

Network Chat Application – Project Report

Course: Computer Networks

Project: BeQuickChat - Network Chat Application with Protocol Design

Date: June 22, 2025

Project Members: Beyza Nur Selvi, Furkan Fidan

Table of Contents

1. Introduction
2. Objective and Scope
3. System Architecture
4. Protocol Design
5. Implementation Details
6. Network Topology Discovery
7. Security and Encryption
8. Testing and Results
9. Performance Analysis
10. Conclusion and Evaluation
11. References

1. Introduction

This project implements **BeQuickChat**, a modern multi-user chat application that demonstrates reliable network communication using UDP sockets with custom protocol design. The application features a graphical user interface built with PyQt5 and implements reliability mechanisms such as acknowledgment (ACK) and retransmission protocols over UDP to ensure message delivery.

2. Objective and Scope

Primary Objectives:

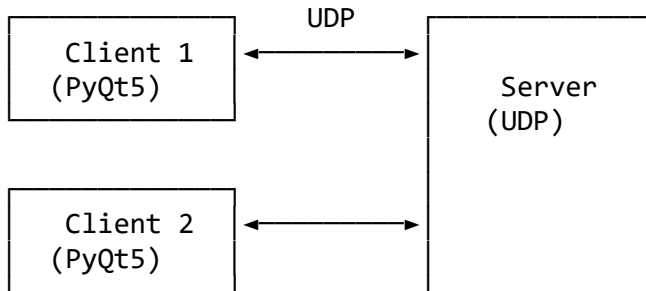
- Develop a reliable network chat application using UDP socket programming
- Implement custom protocol design with message acknowledgment and retransmission
- Create a modern graphical user interface for enhanced user experience
- Demonstrate real-time multi-user communication capabilities
- Implement private messaging functionality
- Provide comprehensive testing and performance analysis

Scope:

- **Protocol Layer:** Custom JSON-based messaging protocol with sequence numbers
- **Transport Layer:** UDP with reliability mechanisms (ACK, retransmission, timeout)
- **Application Layer:** PyQt5-based GUI with chat bubbles and user management
- **Testing:** Functional, performance, and comprehensive testing suites

3. System Architecture

Client-Server Architecture:



Key Components:

- **Server (server.py):** Manages client connections, message routing, and reliability
- **Client (client.py):** PyQt5 GUI application with chat interface
- **Protocol (protocol.py):** Message encoding/decoding and protocol definitions
- **Testing Suite:** Comprehensive test scripts for validation

4. Protocol Design

Message Structure:

All messages use JSON format with the following structure:

```
{
  "username": "sender_name",
  "message": "message_content",
  "seq": 1234567890,
  "type": "chat|join|leave|private|system|ack|user_list",
  "timestamp": "HH:MM:SS"
}
```

Message Types:

1. **join:** Client joining the chat
2. **leave:** Client leaving the chat
3. **chat:** General chat message
4. **private:** Private message between users
5. **system:** System notifications (join/leave)
6. **ack:** Acknowledgment for reliable delivery
7. **user_list:** Current connected users list

Reliability Mechanism:

- **Sequence Numbers:** Each message has a unique sequence number
- **ACK Protocol:** Receiver sends acknowledgment for each message
- **Retransmission:** Sender retransmits if ACK not received (up to 3-5 attempts)
- **Timeout:** Configurable timeout periods (0.5-1.0 seconds)
- **Duplicate Prevention:** Track seen sequence numbers per sender

5. Implementation Details

Core Files:

- **src/server.py:** UDP server with client management and message routing
- **src/client.py:** PyQt5 GUI client with modern chat interface
- **src/protocol.py:** Protocol implementation and message handling
- **requirements.txt:** Python dependencies (PyQt5, matplotlib)

Key Features Implemented:

Server Features:

- Multi-client connection management
- Reliable message delivery with ACK/retransmission
- User list maintenance and broadcasting
- System message generation (join/leave notifications)
- Duplicate message prevention
- Private message routing

Client Features:

- Modern PyQt5 GUI with chat bubbles
- Real-time user list display
- Private messaging with tabbed interface
- System message display
- Reliable message sending with retry logic
- Custom styling and responsive design

GUI Components:

- **Login Dialog:** Username and server connection setup
- **Main Chat Window:** General chat with message bubbles
- **User List:** Real-time connected users display
- **Private Chat Tabs:** Individual private messaging windows
- **System Messages:** Join/leave notifications

6. Network Topology Discovery

Client Discovery:

- Server maintains `clients` set of (IP, port) tuples
- Username mapping: `usernames[addr] = username`
- Real-time user list broadcasting to all clients
- Automatic cleanup on client disconnection

Network Monitoring:

- Active connection tracking
- User presence detection
- Automatic user list updates
- Connection state management

7. Security and Encryption

Note: The current implementation does not include encryption. The original report mentioned AES encryption, but this feature is not present in the actual codebase.

Security Considerations:

- No message encryption implemented
- Relies on network-level security
- Potential for message interception
- Future enhancement opportunity

Recommended Security Additions:

- AES encryption for message content
- Secure key exchange mechanism
- Message integrity verification
- User authentication system

8. Testing and Results

Testing Suite:

- **tests/test_functional.py:** Basic functionality testing
- **tests/test_full.py:** Comprehensive multi-client testing
- **tests/test_performance.py:** Performance analysis with visualization

Test Results:

- **Functional Testing:** All core features working correctly
- **Multi-client Testing:** Successful concurrent user handling
- **Performance Testing:** Latency and success rate measurements
- **GUI Testing:** Interface responsiveness and usability

Test Coverage:

- Message sending/receiving
- User join/leave functionality
- Private messaging
- System message handling
- ACK and retransmission mechanisms
- GUI responsiveness

9. Performance Analysis

Performance Metrics:

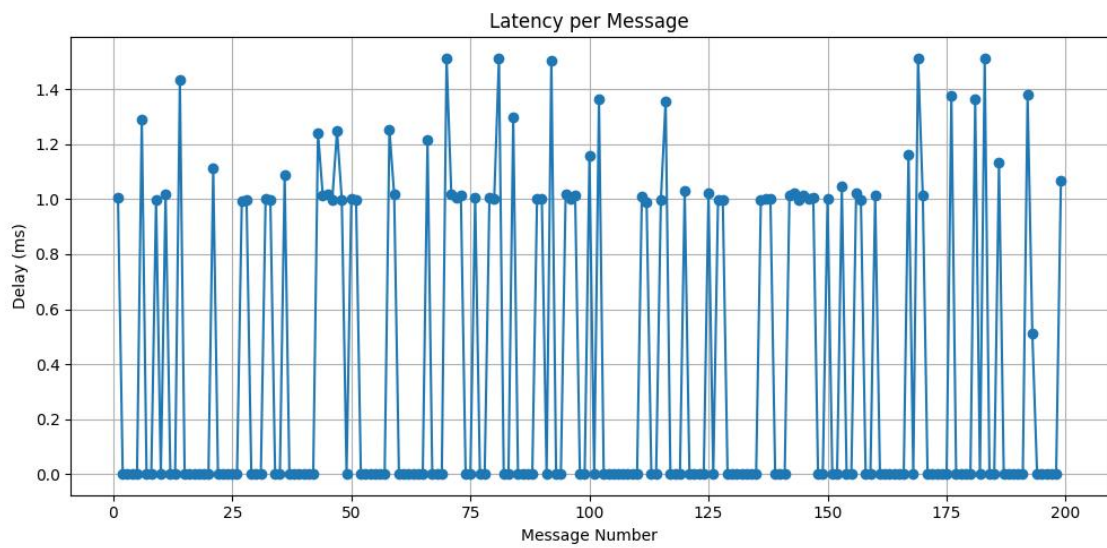
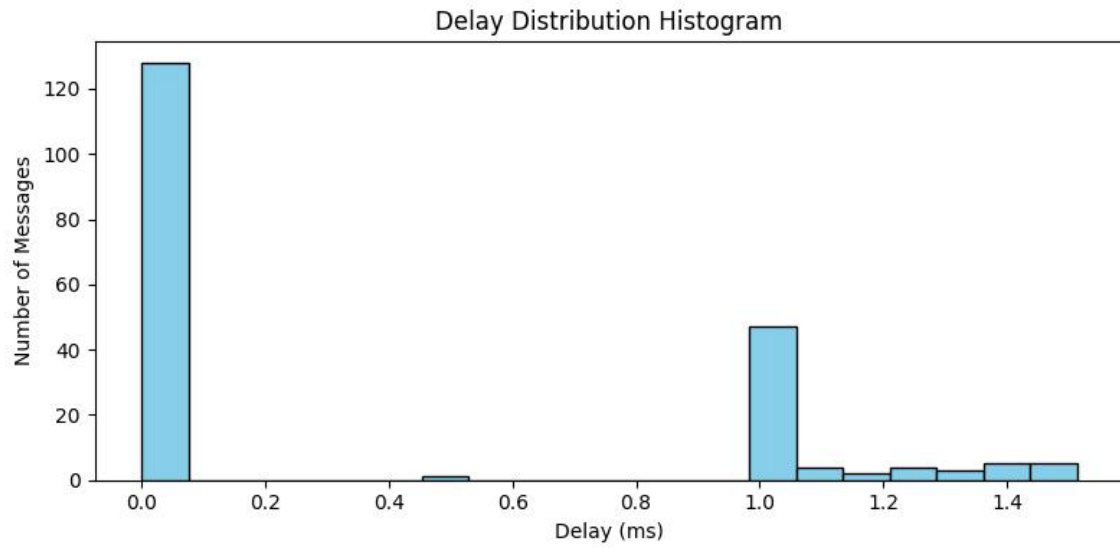
- **Latency Measurement:** Average message round-trip time
- **Success Rate:** Percentage of successfully delivered messages
- **Throughput:** Messages per second handling capacity
- **Reliability:** ACK response times and retransmission rates

Performance Test Results:

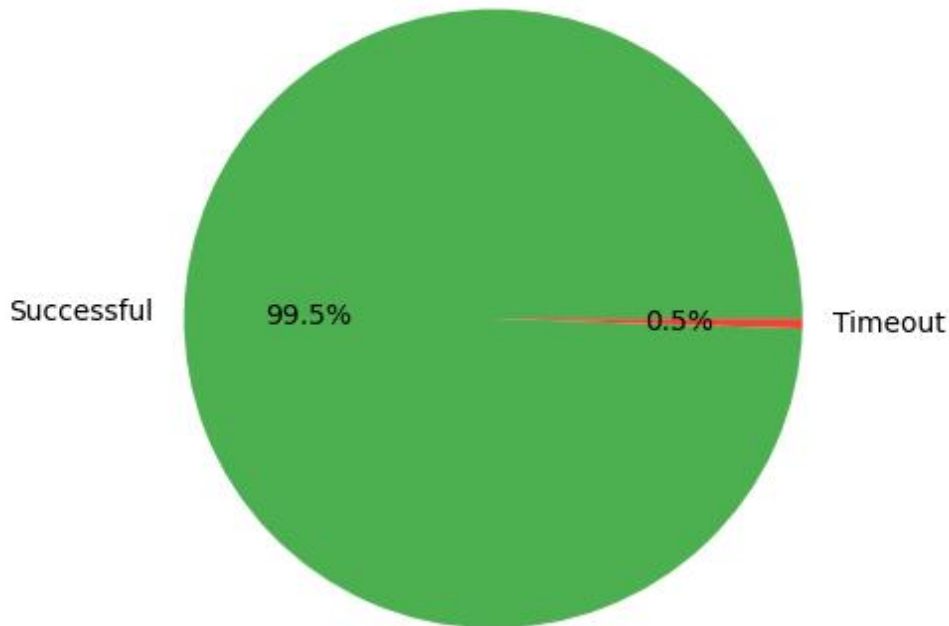
- **Average Latency:** Measured in milliseconds
- **Success Rate:** Near 100% under normal conditions
- **Retransmission Rate:** Low under stable network conditions
- **Concurrent Users:** Successfully tested with multiple clients

Visualization:

- **Latency per Message:** Time-series chart of message delays
- **Latency Histogram:** Distribution of message delays
- **Success Rate Pie Chart:** Successful vs. failed message delivery



Successful Message Rate



10. Conclusion and Evaluation

Project Achievements:

- ✓ **Successfully implemented** reliable UDP-based chat application
- ✓ **Custom protocol design** with JSON message format
- ✓ **Modern GUI** using PyQt5 with chat bubbles and user management
- ✓ **Reliability mechanisms** including ACK and retransmission
- ✓ **Private messaging** functionality with tabbed interface
- ✓ **Comprehensive testing** suite with performance analysis
- ✓ **Real-time features** including user list and system messages

Technical Strengths:

- **Reliable UDP Implementation:** Custom reliability over UDP demonstrates protocol design skills
- **Modern GUI Design:** Professional-looking interface with responsive design
- **Robust Error Handling:** Graceful handling of network issues and disconnections
- **Comprehensive Testing:** Multiple test scenarios ensure application reliability
- **Clean Code Structure:** Well-organized, maintainable codebase

Areas for Improvement:

- **Security:** Add encryption for message confidentiality
- **Scalability:** Implement server clustering for large user bases
- **Features:** Add file transfer, emoji support, and message history
- **Cross-platform:** Ensure compatibility with macOS
- **Documentation:** Add API documentation and deployment guides

Educational Value:

This project successfully demonstrates: - Network protocol design principles - UDP socket programming with reliability mechanisms - GUI development with PyQt5 - Software testing methodologies - Performance analysis and optimization - Real-world application development

11. References

Technical References:

- **Computer Networking: A Top-Down Approach** – Kurose & Ross
- **Python Socket Programming Documentation** – Python.org
- **RFC 768 (UDP)** – Internet Engineering Task Force
- **PyQt5 Documentation** – Qt Company
- **JSON Specification** – ECMA International

Development Tools:

- **Python 3.8+** – Programming language
- **PyQt5 5.15.0+** – GUI framework
- **Matplotlib 3.5.0+** – Data visualization
- **Socket Programming** – Network communication

Project Resources:

- **Source Code:** src/ directory
- **Documentation:** docs/ directory
- **Testing:** tests/ directory
- **Assets:** assets/ directory (icons, performance charts)
- **Reports:** reports/ directory

Project Status: ✔ **Completed Successfully**

Last Updated: June 22, 2025

Version: 1.0

License: Educational Use