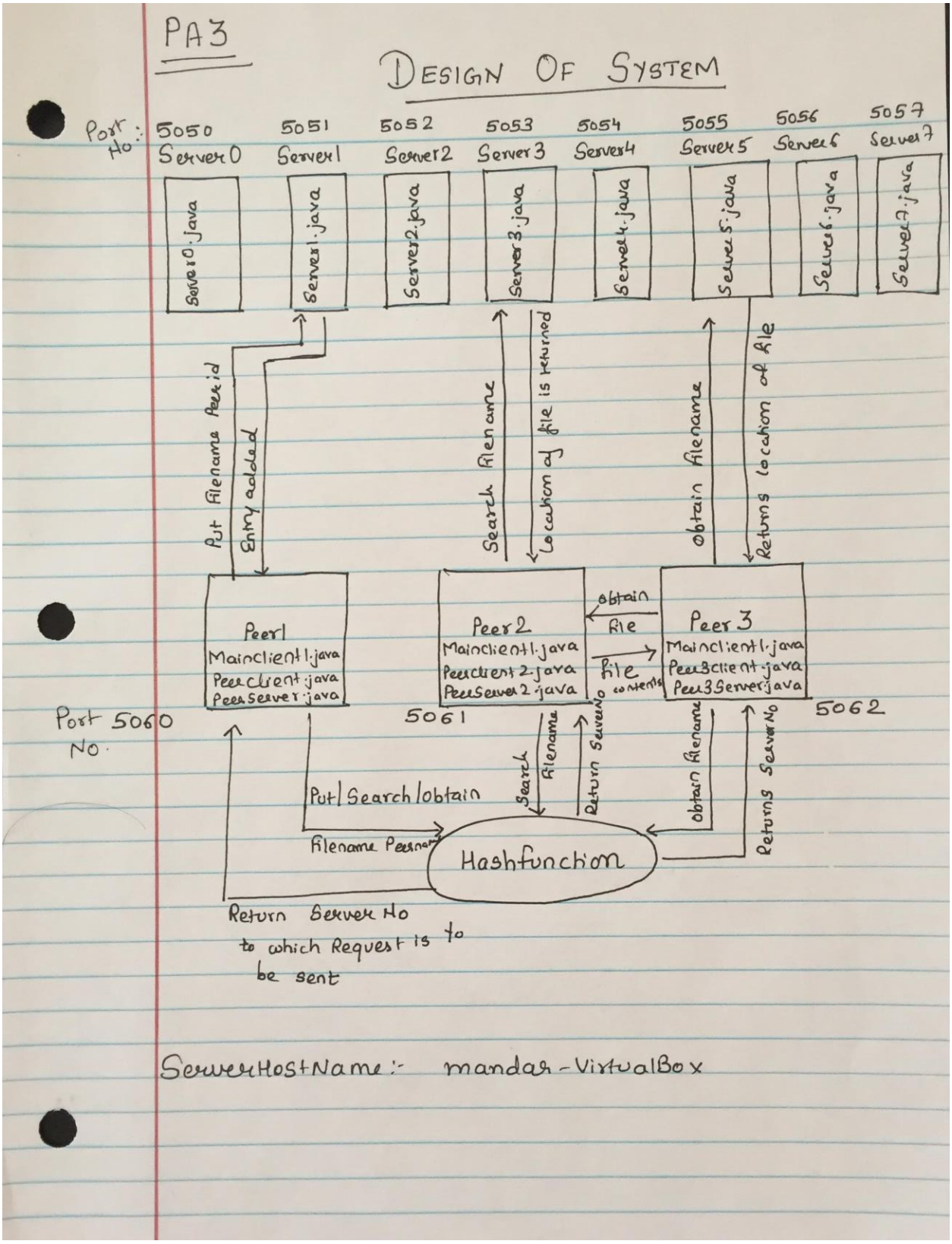
Design Document



ServerHostName :- mandar-VirtualBox

- Figure shows the design plan of the decentralized p2p file sharing system.
- System is implemented on Oracle VB-Ubuntu.
- The 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.
- 8 Servers are kept under different directories and 3 peers are also kept under different directories to make system distributed.
- Each Server has a concurrent hashmap which allows the peerclient to register and search.
- Each peer has 3 files: MainClient1.java, PeerClient.java and PeerServer.java. MainClient gives user the interface to request the query.
- On put command, using filename, the hash function locates a server on which this request should be forwarded i.e. the server on which filename and peername entry should be added. Before sending the actual query a socket connection is established with that Server from MainClient.
- **HASH FUNCTION**:
  Hash function used by this system is java's hashCode().It is implemented using a product sum algorithm over the entire text of the string. For a string s

$$h(s) = \sum_{i=0}^{n-1} s[i] \cdot 31^{n-1-i}$$

  Where s[i] denote the ith character of string s
  And n denotes the length of string s
- Server is located using:
  Server_location = (Math.abs(filename.hashCode()))%noofservers
  Absolute value is taken as sometimes hashCode() gives negative values.
- Similarly for search query, file location is returned from that server after locating the server and establishing a connection.
- For Obtain query after getting file location of that file from the indexing server, MainClient internally calls PeerClient of that respective peer which has the requested file. This PeerClient is connected to the respective PeerServer and hence downloads that file from the PeerServer. This makes downloading file on the requesting Peer very easy and fast.
- PeerServers and Indexing Servers both can handle multiple clients at a time. This is done using multithreading in java.
- Text as well as binary files are supported by the system.

- In case, one of the indexing server is down, request of client is fulfilled by immediately next server. This metadata replication is done using following method:
    - All the requests of client for put command is also forwarded to the server next to the server located using hashfunction. So 2 entries for each registering is done.
    - Server_location = (Math.abs(filename.hashCode()))%noofservers;
    - Server_location_rep = Server_location + 1;
- In case, one of the peer server is down, file replication is implemented.
    - All files of peer1 are also copied to peer2. Files of peer2 are copied to peer3 and of peer3 are copied to peer1.
    - So when peer1's server is down peer2's server completes the request. Similarly for other peers.
- **TRADEOFFs:**

    - When multiple locations of a file exists by default Client takes the 1st Peer name in the list and goes to that peer for obtaining a file.

- **IMPROVEMENTS:**

    - The Port numbers and Local Hostnames are hard coded in the program. So making these dynamic would make it easy for running it on other machines.