Advanced Operating Systems

CS550

Programming Assignment 1

Simple Napster Style Peer to Peer File Sharing System

Student

Nikhil Nere

A20354957

nnere@hawk.iit.edu

Table of Contents

1. Introduction

1.1. Purpose

This document provides a detailed design of the Distributed File Sharing System. It provides an overview how different components in the system interact with each other

1.2. Problem Statement

A simple P2P system is to be designed which has a central indexing server. The peers in the network will register the files on the indexing server.

Central Indexing Server: The central indexing server will maintain an index of the files registers by peers. When a peer will make a request for file search it will return the peerId for the peer who has register that file on the server.

Peer: A peer will register the files on the indexing server to make them available for other peers. A peer will make a file search request to the indexing server and will get peerId of the peer having that file. It will then connect to the peer to get the file.

2. Design Overview

2.1. Topology Diagram

The system has a central indexing server and peers

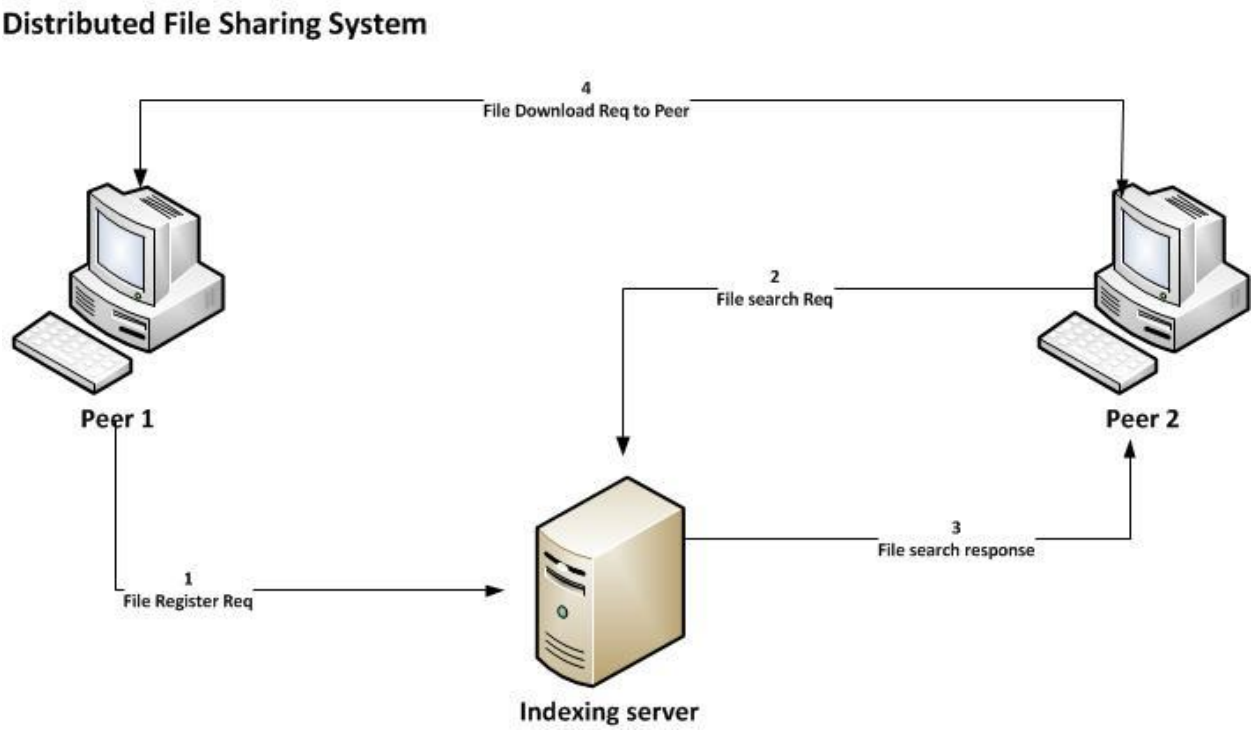

Figure-1: Distributed File Sharing System

2.2. Detailed Design of the System

The distributed file system has a central indexing server which indexes the files registered by the peers. The peers in the system register their files on the indexing server. The peers can also search for a file on the indexing server. The indexing server returns the peerId of the peer having that file. With the peerId the requesting peer then connects to the other peer to get the file.

Below is detailed communication between different components. Please refer to **figure-1**.

1) Peer 1 sends a request to the indexing server to register the files it has. It sends the list of file names and its own ip address and the port where it will be listening on for file download requests.
   On receiving the request the indexing server makes entries for the files in its index table. It also stores the IP address and port number of the peer who has the files

2) Peer 2 sends a search request to the indexing server. It sends the file name to be searched in the index.
3) On receiving the search request the indexing server searched for the file in the index and returns the IP address and port number of all the peers who has the file
4) In the search request response Peer 2 gets the list of peers who have the file. It then picks first peer from the list in this case Peer 1 has the file so Peer2 connects to Peer 1 and gives him the name of the file. Peer 1 reads the file from its memory and sends the file to Peer2

This is how the components in the system interact with each other. The indexing server is always listening for the requests from other peers and it can distinguish between the file register requests and file search requests. Any peers in the system can register its files on the indexing server and can search file on the indexing server.

3. Implementation

3.1. Server Component

- The indexing server handles the clients independently. It creates independent thread for each client that connects to it.

```
while (true){
    Socket client = serverSocket.accept();
    Thread clientThread = new Thread(new HandleClient(client));
    clientThread.start();
}
```

- The server then identifies the type of request if it is a register request or a search request

```
if(obj instanceof RegisterDTO){
    RegisterDTO regDTO = (RegisterDTO)obj;
    registerFiles(regDTO);
}else if(obj instanceof String){
    String fileName = (String)obj;
    searchFile(fileName);
}else{
    System.out.println("Invalid request from client..\n");
}
```

- The registerFile method then registers the files on the index.

```
for(int i = 0; i< regDTO.getNoOfFiles(); i++){
    String fileName = regDTO.getFileList().get(i);
    if (fileIndex.containsKey(fileName)) {
        peerList = fileIndex.get(fileName);
        peerList.add(regDTO.getPeerId()+"|"+regDTO.getPortNo());
    }else{
        peerList = new ArrayList<String>();
        peerList.add(regDTO.getPeerId()+"|"+regDTO.getPortNo());
        fileIndex.put(fileName, peerList);
    }
}
```

- The searchFile method searches for the file in the index

```
ArrayList<String> peerList = fileIndex.get(fileName);
```

3.2. Client Component

There are two parts of the client component –
One that communicates with indexing server for file register and file search/download
Second accepts requests for **file download**

- **File Register :** The peer sends file names and its own IP and port to the indexing server to register the files

```
RegisterDTO regDTO = new RegisterDTO();
regDTO.setNoOfFiles(noOfFiles);
regDTO.setPeerId(getIPAddress());
//regDTO.setPeerId(InetAddress.getLocalHost().getHostAddress());
regDTO.setPortNo(prop.getProperty("peer.port"));
System.out.println("\nEnter file Names :");
for (int i = 0; i < noOfFiles ; i++){
     regDTO.getFileList().add(scan.next());
}
System.out.println("\nConnecting to the server at IP:"+ serverHostname + ", Port:" + serverPort);
clientFileSystem.registerFile(serverHostname, serverPort, regDTO);
System.out.println("\nRequest sent to the server to register files.. \n");
```

- **File Search :** The peer sends the file name to the indexing server and gets a list of peers having that file

```
ArrayList<String> peerList = clientFileSystem.searchFile(serverHostname, serverPort, fileName);
```

- **File Server:** This is the peer acting as a file server. It accepts the file downloading requests from other peers. The file server can handle the multiple download requests simultaneously by creating an independent thread per peer.

```
while (true){
    Socket client = fileServerSocket.accept();
    Thread clientThread = new Thread(new FileSender(client, fileDir));
    clientThread.start();
}
```

4. Future Enhancements

- Currently the peer has to send the exact file name to the indexing server. The enhancement can be done in future to apply sophisticated search algorithms to find the partial matches.
- In further enhancements the peer can be able to fetch the list of available files on the indexing server based on an alphabet or a partial match.
- Currently peer can only register the file. In future enhancements the peer can send a request to delete the entries of the files from the index which it has previously registered.

5. Assumptions and Constraints

- Peer has to keep the files which it is sharing under one folder and the folder path should be updated in the config.properties file
- Peer should type exact name of the file for searching else the server will return message saying “File not found”.
- If the client application is closed the other peers will not be able to connect and download the file.
- If a peer has registered a file on the server it has to keep listening on the port it has given in the register request