University of Dhaka
Department of Computer Science and Engineering

Encrypted Chatting and File Transfer System with Priority Messaging, Resume Support, and Group Chatting

CSE 3111: Computer Networking Lab

Submitted to:

Dr. Shabbir Ahmed, Professor
Dr. Ismat Rahman, Associate Professor
Palash Ray, Lecturer

Submitted by:

Farhan Bin Rabbani (35)
Tamal Kanti Sarker (39)

Submitted on:

18 November 2025

1 Introduction

1.1 Project Title

Encrypted Chatting and File Transfer System with Priority Messaging, Resume Support, and Group Chatting

1.2 Problem Domain and Motivations

Modern communication systems require secure, reliable, and efficient message and file handling. This project addresses gaps in typical chat applications, including security vulnerabilities, unreliable file transfers, and lack of prioritization.

- **Mo1: Lack of Security**
  Many applications lack robust **end-to-end encryption**, exposing sensitive data to interception.

- **Mo2: File Transfer Reliability**
  Large files such as video, audio, and PDF are often disrupted due to network instability, motivating **interruption detection and automatic resume**.

- **Mo3: Usability and Prioritization**
  Organized group chat and **priority-based message delivery** enhance real-time collaboration and communication efficiency.

1.3 Objectives / Aims

The project aims to:

Ob1: Develop an encrypted multi-client chat system supporting **group and direct messaging**.

Ob2: Implement encrypted file transfer for audio, video, and PDF files with **interruption detection and automatic resume**.

Ob3: Integrate a **priority-based messaging and file-handling system** for efficient delivery of urgent communications.

2 System Features

The system provides the following key features:

1. **Encrypted Text Messaging:** AES-256 encryption is applied to all messages to maintain privacy and confidentiality.
2. **Group Chat Support:** Users can join multiple chat rooms, with server-managed group memberships and efficient message routing.
3. **File Transfer (Audio, Video, PDF):** Supports large files such as `.mp3`, `.mp4`, and `.pdf`.
4. **Interruption Detection and Resume:** Files are segmented, with the last received byte tracked to resume transfer from the point of failure.
5. **Priority Messaging System:** Server-side priority queue ensures urgent messages and control signals are processed ahead of bulk data.
6. **User Management:** Authenticated sessions, active user lists, and routing information are maintained by the server for reliable communication.

2.1 Block Diagram / Workflow

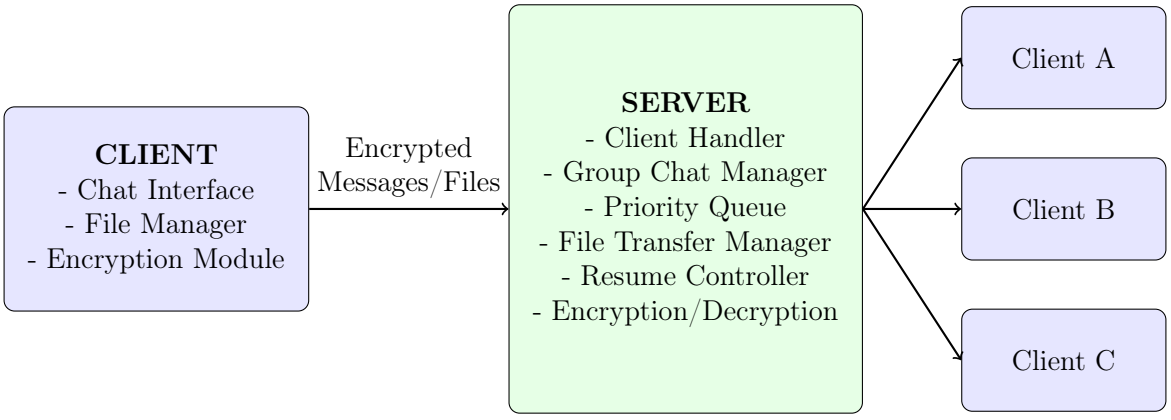

Figure 1: System Block Diagram showing the multi-client network architecture.

2.1.1 Client Components

- **Chat Interface:** The GUI layer for user interaction.

- **File Manager:** Handles file input/output, segmentation for transfer, and reassembly.
- **Encryption Module:** Manages key exchange and applies AES/RSA encryption and decryption.

2.1.2 Server Components

- **Client Handler:** Dedicated thread/task per client for concurrent management.
- **Priority Queue:** Min-heap structure for prioritizing incoming packets.
- **File Transfer Manager & Resume Controller:** Stores transfer state (File ID $\rightarrow$ LSTCI) and coordinates restarts.
- **Group Chat Manager:** Controls group membership and multicasts messages efficiently.

2.2 Security Protocol Design

- **Key Exchange (RSA):** Asymmetric encryption secures the initial connection handshake to exchange session keys.
- **Bulk Data Encryption (AES-256):** Symmetric encryption is used for all messages and file chunks for efficiency.

3 Tools and Technologies

1. **Programming Language:** Python/Java for server-client implementation.
2. **Networking Protocol:** TCP socket programming ensures reliable, ordered data delivery.
3. **Concurrency Model:** Threading/Async I/O handles multiple simultaneous client connections.
4. **Cryptography Library:** Provides secure implementations of AES-256 and RSA.

4 Detailed Implementation Focus

4.1 Priority Queue Implementation

$$\text{Priority Level } (P) = \begin{cases} 1 & \text{Highest: Control packets, session key exchange, short text messages} \\ 2 & \text{Medium: Group chat messages, user status updates} \\ 3 & \text{Lowest: Large file transfer chunks} \end{cases}$$

4.2 File Transfer Reliability and Resume Logic

- **Chunking:** Files are segmented into fixed-size segments C.
- **Tracking:** Resume Controller stores File ID and Last Successfully Transferred Chunk Index (LSTCI).
- **Resume Protocol:** Client reconnects, requests LSTCI, and resumes from chunk LSTCI+1.

5 Conclusion

This project proposes a complete solution for secure and reliable communication. Hybrid encryption, TCP-based file chunking with automatic resume, and server-side priority scheduling ensure a robust platform for real-time collaboration.