Semester Project

P2P Communication Application

**Deadline: 8-12-2024, 11:50 PM**

# Introduction

The term P2P refers to peer-to-peer networking. A peer-to-peer network allows computer hardware and software to communicate without the need for a server. Peer-to-peer file sharing refers to the distribution of digital media over a P2P network, in which the files are located on individuals' computer and shared with other members of the network, rather than on a centralized server.

Software’s that are already using this method are: **Napster**, **Gnutella** etc.

# Description

* 1. Use **Hybrid Peer-to-Peer** architecture for P2P communication application.
  2. Language/Technology: you are free to use any language/technology to implement this application.
  3. Implement Graphical User interface (you can use WebSocket too)
  4. P2P Communication Application Modules description:
     1. **File sharing**
        1. There is a single central server (**server.c**) and at least 4 peers. Peers are different instances of the same code (**client.c**). The server should be able to handle multiple peers simultaneously and also maintains the joined peers list to show to all the connected peers in the network.
        2. All peers know the central server details. This can be accomplished via a command- line argument. Example: **./client server-ip server-port**
        3. Each peer also has its own local folder named **'p2p-files'** which is used to store files it wishes to publish as well as one’s which it fetches from others.
        4. Each peer supports **duplex operation** i.e. it should be able to transmit (upload to another peer) as well as receive files (download from another peer) simultaneously. Multiple uploads and downloads are also be supported.
        5. In the network, each peer will **act as a server and/or a client** based on the requirement.
        6. A peer can send file without any request from the other peer(s) (self-attachment) while the receiver will get a prompt on screen having discarding or saving the file option. (Just like in whatsapp application
        7. Code works over any type of file: **pdfs, doc, jpg, png, txt etc**.
        8. **File Version Control:** Allow peers to share different versions of the same file. Each version should be tracked, and peers can request specific versions.
        9. **File Preview**: Before downloading a file, peers can preview a portion (for text files or images) to verify its contents
        10. **Automated File Sync**: Peers can opt to automatically sync updated files from others, ensuring they always have the latest version.
        11. **Peer Authentication**: Before sending or receiving a file, each peer should authenticate via a unique key or password.
     2. **Group Chat Room**
        1. You have to perform **encryption and decryption** while you are exchanging messages between peers. Note: you can use any encryption/decryption algorithm for this purpose.
        2. You have to create a group chat room for the peers in which the peer(s) can communicate with all the peers present in that group.
        3. Each peer can **send**, **receive** and **delete** the message(s).
        4. In chat, peer can use **simple text**, different **emoji’s** and can also **share file.**
        5. Central Server functionality in this module should be just to provide the all-peers list. The Peer will use this list to create a group for the chat and name that group. (You can choose a default name for group.)
        6. **File Version Control**: Allow peers to share different versions of the same file. Each version should be tracked, and peers can request specific versions.
        7. **File Preview**: Before downloading a file, peers can preview a portion (for text files or images) to verify its contents.
        8. **Automated File Sync**: Peers can opt to automatically sync updated files from others, ensuring they always have the latest version.
        9. **Peer Authentication**: Before sending or receiving a file, each peer should authenticate via a unique key or password.

# Operation of Peers:

* 1. **Join**: A Peer connects to the network
  2. **Publish**: Peer informs peer(s), information about their shared files (name and location)
  3. **Query**: Peer queries the peer(s) about the file index
  4. **Download**: Peer randomly chooses any of those peer(s) who has the required file and download directly from it. On finishing download, it publishes that it also has a copy of that file to other peers.
  5. **Leave**: Peer releases connection to other peer(s) **File Version Control**: Allow peers to share different versions of the same file. Each version should be tracked, and peers can request specific versions.
  6. **File Preview**: Before downloading a file, peers can preview a portion (for text files or images) to verify its contents.
  7. **Automated File Sync**: Peers can opt to automatically sync updated files from others, ensuring they always have the latest version.

# Peer Authentication: Before sending or receiving a file, each peer should authenticate via a unique key or password.

**4 Network Enhancements**

* **Load Balancing**: Add a feature where the central server distributes file-sharing and chat traffic evenly across peers to avoid overloading any single peer.
* **Failover Server**: Implement a backup central server that takes over if the primary server fails, ensuring minimal disruption.

**5 Security Features**

* **End-to-End Encryption**: Expand encryption to include all types of file transfers, not just chat messages.
* **Multi-Factor Authentication**: Before a peer can join the network or share files, require multi-factor authentication (e.g., password and email/SMS code).
* **Intrusion Detection**: Add a system that monitors and alerts when suspicious peer behavior occurs, like excessive file requests or potential flooding.

This enhanced version introduces more comprehensive functionality while preserving the core ideas of file sharing and peer-to-peer communication.

Let me know if you'd like further elaboration on any specific module!

**NOTE**: you can use special characters (#, $, @ etc.) to differentiate between operations. (For example: for any attachment use $- it will alert the system that user is going to attach something.)