|  |  |
| --- | --- |
| **CS 550 Assignment 1** | **Prachi Chachondia - A20260722** |
| **January 29, 2017** | **Rekha Seetham – A20353704** |

**Introduction**

The goal of the project was to design a peer-to-peer file sharing system with a central indexing server and three peers. The index server was built to maintain and store a registry that holds information on all the files owned by a peer. It was used to provide a search facility to peers. Peers were implemented as a combination of a client and a server, with the client facing the user and taking user input such as filename and IP, and the server facing the index server. The project code was written in Java v1.8, and used sockets for establishing connections, and threads to ensure requests from multiple peers could be completed simultaneously.

**Basics Design and Implementation**

The communication within the system starts with the user at one end, who sends a lookup request as a client. The indexing server then returns a list containing all the peers in the system which have the requested file. The client then picks one among these peers, and sends a retrieve file request to the server of that peer.

**Server –** Two servers were implemented to form the central server unit, one to pair with the client for the peer, and an indexing server which keeps a registry associating all files in the system with a list of all peers that hold them. The lookup request is sent to the index server, which returns the server IP address of the peer that has the file. This allows the client peer to send a direct request to the server of the peer with the file.

The index server stores its information as a hash map, mapping the filenames to an arraylist of the peer ids, and a second hashmap for the peer ids and peer objects such as IP address. Each peer must be registered to the remote server so that the server has its IP address on record, and is able to assign a peer ID. This is followed by every file that the peer holds being added to the registry.

**Client –** The client was implemented with a function to check if the client was already registered with the server or not, and to register it if the latter. A file list was used to keep track of all the files on the client side, so they could be added to the server registry. It also kept track of any local file deletions that happen, so that the list on the server can be kept up-to-date.

**Threads –** Since multiple peers could send requests to the server at any time, threads were implemented to handle the requests from each peer.

**Design Trade-offs**

**Improvements and Extensions**

The deleting of files currently does not automatically update the server registry, since it is tracked using an arraylist containing details of all files. A different tracking method such as a flag for every file could allow the tracking of files individually so that the information can be sent to the server as soon as the modification is made, instead of having to re-send the entire directory’s worth of file information. Some additional features that could be implemented in this project would be to set up multiple servers, and allow peers the choice to connect to any one of them by entering the server IP address.