TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING THAPATHALI CAMPUS

Proposal

On

Client-Server Chat Application

Submitted by:

Saksham Neupane THA081BCT032

Sandeep Dhungana THA081BCT034

Sandesh Acharya THA081BCT035

Submitted to:

Department of Electronics and Computer Engineering
Thapathali Campus
Kathmandu, Nepal
12 March, 2025

ABSTRACT

This project aims to develop a robust client-server chat application using C programming language and socket programming. The application will allow multiple clients to connect to a central server, enabling real-time communication between users through text messages. The system will implement TCP/IP protocols to ensure reliable data transmission and employ threading to handle multiple concurrent client connections.

Keywords: Client-server architecture, C programming, Socket programming, TCP/IP, Multi-threading, Network communication, Real-time messaging

Table of Contents

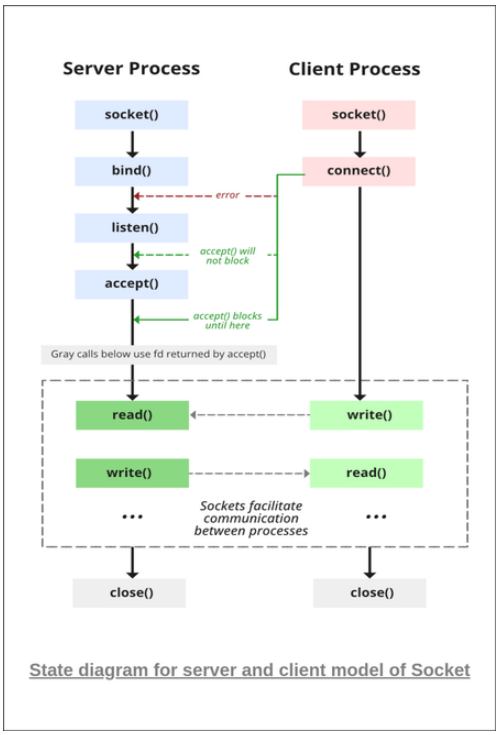


Figure 1

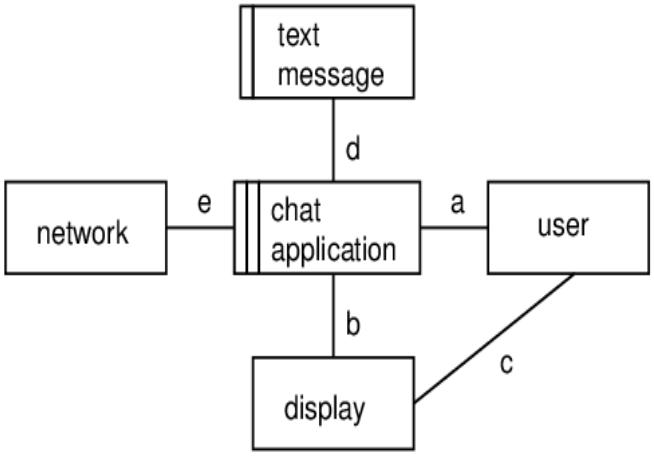


Figure 2

List of Abbreviations

API: Application Programming Interface
IP: Internet Protocol
TCP :Transmission Control Protocol
UDP: User Datagram Protocol
GUI: Graphical User Interface
I/O :Input/Output
OS: Operating System
HTTP:Hypertext Transfer Protocol
IDE: Integrated Development Environment
LAN :Local Area Network
WAN :Wide Area Network
ASCII :American Standard Code for Information Interchange

1. INTRODUCTION

This project focuses on developing a client-server chat application using C programming language that allows multiple users to communicate in real-time. The application utilizes socket programming to establish connections between clients and the server, enabling message transmission across a network.

1.1 Background Introduction

Communication systems have evolved significantly with the advancement of computer networks. Client-server architectures provide a foundation for many networked applications, including chat systems, which enable users to exchange messages over a network infrastructure.

1.2 Motivation

The motivation behind this project is to understand and implement core networking concepts in C programming, particularly focused on socket programming and client-server architecture. It provides practical experience in handling network connections, concurrent processing, and data transmission in a distributed environment.

1.3 Problem Definition

Traditional communication methods may be restricted by geographical limitations or require dedicated hardware. A software-based chat system can overcome these limitations by utilizing existing network infrastructure. The challenge lies in implementing a robust system that can handle multiple concurrent connections while ensuring reliable message delivery.

1.4 Objectives

The main objectives of our project are listed below:

- To develop a functional client-server chat application using C programming and socket communication
- To implement concurrent client handling on the server side to support multiple simultaneous users

1.5 Scope and Applications

The project will implement basic text-based chat functionality with support for multiple clients. It will include user authentication, private messaging, and basic error handling. The application can be used for local network communication in educational institutions, small businesses, or as a learning tool for understanding network programming concepts.

2. LITERATURE REVIEW

Socket programming has been a fundamental aspect of network communication since the development of the Berkeley Socket Interface in the 1980s. Various implementations of chat applications have explored different architectures and protocols to achieve efficient and reliable communication.

2.1 Socket Programming in Network Applications

Socket programming provides an interface for network communication in most operating systems. It enables programs to establish connections over a network using standardized protocols such as TCP/IP. In the context of chat applications, sockets facilitate the exchange of messages between clients and servers through established connections.

2.2 Client-Server Architecture in Communication Systems

Client-server architecture represents a computing model in which server software accepts and responds to requests from client programs. This model is widely used in networked applications due to its centralized nature, which simplifies management and security implementation. In chat applications, the server acts as a central point that relays messages between connected clients.

3. PROPOSED SYSTEM ARCHITECTURE

The proposed chat application follows a centralized client-server architecture where multiple clients connect to a single server that manages message routing and user authentication.

3.1 Block Diagram and System Architecture

The system consists of two main components: the server and multiple client applications. The server manages client connections, authenticates users, and routes messages between clients. Each client application provides a user interface for sending and receiving messages while maintaining a persistent connection to the server.

3.2 Data Flow Diagram

The data flow in the system begins with client connection requests to the server. After successful connection, clients can send messages to the server, which then processes and forwards them to the intended recipients. The server maintains a list of active connections and handles disconnections gracefully.

3.3 Tools and Environment

The project will be developed using the following tools and environment:

- C Programming Language (C99 standard)
- GCC Compiler
- Visual Studio Code / Code::Blocks IDE
- Linux/Windows operating system
- Standard C libraries for socket programming

4. METHODOLOGY

The implementation methodology focuses on creating reliable socket connections and efficient message handling between clients and the server.

4.1 Socket Implementation

The socket implementation will use TCP sockets to ensure reliable data transmission. The server will create a socket that listens for incoming client connections on a specified port.

4.1.1 Server Socket Implementation

The server implementation will utilize multi-threading to handle multiple client connections simultaneously. Each client connection will be assigned a separate thread to process incoming messages without blocking other connections.

4.2 Client Implementation

The client application will establish a connection to the server using the server's IP address and port number. It will have separate threads for sending and receiving messages to provide a seamless user experience.

4.2.1 Client Socket Configuration

The client socket will be configured to connect to the server's listening socket. It will handle connection establishment, message exchange, and proper disconnection procedures.

4.2.2 Client User Interface

A simple text-based user interface will be implemented to allow users to input messages and view incoming messages from other users.

4.3 Message Handling and Transmission

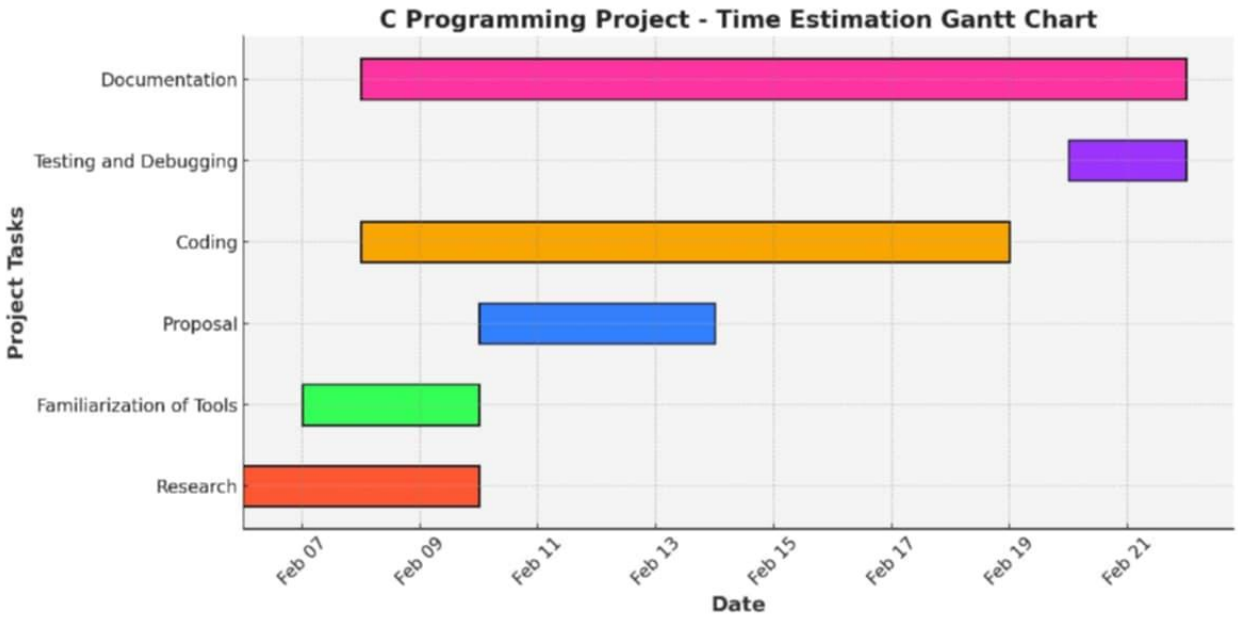
Messages will be transmitted as text strings with appropriate headers to identify the sender and intended recipient(s). The server will parse these headers to route messages correctly.

5. SCOPE AND APPLICATIONS

The chat application can be used in various scenarios including:

- Internal communication within organizations
- Educational settings for demonstrating network programming concepts
- Small group collaboration in project settings
- Learning tool for understanding client-server architecture

6. TIME ESTIMATION



7. FEASIBILITY ANALYSIS

The project is feasible in terms of technical implementation as it relies on well-established socket programming techniques in C. The hardware requirements are minimal, requiring only computers with network connectivity. The development can be completed within the allocated timeframe using available resources and open-source tools.

References

[1] YouTube - https://www.youtube.com/watch?v=uagKTbohimU

[2] GITHUB - https://github.com/aaaaaaayush-no/c-project

[3] SOCKET PROGRAMMING IN C - https://www.geeksforgeeks.org/socket-programming-cc/