CS494 Matt Fleetwood
Request for Comments: #494

December 2020

Internet Relay Chat (IRC) Protocol

Executive Summary

Here, a protocol is defined that allows for multiple clients to message and chat with each other using one server. The protocol uses TCP sockets and is demonstrated in the Python programming language with independent worker threads. Users are able to create, join, or leave chatrooms that facilitate communicating with other users. Private messaging and file transferring are also described. In the attempt of either a server or client failure, then a graceful disconnect occurs.

Table of Contents

Servers

The server establishes a process to handle client connection methods (e.g. private message, file transfer, create/join/leave chatrooms). In this protocol there is some configuration that needs to be set for the server. Currently, the IP and port for the server are set to local host (i.e. 127.0.0.1) and 8080, respectively. However, there is no reason that these values couldn't change depending on the needs of the user, such as scalability. The server listens on a TCP socket for new client requests and uses a thread to serve a new connection. Each thread is responsible for determining their respective client's requests to perform an action like leave or join a room, send a message, etc.

Clients

The client is expected to be a user running the client-side process that interacts with the user to fulfill their needs with communicating to other users (or clients). Users are required to specify a username for themselves upon starting the client process. There are no limitations on usernames except for: usernames containing spaces, which are invalid; and a max character length of 1028 / 8 or 128 characters.

The client process receives the input from the user alongside an independent thread that reads and processes responses from the server. This independent thread can detect server responses for crashes, messages from other users and chatrooms, and file transferring.

Chatroom

Chatrooms connect multiple users together in the sense that any users in a chatroom will receive messages from other users in the chatroom. Chatrooms have naming restrictions identical to username restrictions. If a chatroom does not exist, then by joining it, the user also creates the chatroom. The chatroom name provides a reference to the chatroom of interest (e.g. /join chatroom1, /msg chatroom1, /leave chatroom1).

Currently, the server does not clean up chatrooms after all users have left them. Similarly, creating multiple chatrooms with the same name is not prohibited or differentiated by the server. A user must "/create" a chatroom if the chatroom does not already exist. At present there are no restrictions for users wanting to join any chatroom. This means any user can join any chatroom.

Graceful disconnects

In the event that a client crashes, the server is notified and removes the user from any rooms they are in before closing the socket for the crashed client. The server follows a similar pattern for graceful

disconnects. That is, if the server crashes then the clients are informed before they are disconnected from the server. Clients typically must enter any alphanumeric value to quit from the client process when the server unpredictably crashes (e.g. from a KeyboardInterrupt error made by the user running the server process). When the server disconnects from clients, it sends the message "/DC" to the client. If the server crashes then

IRC Specification

Overview

This protocol describes both server to client as well as client to server connections and methods. At times there are requests or commands sent from the client to the server and other times commands or responses are sent from the server to the client. The client process asks the user for their input based on the available commands shown by the print_help_menu() function.

Character encoding

There is no explicit limitation on character sets. Since TCP sockets are used then one of the only significant concerns with respect to character codes is how the byte string message is sent to or from the server. For most commands, they are sent as a byte encoded message using UTF-8 encoding. The exception command is for file transferring, which sends data encoded only as a byte string.

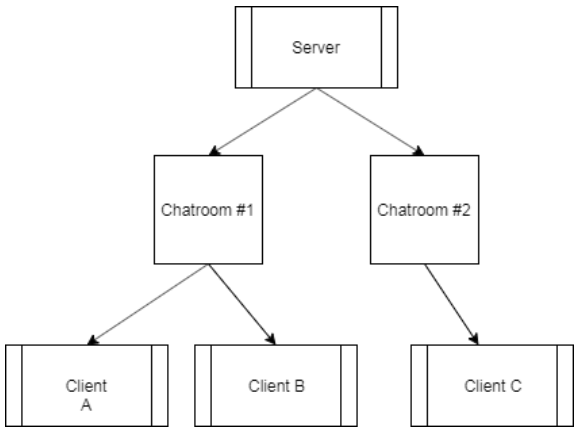
Commands

Due to the asynchronous state of the multi-client / server application in this instance then commands sent to or from the server may or may not cause anything to happen. For example, if a user is in a chatroom and enters the command to message a different chatroom that they are not in then they will only continue to see messages from their current room; their message to the chatroom they are not in will do nothing as a command.

Most commands generally use the following form: /command <string value1> <string value2>, where value1 and value2 can either be a username, a chatroom name, or the message to be sent to a user. Certain other commands in contrast use the form: /command <string value1> or /command. In the former case the second parameter value1 is either a chatroom name or a user name. In the latter case the only parameter is the command itself (for instance with "/dc" to disconnect). All commands begin with a "/".

Chatroom visualization and command definitions

It is briefly noted here how the server process can be visualized.



[Fig. 1. Visualization of the server handling two chatrooms with three clients / users.]

Private message

Users are able to privately communicate with each other. This is facilitated by the "/pmsg" or private message command. Users are required to be in a chatroom prior to being able to send or receive private messages. That is, once a client connects to the server then they need to join a chatroom before they are eligible to privately message other users (or receive other user's private messages). The length of the message is restricted by the size of the data sent over the TCP socket. Currently this length is set to 1024 bytes, which means 1024 / 8 or 128 characters can be sent.

Example: User sends a message, to another user with username "Tim": "Hello!"

```
/pmsg Tim Hello!
```

Chatroom message

The client is able to send messages to a certain chatroom they occupy. This is accomplished by the server maintaining a list of chatrooms, where each chatroom contains another list of users within it. When a client message is to be sent to a chatroom, the server detects this and cycles through all chatrooms until it finds the one where the message is destined for. The server then cycles through all users in the room, sending each of them the source message. The length of the chatroom message has the same restriction of a private message. This means only 128 characters can be sent per each message.

Example: User sends a message to the chatroom "cs494" the message "Hello!"

```
/msg cs494 Hello!
```

Joining, creating, and leaving chatrooms

The user is able to join, create, or leave a chatroom. If the user is not in a chatroom when they leave, then this does nothing.

Example: User leaves the room "cs494".

```
/leave cs494
```

If the user joins a room that does not exist, then that room is first created before appending the user to its user list.

Example: User joins the room "C++ > Python".

```
/join C++ > Python
```

The command to create a room is nearly identical to the "/join" command. However, if the user attempts to create a chatroom with a name already used by an existing room then the user only joins the userlist for that chatroom.

Example: User creates the room "Chatroom101".

```
/create Chatroom101
```

Listing by chatroom name and by all chatrooms

Either all existing chatrooms can be listed or the users in a certain chatroom can be listed. Listing all available chatrooms is a command that takes no extra parameters other than its name. The syntax "/ls_all" means "ls_all" is a command due to the "/" that means to "list all (available chatrooms)". If there are no chatrooms then the command returns an empty list "[]".

Example: User lists all available chatrooms.

```
/ls_all
```

To determine which users (identified by their username) are in a certain chatroom, the user can use the command "/ls <str chatroom_name>". The string "chatroom_name" must match the string that was used when the first user created the chatroom.

Example: User lists all users in the chatroom "cs163".

```
/ls cs163
```

File transferring

There are restrictions on the files sent by the file transferring command "/fsend <str username> <str file.extension>". First, the file must be in the current working directory of the client process. Second,

the file's name and the file's extension type (e.g. .png, .gif, .c) must be specified as it matches the file's true saved name and extension type. For instance, if the file is an image saved in the current directory as "colorful.png" then the file's name "colorful" and the extension ".png" must be provided in the command. The sender, receiver, and the server will all indicate when the file transfer begins, is in process, and ends by showing the following respective messages:

Sender: "Sending file to sever… Done sending file to server.";

Server: "Receiving… Sending… … Done sending …";

Receiver: "Receiving file from user: (user). Downloading file … Receiving file from user… … Done receiving file from user: (user)."

Example: User sends the file "colorful.png", which is in their current working directory of the client process, to the user "Hilde".

```
/fsend Hilde colorful.png
```

Help

One client-side command that doesn't involve the server is "/help". This command displays to the user the available commands.

Example: User displays the help menu in order to see all available commands.

```
/help
```

Disconnecting from the server

While the server and client can gracefully handle crashes from each other, the client can disconnect directly from the server. This uses the command "/dc". Like the "/help" and "/ls_all" commands, the "/dc" command takes no other value when it is invoked.

Example: User disconnects from the server process.

```
/dc
```

Disconnecting from clients

When the server disconnects from clients, it sends the message "/DC" to the clients to indicate it is disconnecting from them.

Example: Server disconnects from clients.

```
/DC
```

Author's Address

Matt Fleetwood
Portland State Maseeh College of Engineering and Computer Science
E: fleet@pdx.edu