High Level Design
Instant Chatter

Group 1
Laxmi Kant Singh
Licency Patel
Jangnam Jonas Edward
Sukanya Pradhan
Vibhu Bhardwaj

Last date of revision: 18-10-2022

Contents

1. Introduction

1.1. Why this High Level Design document?

The purpose of this High Level Design (HLD) document is to add the necessary detail to the current project description to represent a suitable model for coding. This document is also intended to help detect contradictions prior to coding, and can be used as a reference manual for how the modules interact at a high level.

1.2. Scope

The HLD documentation presents the structure of the system, such as the application architecture (layers), application flow (Navigation), and technology architecture. The HLD uses non-technical to mildly-technical terms which should be understandable to the administrators of the system.

1.3. Definitions

- Client/Server Programming Model - Whether they are running on the same machine or another, a server application anticipates and responds to requests from client programs. Depending on the application, a computer may act as both a client and a server to other programs requesting services from it.
- Concurrent Server - A concurrent server accepts a client connection, delegates the connection to a child process of some kind, and immediately signals its availability to receive the next client connection.
- POSIX threads- The C/C++ thread API used by the POSIX thread libraries is based on standards. It makes it possible to create new concurrent process flows. It functions well on systems with several processors or cores, where the process flow can be scheduled to run on a different processor to speed up computation through parallel or distributed processing.
- Mutexes- a mutex (mutual exclusion object) is a program object that is created so that multiple program thread can take turns sharing the same resource, such as access to a file.
- Pipe Signals -Pipe is a communication medium between two or more related or interrelated processes. It can be either within one process or a communication between the child and the parent processes.

1.4. Overview

The HLD will:

- present all of the design aspects and define them in detail
- describe the user interface being implemented

- describe the performance requirements
- include design features and the architecture of the project
- list and describe the non-functional attributes like:
  - security
  - portability
  - reusability
  - application compatibility
  - resource utilization

2. General Description

2.1. Product Perspective

- The Instant Chatters chatting application will consist of two different parts: Server and Client.
- Both of these have different .c files for different functions wherever required. The role of a server is to store, send and receive data. It provides services to the clients that get connected to it.
- As usual, the server binds to a particular IP address and port no. and waits for connections from clients (using listen()). It accepts the request from the client that they want to get connected with and all of them get added up in a queue. The server has a record of all the clients that are connected to it.
- If the client is a new user, the server sends a message for registration. If an old user, the server retrieves the username. Multithreading is used to handle multiple clients with the help of POSIX threads. The role of a client is to avail of a service that a server provides once it gets connected to the respective IP and port no. The client can avail the functionality of sharing text messages between two or more users through the server. The client can exit anytime from the application.

2.2. Tools used

- Vim - text editor to write C programs.
- GCC - GNU Compiler Collections which is used to compile C programs.
- gcov - For examining how much of the code is run for different case
- gprof - For examining which function is being called how many times, for how long and by which function
- Valgrind - Valgrind is a programming tool for memory debugging, memory leak detection, and profiling.
- Splint - Splint is a tool for statically checking C programs for security vulnerabilities and coding mistakes.

2.3. General Constraints

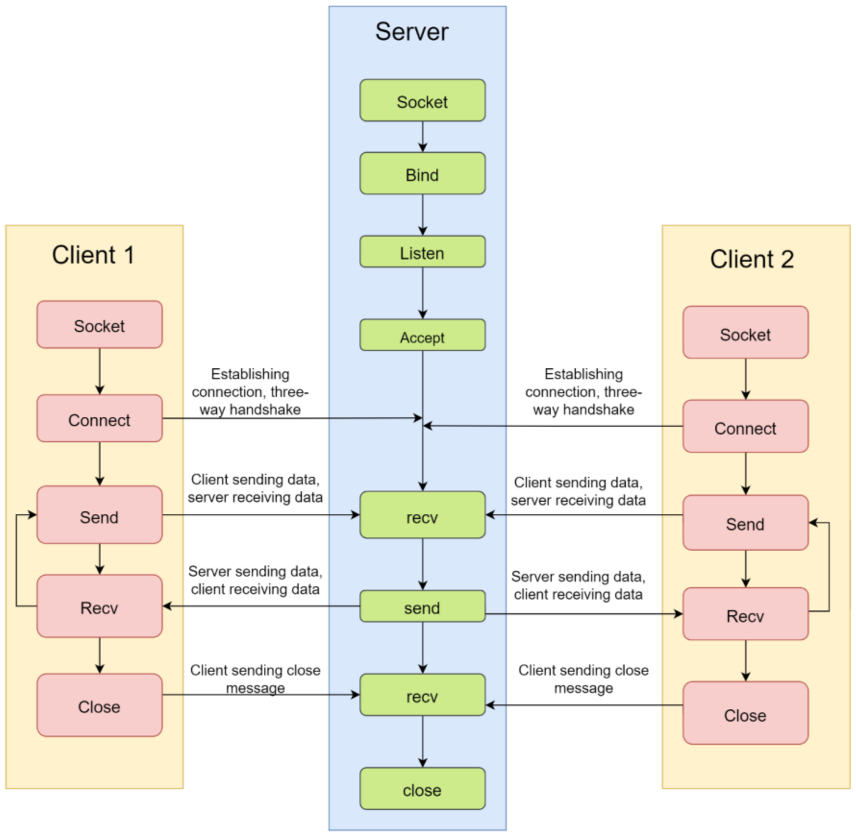
2.3.1 No encryption facility

3. Design Details

3.1. Main Design Features

The main design features include two major parts: the architecture and process communication. In order to make these designs easier to understand, the design has been illustrated in attached diagram.

3.2. Dataflow



Server
Socket
Bind
Listen
Accept
recv
send
recv
close
Client 1
Socket
Connect
Send
Recv
Close
Client 2
Socket
Connect
Send
Recv
Close
Establishing connection, three-way handshake
Client sending data, server receiving data
Server sending data, client receiving data
Client sending close message

3.3. Technology Architecture

3.3.1. Server Architecture

The role of a server is to share data as well as to share resources and distribute work. In this project, the server will be handling clients concurrently.

3.3.2. Client Architecture

The clients have to authorize themselves using a username and a password. All the messages sent and received get stored in a log file by the server.

3.3.3. <u>Tools Used</u>

See section 2.2 for tools used in the design of this project.

3.4. <u>Standards</u>

Inputs – entered through a text field.

Quality – by keeping the interface simple and direct, quality should be kept at a maximum.

3.5. <u>User Interface</u>

The user interface is a command line interface, which has no graphics and a very plain, unadorned layout. It processes the data after receiving the appropriate input from the administrator or user.

3.6. <u>Error Handling</u>

Should errors be encountered, an explanation will be displayed as to what went wrong. An error will be defined as anything that falls outside the normal and intended usage.

3.7. <u>Performance</u>

The performance of the application is at par with a normal chat program.

3.8. <u>Portability</u>

Code and program portability should be possible between kernel-recompiled Linux distributions. For everything to work properly, all components should be compiled from source.

3.9. <u>Reusability</u>

The code written and the components used should have the ability to be reused with no problems. Should time allow, and detailed instructions are written on how to create this project, everything will be completely reusable to anyone.

3.10. <u>Application compatibility</u>

All components in this project will be using C language and sockets will be the interface for client server communication (connection oriented).

3.11. <u>Resource utilization</u>

Any task that is carried out is likely to consume all available processing power until it is completed.