# File Sharing System Design

**Si Liu**
**A20334820**

## 1. Introduction

This File Sharing System is a simple P2P system. It mainly includes two components: index server and peer. The index server indexes the contents of all peers that register with it, which is invoked by a peer to register all its files to the server. Then the server builds the index for the peer accordingly. The index server also provides search facility to peers, which searches the index and return all the matching peers to the requestor. A peer is both a client and a server. As a client, the user specifies a file name with the indexing server using "lookup". The indexing server returns a list of all other peers that hold the file. The user can pick one such peer and the client then connects to this peer and downloads the file. As a server, the peer waits for requests from other peers and sends the requested file when receiving a request.

## 2. Environments:

a. Operating System: Linux (Ubuntu), 12G Memory, 8 CPU cores
b. IDE: Eclipse
c. Language: Java
d. Complier: Ant

## 3. Analysis of System Requirements (Use Case)

Based on the system requirements, we analyzed the basic system functions that should be provided to users as shown in Figure 1.

For the peer, it should provide six basic functions, which are triggered by commands from users.

- **Register**: When user types in “register” in the peer client, the peer starts sending register request to the index server to register all files in its shared folder.
- **Search**: When user types in “search” and file name in the peer client, the peer starts sending search file request to the index server to get the locations of the file.
- **Lookup**: When user types in “lookup” and peer ID in the peer client, the peer starts sending loop up request to the index server to get the peer address (IP and port).
- **Obtain**: When user types in the “obtain” and peer ID in the peer client, the peer starts sending download request from remote peer to achieve the targeted file.
- **Delete**: When user types in “delete” and file name in the peer client, the peer starts sending delete file request to the index server to delete the entry in the registration table.
- **Send file**: A peer is also a file download server, which sending file to the requestor when receiving “obtain” file request.

For the index server, there are four basic functions provided with responding to the requests from peers: add file-peer entry to registration table; search file location from registration table; loop up peer address from peer management table; delete file-peer entry from registration table.

In addition to the basic functions stated above, both index server and peer should be able to handle multiple requests at the same time without blocking.

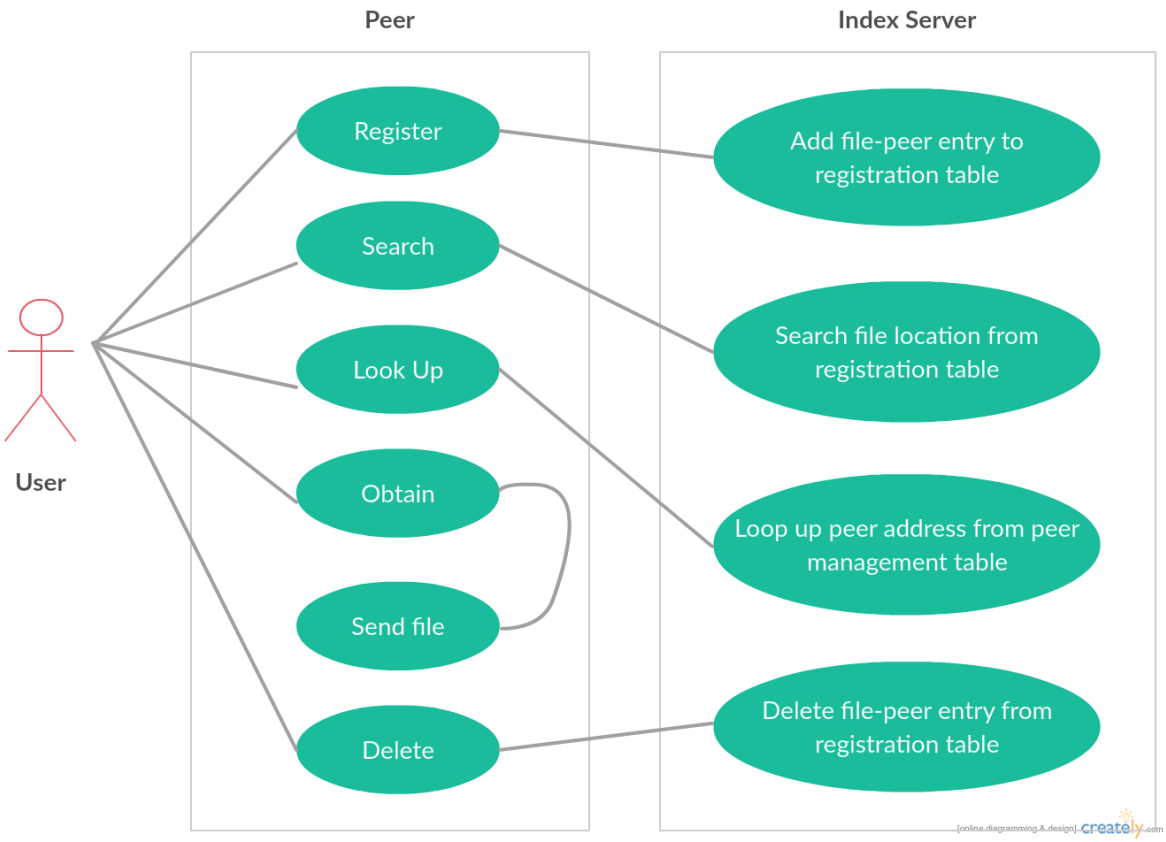


Figure 1 System functional analysis Use Case

## 4. Data Structures

There are two data tables maintained in this system: file-peer map table and peer id-address map table. Hash Map and Hash Set are the two main data structures used in this system.

a. **File-peer map table**: This is used to keep track of file locations. Specifically, it is a map of files to different peer servers because multiple peers can own a file simultaneously. Therefore, we use a Hash Map to store the information. The key is the file name and the value is a Hash Set containing peer server IDs.
b. **Peer id-address map table**: This is used to maintain the peer metadata mapping from peer id and peer address (IP and port).

## 5. Functional Modules Design (Class)

As it has been analyzed above, we designed the functional modules that should be implemented to realize this system. Figure 3 shows the modules and their relations (connections).

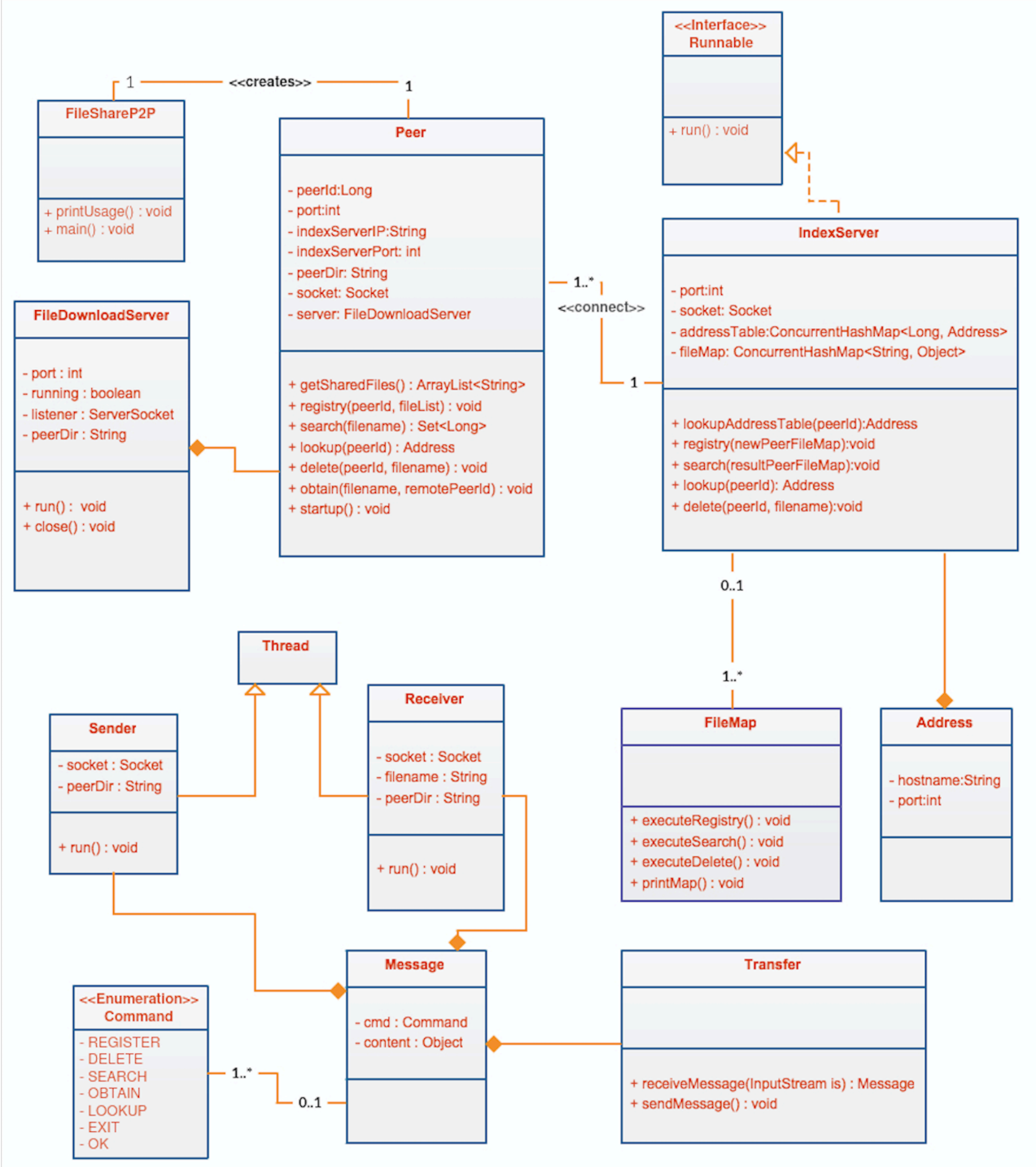


Figure 2 System functional modules design class

## 6. Techniques:

### a. Socket

The communication protocol adopted in this system is socket. The index server keeps listening new requests from peers on a dedicated channel while it is running. This dedicated listening channel is on a fix port (e.g. 5800) at the index server side.

Once the request is accepted, the peer will setup its own communication channel with the index server, which means each peer has dedicated work channel while communicating with the index server.

**b. Multi-thread**

In order to support multi-client requests at the index server side, multi-thread technique is adopted to handle it. Each request will trigger a thread in the server to specifically handle this request. Similarly, each peer file download server also listens on a specific port for potential download request, and a new thread is created for each download request.

**c. Synchronization**

In this system, multiple requests would go to the index server to modify the file-peer map table and peer id-address map table, which may cause inconsistency. To handle this inconsistency, synchronization technique is adopted for each operation on those two tables. As long as there is a "write" operation on the table, the operation acquires a lock, which prevents the other "write" attempt before it finishes. Once the current operation finishes, the lock will be released.

**d. Gson**

Gson is an open source Java library to serialize and deserialize Java object to JSON. In this system, Gson is adopted to handle different messages transferred between peers and index server. Although the content of each message differs from each other, all of them maintain similarly key-value structures, which is perfect suitable for JSON format. Besides, the transformation between message and JSON object is convenient in Java.

**e. Automatic registration**

In this system, any change in the peer local file system will be reported to the index server and renew the registration. On startup, peers automatically register all files in their local file system to the index server, which is comparable to the registration triggered by users. In addition, any change (e.g. modified, deleted, added, etc.) in the local file system is monitored all the time and registration is triggered once changes occur. A virtual file system (VFS) or virtual file system switch is an abstraction layer on top of a more concrete file system. The purpose of a VFS is to allow client applications to access different types of concrete file systems in a uniform way. In this system, we leverage VFS' automatic file system monitoring function and event triggering mechanism. Each peer has its own implementation of VFS' monitor. The monitor will trigger a corresponding method and send out the most updated local file system files to the index server, thus keeping the global metadata updated and accurate.

## 7. Program List

Figure 3 displays the program list of this system.

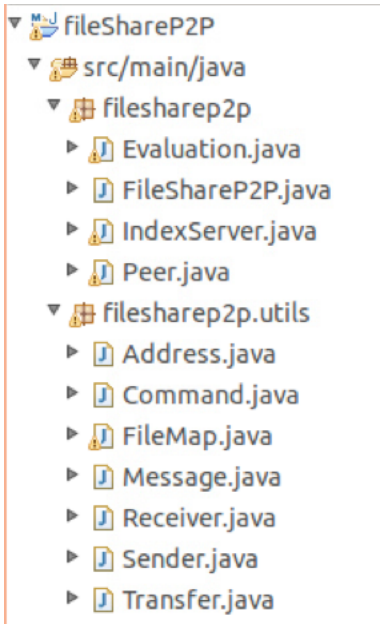


Figure 3 System Program List

## 8. Discussion (Problem)

This system is designed and implemented with basic functions as the assignment requires. There are still several potential improvements that could be done to improve the system.

- Other communication protocols: For peer-to-peer communication, there many other protocols can be used, such as RMI and RPI, other than socket. It is not trivial to assert that which is better. We would like to quantitatively measure the tradeoffs between them in the future.
- The current system interaction interface is command line. Definitely, a GUI is missed to it.
- As system increases in size, we expect a centralized design would degrade the system performance dramatically, this is shown in the separate system performance test report. A possible alternative is to design a distributed metadata management mechanism. We would like to include this feature in future.
- As the system scales larger, the data structure may turn out to be inefficient. We would like to try out different alternatives for different purpose in future design.
- As future development, the system will include fault-tolerance feature. This is a key for a distributed file sharing system to go for production.