Distributed Systems

Protocol for messaging service
COMP28112

Pavel Ghazaryan

University of Manchester
March 1 2023

Created Using LaTeX

0.1 Main idea

I propose creating a file that can be run on two different terminals, enabling users to exchange messages with each other. To ensure the sending and waiting states of two users, I have devised a plan: the first user to connect to the server (by running the file) will be in the write state, while the second user will initially be in the waiting state.

0.2 Technical aspect

To implement this idea, I will use server keys to maintain communication order. There will be three main server keys: server[‘status’], server[‘received’], and server[‘message’]. The server[‘status’] key will hold the current state of the server, which will be one of three possible values: ‘off’, ‘one’, or ‘two’. When the first user runs the file, server[‘status’] will be set to ‘one’, indicating that only one user is currently connected and another connection is required. Once the second user connects, server[‘status’] will be set to ‘two’, indicating that the connection is established. Any new user attempting to connect to the server will be notified that the server is currently in use, and a connection cannot be established.

After successful connections are established, chatting can commence. The first user who connected will be able to type their message, while the second user will receive a prompt that the other user is typing. When the message is received by the server, server[‘received’] will be set to ‘1’, indicating that the second user has a message to receive. The message will be presented to the user, and they will be able to type their response while the first user is notified that the other user is typing. Each user’s state will be stored in a variable called ‘state’ within the file, which can have two values: ‘send’ or ‘wait’. Initially, the first user will have the state ‘send’, while the second user will be in the state ‘wait’. The state will be flipped after sending and receiving a message, enabling the messaging service to meet the specified requirements between two users.

Furthermore, I will ensure that all server errors are caught using ‘try’ and ‘except’ blocks in Python. I will also consider keyboard interrupts and notify users when the server connection is interrupted, providing a reason for the interruption. Additionally, if one user leaves the conversation, the other user will be notified and the connection will be closed.

In conclusion, I have created a safe and reliable messaging service that meets the specified requirements and accounts for all possible errors that may occur.

0.3 Instructions

Open two separate terminals. Run the file(imclient_j80841pg.py) by python3 imclient_j80841pg.py command on the first terminal and then run it on the second terminal. After that the connection will be established and the user

using the first terminal will be prompted to write a message. Make sure that the im.py file is in the same folder.