Final Project Proposal

For

Network-Optimized SMTP Server

Submitted By

Fiza Amjad
2022-CS-172

Abdullah Chaudry
2022-CS-204

Submitted to

Professor Tehseen

Table of Contents:

1. Project Overview

The goal of this project is to design and implement a **Simple Mail Transfer Protocol (SMTP) server** that facilitates email communication between a client sender, a server, and a client receiver. The server will handle key operations such as email transmission, content verification, directory management, and error handling. This project will not only implement the SMTP protocol but also enhance it with additional features to improve its functionality, security, and usability.

---

2. Scope of Work

2.1 Key Features of the SMTP Server

- **Connection Establishment**: Ensure the server can establish connections with both client senders and receivers, with acknowledgment (ACK) for every step.
- **Email Creation and Transmission**: Allow the client sender to compose an email, attach files, and transmit it securely.
- **Content Verification**: Verify the integrity and format of email content before delivery.
- **Directory Management**: Save emails to appropriate directories based on user or host mappings.
- **Error Handling and Notifications**: Provide immediate feedback for errors in email content, recipient addresses, or server issues.
- **Termination Protocol**: Gracefully terminate the connection when requested by the client sender or receiver.

---

3. Detailed Implementation Plan

3.1 Technologies and Tools

- **Programming Language**: Python (preferred for its networking libraries) or Node.js (as per familiarity).
- **Libraries/Frameworks**:
  - `smtplib` and `socket` in Python for SMTP and low-level network handling.
  - File handling libraries for managing local directories.
  - Security libraries (e.g., OpenSSL for encryption).

- **Platform**: Localhost initially, with potential deployment on a cloud server like AWS or Azure.

3.2 Architecture

- **Client-Server Model:**
  - **Client Sender**: Responsible for composing and sending the email.
  - **Server**: Verifies, processes, and stores the email, then forwards it to the client receiver.
  - **Client Receiver**: Receives, verifies, and stores the email locally.

3.3 Development Phases

1. **Research:**
   - Study the SMTP protocol in-depth and existing implementations.
   - Learn about client-server communication techniques.
2. **Initial Setup:**
   - Set up the development environment and implement a basic SMTP communication model.
   - Test simple email transmission between sender, server, and receiver.
3. **Email Handling:**
   - Add support for attachments and large emails.
   - Implement verification of email content (headers, body, attachments).
4. **File Management:**
   - Implement a system for saving emails to specific user directories.
   - Create a mapping mechanism for email addresses to hostnames.
5. **Error Handling:**
   - Create mechanisms for handling invalid addresses, connection errors, and timeouts.
6. **Security Features:**
   - Encrypt data during transmission (TLS/SSL).
   - Implement authentication mechanisms to verify senders and receivers.
7. **Testing and Debugging:**
   - Test the server with various email scenarios, including edge cases.

4. Enhancements

4.1 Features to Add

- **Email Queuing**: If the recipient server is unavailable, queue the email for later delivery.
- **Graphical User Interface (GUI)**: Develop a user-friendly interface for composing, sending, and viewing emails.
- **Email Filtering**: Implement spam detection and filtering mechanisms.
- **Logging and Monitoring**: Maintain logs of all email transactions for tracking and debugging.
- **Multi-threading**: Enable the server to handle multiple client connections simultaneously.

4.2 Advanced Enhancements

If feasible, advanced enhancements may include:

- **Database Integration**: Store email data in a database for better scalability and data retrieval.
- **Mobile App**: Create a mobile version for better accessibility.

---

5. Benefits

5.1 Practical Applications

- Provides hands-on experience with real-world email communication systems.
- Demonstrates understanding of networking protocols and client-server architecture.
- Offers potential integration into larger systems, such as enterprise email servers.

5.2 Academic and Professional Benefits

- Enhances knowledge of SMTP and email communication.
- Develops expertise in network programming, security, and protocol handling.
- Builds a foundation for working on more advanced projects, such as chat applications or distributed systems.

5.3 End-User Benefits

- Easy and secure email communication.
- Additional features like queuing and filtering improve usability.

---

6. Potential Challenges and Strategies

Challenge	Strategies
Ensuring reliable transmission	Use acknowledgment mechanisms at every step.
Managing file attachments	Implement file size limits and compression.
Securing email data	Use TLS/SSL encryption and authentication.
Handling multiple clients	Use multi-threading for simultaneous connections.

---

7. Conclusion

Building an SMTP server provides a robust learning opportunity in network programming and protocol implementation. Enhancing the project with additional features like encryption, queuing, and AI-based spam detection ensures that the system is not only functional but also practical and scalable. The project also opens doors for integration with enterprise-level systems or future research.
