Group Chat Application Using Java (Multicast Communication)

Abstract

The Group Chat Application is a command-line-based network communication tool developed in Java. It uses multicast networking to enable real-time communication between multiple users in the same chat group. By leveraging Java's MulticastSocket and threading capabilities, the application facilitates seamless group messaging. Each user can send and receive messages in a synchronized environment, identified by unique usernames. The project demonstrates the application of socket programming, multithreading, and network communication protocols in Java. Its simplicity and scalability make it an excellent tool for learning and practical use in local or subnet-based networks.

Objective

The primary objective of the Group Chat Application project is to design and develop a real-time communication platform using Java, focusing on the following goals:

1. **Enable Group Messaging:**
   Facilitate seamless real-time communication between multiple users within a multicast group.
2. **Implement Multicast Networking:**
   Utilize Java's MulticastSocket to broadcast messages efficiently across all users in the group.
3. **Demonstrate Concurrent Communication:**
   Leverage multithreading to allow simultaneous message sending and receiving without interruptions.
4. **Simplify User Interaction:**
   Provide a straightforward, command-line-based interface for ease of use and accessibility.
5. **Promote Scalability and Flexibility:**
   Allow multiple users to join the chat without the need for complex configurations or server setups.
6. **Enhance Practical Knowledge:**
   Serve as a practical demonstration of Java networking concepts such as sockets, datagram packets, and multithreading.

System Requirements

1. **Java Development Kit (JDK):** Version 8 or higher.
2. **Command-Line Terminal:** Compatible with Windows CMD, Linux Terminal, or macOS Terminal.

3. **Network Connectivity:** Users should be on the same network or configured subnet.

Technologies and Concepts Used

1. **Java Networking:**
    - **MulticastSocket:** For joining a multicast group and sending/receiving messages.
    - **DatagramPacket:** To encapsulate data sent or received in the multicast group.
2. **Multithreading:**
    - A dedicated thread reads incoming messages, ensuring the main thread remains responsive for user input.
3. **I/O Streams:**
    - Handles input from the user and processes network communication.
4. **Character Encoding:**
    - Uses UTF-8 to ensure compatibility and proper display of messages across different systems.
5. **Group Communication Protocols:**
    - Configures TimeToLive for controlling the scope of message propagation.

How It Works:

1. **Initialization:**
    - Users run the program with two command-line arguments: multicast group address (e.g., 224.0.0.1) and port number.
    - Each user enters their name to join the chat.
2. **Message Sending:**
    - Users type messages in the terminal, and the messages are broadcast to the multicast group.
    - Messages are prefixed with the sender's name for identification.
3. **Message Receiving:**
    - A separate thread (ReadThread) continuously listens for messages from the group.
    - Incoming messages are displayed in the terminal, except those sent by the current0 user.
4. **Exit Mechanism:**
    - Typing Exit allows users to leave the chatroom gracefully by closing the socket and leaving the group.

Code Workflow

1. **Main Class (GroupChat):**
    - Handles user input and outgoing messages.
    - Joins the multicast group and starts the reading thread.
2. **Reading Thread (ReadThread):**
    - Continuously listens for incoming messages.
    - Displays messages that are not from the current user.
    - Handles socket closure upon termination.

Sample Commands

- **Start the Program:**

  java GroupChat <multicast-host> <port-number>

  Example: java GroupChat 224.0.0.1 5000
- **Exit the Chat:**
  Type Exit in the terminal and press Enter.

Strengths of the Project

1. **Simple and Lightweight:** Runs entirely on the command line without additional dependencies.
2. **Real-time Communication:** Uses UDP multicast for efficient message broadcasting.
3. **Scalability:** Multiple users can join the same multicast group without additional server configuration.