|  |  |  |
| --- | --- | --- |
|  | National University  of Computer and Emerging Sciences  Chiniot-Faisalabad Campus |  |

**Computer Networks**

**P2P File Sharing & Group Communication**

**Fall 2024**

|  |
| --- |
|  |

**Mustehsan Nisar (22F-3281)**

**Umar Anjum (22F-3292)**

**Imran Ali (22F)**

**Submitted To**

“Mam Rabia Anwar”

1. **Introduction**

Peer-to-Peer (P2P) communication networks provide a decentralized method of connecting devices to share resources or data directly without the need for an intermediary server. This project implements a P2P communication application using C#, emphasizing efficient and secure data transfer. The core objective was to create a robust system capable of file sharing, group communication, and enhanced security without relying on external APIs.

The application employs a hybrid architecture combining central server functionalities for coordination and direct peer-to-peer interactions for communication and data transfer. The system was designed to optimize performance and reliability, ensuring it can handle failures and provide secure operations.

1. **Features**
2. **Utilization of C#**: The entire application was developed in C#, leveraging its low-level features to ensure efficient execution and fast processing.
3. **No External APIs**: All functionalities, including networking and encryption, were developed from scratch without relying on third-party APIs, providing full control over the implementation.
4. **Backup Server**: A failover mechanism ensures continuous operation by seamlessly switching to a backup server if the main server fails.
5. **Email Authentication**: Before joining the network, peers are authenticated via email for added security.
6. **Encryption and Decryption**: Messages and file transfers are encrypted to ensure data security during transmission.
7. **File Transfers**: Supports transferring various file types, including text files, images (JPG, PNG, JPEG), and more.
8. **Group Communication**: Enables real-time group chats with multiple peers, including file sharing and emoji support.
9. **Broadcasting Approach**: The system efficiently broadcasts messages and updates across peers in the network.
10. **Challenges and Solutions**
11. **IP Address Conflict**:
    * **Challenge**: When running the application on the same device, all forms generated the same IP, preventing communication between peers.
    * **Solution**: A unique number was appended to the IP address before storing it in the server’s dictionary, allowing proper differentiation.
12. **Peer List Issue**:
    * **Challenge**: Peer lists were not saving due to identical IP addresses across instances.
    * **Solution**: The same approach of appending a unique identifier to IP addresses resolved the issue.
13. **Group Chat Race Condition**:
    * **Challenge**: Without external APIs, establishing peer-to-peer connections during group invitations caused a race condition. This was especially challenging as all operations occurred on a single device, preventing proper socket creation.
    * **Solution**: Threading was employed with a 100-millisecond wait introduced for each invite to prevent race conditions and ensure successful socket creation.

#### ****Encryption and Decryption****

* **Functionality**: Implements secure data transfer by encrypting and decrypting messages and files exchanged between peers.
* **Library**: using System.Security.Cryptography;

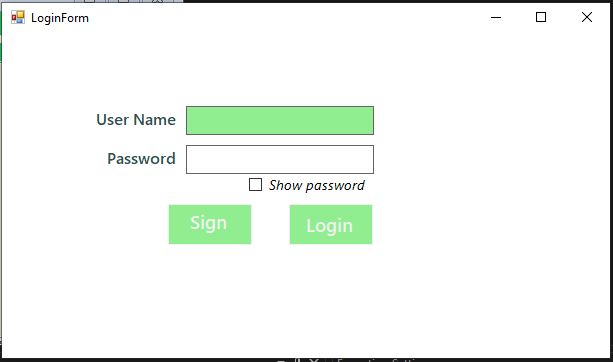
Asymmetric Encryption (RSA or ECC) for Key Exchange

Use Case: RSA and ECC (Elliptic Curve Cryptography) are ideal for securely exchanging encryption keys over an untrusted network.

Why: With asymmetric encryption, a peer uses a public key to encrypt data, and only the peer with the **corresponding private key can decrypt it.**

1. **Workings and Outputs**

**Login & Sign-Up Form:**

****

**A screenshot of a computer

Description automatically generated**

**Peer Interface:**

**A screenshot of a computer

Description automatically generated**

**Peer List:**

**A screenshot of a computer

Description automatically generated**

**Group Chat Setup:**

**A screenshot of a computer

Description automatically generated**

**P2P Communication:**

**A screenshot of a computer

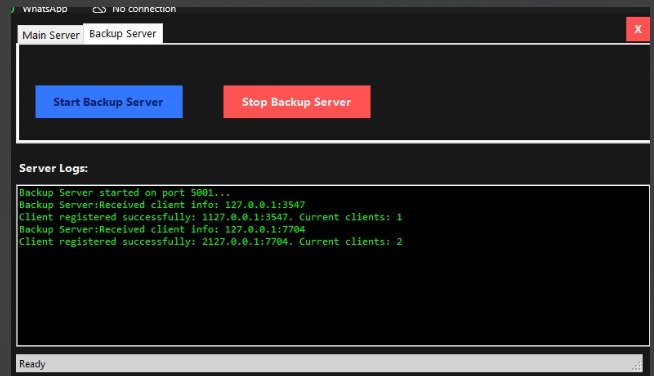
Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Back UP Server:  
**

**Main Server:**

**A computer screen shot of a computer program

Description automatically generated**

**File Sending From One Peer To Other:**

**A screenshot of a computer

Description automatically generated**

**Communication Between Peers:**

**A screenshot of a computer

Description automatically generated**

**File Received:**

**A screenshot of a computer

Description automatically generated**

**Server End:**

**A screen shot of a computer

Description automatically generated**

1. **Conclusion and Future Work**

Developing this P2P communication application in C# presented significant challenges due to its low-level nature. Resolving errors and implementing complex functionalities from scratch often required substantial time and effort, including rethinking and reimplementing entire logic structures when issues arose. Despite recommendations to use higher-level languages like Python or JavaScript, we persisted with C# to prove its viability for such projects.

This project served as an excellent learning opportunity, highlighting the intricacies of networking, multithreading, and secure communication. Moving forward, we aim to explore similar P2P applications using high-level languages for easier development and integrate advanced AI/ML algorithms to enhance functionality and user experience.