Project # 1 Documentation

Scott Godwin

1 Summary

The features supported by the application are just what are required for the project. Specifically, you can register, login, and logout and can send public and private messages both anonymously. The server logs all attempts of registration, logging in, and logging out as well as public and private messages even if they are anonymous. The last supported command is the ability to list the online users.

As far as I am aware, there are not any known bugs or problems with my implementation of the project.

2 Build Instructions

The implementation is written in modern C++ (specifically C++11) and as a result requires a C++11 compliant compiler. Since the builds are done using Makefile, the default compiler used is gcc. And as C++11 is required, gcc v4.8.x or higher is required to compile the project.

The project structure is divided into three folders: client, common, and server.

To compile the client, enter the client directory and run make. By default, the client will be compiled in release mode, but can be compiled in debug mode using "make debug". The outputted binary will be "client/bin/*mode*/chatroom_client" where *mode* is either "debug" or "release" depending on how it was compiled.

To run the client binary, two arguments are required. The first argument is the address of the server, and the second argument is the port.

Compiling the server is similar to the client, but the outputted binary will instead be "server/bin/*mode*/chatr where *mode* is again either "debug" or "release". It is recommended to compile in debug when testing since it allows for reusing the port number.

The only argument needed in order to run the server binary is the port number.

3 Usage

First, compile and start the server with a given port number like so: "chatroom_server PORT"

Next, compile and start one or many clients with the address and port of the server like so: "chatroom_client ADDRESS PORT"

The client will output messages differently depending on the source. If it is a message from the client (like if an unknown command was inputted), it will appear like "<*CLIENT*> *message*". If it is a message from the server (like if login failed or invalid arguments were given), it will appear like "<*SERVER*> *message*". Public messages from users will appear like "<*name*> *message*", but if it is an anonymous message, the name will instead be "ANONYNMOUS" like "<*ANONYMOUS*> *message*". Lastly, if it is a private message, it will appear like "<~*name*~> *message*" where *name* will be "ANONYMOUS" if it is an anonymous private message.

When running the client, the following commands are available (commands prefixed with * require you to be logged in). Arguments in brackets are required.

- * **list** - Lists all of the users currently logged in to the server. The server allows you to login with the same user on multiple clients, but only one user will be shown in the list. For

example, if I login to the same account using two clients and run this command, only one user will be shown.

- **login** [**name**] [**password**] - Attempts to login to the chat server using the given name and password. The client will output whether or not the login was successful, and if not, why it was not successful. This command is inaccessible when already logged in.
- * **logout** - Logs out of the chat server. This command will keep the client open unlike **quit**, so you can login again later.
- **quit** - Logs out of the chat server and closes the client.
- **register** [**name**] [**password**] - Attempts to register a new account with the given name and password. The client will output whether or not the registration was successful, and if not, why it was not successful. This command is inaccessible when already logged in.
- * **send** [**message**] - Sends a public message to everyone who is logged in. This command will tell others who sent the message.
- * **senda** [**message**] - Sends a public anonymous message to everyone who is logged in.
- * **sendpriv** [**name**] [**message**] - Sends a private message to the user with **name**. You cannot send a message to yourself.
- * **sendpriva** [**name**] [**message**] - Sends a private anonymous message to the user with **name**. You cannot send a message to yourself.

4 Protocol Specification

The protocol is a binary-based protocol that uses persistent connections on top of TCP. When you login, you are authenticated for only that connection, and if the connection is closed, you will have to login again.

4.1 Header

The messages sent by the server and client all have the same header format.

Data Type	Explanation
1 Byte Integer	The type of message that is being sent. The types that can be used here depend on whether it is the client or server sending the message
2 Byte Integer	The size of the message to be followed.

The client and server will first read in the 3 byte header and validate the contents. The message type has to be a valid message type depending on whether it is the client or server parsing the header. The message size must be small enough to fit in the read buffer which is defined as a 8192 byte buffer.

After ensuring the header is valid, the client or server will read the rest of the message as defined by the message size. After reading the entire message, it will then be parsed and handled which is dependent on the message type.

The following sections will focus on the different message types that can be sent by the server and client.

4.2 Server

The following sections focus on messages that can be sent by the server. As an enumeration of type **ServerMessageType**, the types of server messages are the following:

- **HeaderErrorResponse**
- **ListUsersResponse**
- **LoginResponse**
- **LogoutResponse**
- **RegisterResponse**
- **SendPrivateMessageEvent**
- **SendPrivateMessageResponse**
- **SendPublicMessageEvent**
- **SendPublicMessageResponse**

4.2.1 HeaderErrorResponse

The header error response message is sent whenever an invalid header is received by the server. For example, if a client were to send a message with a header that contained an unknown client message type, the server would response with this message and the corresponding error code.

Data Type	Size (in bytes)	Explanation
ServerMessageType	1	The server message type is always **HeaderErrorResponse**
Unsigned integer	2	The size of the message is always 1.
HeaderErrorCode	1	An error enumeration code that represents the reason why the header was invalid.

The enumeration of type **HeaderErrorCode** can be one of the following:

- **MaximumMessageSizeExceeded** - The size specified in the header is too large.
- **UnknownMessageType** - The message type is not recognized.

4.2.2 ListUsersResponse

The response message that is given when a client requests for the current user list.

Data Type	Size (in bytes)	Explanation
ServerMessageType	1	The server message type is always **ListUsersResponse**
Unsigned integer	2	If the request was successful, this will be large enough to contain the user list in the message, otherwise it will be 1.
ListUsersResponseCode	1	A response enum code that represents the status of the request. If it was successful, the user list will follow this field.
Unsigned integer	1	An integer defining the size of the following string. This string is the name of a user in the list.
String	Varies	A name of a user in the list. The size of this string is the value of the field prior to this one. The prior field and this field are repeated for every user in the list.

The enumeration of type **ListUsersResponseCode** can be one of the following:

- **Success** - The request was successfully handled and the response contains the user list.
- **Uauthenticated** - The user making the request is not logged in.

4.2.3 LoginResponse

The response message that is given when a client makes a login request.

Data Type	Size (in bytes)	Explanation
ServerMessageType	1	The server message type is always **LoginResponse**
Unsigned integer	2	The size of the message is always 1
LoginResponseCode	1	A response enum code that represents the status of the request.

The enumeration of type **LoginResponseCode** can be one of the following:

- **Success** - The user was successfully logged in.
- **IncorrectPassword** - The client gave an incorrect password.
- **InvalidName** - The name given was invalid (can only contain alphanumeric characters).
- **InvalidNameLength** - The length of the name in the request was of an invalid size (can only be between 4 and 8 characters).
- **InvalidPassword** - The password given was invalid (can only contain alphanumeric characters).
- **InvalidPasswordLength** - The length of the password in the request was of an invalid size (can only be between 4 and 8 characters).
- **MissingName** - The name was not in the request.
- **MissingNameLength** - The length of the name was not in the request.
- **MissingPassword** - The password was not in the request.

- **MissingPasswordLength** - The length of the password was not in the request.
- **Unauthorized** - The user is already logged in.
- **UserDoesNotExist** - No such user with the given name exists.

4.2.4 LogoutResponse

The response message that is given when a client makes a logout request.

Data Type	Size (in bytes)	Explanation
ServerMessageType	1	The server message type is always **LogoutResponse**
Unsigned integer	2	The size of the message is always 1
LogoutResponseCode	1	A response enum code that represents the status of the request.

The enumeration of type **LogoutResponseCode** can be one of the following:

- **Success** - The user was successfully logged out.
- **Unauthenticated** - The user making the request is not logged in.

4.2.5 RegisterResponse

The response message that is given when a client makes a registration request.

Data Type	Size (in bytes)	Explanation
ServerMessageType	1	The server message type is always **RegisterResponse**
Unsigned integer	2	The size of the message is always 1
RegisterResponseCode	1	A response enum code that represents the status of the request.

The enumeration of type **RegisterResponseCode** can be one of the following:

- **Success** - The user was successfully logged in.
- **InvalidName** - The name given was invalid (can only contain alphanumeric characters).
- **InvalidNameLength** - The length of the name in the request was of an invalid size (can only be between 4 and 8 characters).
- **InvalidPassword** - The password given was invalid (can only contain alphanumeric characters).
- **InvalidPasswordLength** - The length of the password in the request was of an invalid size (can only be between 4 and 8 characters).
- **MissingName** - The name was not in the request.
- **MissingNameLength** - The length of the name was not in the request.

- **MissingPassword** - The password was not in the request.
- **MissingPasswordLength** - The length of the password was not in the request.
- **Unauthorized** - The user is already logged in.
- **UserAlreadyRegistered** - A user already exists with the given name.

4.2.6 SendPrivateMessageEvent

The event message that is sent when a private message is sent to a user.

Data Type	Size (in bytes)	Explanation
ServerMessageType	1	The server message type is always **SendPrivateMessageEvent**
Unsigned integer	2	The size of the message depends on whether it is an anonymous message and the length of the user and message.
Boolean	1	Whether the message is anonymous.
Unsigned integer	1	The size of the name of the user sending the message. This field is not sent if the prior field is true.
String	Varies	The name of the user sending the message. The length of this string is the prior field. This field is not sent if the message is anonymous.
Unsigned Integer	2	The size of the message that is being sent.
String	Varies	The message that is being sent to the user. The length of this string is the prior field.

4.2.7 SendPrivateMessageResponse

The response message that is given when a client sends a private message.

Data Type	Size (in bytes)	Explanation
ServerMessageType	1	The server message type is always **SendPri sageResponse**
Unsigned integer	2	The size of the message is always 1
SendPrivateMessageResponseCode	1	A response enum code that represents the status quest.

The enumeration of type **RegisterResponseCode** can be one of the following:

- **Success** - The private message was successfully sent.
- **CannotMessageSelf** - The user attempted to send a message to him/herself.
- **InvalidMessage** - The message given was invalid (can only contain printable characters).
- **InvalidMessageLength** - The length of the message in the request was of an invalid size (can only be between 1 and 4096 characters).
- **InvalidName** - The name given was invalid (can only contain alphanumeric characters).

- **InvalidNameLength** - The length of the name in the request was of an invalid size (can only be between 4 and 8 characters).
- **MissingMessage** - The message was not in the request.
- **MissingMessageLength** - The length of the message was not in the request.
- **MissingName** - The name was not in the request.
- **MissingNameLength** - The length of the name was not in the request.
- **MissingOptions** - The options byte (containing the anonynmous boolean) is missing.
- **Unauthenticated** - The user is not logged in.
- **UserNotOnline** - The user being sent the message is not online (or not even registered).

4.2.8 SendPublicMessageEvent

The event message that is sent when a public message is sent.

Data Type	Size (in bytes)	Explanation
ServerMessageType	1	The server message type is always **SendPublicMessageEvent**
Unsigned integer	2	The size of the message depends on whether it is an anonymous message and the length of the user and message.
Boolean	1	Whether the message is anonymous.
Unsigned integer	1	The size of the name of the user sending the message. This field is not sent if the prior field is true.
String	Varies	The name of the user sending the message. The length of this string is the prior field. This field is not sent if the message is anonymous.
Unsigned Integer	2	The size of the message that is being sent.
String	Varies	The message that is being sent. The length of this string is the prior field.

4.2.9 SendPublicMessageResponse

The response message that is given when a client sends a public message.

Data Type	Size (in bytes)	Explanation
ServerMessageType	1	The server message type is always **SendPu sageResponse**
Unsigned integer	2	The size of the message is always 1
SendPublicMessageResponseCode	1	A response enum code that represents the status quest.

The enumeration of type **SendPublicMessageResponseCode** can be one of the following:

- **Success** - The private message was successfully sent.

- **InvalidMessage** - The message given was invalid (can only contain printable characters).
- **InvalidMessageLength** - The length of the message in the request was of an invalid size (can only be between 1 and 4096 characters).
- **MissingMessage** - The message was not in the request.
- **MissingMessageLength** - The length of the message was not in the request.
- **MissingOptions** - The options byte (containing the anonynmous boolean) is missing.
- **Unauthenticated** - The user is not logged in.

4.3 Client

The following sections focus on messages that can be sent by the client. As an enumeration of type **ClientMessageType**, the types of server messages are the following:

- **ListUsers**
- **Login**
- **Logout**
- **Register**
- **SendPrivateMessage**
- **SendPublicMessage**

4.3.1 ListUsers

The request message that is given when a client requests the user list.

Data Type	Size (in bytes)	Explanation
ClientMessageType	1	The client message type is always **ListUsers**
Unsigned integer	2	The size of the message is always 0

4.3.2 Login

The request message that is given when a client attempts to login.

Data Type	Size (in bytes)	Explanation
ClientMessageType	1	The client message type is always **Login**
Unsigned integer	2	The size of the message is large enough to include the name and password.
Unsigned integer	1	The length of the name.
String	Varies	The name to login using. The length of the name is the value of the prior field.
Unsigned integer	1	The length of the password.
String	Varies	The password to login using. The length of the password is the value of the prior field.

4.3.3 Logout

The request message that is given when a client attempts to logout.

Data Type	Size (in bytes)	Explanation
ClientMessageType	1	The client message type is always **Logout**
Unsigned integer	2	The size of the message is always 0.

4.3.4 Register

The request message that is given when a client attempts to register.

Data Type	Size (in bytes)	Explanation
ClientMessageType	1	The client message type is always **Register**
Unsigned integer	2	The size of the message is large enough to include the name and password.
Unsigned integer	1	The length of the name.
String	Varies	The name to login using. The length of the name is the value of the prior field.
Unsigned integer	1	The length of the password.
String	Varies	The password to login using. The length of the password is the value of the prior field.

4.3.5 SendPrivateMessage

The request message that is given when a client attempts to send a private message.

Data Type	Size (in bytes)	Explanation
ClientMessageType	1	The client message type is always **SendPrivateMessage**
Unsigned integer	2	The size of the message is large enough to include the target user name and message
Boolean	1	Whether the message is anonymous.
Unsigned integer	1	The length of the target user name.
String	Varies	The name of the user to send the message to. The length of the name is the value of the prior field.
Unsigned integer	2	The length of the message.
String	Varies	The message to send. The length of the message is the value of the prior field.

4.3.6 SendPublicMessage

The request message that is given when a client attempts to send a public message.

Data Type	Size (in bytes)	Explanation
ClientMessageType	1	The client message type is always **SendPublicMessage**
Unsigned integer	2	The size of the message is large enough to include public message
Boolean	1	Whether the message is anonymous.
Unsigned integer	2	The length of the message.
String	Varies	The message to send. The length of the message is the value of the prior field.