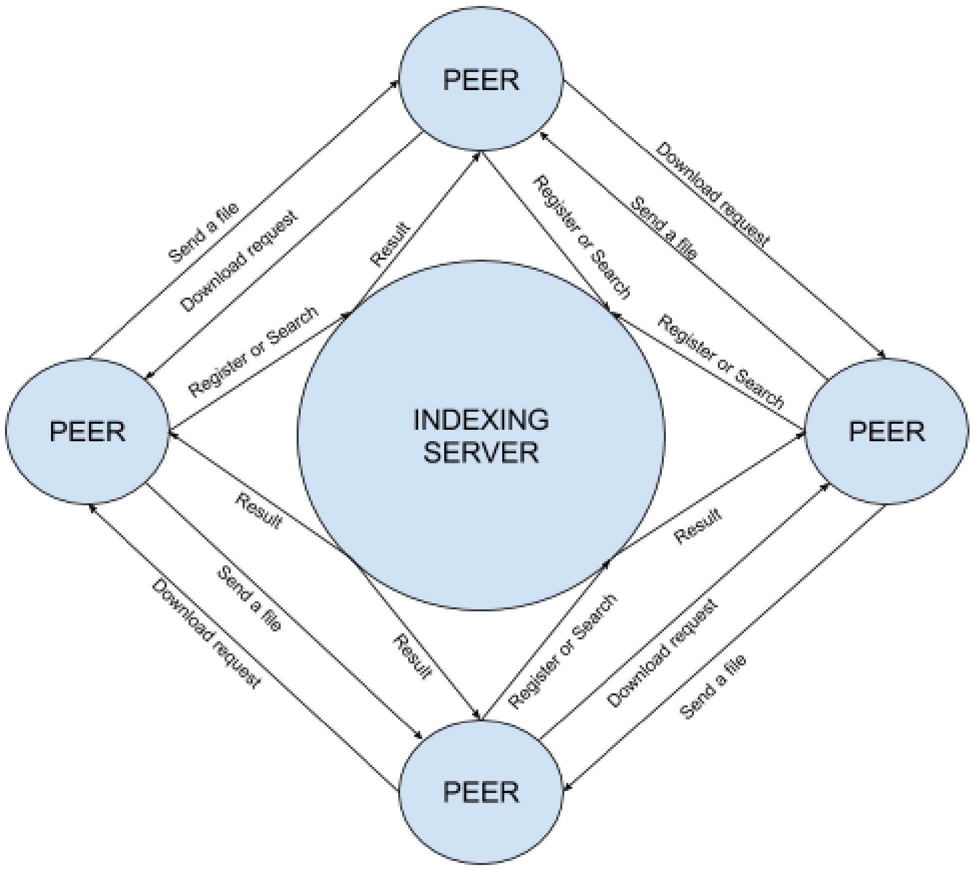
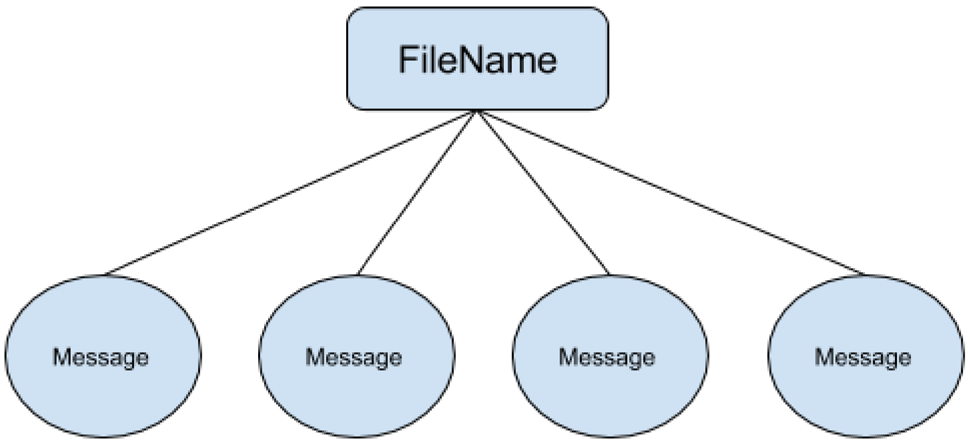
- **Overview**.
  - This project aims to provide a coordinator server that store and process file indexes of peers, which are pre-registered by peers. Moreover, those stored indexes may be shared with peers ,with location information, based on a peer request. On the other hand, peers in this system have the ability to register and downloads files. The system accept N number of peers since the server is using a multi-threaded listeners.

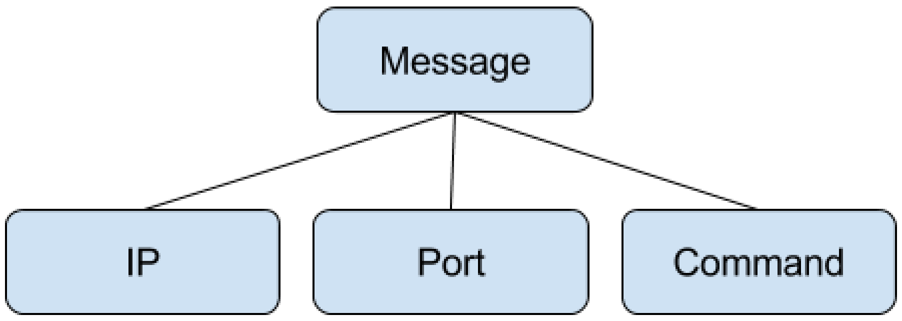
- **Workflow**:



- **Indexing Server Architecture**:
  - Indexing server is the entity that stores the files indexes which are registered by peers. It is the heart of the system that provides all the information from peers to download files.
  - The server stores indexes in a concurrent hash table which uses the filename as an ID and an array content.
  - The array holds the messages object.

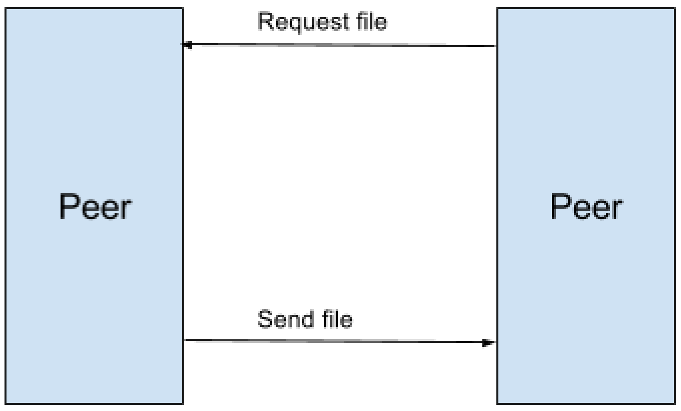


- Each message contains a peer information (IP, port, command..).
- This hash map can be accessed by many threads in order to lookup. However, it will be blocked if there a writing operation on it.

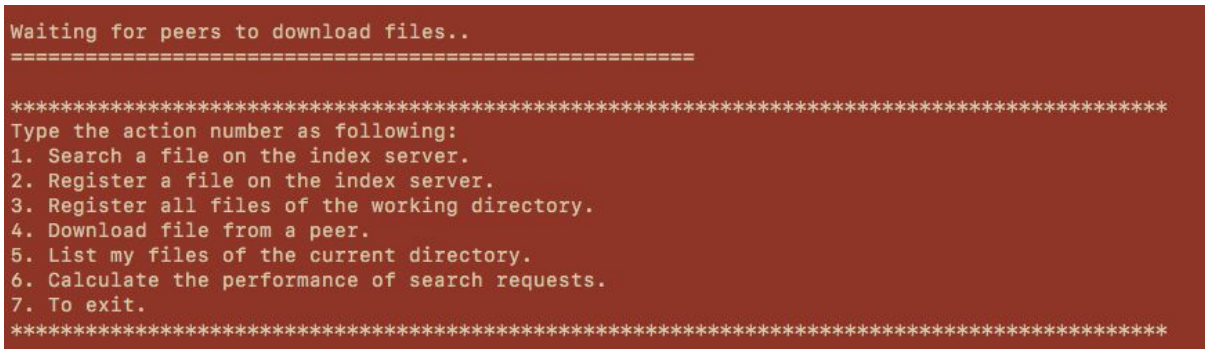


- The server listens for peers all the time. The port ‘60000’ is reserved for registration requests while port ‘60001’ is reserved for lookup requests.
- Once a request is received, the server will accept the connection immediately and create new thread to handle that request.

- **Peer Architecture**:
  - Each peer has three major functionality, listening to other peer, making request to the server, and requesting from another peer.
  - In term to get a file from a peer, a peer should be listening to others requests, and send the requested files to them.



- Peer can also many functions as listed on the following image:



```
Waiting for peers to download files..
=======================================================

*********************************************************************************************
Type the action number as following:
1. Search a file on the index server.
2. Register a file on the index server.
3. Register all files of the working directory.
4. Download file from a peer.
5. List my files of the current directory.
6. Calculate the performance of search requests.
7. To exit.
*********************************************************************************************
```

- Searching a file required user input to get the required file name, and returns list of (peer id and IP), which is the locations that a file existed.
- Register a file allow user to register a file index in the server. It is requires the file name, and this file should be existed on the current machine.
- Register all files, allows the user to register all the files existed on the resources folder.
- Download file from a peer, allows user to get a file from another peer. It requires the user to type the peer ID, IP, and the file name in this format (pppp-IP-filename.ext).
- List my files, displays the files which are on the resources folder, to the user.
- Calculate the performance, let the user measure the search request performance using N number of requests. It requires the file name and the number of needed search request, and return time in milliseconds.

● **Future enhancement:**
  - The system needs more specific exceptions handlers and messages in term to be readable for the end users. For example, instead of showing system messages, we may display some short and readable messages like “host is not available”.
  - It needs GUI to be user friendly.
  - Dynamic ports allocation needed in the system to avoid overlapping reserved ports.
    - We may achieve this by creating a method that looks for the unreserved ports numbers, and return one of them. Then, this port will be the node listening port and its ID.
  - Enhancing the performance by apply load balancing approach on the server side.