COMP28112 Distributed Systems Exercise 1

SIMPLE MESSAGING SYSTEM FOR HEALTHCARE PROFESSIONALS

March 1, 2024

Joseph Rhodes

University of Manchester

Description of the Code and Protocol

The protocol in this application allows two clients to message with each other with synchronous communication that prevents deadlocks.

Initialization and Connection Establishment

Initialization

To initialize the server im.IMServerProxy was used with the URL giving to us and with my specific username. The URL reads `https://web.cs.manchester.ac.uk/n72011jr/comp28112_ex1/IMserver.php`

Connection Establishment

The connection establishment is the fundamental step for initiating communication between the client and the server. This allows a smooth and organized way into the messaging system. When the first client attempts to connect to the server it will check with the server's availability which is represented by the connection status of '0'. '0' meaning that no one is connected. When the server is available it will notify the request to connect by changing the status to '1' which means that the connection was successful. '1' meaning that one client is connected. That client will be set to the self.status of True meaning that it will have sending status when the second client connects. Once the second client connects to the server the self.status will be set to False meaning it will have waiting status and will have to wait until the other client sends a message. This function also includes a condition when a third client tries to connect to the server when two other clients are already connected. When the third client attempts to connect the client will not be allowed to connect and receive a notification that the Server is full. This is due because a full server condition '2' restricts more than 2 clients to access to the messaging system.

Within the connection function I have included a timer system that restricts a client from waiting indefinitely for a second client to connect. When the first client connects the other client will have approximately 15 seconds to connect otherwise the first client will be kicked from the server and given a message “Second client not found.” This allows two separate clients to have access to the messaging system if the first client forgets to terminate its search for a second client.

Exchanging of Messages

The messageSystem function handles both the sending status and waiting status of the clients along with the exchange of messages.

Sending Status

The sending status part within the protocol controls the transmission of messages from the client to the server, enabling effective communication between the two clients. A client is classified to be in the sending status when the self.status = True and this allows the program to know that this client is available to send messages to the other client. The sending status client will receive an area of input saying “Enter your message: ” then the client can input whatever they would like to the second client. One restriction is that the client cannot send an invalid input such as an empty message or they will be kicked from the server and be asked to start again.

Given that the client sent a valid input the sending status client will then wait until they have received a message “Sent” that their message has been successfully sent to the other client. After the confirmation of the successful transmission of the message the client will receive a status change from True to False converting the client into waiting status. The handling of the messages ensures the reliability of communication but also promotes a synchronized interaction model, where the client and server have a fluid exchange of communication.

Waiting Status

The waiting status part within the protocol controls the retrieval and processing of incoming messages from the server, enabling timely and responsive communication. A client is classified to be in the waiting status when the self.status = False and this allows the program to know that this client is only able to receive messages from the other client. The waiting status client will have a message saying “Wait for a message” meaning exactly what the message says. The client has to wait for the sending status client to send a message then the waiting status client will be able to respond.

Once the sending status client sends the message the waiting status client will receive a message stating “You have received a message:” followed by the message sent. Upon the successful presentation of the message the protocol will update the server knowing that the message has been received and will change the status of the waiting client from waiting to sending and know that client will have the functions of the sending status client. With the integration of the message retrieval, the protocol promotes efficient communication and enables two clients to engage messages without disruption or delay.

Termination of the Connection

The terminate function is the connection termination mechanism within the protocol and helps maintain the synchronization between the two clients. This function allows either client to terminate the connection between the two of them by setting the connection status to ‘closed’. Terminating the connection will allow the client to promptly exit the messaging system and prevent anymore messages to be sent. If the waiting status client leaves the system when the sending status client attempts to send a message it will be kicked from the system and vice verse. A problem I ran into was that when the sending status client leaves the waiting status client will not be notified and just wait endlessly. A fix I proposed was to have a counting system so it will wait for a certain amount of seconds and then be kicked if a message has not been received. I just could not get it to work.

Along with the termination function one of the clients may interrupt the connection with a keyboard interrupt (e.g., Crtl+C) and this is handled by terminating that clients connection to the server. The other client will have to terminate its connection as well and reconnect to the server if the client wants to message again.

Achieving Synchronization

To achieve synchronization, the application alternates between sending and receiving messages. It allows only one client to send a message and one client to wait for a message. The client that starts in the sending status receives a prompt to input a message. Once the client sends the message, it receives a notification confirming that the message has been sent and then transitions to the waiting status. The client in the waiting status will see a message on their screen that reads "Wait for a message." During this time, the client has no functionality and must wait until the sending status client sends a message. Once this occurs, the waiting status client receives the message and switches to the sending status, allowing it to send a message back. This alternation ensures that both clients have the opportunity to send and receive messages without encountering deadlock.

Another aspect included in the protocol is continuous polling for incoming messages while waiting because the messaging system requires prompt message retrieval. Polling allows clients to send, respond to, and receive messages as soon as they are sent.

One last feature that contributes to the synchronization of the system is a timing mechanism: when the first client connects to the server, it waits for 15 seconds for the second client to connect before terminating the connection to the server. This ensures that no client is indefinitely waiting for a connection, allowing two other clients to connect if the first client forgets to interrupt the connection.

Running and Testing Description(max 800 words)

Running the Code

Step 1: First you have to go into the directory of ex1 using the command either **~/COMP28112/ex1/** or **cd ~/COMP28112/ex1/**.

(a) I am putting both because for me who was using a Macbook I had to use the second one.

Step 2: Once in the directory open two separate terminals and in both type **python3 imclient_n72011jr.py** and have fun messaging away.

Step 3: If you would like, you can open a third terminal and try to access the messaging system with a third client. Just follow the Steps 1 and 2.

Testing the Code

Test 1: Connect one client you will see the message "Searching for another client..." and then wait 15 seconds to show that the server will terminate if no second client was found.

Test 2: Connect both clients using two different terminals and see on the terminal of the first client that you connected, there will be an input area saying "Enter your message: " and on the waiting client terminal it will say "Wait for a message"

Test 3: Open a third terminal and try to connect a third client and you will receive a message saying "Server is full" and you will be able to continue messaging with the first 2 clients.

Test 4: Once two clients are connected successfully try to send an empty message and the client will be kicked with a message saying "Start again and please try to input something"

Test 5: Once two clients are connected successfully send a message from the sending status client with any message you would like and you will receive a notification that the message was sent and now you are the waiting status and it will say "Wait for a message" On the waiting status client after the message has been sent you will see a message saying "You have received a message" followed by the message the other client sent. Below the received message you will see an input area saying "Enter your message." This is because the waiting status client became the sending status client after it received a message. You are now free to repeat this over and over again.

Test 6: Given both clients are connected, use a keyboard interruption on the waiting status client and when you attempt to send a message on the sending status terminal you will be kicked when you click enter to send your message.

Test 7: Given both clients are connected, use a keyboard interruption on the sending status client and after a given amount of time the waiting status client will be kicked. (Doesn't work in my program but the solution I proposed is at the end of page 3 and the start of page 4 of the report.)