

Programming Project 2

CS5352 Advanced Operating Systems Design

Spring 2018

*Napster Style Peer to Peer File Transfer*

(Design Documentation)

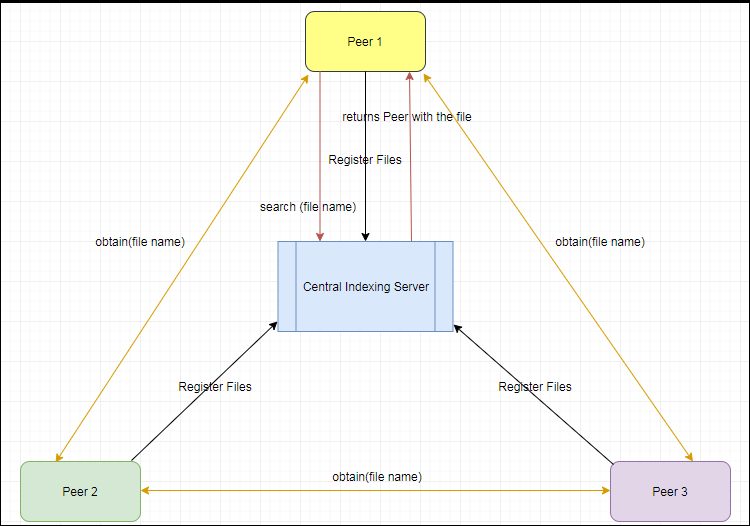
by

Abhishek Kumar

**Implementation:**

For this project we have used sockets to connect clients (peer) to server (central indexing server). So, when the server is online, the clients (peers) can connect with it to register their files. Once the clients register all its files on to the server it can start the search method using a filename to search the peer/peers containing that file.  
Client can choose which file to download among the list of peers containing that file and also should enter the port number of that particular peer where it acts as a server.

File is then downloaded which is reflected in the shared directory.

**Design**:

**Working:**

*Server side:*

File: Server.java

Server gets online using a hardcoded port number which is common to both clients and server. Once the server is online it waits for a client to connect.  
After connection is established, the client sends serializable object to server's InputStream. The server deserializes the object containing information on whether client has invoked registration of files or a search.  
Server performs the requested functionality and returns the result using the same serialized object back to the client.

Registering Files:

To register files Server creates an ArrayList of the Data class containing information fields for peers to keep track of all the files registered by each peer. Each peer invokes the same register method.

Search Files:

To search files server iterates through the ArrayList to obtain all the peer ids containing the file passed to this method. All the peer ids are then passed to the client.

*Client-side:*

File: Client.java

Once the client connects to the server user chooses whether to search for a file or register files to the Server. On registering files, user enters the shared directory of the file from which all files are fetched and registered in the Server.   
When user selects to search for a particular file, filename is sent to the Server in the form of Serialized object, clients InputStream receives a list of all the peers containing the file and displays this list to the user.

On selecting from which Peer the file needs to be downloaded and entering the port number for that peer at which it will act as a server, establishes connection with Peer containing the file which is acting as a server and downloads the file by using byte stream to receive file as a stream of bytes from the peer Server.

Download File:

For downloading a file user selects from the peer list to download the file. Then user needs to enter the port number at which that client would act as a server. Based on that port number client creates a connection by creating a socket with the peer containing file and downloads the file.

**Trade-offs and Improvements**

Client and Server would form a connection regardless of the platform in which they are implemented, since Implementation of sockets makes this Client Server model platform independent. But in general, when a client connects to the server upto that time server continuously waits for request from a client leading low performance.

Also the InputStream of client continuously waits until it receives a data from client. To avoid this, client and server are sync in a loop such that after every data sent from the client control goes to the server input to store that data, this operation is performed in a loop until client has no data to send to the server.

**Improvements:**

We can use efficient threading concepts to handle multiple concurrent requests from each clients, to decrease the response time.

It needs GUI to be user friendly.

Enhancing the performance by apply load balancing approach on the server side.