is currently signed in.
3. Your server should wait for the client to send message. Each time a new message is received, the server add the user's name to the message and send it out to the chat group.
4. When a server received a user sign-off message, the server will send back the confirmation to the user, and inform all other clients that user has just left the chat group.
5. You need to design your chat message format, including the sign-in message, sign-off message and regular message.

2. [Extra credit: 5pts] Design a multi-person, simplified, <u>P2P</u> chat program application using UDP sockets

In volatile environments, P2P is a better architecture for implementing a chat program. You will design a simple P2P based chat program using UDP sockets in JAVA. The important difference is that the P2P chat program does not have a server that accepts and distributes the message. In the P2P program, each peer sends the messages directly to all other peers. Instead of the server (as in the client server architecture), each peer registers itself with a “tracker”. The tracker is a separate program that keeps a list of all the peers in the group chat. Remember all connections are through UDP.

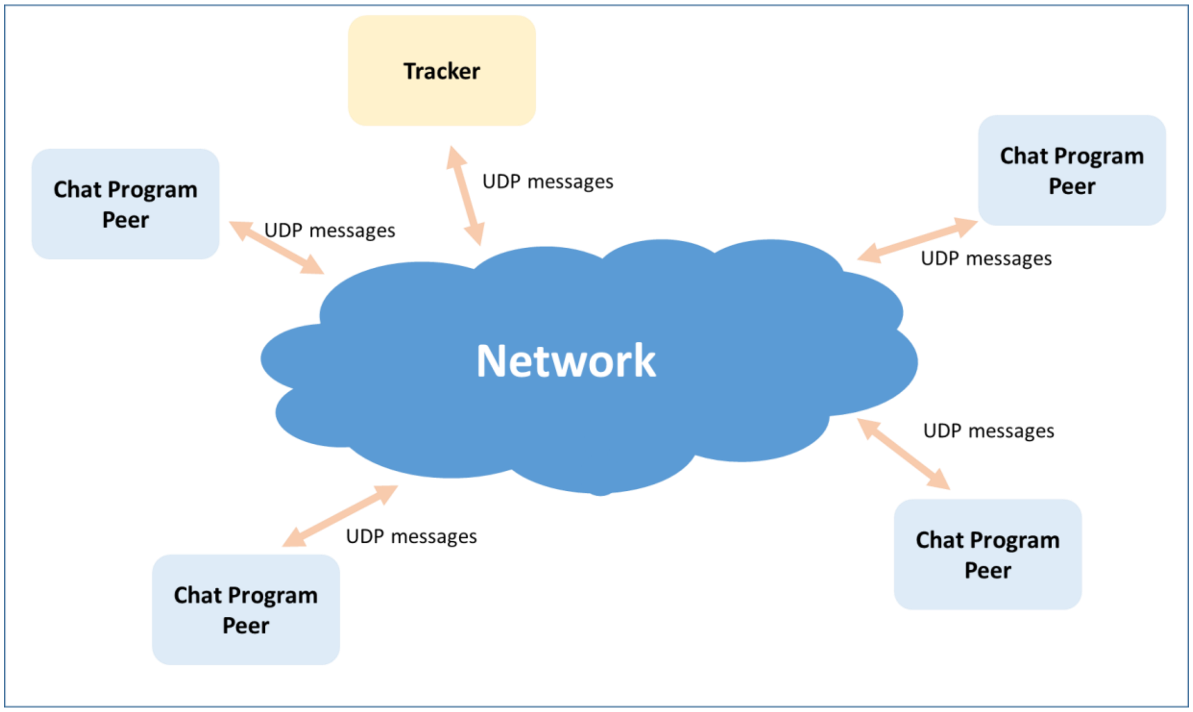

Figure 2 P2P Chat Program

Here are the main specifications of the P2P architecture.

1. The tracker receives UDP messages on a well-known port.
2. A new peer registers by sending an initial message to the tracker. The tracker adds the peer’s information (IP address and port) to its list. The tracker sends back the information of all the peers to the new peer and the updated list to all the other peers.
3. Each peer accepts messages from the user and directly sends the message to all other clients using UDP.
4. The peer also accepts the messages from all other clients and displays to the user (i.e., user name, time stamp and message).
5. When a peer disconnects from the chat, it simply sends a message to the tracker. The tracker then sends the message regarding the departure of the peer to all the other peers.

With respect to how the users interact with the program, the chat program is similar to the previous section (see client server program in #1).

3. Deliverables:

All submissions, unless otherwise specified, will be done through Blackboard using the link for “Project”

<u>Teaming information</u>
Deadline: Nov. 11th
You will form a team (up to 4 members). One of the team members will email the instructor with the teaming information and a team name by Nov. 11.

Initial presentation
Presentation Submission Deadline: Nov. 17th
Presentation: Week of Nov. 18th

- You will do an initial presentation on your plan for implementing the chat program. This should include pseudocode; ideas for implementing the message passing using sockets and the front end; current status; and task allocation among team members. Your presentation will have 5 slides at the most. For the extra credit part, you can have an additional 2 slides.
- Each member of the team will submit the slides (same copy) on blackboard using the link for initial presentation under the "Project" folder.
- Non submission will lead to a reduction of 3 pts from the final score.

Final Submission and presentation
Deadline for submission: Dec 8th 11:59 PM

For both problems #1, #2 submit the following:

- A PowerPoint file that provides a description of your design and features. You may use flowcharts, UML etc. to provide details of your implementation. You will also provide the tasks performed by each team member.
- The source code files.
- The compiled files (unless your program does not compile).
- Each member of the team will submit the slides (same copy) on blackboard using the link for "final submission" under the "Project" folder.

Each team will schedule a presentation with the instructor (Dec 09 - 13). Your presentation will last about 15 minutes. The instructor will email you to set up the presentations.

During the presentation, you will be asked to run the program on your laptop. Be prepared to answer questions about your application.

You can get partial credit based on your design and your code even if your classes are not compiled or the project is not producing the correct. You should be able to explain your work. However, for the extra credit problem, your code has to work to specifications.

Anyone who misses the final presentation will not receive a grade for the project.