MANUAL

- A simple decentralized peer to peer file sharing system designed here comprises of 3 peers and 8 indexing servers. Each peer acts as a server as well as a client.
- Each indexing server provides each peer following functionalities:
  i. Register: Each peer can register all the files it has with the indexing server using put command. Syntax: Put filename Peerid.
  ii. Search: Each peer can search for a particular file's location on indexing server using search command. Indexing server returns the peerid with which that particular file is. Syntax: Search filename.
- Each peer provides other peers following functionality:
  i. Obtain: Each peer can download a file from other peer using Obtain command. Syntax: Obtain filename.
- **Functions of each java file:**
  - Server0.java: Indexing Server1 program.
  - Similarly Server1.java to Server7.java contains indexing server code.
  - MainClient1.java: Acts as Client with request of Put, Search and Obtain. Connects to any of the Indexing Server (Server0.java to Server8.java) located using hash function for Registry, Search, and Obtaining file request and then connects to respective PeerServer via respective PeerClient for downloading the file.
  - PeerServer.java: Transfers file/s with Peer1 to any other Peer on request.
  - PeerServer2.java: Transfers file/s with Peer2 to any other Peer on request.
  - Peer3Server.java: Transfers file/s with Peer1 to any other Peer on request.
  - PeerClient.java: Goes to Server of Peer1 with request of file/s with Peer1.
  - PeerClient2.java: Goes to Server of Peer2 with request of file/s with Peer2.
  - Peer3Client.java: Goes to Server of Peer3 with request of file/s with Peer3.
- **To create a decentralized p2p file sharing system perform following steps:**

1. Create 8 folders: Server0, Server1, Server2,……, Server8. Put files Server0.java, Server1.java, Server2.java,……, Server8.java in respective Server folder. These java files contain code for indexing server.
2. Create 3 folders: Peer1, Peer2, and Peer3.
3. Put following files under Peer1:
   PeerClient.java, PeerServer.java, PeerClient2.java, Peer3Client.java, MainClient1.java, file1.txt, file2.txt and file3.txt.
4. Put following files under Peer2:
   PeerClient2.java, PeerServer2.java, PeerClient.java, Peer3Client.java, MainClient1.java, file5001.txt, file5002.txt and file5003.txt.
5. Put following files under Peer3:
   Peer3Client.java, Peer3Server.java, PeerClient.java, PeerClient2.java, MainClient1.java, file7501.txt, file7502.txt and file7503.txt.

As the program is hardcoded, running it on different machine requires to change local serverhost name from “mandar-VirtualBox” to respective local serverhost’s name.

Portnumbers are also predefined. PeerServer and PeerClient are running on 5060, PeerServer2 and PeerClient2 on 5061, Peer3Server and Peer3Client on 5062, Server0 on 5050, Server1 on 5051, Server2 on 5052, Server3 on 5053, Server4 on 5054, Server5 on 5055, Server6 on 5056, Server7 on 5057.

Now after setting up the environment the system is ready to run:

I. Open terminal.
II. Cd to Server0
III. Run server0.java
IV. Perform steps 1 to 3 for 8 severs.
V. Open terminal.
VI. Cd to peer1
VII. Run PeerServer.java
VIII. Perform steps 5 to 7 for peer 2 and peer 3 so that peer servers are all up.
IX. Open terminal.
X. Any peer can now request for registering, searching or obtaining a file with the indexing server and peer server.
XI. Suppose peer1 wants to register its files. Using put command this can be done as: “Put file1.txt peer1”
XII. Similarly for searching a file, “Search file1.txt”.
XIII. For obtaining a file, “Obtain file1.txt”.

The decentralized p2p file transfer system supports data replication too. In case one of the Indexing Server or Peer Server is not up then also the request is served.

Also, this system supports both text as well as binary files.