Client-Server Protocol

Course Name: Basic Computer Science Practice

Professor: Li Yugang

Class Day/ Time: Monday - Friday / 08.00 - 11.30, 14.10-16.40

Leader: 1820222041 – 温富勝 Alexander Darryl Kristiawan

Member: 1820222021 – 郑国強 Darren Tejaatmaja

1820222030 – 林哲豪 Wilbert Jaya Sucipto

1820222040 – 冯明想 Jesslyn Clarissa Hermanto

1. Introduction

Internet traffic consists of numerous data transfers between servers and devices, facilitated by two main protocols: TCP and UDP. Each of these protocols has its own strengths and limitations, which users can utilize to enhance their browsing experience.

The key distinction between TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) is that TCP requires a connection, while UDP does not. TCP is more reliable but transmits data at a slower pace, whereas UDP is faster but less dependable. As a result, each protocol is better suited for different kinds of data transfers.

2. Choosing TCP for Instant Messaging

TCP is a connection-oriented protocol, meaning it establishes a connection between the client and server before any data is transferred. It provides reliable, ordered, and error-checked delivery of data. These features make TCP a suitable choice for applications where data integrity and delivery order are critical, such as instant messaging.

Key Features of TCP:

- **Reliability:** TCP ensures that all data packets are delivered to the destination in the correct order. If any packet is lost during transmission, TCP will retransmit it.
- **Connection-Oriented:** TCP establishes a persistent connection between the client and server, maintaining the connection until the communication session is complete.
- **Error Detection and Recovery:** TCP includes mechanisms for detecting errors in data transmission and recovering from them, ensuring that the data received is accurate.

3. Building the Application with JavaFX and Sockets

3.1 JavaFX for GUI

JavaFX is a modern framework for building graphical user interfaces (GUIs) in Java applications. It offers a rich set of components, including buttons, text fields, labels, and panels, allowing developers to create visually appealing and interactive UIs with advanced styling capabilities using CSS.

In the context of an instant messaging application, JavaFX can be used to create:

- **Chat Windows:** A window where users can view messages and type their responses.
- **User List:** A panel displaying the list of currently connected users.
- **Conversation Tabs:** Tabs that allow users to switch between multiple chat rooms or private conversations.
- **Notification Pop-ups:** Alerts for new messages or users joining/leaving the chat.

3.2 Integrating Sockets with JavaFX

To integrate socket communication with the JavaFX interface:

- **Event-Driven Programming:** JavaFX operates on an event-driven model, where user actions (e.g., clicking a button) trigger events. These events can be used to send messages through sockets.
- **Background Threads:** Socket communication should run on background threads to avoid blocking the JavaFX Application Thread (FX Application Thread). This ensures that the UI remains responsive while waiting for messages from the server or sending messages to other users.
- **Real-Time Updates:** The client application listens for incoming messages from the server and updates the chat window in real-time. Similarly, when a user sends a message, it is transmitted to the server and broadcasted to other clients.

3.3 Example Workflow

User Logs In:

- The client establishes a TCP connection to the server using a socket.
- The user interface displays the chat window and user list.

Sending a Message:

- The user types a message and clicks the "Send" button.
- The message is transmitted to the server through the socket.
- The server broadcasts the message to all other connected clients.

Receiving a Message:

- The client listens for incoming messages from the server.
- When a new message is received, it is displayed in the chat window in real-time.

4. Conclusion

In conclusion, TCP is the ideal choice for the networking protocol in an instant messaging application due to its reliability, connection-oriented nature, and error-handling capabilities.

Java's socket programming, combined with the rich UI capabilities of JavaFX, provides a solid foundation for developing a responsive and user-friendly instant messaging application. By leveraging these technologies, the application can offer real-time communication with a reliable, consistent user experience.