MP3 Report / Design Proposal

Andrew Johnson

Design

The design I choose for my server is: concurrent threaded design.

Ports used:

Command Server: There is a master port for the main server (Command Server) .

Chat Room Server: For each chat room created a new port is registered on the server. This allows the chat communication and the server command communication to be kept completely separate.

Sockets used:

Command Server: The master port has a master socket bound to it. From that master slaves are created every time a command is received. The slave socket handles all the communication for the client

Chat Room Server: The master port for the chat room has a master port bound to it. From that each client connected to the chat server has a slave socket derived from the master socket.

Clients: Clients connect and send command through the same master socket bound the communication port. If they join a chat room they bind a new master socket to the chat room server port.

Threads used:

Command server: The main command server spawns thread to handle the commands once they are received. The newly created thread handles the command, and in the case of creating a chat server actually becomes the Chat Room Server.

Chat Room Server: The chat room server spawns threads to handle the communication between IM clients. Specifically each IM client has its own thread on the server. The thread listens to incoming messages from the client and then broadcasts them out to everyone else.

Clients: The clients only spawn threads when they enter a chat room, at that point the spawn a thread that listens to the communications socket and prints out anything that comes in. This way they can listen and receive at the same time.

The code for the Command Server and the Chat Room Server is located in the crsd.c file. The code for the client compute is located in the crc.cpp file.

Compilation

The server is compiled with the command:

```
 g++ crsd.c –o server –lsocket –lpthread
```

The server is started without any argument i.e:

```
 ./server
```

The client is compiled with the command:

```
g++ crc.cpp –o client –lsocket –lnsl –lpthread
```

The client only needs a command when it is not connecting to the unix.cs.tamu.edu machine in which case it needs the server name as an address

./client or ./client linux.cs.tamu.edu

Commands

The server does not accept any stdin commands. It is managed remotely.

The client has four commands:

CREATE <name> : will create a chat room with the given name. Once the command completes clients will be able to join the chat room

JOIN <name> : The client will join the chat room given by the name, and the server will add the client to the chat room. The client will be able to send and receive messages with everyone in the room

DELETE<name>: The client will delete the given chat room. All the people currently in the chat room will be notified and there connections will be terminated.

QUIT: Can be called from inside a chat room to return to the lobby where they can issue commands against the server. It can also be called in the lobby to exit the program and close the sockets correctly.