IRC Server

Tommaso De Nicola 2006686

January 2024

1 Introduction

The system under consideration is an IRC (Internet Relay Chat) server, implemented in C++ and equipped with a PostgreSQL database for managing channels and activity logs. Users interact with the server using their preferred IRC client (in this case, we will use KVIrc) where they can use the classical IRC commands to join/leave/chat in channels and even more.

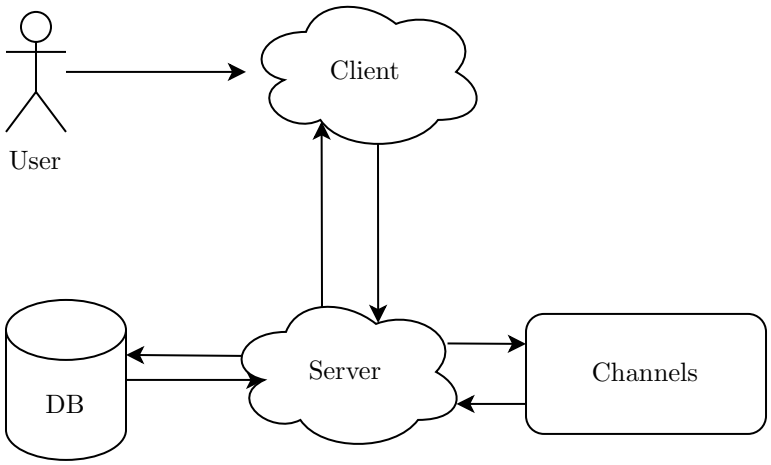

Figure 1: Environment

2 User Requirements

This is a list of commands that can be used in the server to make easier the communication and the utilization of the server itself.

Non Registered User

1. Nick: the nickname for the registration
2. Pass: the password of the server
3. Quit: quit from the server
4. User: the username for the registration

User

1. Join: joins a channel with optional password
2. Notice: sends a message only at one user in a channel
3. Part: quits from a channel
4. Ping: send connection check
5. Pong: returned connection check
6. PrivMsg: opens a private message chat with another user

Admin

1. Kick: kicks from a channel

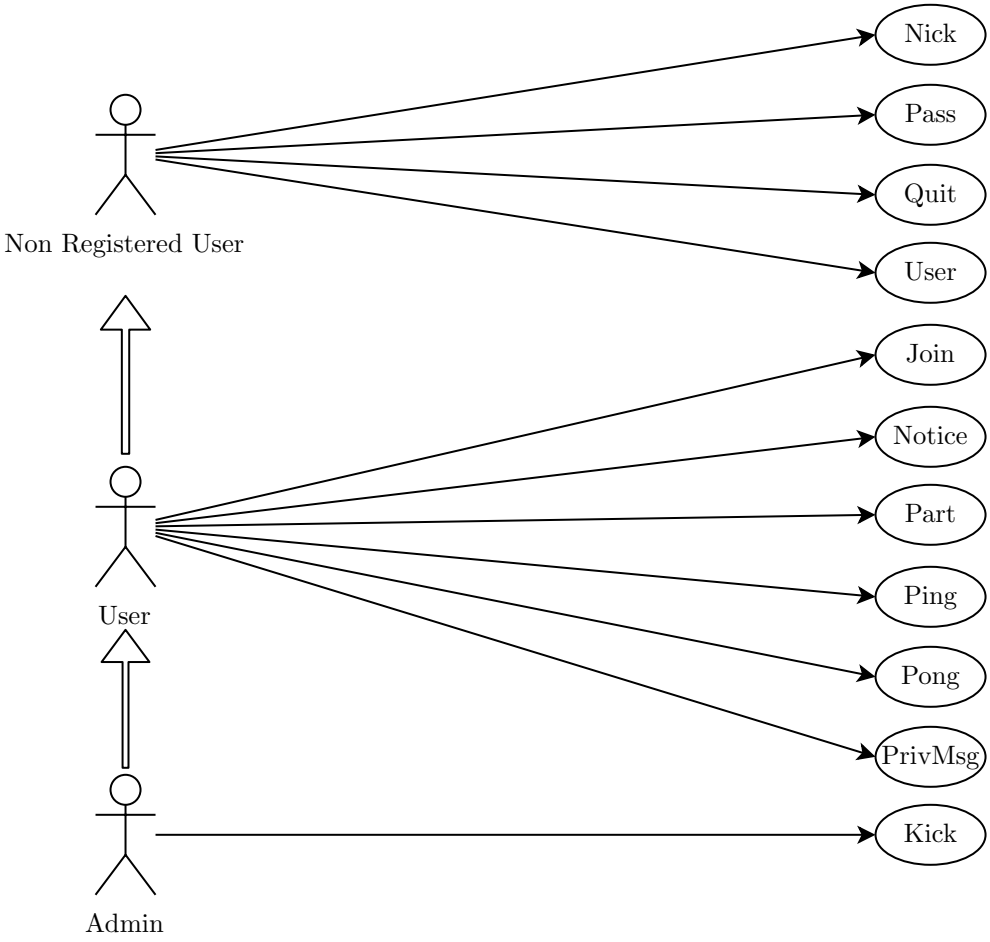

Figure 2: Use Case

3 System Requirements

The system requirements we need, are all connected by a main component which we will call "Server", this one connects the database connection module to the others components, like the channel one, that can load previously saved channels from the database, and handles all channel correlated tasks, then we have the client component which represents the connected clients as users of this system, and they can also use commands via the command handler component.

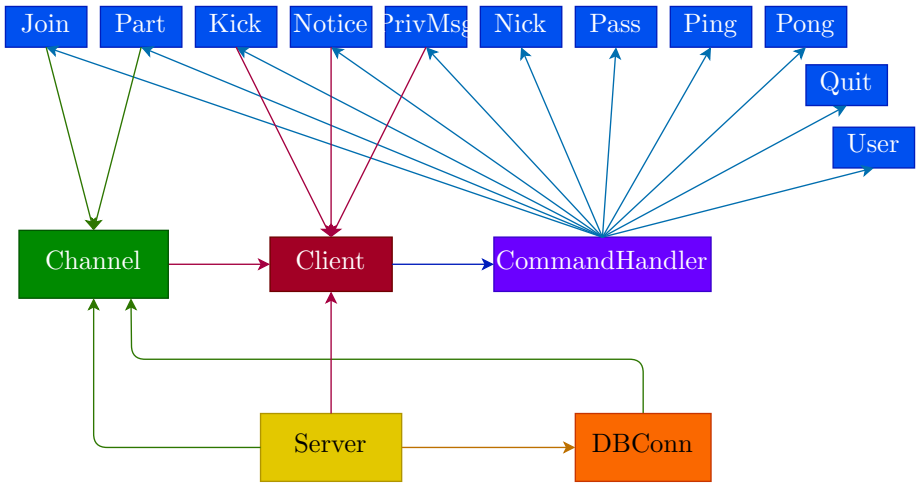

Figure 3: Components

Here an example of a private message from a sender user to a receiver user, and how the server components handles it.

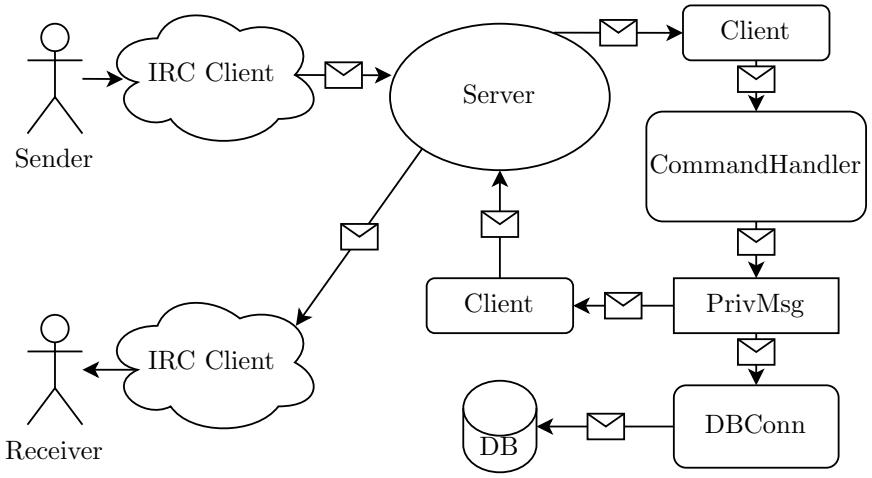

Figure 4: Private Message Send Example

4 Implementation

4.1 Components

4.1.1 Server

The Server Component on creation, loads from the database all the previously saved channels and creates them, then when started, it opens the main socket where it will listen for incoming connections, all the server sockets interaction are handled by a polling event, that calls, the handlers for connection, disconnection and messages; in the message handler calls the command handler to execute possible commands.

4.1.2 Client

When a connection poll event is triggered, a new client is created, with his own socket and identifiers; it can reply messages or join/leave channels.

4.1.3 Channel

A channel can be created by a user or loaded from the db, it can have an optional password for security reasons; a normal message in a channel is broadcasted to all others users in that channel, it can else add/remove clients or even kick them (admins only); by default an admin is the creator of the channel.

4.1.4 CommandHandler

It stores all the commands instances, when receiving a message from a client, it analyze it, replying with an error if it's an unknown command, else if the user has enough permissions it will execute the received command.

4.1.5 Command

Command is an abstract class that defines an interface for commands, they all have, the authorization level needed to be invoked and the execute function; every command does also the needed checks before been invoked.

4.1.6 DBConn

On creation it connects to the Database, it can execute commands, like insert, delete or update, or can execute queries, it also provide a logging method to monitor the server activities that are also saved on the database.

4.2 Database

The database schema is composed by 2 tables, the first one, for the logs, which have a unique id used to differentiate them, the log, and the date, which is the time stamp of that log; the second table is the table of channels, this is used when the server is started, there the channels in the table will be loaded, on a new channel creation, it's automatically added to the table.

```
CREATE TABLE logs (
  id SERIAL PRIMARY KEY,
  log VARCHAR(1023),
  date TIMESTAMP
);

CREATE TABLE channels (
  name VARCHAR(255) PRIMARY KEY,
  password VARCHAR(255) NULL
);
```

Listing 1: DB Schema

5 Results

This server provides a basic IRC experience, with all the needed features, instead of being only experimental, it can really be used as a server (I recommend using KVIrc as client).