Report- Peer to Peer Distributed File Sharing System
AOS Assignment 3

1. Socket Programming
Socket programming forms the backbone of communication between clients and servers in a peer-to-peer (P2P) file sharing system. It allows clients to connect with each other and transfer data over a network. Here are key components of socket programming used in this system:

Socket: This represents the communication endpoint. In a client-server model, both the server and the client create sockets for network communication.
Bind: The server binds its socket to a specific IP address and port, allowing it to listen for incoming client connections.
Listen: The server listens for incoming connections on the bound socket.
Accept: Once a client tries to connect to the server, the server accepts the connection, which returns a new socket dedicated to communication with that specific client.
Connect: A client uses connect() to establish a connection to the server's IP and port.
Send/Recv: Both client and server use send() and recv() to transfer data through the socket once the connection is established.
Close: Once the communication is complete, the socket is closed.

2. Multithreading
Multithreading is essential to handle multiple simultaneous connections in the system. For example, a server needs to handle multiple clients downloading different parts of a file at the same time, and a client may download different chunks of a file from multiple peers concurrently.

Thread Creation: When a client connects to a server, a new thread is spawned to handle communication with that specific client. This allows the server to accept other connections simultaneously.
Thread Detachment: The thread responsible for handling a specific client can be detached, allowing it to run independently, so the server can handle new incoming connections without waiting for previous connections to finish.
Synchronization: Mutex locks are used to avoid race conditions when multiple threads access shared data, such as the file chunks being downloaded or the list of available peers.

Implementation Details for following commands:
1. Create User Account: A map is maintained to list the users
2. Login: A map is maintained to list the users already logged in
3. Create Group: A group class is create which contains various details regarding the group
4. Join Group: Within group class a map named peers_requesting stores the data.
5. Leave Group: Within group class a map named peers_connected removes the user from group
6. List pending join: peers_requesting map is iterated
7. Accept Group Joining Request: The user is moved from peers_requesting to peers_connected.
8. List All Group In Network: A map called all_groups stores all the groups
9. List All sharable Files In Group: Within group class a vector named sharable_file stores the file names.
10. Upload File: Within tracker.cpp the Group class object is updated with the informatiion provided along with that SHA-1 is also calculated in client file to ensure intergrity of data send
11. Download File: One entire file can be downloaded from one client to another at a time. Using socket programming concepts. Connection between two peers is establised and then files contents are send.