Technical report on Offline Messenger

Bazon Bogdan E3

December 11, 2024

1 Introduction

In this report I will present my idea and the current progress for the Computer Networks project: "Offline Messenger", a project consisteng a server/-client based application that allows users to send messages eachoder, even if the specific user is offline, that has functionalities such as reply messages and capability to see previously sent messages. I will also specify some auxiliary features to improve the user experience.

2 Applied Technologies

For this project, I went for a pre-threaded concurent aproach with mutexes since the project requires usage with at least 1 running client apps and to make sure that a certain line of code is not executed by different threads created at the same time, avoid race conditions or even deadlocks. Since the aim of the project is to send messages between 2 clients, is important to make sure that the contents of the message is sent to the destination without any data loss no matter the performance of the server, hence the TCP aspect of the server.

Besides that I want to experiment with other technologies such as storing student credidentials, as well as saved conversations and notifications in an SQL database, a semi-intuitive interface using the ncurses and advanced login system with username and password system.

3 Application Structure

For each time a client is connected, it goes through the TCP Client/Server connection. After that, the the client is assigned with a guest of the format $Client[id]$. This is a temporary implementation, and a register and sign in functonality with usernames and passwords will be implemented to the final result.

After the initialization phase with the client assignation, it will be added to an array of clients of a structure type containing adress, uid and the socket descriptor.

And then the user is allowed to send messages no matter a seccond client is connected. For every message sent by the client to the server the following operations are performed:

The message written is first stored in a log file. The log file serves the purpose of, every time a client is connected, the server will readthe contents to the log file and redirects to the client where is outputed

If there are more that 1 clients connected to the server, the message will be broadcasted to all the clients connected to the server, besides the sender client.

If the message is begins with the "$/r :$" at the begining of the message, it allows the client to reply to the last line sent only (by reading the log file and getting the last line only). After that, the message that will be both broadcasted and stored to the log file will be displayed as:

```
Replied to [Last Message] : [Message]
```

Durring the broadcasting, the server will write the message to the client, the displaying of the message is done by a thread running outside the main client program

And finally, if one client application ends running. The client is popped out of the clients queue, and the connection for the other clients is not terminated for all the online clients. as it allows to broadcast eachoder, or in the case when there is only 1 client, append the message to the log file.

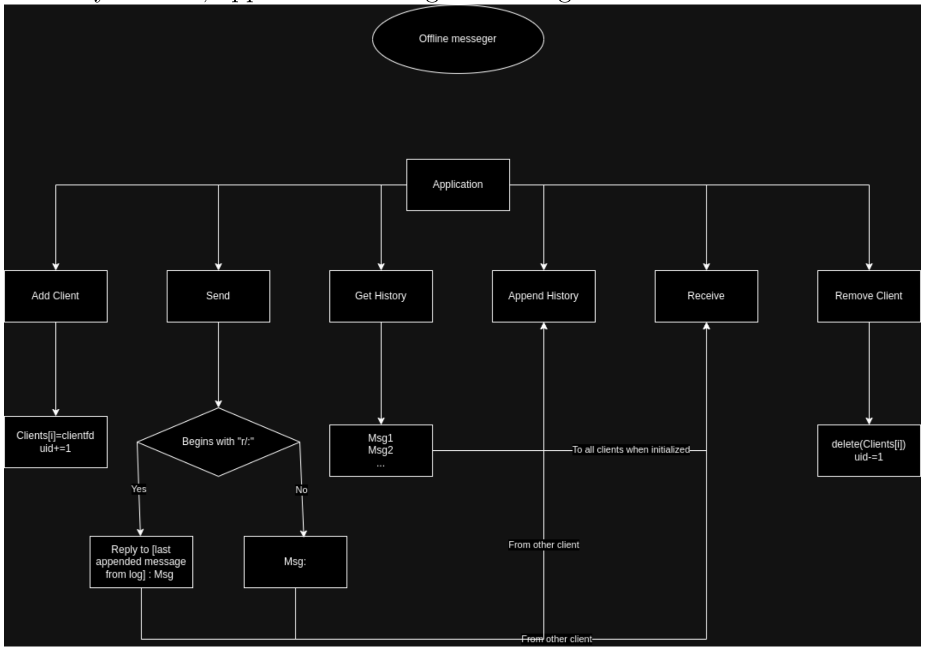

Figure 1: Application structure of the current client

4 Implementation Aspects

The client implementation is a main program that performs the basic client connection to a server with a separated thread that is responsible to receive the answers from the server
The server implementation is a TCP concurent pthreaded with mutexes for the client threads not interfering eachoder, a layer of protection. Which for every connected client, is added to an array of clients. Which can pose some issues down the line:

```
 #define LOGFILE "chat_log.txt"
 #define MAX_CLIENTS 100
```

Firstly the messages are stored in a single file and up to 100 clients can connect to the server
Which poses the following question: Which message is which? Who send this message to whom?
This will be resolved by implementing the login system and for a session between distinct two clients a different
Another issue is other messages besides the buffers sent by clients are added in the log file. The server outputs messages to indicate some status such as: Client has logged in , or client has logged out. And these buffers are stored in the log file. The solution of this problem: Rejecting all messages that start with "Client"

```
 void log_message(char *message) {
  if (strncmp(message, "Client", 6) != 0) {
   FILE *file = fopen(LOGFILE, "a");
   if (file) {
    fprintf(file, "%s\n", message);
    fclose(file);
   }
  }
 }
```

Which causes the following issue. A message sent by a client that happen to start with "Client" prefix is not appended to the log file. Hence not present when the previous messages are loaded.
Another issue is the client sending the character attributed to the SIGINT signal sending. Which is prevented by this line of code:

```
void handle_sigint(int sig) {
 printf("\nCaught signal %d\n", sig);
 exit(0);
}
```

And lastly, the the reply functionality suffers from two issues:
• The client who sends the reply message will have the command displayed while the client has the intented reply text shown.
• The issue that a replied message can also be replied, which causes to display the following this message;

```
Replied to Replied to..... : ...
```

Both can be remediated by implementing the interface with ncurses library.

5 Conclusion

In other words, the implementation of the Offline Messenger project is far from perfect. Some aspects of the project aren't implemented such at chatting between different clients and a login system with usernames are not implemented, as such, these will be the main priority of the project. Alongside, as another priority, the project will have a semi-intuitive interface using the ncurses library. Besides all the main priorities done, I will attempt experimenting by implementing some auxiliary features such as, storing users informations and messages in SQL database, Password system (with encryption). With many implementation will be added to the project is certain that the report is going to be updated conformed to the changes.

References

[1] TCP/IP concurent server implementation using threads "https://profs.info.uaic.ro/georgiana.calancea/Laboratorul_12.pdf"

[2] Linux man page for pthreads.h C library https://www.man7.org/linux/man-pages/man7/pthreads.7.html

[3] Mutex lock for Linux Thread Synchronization (Geeks For Geeks) https://www.geeksforgeeks.org/mutex-lock-for-linux-thread-synchronization/;

[4] Linux man page for pthread_mutex_lock https://www.man7.org/linux/man-pages/man3/pthread_mutex_lock.3p.html

[5] Initialization of Pthread Mutex: https://linux.die.net/man/3/pthread_mutex_init

[6] Code Listing (Overleaf) https://www.overleaf.com/learn/latex/Code_listing

[7] Hyperlinks (Overleaf) https://www.overleaf.com/learn/latex/Hyperlinks

[8] Inserting Images(Overleaf) https://www.overleaf.com/learn/latex/Inserting_Images

[9] Bibliography management with bibtex (Overleaf) https://www.overleaf.com/learn/latex/Bibliography_management_with_bibtex#Introduction

[10] Website to create the Application structure graph (diagrams.net) https://app.diagrams.net/