CSC 469 / CSC 2208: Advanced Operating Systems

ASSIGNMENT 3: DISTRIBUTED PROGRAMMING & FAULT TOLERANCE

Client/Server Chat System

Binuri Walpitagamage

April 7, 2016

1. OVERVIEW

When initializing the chat client, the client first establishes communication networks for control (TCP) and chat (UDP) messages. It also creates the receiver process used to display the chat messages to the user and registers it with the server. Upon a successful connection with the server, the client then listens to user input with 5 second timeouts. If the user has not typed anything within that interval, then the client confirms that the connection with the server still exists. If at any point, the client has to terminate whether it is due to a lack of network connection or a user exit request, the client closes the receiver and notifies the server(when possible) before exiting.

2. MAIN CLIENT FUNCTIONALITIES

2.1. CONTROL MESSAGES

All control messages are sent using a single method where the control message header is adjusted according to the type of message being sent to the server. The control request handlers call on the previously mentioned function with the request type, a buffer for the server response as well as a room name (Applicable to member list, room list and switch room requests). The handlers then parse the response from the server in order to determine if the request was completed and notifies the user of this status. Except in the case of a registration request, the server connection is tested when a handler method fails to successfully complete a control message request to the server. If an error occurs when sending control messages while registering, the client is safely terminated.

2.2. CHAT MESSAGES

As mentioned in the hand out, failures in the UDP were ignored after notifying the user, due to its unreliability. If the user input is determined to be a chat message, then it is sent to server using the UDP protocol. The server then broadcasts the message to the registered receivers in the same chat room.

2.3. FAULT TOLERANCE

In order to avoid extended periods of server outage for users, the client periodically checks the network connection by sending a MEMBER_KEEP_ALIVE request to the server. Additionally, due to the presence of users who wish to observe the chat network without actively interacting, user input is listened to by 5 second intervals with timeouts. This allows the program to silently check for connection and only alert the user when needed. Upon discovering a disconnected server, the chat client is given 3 tries to reconnect with the server. If a connection cannot be established, then the receiver and the client are terminated. All this is done without any user input.

3. RECEIVER

After creating the receiver, the port number is communicated back to the parent by using the send_ok method. This allows the client to register the receiver with the server. Moreover, when waiting for chat messages, the receiver also checks for control messages by using a timeout interval of a second. This ensures that the receiver promptly exists when required by the chat client.

4. LOGGING

Log messages for the chat client and receiver are recorded in “chatclient.log” and “receiver.log” respectively. In order to enable the functionality, set the DEBUG flag found in client.h file as 1.

4. UNIMPLEMENTED FEATURES

All the features mentioned in Stage 1, 2, and 3 of the assignment are fully implemented. In fault tolerance, when a user reconnects to a server, he/she is not placed back in the same chat room. Furthermore, if a username is taken in the new server, the current implementation informs the user to reconnect with a different username and terminates the client. An alternative solution would have been to prompt the user for a different username and attempt to reconnect again.