Ashwin Chati **A20354704**

Design Document:

Napster style p2p file sharing system

Problem Statement:

To design a Napster style p2p file sharing system. The system should be multi threaded which will make server and client handle multiple requests at the same time.

The project should contain simple p2p system with two components:

1. A central Indexing Server : Indexing server should be able to perform following functions:

a) **Register** : Centralized server should hold the information about all the peers connected to it at a given time. The server should also hold the information about the files on the peers.

b) **Search :** The centralized server should able to search requested file location in the index & return the addresses of the peers holding that file to the requesting client.

2. A Peer : A peer can act two roles

a) Client : Client should be able to make the request to the centralized server and register the filenames. It should also be able to make the request to the server for the file it is searching. The server will return the the peer information on which the file is located. The client should be able to make the connection with the specified peers and download the specified file.

b) Server : As a server,on getting request for a specified file the client should be able to send the file to requesting peer.

System Architecture

The high level architecture is shown in the given figure.

Server : Maintains directory of files and peer IDs who contain those files and peer IDs . It can handle multiple clients at the same time.

Peer (Client) : Peer connects to the server and registers its files and make them available for downloading for other peers. Also a peer can search for a file on the server and request to download that file from other peer who is holding that file.

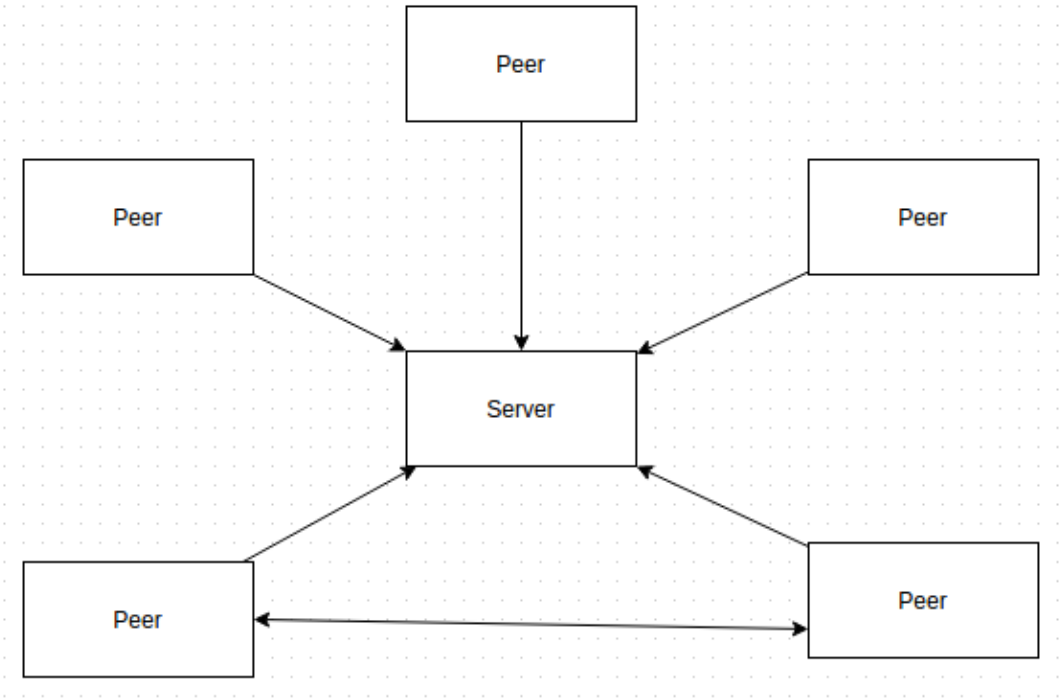

Fig 1.1 System Architecture

Functionality:

1. **Server :** Server can perform below two functions.

a. Register: This function is used to register the information about files and the peers on which those files exists. To implement this functionality a hashmap is created. Hashmap holds key-value pairs . Here key is the file name and value is the address of the file that is the peer information on which the file exists. When client is connected and wants to register a new file then entry map is created.

b. Searching: This functionality is used to process search request given by the peer. When server received request to search a file, it searches the hashmap with key as a filename. If the file entry is present in the hashmap then it sends the peer details to the peer.

2. Peer:

a. Connect to Server: Peer connects to the server and registers its files or search for the files. All the communication is done using sockets.

b. Act as a server for other peers: Peer and server are multi threaded so that they can handle multiple request at the same time.

c. Downloading: Streams are used to send and receive download the files from one peer to another.

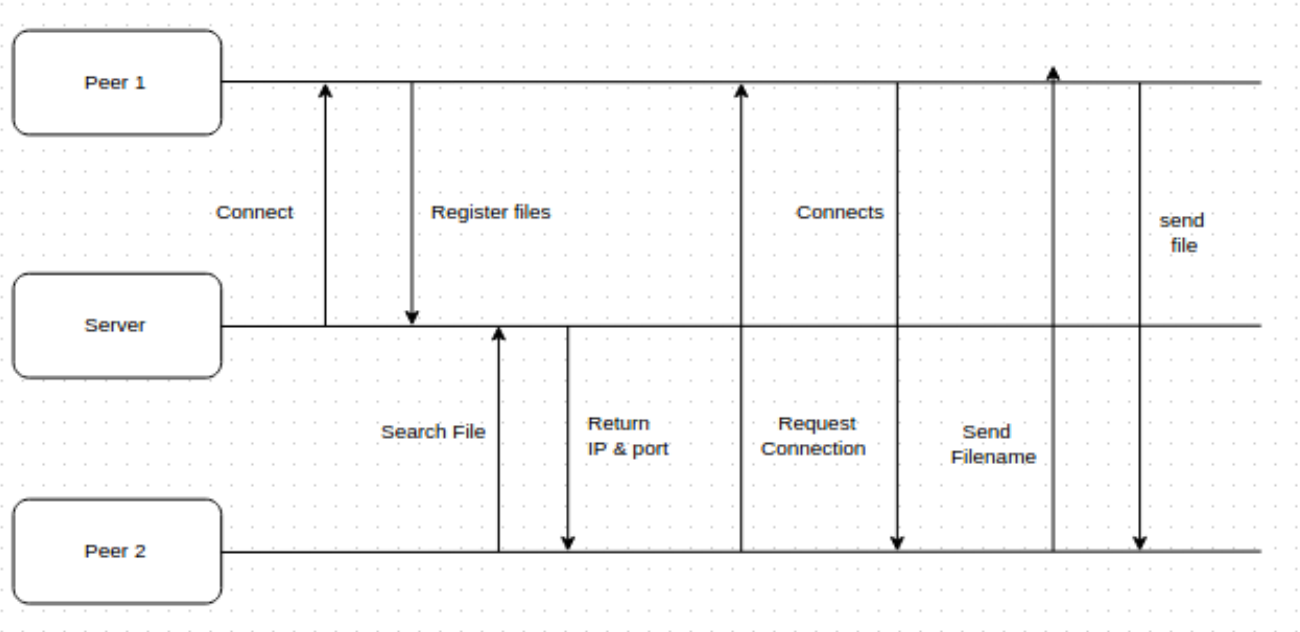

Fig 1.2 functionality

Class Diagram:

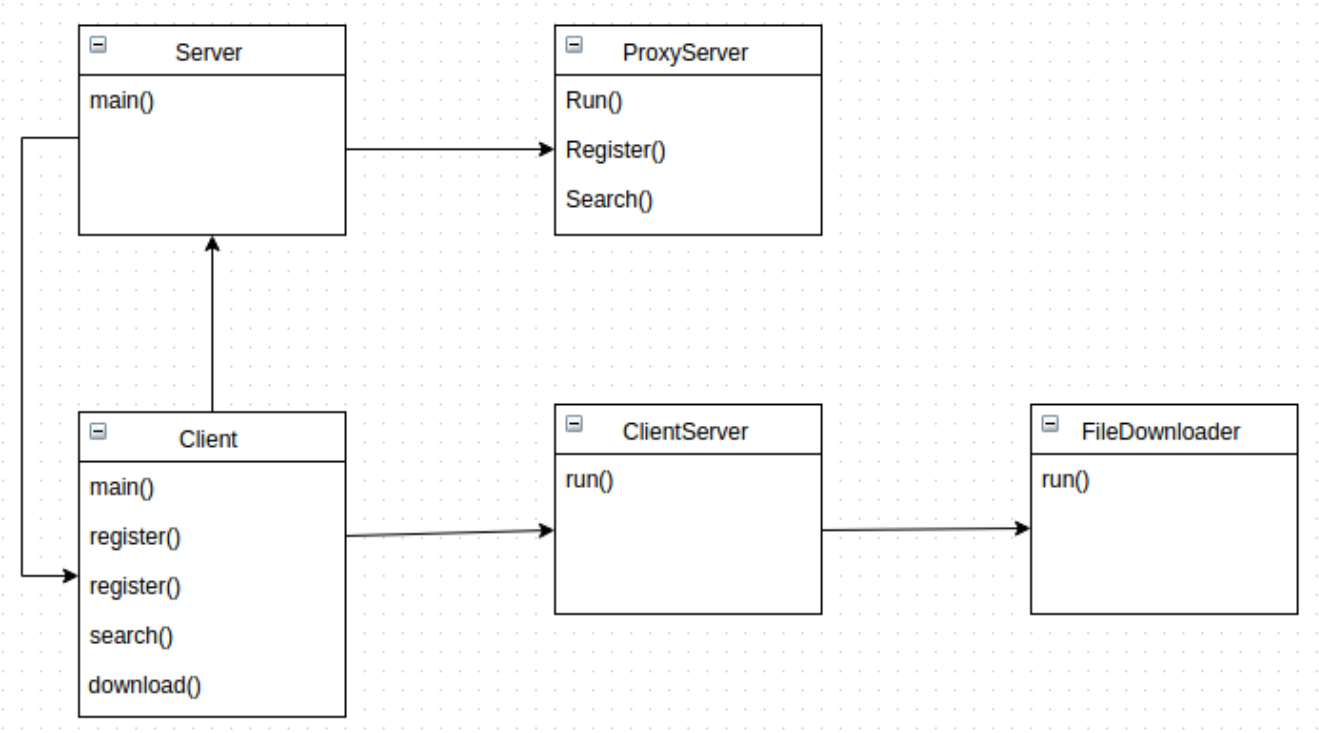

Fig 1.2 Class Diagram

There are five main classes in the system:
1. **Server:** This is the main server class. When it up and running it keeps listening for the peer request to connect. When the peer gets connected it generates a new thread and keeps listening for other peers.

2. **ProxyServer:** This class is used to process the threads initiated by the main Server class. Register and search requests are processed by the this class.

3. **Client:** This is main Client class. It registers its files to the server. Also it can search for the files on the server.

4. **ClientServer:** This class is to implement the client as a server. It processed the requests from the peers.

5. **FileDownloader:** This class is used to process the thread form the peers for downloading the file.

System Requirement:

1. Operating System: Windows/Linux
2. Softwares: a. java
b. Ant

Future Scope:

1. Implementation of a better algorithm to search the files.
2. Implementation of data replication.
3. Implementation of the partial match functionality of the file.