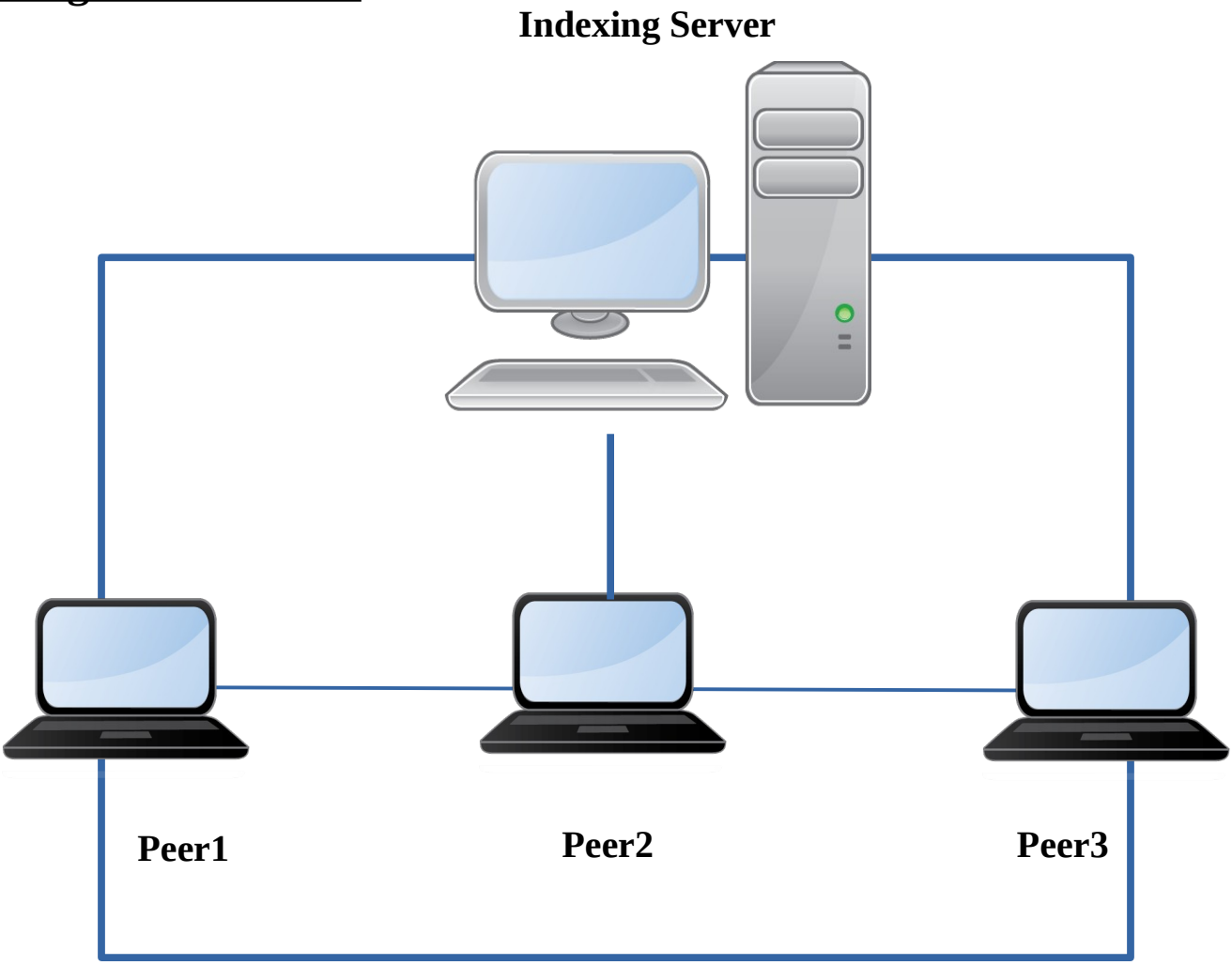
Aditya Kale – A20355807

Design Document :



Peer To Peer System.

Its a Peer to Peer system where all the all the Clients (Peer) are connected to an Indexing Server. Also Each Peer can communicate with another Peer when needed.

Aditya Kale – A20355807

Classes in Project:

IndexingServer.java

1.IndexingServer.java
It's a main class which runs the Indexing Server and wait for client requests.

It calls methods of **IndexingServerImpl.java** . Which does three tasks.

i. Add all file names in IndexingMap when client Registers.
ii. Excepts a file name from client and send list of all Peer who has that file.
iii. Returns all file names which are registered at server.

A **ServerSocket** is created on a particular port and server waits for requests on that port.

Client.java

2.Client.java
It's an another main class which runs the Client.

It calls methods of **ClientImpl.java** .
Which can perform four tasks.
i. Register a client to a Server.
ii. Request Server to show all files which are available on server.
iii. It can request a file for downloading to another peer.
iv. Itself its a server which provide his files to another peer for downloading.

A **Socket** is created using an IP and a Port to make connection with server and another client.

3. ClientDetails.java

Its a simple POJO class. It has three parameters for storing ipAddres, list of files and port number. I have used this class object to send data from Client to a Server while registering a Client.

Overview Of a System:

IndexingServer

- When Server gets up, it **creates** a **ServerSocket connection**. It is **continuously running** and ready to accepts continuous requests from different Client.
- When any Client connects to a server, e**very time it accepts the connection and create a new Thread to process each request**. And thus **able to** continuously **listen concurrent request**.
- If a Client want to Register OR Search a file name in indexing map OR want to check all file names which he can download Indexing server will make a seperate thread for each request.
- **Task :** Its only task is to **add file names and their corresponding machine names in an Indexing Map** and **gives the machine address for a particular file if requested by a peer.**

Client as a user for Downloading a file:

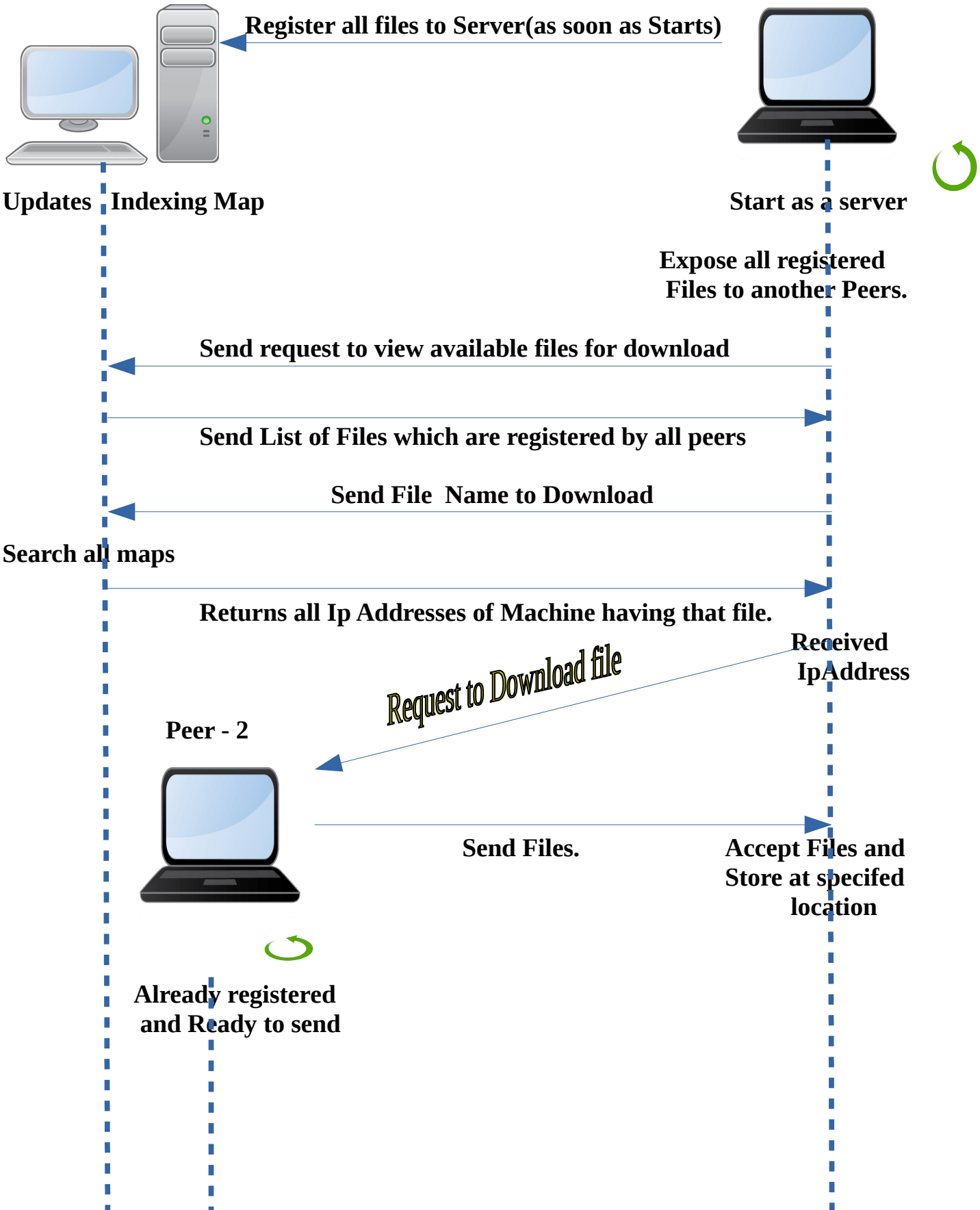
- Client will send a request for searching a file name to indexing server. In return It get list of servers where that file is located. Now it creates a new connection with that IP and fetches the file.
- **Task : Create a connection with peer and downloads a file.**

Aditya Kale – A20355807

Client as a Server for Sending a file:

- **Client creates** a new **ServerSocket** and is continuously up for Sending the files. As soon as any other client makes a request to download a file, it creates a new Thread for performing send Operation. **SendFiles.java** is a class which has **method to send a file** to a client. As for **every request** Client **creates a new thread**,**concurrent send operations** is **handled** by a Client.
- **Task : Accepts a connection from another peer and send file to that peer.**

Sequence Diagram :



Improvement and Extensions:

1. **Folder Path configuration** can be moved to config.properties, so that user will be able to register files from any location whichever he wants.

   Achieved by : Reading folder path from config.properties. And if path is not mentioned in property file then register files from default path.

2. Handle scenario where user inputs other than the mentioned numeric values while selecting a server to download a file.
   Achieved by : Reading an input and checking it as numeric or not. The numeric value should be less than or equal to no of servers having that file.

3. **Running clients on different port numbers** and communicating between them.
   Achieved by : Storing the port number with the Ip Address. While sending fileNames and IpAddress to server also add port number. And while retrieving Ip address also get exact port number on which Client is running for communication.
   Ex : IP_PORT

4. **Registering newly added files Or Removing Deleted files**: currently system does not allow a client to register newly added files without a restart. Also if we delete some files from the path we do not re-index the server map.
   Achieved by : Give an option to re-register/refresh files in map.

5. **Data Replication is not handled.** So the system is not fault tollerant.
Achieved by : Replicating the IndexingMap on some other server. And if the server is down it should give an Ip address of another Server where IndexingMap is replicated.

References Use:

- **http://www.javaworld.com/article/2077322/core-java/core-java-sockets-programming-in-java-a-tutorial.html?page=2**
- **http://www.mkyong.com/ant/ant-how-to-create-a-java-project/**
- **http://www.rgagnon.com/javadetails/java-0542.html**