Daniel Hadaya, 0547045 – Distributed Systems

Text Chat System with Channel Support

Introduction

This report presents the architecture of a simple text-based chat system that allows clients to connect to a server and exchange messages within two text channels (text1 & text2). The system is lightweight and easy to use, providing a basic foundation for understanding chat systems and network. The project is done using Python.

System Architecture

The chat system consists of two components, client application and server application. The client application is responsible for connecting to the server, setting nickname, choosing the channel, sending messages, and receiving messages from other clients through the server. The server application manages connected clients, receives messages from clients, and broadcasts messages to the appropriate channels.

Client Application

The client application utilizes the socket and threading libraries. The following functions are implemented in the client:

receive_messages(): Receives and prints messages from the server in a separate thread.

send_messages(): Listens for user input, encodes the message, and sends it to the server in a separate thread.

The client application prompts users for the server IP address, a nickname, and a channel choice (text1 or text2). Once connected, users can send and receive messages within the chosen channel.

Server Application

The server application leverages the socket, threading, and datetime libraries. The server has a simple Client class that stores information about connected clients, including their nickname, connection, address, and channel.

The server application contains the following functions:

handle_client(client): Manages the connection of each client and listens for incoming messages in a separate thread.

broadcast(message, sender_client): Sends a received message to all other clients in the same channel.

remove_client(client): Removes a client from the list of connected clients and closes the connection.

The server listens for incoming connections and spawns a new thread to handle each client connection. It also stores the clients' information, including their chosen channel, in a list.

Design Considerations and Assumptions

During the design and development of this chat system, several assumptions and considerations were made:

The system is designed for a small number of users and channels, making it suitable for educational purposes or small-scale applications.

The system does not implement any authentication or encryption mechanisms, and therefore should not be used for sensitive communications.

The server assumes that clients will provide a valid IP address, nickname, and channel choice. Proper input validation should be implemented for a more robust solution.

The server's error handling is minimal, and there is room for improvement in handling different types of exceptions and edge cases.