LUT

CT30A3401 Distributed Systems

Assignment 4

The Design and architecture of Assignment 4

Report

8.4.2024

Jeremias Wahlsten jeremias.wahlsten@student.lut.fi

Table of Contents

1 Introduction

In this report, I will be going over my design overview and choices on the Assignment 4. This includes things such as an UML diagram of the design, assumption made during the assignment, and explanations on some decicions I made in this assignment. I have used generative AI (Gemini and Codiumate) as a tool to check and improve my code, but all code and text in this assignment is made by me and so is mine.

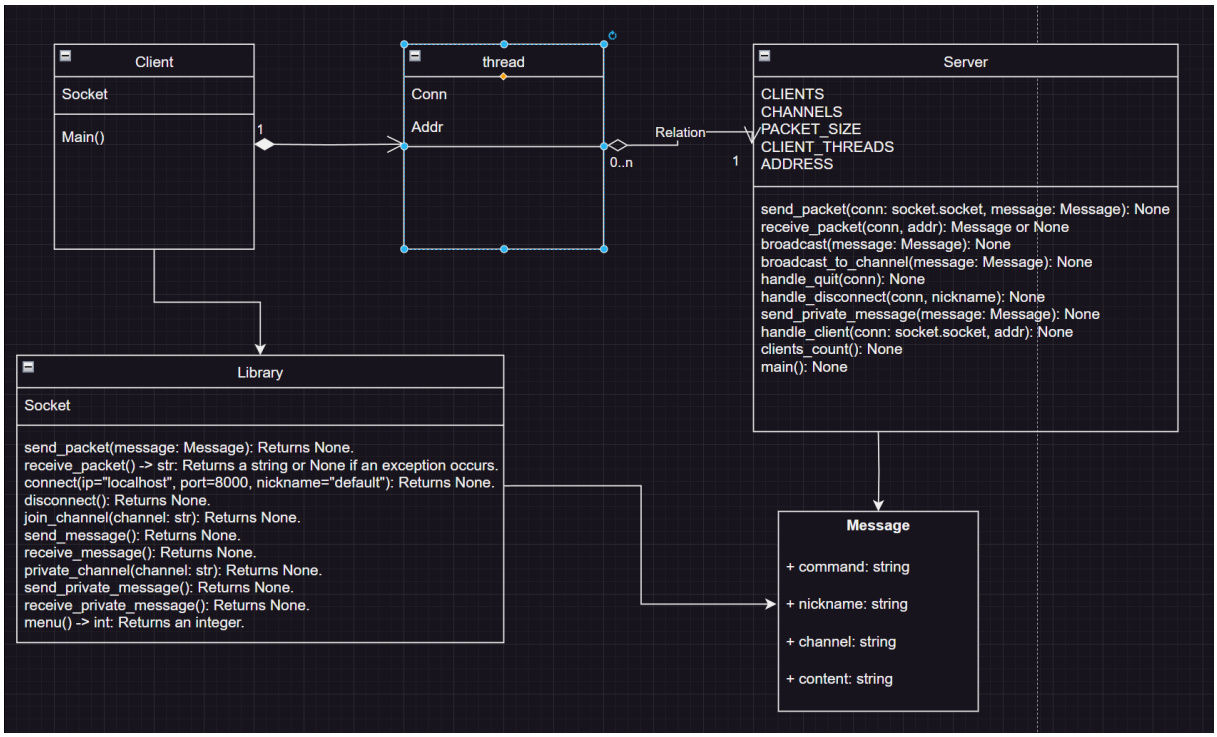
2 Assumptions

Here is a list of assumptions I've made for this assignment:

- Security is not an issue
    - This means that nicknames aren't tied to an account, so after leaving the server somebody else can use the same nickname
    - Messages have no encryption
    - There is not server password, all servers automatically accept new users
    - There is no sanitization of messages
    - Other machine can send packects with your credentials, since server doesen't check the ip with the nickname
- Privacy is not an issue
    - Again, somebody else can use the same nickname as you if you aren't connected to the server
    - Everybody can send you a message, you can't block people
    - Messages have no profanity or other kind of filters
- The made software is not made for distribution
    - In addition to already mentioned security and privacy issues, this software is not being packaged (like in docker) to be run in other machines with easy deployment.
- The max packet size is 2048, so your message can't be over that
- No nickname is the same as a channel

4 Design overview

The following UML-diagram is to give you an overview of my implementation for this assignment.



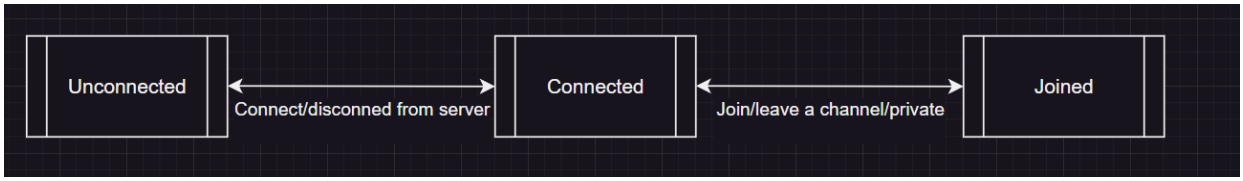
For the client, every method except the main() is in its own library. The same could be done to the server. The server creates a thread for each connecting client. This thread handles every request they have. The server stores the client informastion on the CLIENTS dictionary as “nickname:ip” and their threads are locked in CLIENT_THREADS. CHANNEL is a list of available channels to connect. PACKET_SIZE is a standard packet size variable, which I assignd as “2048”. For easy handling, a Message class was created to include all packet information (more info on 5. Explanation)

5 Explanation

Client:

The client is a menu-based CLI software that lets you send messages to channels and other people in a server.

There are three stages for the client:



- Unconnected
    - This is the default stage for the client. It has not connected to any server and so, hasn't joined any server.
    - You leave this stage by connecting to a server and come back once you've disconnected.
- Connected
    - Here you have established a connection to a server and are connected to it, but haven't hoined any channels and so can't send messages yet
    - You can disconned to go back to Unconnected stage
    - You can join a channel/private message to go to the Joined stage
- Joined
    - You have joined a channel or a private channel and can send messages (your nickname is also set)
    - You can leave the channel to go back to Connected stage
- Since there is not storage for the messages, new users will only see messages sent after joining a channel and private messages disappear after disconnecting from the server.

Server:

The server sends and accept a packet with the following format:

“COMMAND|NICKNAME|CHANNEL|CONTENT”

Where each column being used as follows:

- Command: Tells which kind of a packet it is. There are six different accepted commands:
    - CONNECT: for connecting to the server
    - JOIN: for joining a channel
    - MESSAGE: for sending a message in a channel
    - PRIVATE; for sending a private message)
    - QUIT: for leaving a channel
    - ERROR: for when server has and error to send to the client. Server never receives an error.
- Nickname: has data when asking for a nickname or when sending a message
- Channel: used when sending a message to the channel or asking to join one
- Content: has either messages content or an error message

6 Conclusion

There are certainly improvements to be made, such as Implementation of user authentication authorization for secure access, adding message history functionality to retrieve past conversations and containezation for easy server deployment (as talk about in the video), but the current implementation is more than within the requirements of the assignment.