Programming Assignment#1
A Simple Napster Style Peer to Peer File Sharing System
Design Document
Jinyang Li
A20317851

Over All:

This project consists of two parts

1. Server Side. – An index Server
2. Client Side. – A Typical p2p client that could download from other clients and upload file to other peers.

Server Side Design-:

The Server is designed as a multiple thread server that wait for client to establish connecting over socket. Like common socket Server-Client system, it can handle as much clients as possible in JDK's define.

The Server has Three parts, I abstract them into 3 classses:

1 CentralIndexServer

Handle income Connections, and process each client socket to GateWay. Each Client incoming would initialize a new runnable GateWay instance in a child thread.

2 GateWay

Handle Client requests, for each Client, GateWay filter their reason for connecting by:

A) Registration of Files

invoked by a peer to register all its files with the indexing server. The server then builds the index for the peer.

File is abstracted into a "SharedFile" object, which contains peer's internet address, file size, file path in peer's machine, peer's uploader module port and file md5. Thus, this info can send back to peer when they requested, thus they have all the information that needed to download that file.

After Client invoke this function via socket, GateWay will redirect object to FileServer to store it.

.

B) Search File

This module is used for searching a registered file in Server, however, it only returns files that has a valid peer connection. So the client can truly download it if they are interested in.

C) Download File

This module is used for downloading a registered file in Server, however, it only returns files that has a valid peer connection, filted by checkAlive method. So the client can truly download it without any hassle.

D) List Current File

This module returns all files that registered in Server. For files that has no valid peer sharing point, it would explain it to Client. So the Client know what they can actually download on this P2P network.

E) Exit Connection

This module is just invoked in case Client want to close connection. Server would close its socket and exit thread

3 FileServer.

File server is used for store sharedFiles objects, and making file related operation SharedFile Objects are stored in a hashmap

```
/**
* The map of shared files.
*
* The key is the checksum of a shared file. So, the file would be unique.
* The value is an ArrayList which stores the meta data of the share file
objects
*
*/
private static final HashMap<String, ArrayList<SharedFile>> sharedFiles = new
HashMap<>();
```

File Server has five member functions, called by GateWay, to perform file related operation.

```
public synchronized static String registry(SharedFile sf);
```
This method registers the file with the indexing server, synchronized for thread safe;

public SharedFile[] searchFile(String name)
This method search the file with the indexing server;

public boolean checkAlive (SharedFile sf)
This method check wether the file with a valid peer sharing point;

public String listFile()
public String listFileInArray(ArrayList<SharedFile> al) {
These two methods used for listing files.

Client Side Design:

Each Client can request Server to register, download, search, list files.

Each Client also open a peer server as a child thread that listening on an automatic allocated port to receive and incoming downloads request.

The child peer server's theory is very similar to my gateway design, each incoming socket would start another thread to make download operation, thus a peer can sharing his files to many other peers.

After receive downloading request, peer server would allocate file as byte array, and send this byte array to requesting peer.

Data Structure Design:

I abstract files as SharedFile Object contains file metadata

```
/**
 * The constructor of this class.
 *
 * @param checksum   file's md5
 * @param fileName  the file name of the file
 * @param peer    the peer who share the file
 * @param size      the size in Byte of the file
 */
```

thus it can used for transforming files over socket.

Future Work and Trade off:

In currently implements, when a client request to download a file, the server always return first usable download info to that client. In future, can extend to filter result array, and return the fast download peer info to client.

In currently implements, Client can only share/download/search one file at one time, future work can extend this to multiple files. Also, a GUI is very friendly to real user.

Actually, it can assign custom port and bind to custom address, but for learning propose and convenient, I hardcoded the port for Index Server. This is a tradeoff.

Another tradeoff is I'm using step-by-step communication over Client and Server, when perform actions, Client must send a request that indicating the reason, it may causes delay.

Known Bugs:

When you are on an external internet environment, the server and clients ports are open to public, thus may causes safety issues.

When Client A is downloading file from Client B for File X, if accidently close Client B, Client A would throw exception, although I handle it in a try-catch block, but I haven't cleaned up the sockets..etc.