Internet Relay Chat Project

Abstract

This memo describes the protocol between a client and server in an IRC-style system for the Networking Protocols class at Portland State University.

Table of Contents

Introduction

This RFC specification describes an internet relay chat with a client and a server. Multiple clients will be able to connect to a single server. Once connected to the server, the clients will be able to communicate with each other via the central server.

Users will be able to create virtual rooms. Multiple users can join a virtual room where they can relay messages to all other users in the room. Users will be able to join multiple rooms, create a room, list the rooms, and leave a room. Users will also have the option to list the users in a room.
Users will also be able to send private and encrypted messages to each other.

Basic Information

The communication between the client and server described in the protocol will take place through sockets. Communication between the client and server will be done through port 8080. The server will be listening for connections on this port, and the clients will be able to connect to the server through this port. Once a client connects to the port, they will be asked to enter a unique username. This username is how they will be identified to all of the other connected clients. The clients will be able to send and receive messages to other clients actively connected.

Both the clients and server are able to disconnect from each other at any time. All Client and server crashes will be handled gracefully, and the user will be informed of a crash.

Features

This IRC allows clients and servers to connect to each other. Multiple clients can connect to a single server. Once connected to a server, a client will be able to:

- Create a room
- Join a room/multiple rooms
- Leave a room
- List a room
- List all users in a room
- Send messages to all users in a room
- Send private messages to other users
- Send a secure private message to other users
- Client and server can disconnect from each other
- Both clients and servers detect crashes and respond gracefully

User Commands

Users will be able to input certain commands into the client in order to communicate. The case of the commands do not matter. The commands the user will be able to enter are as follows (note: user enters their own information in replace of parentheses):

1. CREATE “proposed room name”

-Allows a user to create a room with the given name. If the room name is already taken, they will be notified and asked to pick a new name.

2. JOIN “name of room to join”
   -Allows a user to join a given room. If there is no such room, the user will be asked to try again.
3. LEAVE “name of room to leave”
   -Allows the user to leave the specified room. If they are not currently in the specified room, they will be notified and asked to try again.
4. SHOW ROOMS
   -Shows the user all of the rooms available
5. USERS “name of room”
   -Shows all of the users in the specified room. If the user is not currently in the specified room, they will be notified and asked to try again.
6. SENDROOM “name of room”, “message”
   -Allows the user to send a message to the given room. If the user is not currently in the specified room, they will be notified and asked to try again.
7. SENDUSER “name of user”, “message”
   -Sends a private message to the specified user.
8. SENDSECURE “name of user”, “message”
   -Sends a secure encrypted private message to the specified user.
9. MESSAGES
   -Allows the client to retrieve any messages sent to it by the server.

Error Handling

Any time the client sends a command to the server, the server will attempt to carry out the command. If it is unsuccessful, it will return a string stating that it was unsuccessful, and the reason why it was unsuccessful. This exact string will be displayed to the user and the user will be asked to try again. If there are any errors with the server, making it crash, the server will return: “Problem with server”, which tells the client the server has crashed. The client will pass this message onto the user, and terminate its connection with the server.

If there is ever a crash with a client, the server will detect the terminated connection with the server and the server will close all connections with this client. The client will be removed from all chat rooms it was in.

Usage Examples

Creating, joining, or leaving a room

- When the client wishes to create, join, or leave a room, it will send the server a message containing the corresponding user command and the room. The server will check its list of current rooms. If the server is able to successfully execute the command, it will return a string saying it was successful. If the server was not able to successfully execute the command, the server will return a string stating why it was unable to execute the command. The client displays all messages received by the server directly to the user regardless of its content. If there is an error, the user will be able to try again.
- Clients are able to join multiple distinct rooms. The client can join as many rooms as it wants by sending the JOIN command for each room that it wishes to join.
- If the client wishes to create the room *foo*, the user will type “create foo”.
- If the client wishes to join the room *foo*, the user will type “join foo”.
- If the client wishes to leave the room *foo*, the user will type “leave foo”

Sending and receiving messages

There are three ways for a client to send a message: through a chat room, a private message, or a secure private message. When a message from a room is received, the client receives the name of the room that sent the message. When a private, or encrypted private message is received, the user receives the name of the user that sent the message, and is notified if it is a secure message.

- When the user wishes to send a message to a room, the client will send user command *sendroom* along with the message, sender, and recipient to the server. The server

will pass the message to all of the users in that chat room.

- If the client wishes to send the message *hello world!* to the room *foo*, the user will type "sendroom foo hello world!".
- If the client wishes to send a private, or encrypted private message to a user, the client will send the command *senduser* to send a private message, or *sendsecure* to send a private secure message.
- If the client, foo, wishes to send a private message "hello bar" to the client *bar, the user will input "senduser bar hello bar". Bar will receive the message: "Private message from user foo: hello bar"*
- If the client, foo, wishes to send a secure private message "hello bar" to the client *bar, the user will input "sendsecure bar hello bar". Bar will receive the message: "Secure private message from user foo: hello bar"*

When a user is sent a message, it is not received in real time. For a user to receive a message, the user must type the command *messages*. This will retrieve and display all messages, if any. The client will also retrieve any messages whenever any of the commands are used to communicate with the server. For example, if the user attempts to create a room, and there is a message for the user, the user will receive that message, as well as the response of the create room request.

Conclusion

This specification provides the generic framework for the communication between the user, client and server in this internet relay chat. This specification will be updated as changes are made to the design of this application.