Abstract

This memo describes the communication protocol for an IRC-style client/server system for the Internetworking Protocols (CS 494) class at Portland State University.

Table of Contents

1. Introduction

This specification describes a simple Internet Relay Chat (IRC) protocol by which clients can communicate with each other. This system employs a central server which ''relays'' messages that are sent to it to other connected users.

Users can join and leave rooms, which are groups of users that are subscribed to the same message stream. Any message sent to that room is forwarded to all users currently joined to that room.

Users can also create new rooms and list all available rooms.

2. Basic Information

All communication described in this protocol takes place over TCP/IP, with the server listening for communications on port 8000. Clients connect to this port and maintain a persistent connection to the server. The client and server can asynchronously send messages and requests to each other over this channel.

Both the server and client may terminate the connection at any time for any reason.

The server is set to allow only a finite number of simultaneous users.

3. Label Semantics

Identifying both users and rooms involves sending and receiving labels. Labels must follow these rules:

Must be at least 1 character in length.
Can consist of ASCII characters between [0-9a-zA-Z] or special characters(!, @, #, etc).
Must consist of a single ‘word’ (no whitespace).

4. Client messages

4.1. First message sent to server

4.1.1. Usage

Connecting client must provide a user name.

The server must associate the client’s chat name with the socket connection of the user.

This message should only be sent once; the server will ignore duplicate messages from the same client.

4.1.2. Field definitions

user_name specifies the name the connecting client wishes to use. Must follow label semantics.

4.2. Listing rooms

4.2.1. Usage

Client requests a list of all rooms.

4.2.2. Response

Server returns a TCP response containing a list of all rooms and the number of clients in the room.

4.3. Joining and Creating Rooms

4.3.1. Usage

Client joins room of specified name, If no room currently exists with that name, a room is created.

When joining a room, server sends a message to all users currently in the specified room telling them the clients that just joined the room.

4.3.2. Field Definitions

room_name = Name of the room to join or create. Must follow label semantics.

4.4. Leaving a Room

4.4.1. Usage

Client leaves a chat room.

When the server receives the message it removes the client from the specified room. The server sends a message to all users in the specified room with the name of the client that left.

The server ignores leave requests if the client is not currently a member of the specified room.

4.4.2. Field Definitions

room_name = Name of the room to leave. Must follow room semantics.

4.5. Sending messages

4.5.1. Usage

Client sends text message to a room.

4.5.2. Field Definitions

room_name = Name of the room to join/leave. Must follow label semantics.

msg = Message to send.

5. Server Messages

5.1. Listing Response

5.1.1. Usage

Message sent by the server to provide a client with a list. Used for listing rooms.

5.1.2. Field Definitions

item = Name of the room clients are currently in. Must follow label semantics.

item_names = Array of clients currently in the specified room. Must follow label semantics.

5.2. Forwarding Messages to Clients

5.2.1. Usage

5.2.2. Field Definitions

room_name = Name of the room the message was sent to. Must follow label semantics.

User = Name of the user sending the message

msg = Message sent to the room.

6. Error Handling

The Client and Server must detect when the socket connection is terminated.

If the server detects that the connection to the client has terminated, it should remove that client from all rooms and close the socket.

If the client detects that the connection to the server has terminated, it should notify the user and close the socket.