AOS Assignment-3 [P2P File Sharing] Report

This system consists of two main components: Client and Tracker, each with specific roles and responsibilities. Below is a detailed description of the structures, functions, and their usage in the implementation.

Implemented Features

1. Authentication.
   This involves authenticating the clients by the centralized tracker. Only one session can be active at once. Once connected and logged in, ip and port details of the client will be shared with the tracker.

2. Group Creation and Maintenance
   Implemented commands like create_group, list_groups, accept_request, leave_group, join_group are implemented to facilitate for easy sharing of files among peers belonging to same group. Also handled cases of leaving the group and transfer of owner rights to next member. Implemented group features in such a way that any user who wants to join group need to get approval from the owner. All of this is managed by the tracker.

3. File Upload and Download
   Files can be uploaded to a group by any member, once uploaded the client becomes a seeder for that file and all the clients in that group can download that file. While upload implemented a system to store all the meta info of that file including the hashes of 512KB chunks and that of the total file to the tracker. So any other client requests can be served with this data. Downloads are allowed only if the client is a member of that group. Also implemented features like stop_share to remove a client as a seeder for a file if he wants so.
4. Show_downloads command is implemented on the client side to report the status of all the downloaded files.
5. Logout logs the client and also pauses him as a seeder so that any other clients requesting a file from this seeder can save time by not trying to connect with this logged out seeder.

For calculating file hashes, this implementation uses SHA1 of the entire file and the SHA1 of 512KB chunks to preserve integrity of the files at microlevel. These hashes are matched by the sender to verify integrity and try to redownload if there are any corrupt chunks.

Structures Used to store data and manage

A. Client

a) **struct FileInfo**: - Attributes: fileName, fileSize, fileHash, chunkHashes, relativePath - This structure is used to hold the information about files that are shared by clients. It contains the file name, size, the total hash of the file, individual hashes of its chunks, and the relative path where the file is stored.
b) **struct DownloadedFileInfo**: - Attributes: fileName, location, groupName, status - This structure stores information about files that have been downloaded, such as the file's name, its location on disk, the group to which it belongs, and the current status of the download.

B. Tracker

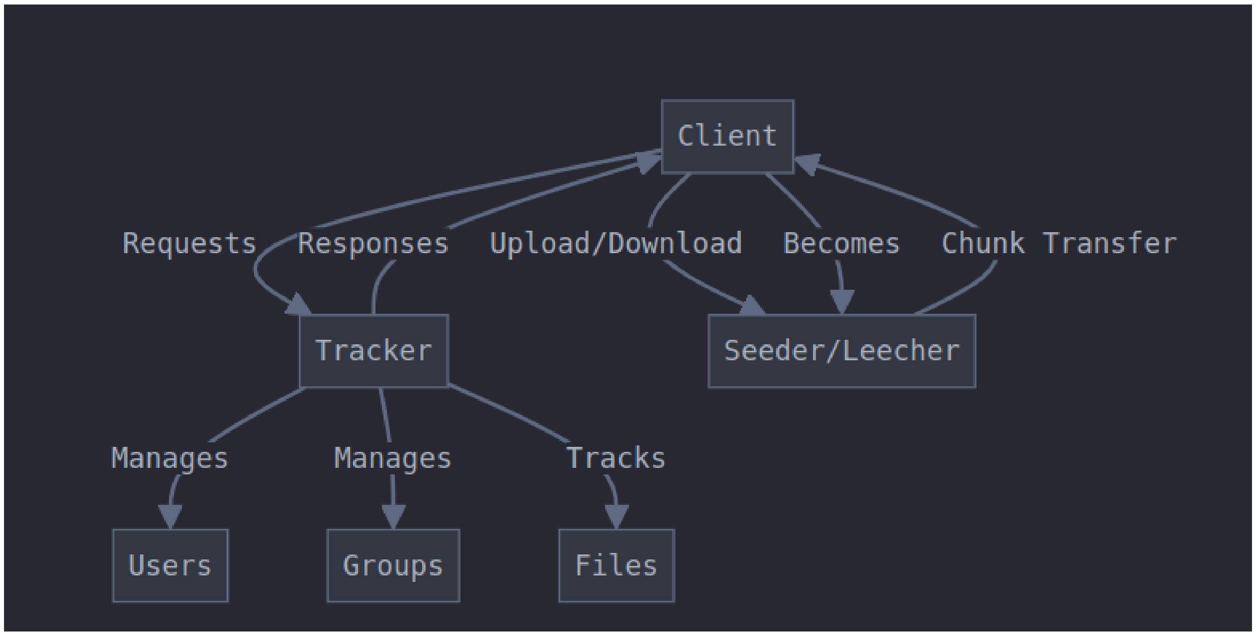
a) **struct User**: - Attributes: username, password, isLoggedIn, groups, ip, port - This structure is used to manage user information such as login status, group memberships, and connection details (IP and port).
b) **struct Group**: - Attributes: groupId, owner, members, pendingRequests, fileHashes - This structure stores details about each group, including its members, the owner, and any pending join requests.
c) **struct File**: - Attributes: fileName, fileHash, fileSize, chunks, seeders, partialSeeders - This structure holds metadata about files in the tracker, including the file's hash, its size, a list of chunks, and the seeders that have the file.

Program Flow

First, tracker starts executing and starts listening for incoming requests and accepts the clients, the client is multithreaded to send and receive commands to the tracker and also listens to any incoming file download requests from its peers.

Tracker only serves as a meta data provider and authenticator for all the peers. It doesn't involve in any downloading operations.

The below flowchart demonstrates how this flow works


Client
Requests
Responses
Upload/Download
Becomes
Chunk Transfer
Tracker
Seeder/Leecher
Manages
Manages
Tracks
Users
Groups
Files