Lab Assignment 4

CSC317: Computer Networks

Submission instructions: Submit the source files containing your code. All files must be submitted on Moodle.

In this assignment, you will develop a UDP based simple application for group chat among friends. The goal here is to enable users sending a group message to all other "active" users. To this end, you will write programs for both the server and client. Use UDP sockets for all communications for this problem.

To send a message to all active users, a client must send it to the server first. A message from a client includes two pieces of information: (A) a nickname for the user (i.e. the originator of the message), and (B) the text that the user wants to send to all other users. The server maintains a data structure (e.g. dictionary or list) to keep track of all the active clients at a given point of time. For the remaining description of this problem, we say the server *lists a client as active* when the client is added to the data structure maintained by the server. Similarly, we say the server *drops* the client when the client is removed from the data structure by the server.

When a client sends a message to the server, the server checks if the nickname chosen by the client is already taken by any other client (i.e. if there is any conflict). If the nickname is already taken by another client, the server discards the message and requests the client to try again with a different nickname. On the other hand, if the nickname is not taken by any other client, and the client is not already listed as active, the server lists the client as active.

Any message received by the server is forwarded to all the active clients including the originator of the message. When the server forwards a message to clients, it adds the following information with the message.

- The nickname of the originator of the message.
- A sequence number for the message.

All clients receiving the same message form a server must receive the same sequence number added with the message. You may use an incremental counter to generate the sequence number. It is okay to use an infinite counter for this problem. However, you are welcome to implement a cyclic (i.e. finite) sequence number generator for a more realistic implementation.

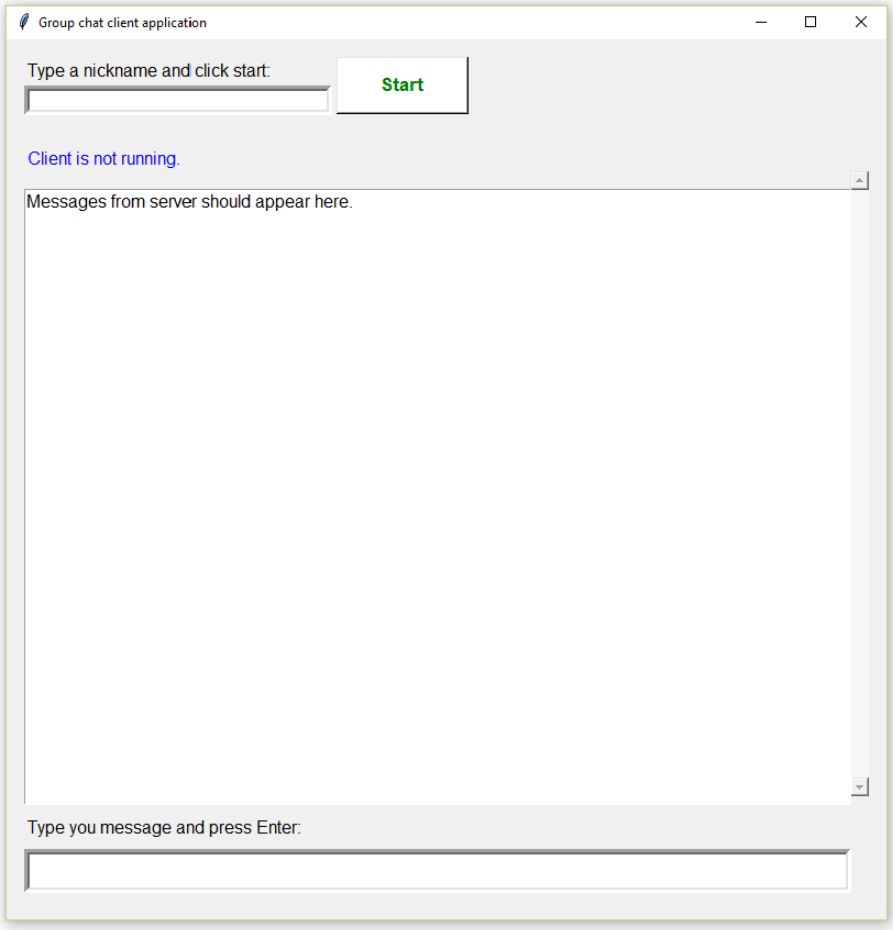

Figure 1: Client user interface.

When a client receives a message from the server, it displays the text to the user and replies the server with an acknowledgement. The acknowledgement contains the sequence number the client received with the message. The purpose of this sequence number is to enable server drop a client when the client stops responding for a while. To this end, while sending a message to a client, the server checks the last sequence number it received from the client as an acknowledgement. If the server finds that no acknowledgements were received for the last 3 consecutive messages, the server drops the client and stops sending any further messages to the client. However, the client can always rejoin by sending a message to the server.

To get you started, you are given the program (on ICON) **"chat_client_UI.py"**. An user interface (UI) is already implemented in this program as shown in Figure 1. When

the start button in the UI is clicked, the client reads the nickname from the UI and sends the following message to the server.

```
<nickname> "Hi, I am joining the room!"
```

Upon receiving this message, the server immediately lists the client as active if there is no conflict with the nickname. Once this happens, the start button in the client's UI converts to a stop button by changing its text from "Start" to "Stop". When the stop button is clicked, the client sends the following message to the server.

```
<nickname> "I am leaving.  Bye!"
```

Upon receiving this message, server drops the client immediately, and the stop button in the client's UI should again change to the start button. Feel free to add necessary verifications for the UI. For example, no message should be sent to the server if no nickname is provided by the user. Also, note that, server forwards both the above messages (i.e. joining/leaving) to all the active clients.

In short, you will implement the client and server for a simple UDP based group chat application. Your application should include the following functionalities:

- The server keeps track of all the active clients at a given point of time, and any message received by the server is forwarded to all the active clients.
- The server ensures that each active client has a distinct nickname (i.e. no conflict).
- A client sends an acknowledgement after receiving each message. The server drops a client if it does not receive any acknowledgement from the client for 3 consecutive messages.
- A clint can join or leave by using the start/stop button.

Test your program by running at least three parallel clients.