Nepal College of Information Technology
Balkumari, Lalitpur
(Affiliated to Pokhara University)

A Project Report
On
LAN Chat Room
With File Transfer using Winsock

Submitted by
Harsh Chaudhary Kalwar (221715)
Harsh.221715@ncit.edu.np
Pranil Poudel (221734)
Pranil.221734@ncit.edu.np

Submitted to
Department of Software Engineering
Network Programming
June 25, 2025

TABLE OF CONTENTS

1. Introduction 3
2. Objectives 3
3. System Architecture 4
4. Tools & Technologies Used 4
5. Implementation Overview 5
6. Key Features 6
7. Challenges Faced 6
8. Testing and results 7
9. Conclusion 8
10. Future Enhancements 8
11. Acknowledgement 8

Introduction

In the world of modern communication, real-time connectivity is crucial. From messaging apps to collaborative platforms, socket-based communication forms the foundation of interaction. This project, *LAN Chat Room with File Transfer using Winsock*, was developed as part of the Network Programming course to provide a practical implementation of TCP/IP networking principles. It focuses on LAN-based multi-client communication using sockets in C++, offering functionalities like real-time messaging, file transfer, and a command-based interface — all within a console application.

Objectives

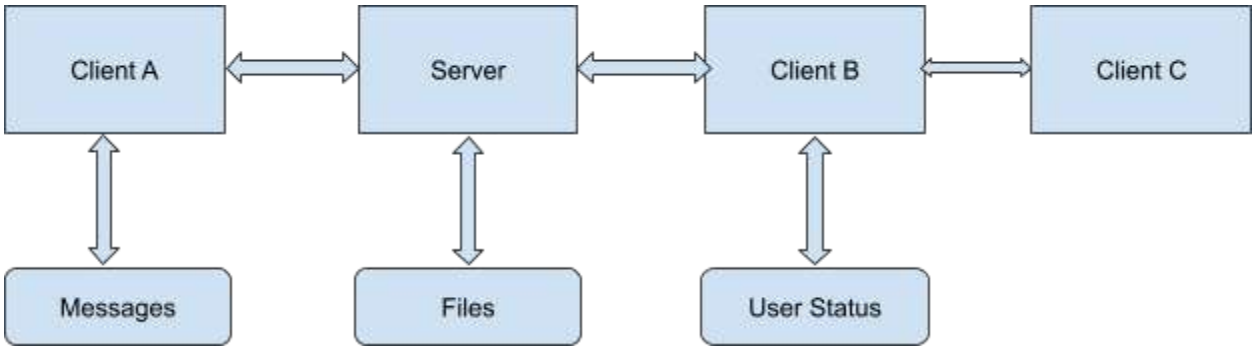
The primary goal of this project is to implement a reliable LAN-based chat system with additional features that mirror real-world messaging platforms. Our objectives include:

- Implementing TCP socket communication using the Winsock API.
- Developing a multi-threaded server capable of handling multiple concurrent clients.
- Creating a client interface for sending and receiving messages in real-time.
- Supporting file transfer between clients via the server.
- Enhancing the console interface with message timestamps and colored outputs.
- Providing essential chat commands like /list, /file, and /exit.

System Architecture

This system is based on a centralized Client-Server model:

- Server:
    - Listens for incoming client connections.
    - Spawns a new thread for each client.
    - Relays messages and file content between clients.
    - Logs all activities (chat and file events) in real-time.
- Client:
    - Connects to the server using IP and port.
    - Sends and receives messages and files.
    - Executes user commands for specific tasks (file transfer, user listing, etc.).



Tools and Technologies Used

Tool/Library	**Purpose**
C++	Core Programming Language
Winsock2 API	Network Programming via TCP/IP
Code::Blocks/VS Code	Development IDE
Windows 11	Operating System
Git	Version Control for code management
MinGW/MSVC	Compiler for building the project
Windows Console API	For colored terminal output

Implementation Overview:

Networking Layer:

The application uses TCP sockets for stable, ordered, and error-checked communication. Winsock2 is initialized on both client and server ends, and proper error handling is implemented.

Concurrency:

The server handles multiple clients using std::thread. Each client connection is managed in its own thread, enabling simultaneous communication without blocking.

Command Interface:

Clients can type standard chat messages or use commands:

- /file filename.txt: Sends a file to other users
- /list: Lists currently connected users
- /exit: Disconnects the client

File Transfer Protocol:

- A special message header FILE:<filename> is sent before sending the actual file.
- The file is sent in binary chunks.
- FILE_END signals the end of file transfer.
- Receiving clients reconstruct and save the file.

User Experience Enhancements:

- Messages are color-coded (e.g., green for own message, yellow for file notices)
- All messages include timestamps like [10:35].
- The client sees their own messages as You: Hello.

Key Features:

Feature	Description
Real-time Chat	Multi-user messaging over LAN
File Transfer Support	Clients can send files through the server
Multi-threaded Server	Each client has a dedicated thread
Colored Terminal Output	Improves clarity and UX
Timestamped Messages	Every message is time-labeled
Self Echo as You:	Sender see You: message instead of name
Safe Disconnection	Clients can gracefully leave with /exit
Server Message Logging	Server sees all messages and file activity

Challenges Faced:

- Concurrency Management: Handling simultaneous clients without data races.
- Partial Transfers: Avoiding incomplete file transfers by enforcing chunking and EOF signaling.
- Crash Prevention: Ensuring the server remains stable even if a client disconnects unexpectedly.
- Text Display: Managing colored output and formatting consistently across different terminals.
- Command Parsing: Cleanly distinguishing between commands and regular messages.

Testing and Results:

Scenarios Tested:

- 2–3 clients chatting simultaneously
- File transfer of .txt, .jpg, and .pdf files
- Stress testing with repeated join/leave
- Edge cases like:
    - File not found
    - File interrupted mid-transfer
    - Invalid commands

Results:

- All messages displayed correctly with timestamps and colors
- Files were successfully transferred between clients
- Commands worked as expected
- Server handled concurrent clients without crashing

Conclusion:

The LAN Chat Room project has successfully demonstrated the power of socket programming using Winsock in C++. We built a multi-user system with messaging and file transfer capabilities, simulating real-time network communication. The enhancements such as colored output, timestamps, and a clean command interface added polish and usability. This project not only met its functional goals but also significantly deepened our understanding of client-server networking and system-level programming.

Future Enhancements:

- Develop a full GUI version of the client using Win32 or Qt
- Implement end-to-end encryption for secure messaging and file sharing
- Add file transfer progress indicators
- Enable LAN auto-discovery using UDP broadcasts
- Store chat logs in files for later retrieval
- Add user authentication or nickname conflict detection

Acknowledgment:

We would like to express our sincere gratitude to our course instructor, Mr. Madan Kadariya, for his invaluable guidance, support, and encouragement throughout this project. His practical teaching methods allowed us to turn theoretical networking knowledge into a functional system that we can proudly demonstrate.