CS 550 Programming Assignment – 2

Umashankar Rajaram A20301260

Design document for Peer to Peer Network

Model realized in code:

- The system has a set of 8 indexing server and 8 peers
- Any peer can join with any server based on its distance from a given server.
- Location co-ordinates (Latitude and Longitude) are used to identify the location of servers and peers.
- Each peer has a set of files and for getting other files, it will connect with other (active) indexing servers.
- The location and configuration information of the system is managed by a config file (serv_commons.h).

System Internals:

- The code is written using C++ and links with two libraries viz. RCF, Boost.
- RCF is utilized for RPC (Remote Procedure Call) communications, which again uses Boost internally for serializing the constructs and files.
- RCF provides clean interfaces to connect and communicate with nodes on a distributed network. They are initialized for every compilation unit.
- Threads are utilized by the servers (Indexing and Peer) to serve concurrent request from the peers.
- Synchronization on critical sections are handled using mutex locks, which again are implemented as scoped locks.

The System has 2 components:

1) Indexing Server
2) Peer

1) Indexing Server:

- The Indexing servers is implemented as a minimal system that exposes an RCF interface with two functions:
    a) *register_peer()* – For peers to register their identity and files. The function indexes the peer files by using hash mapping, in a way that every file is mapped with a set of peers that have it.
    b) *lookup_file()* – For peers' clients to search for a file and retrieve the set of source servers (seeds in p2p jargon) so that they can start downloading the requested file from one of them.
- The Indexing server runs continuously while serving its clients simultaneously by spawning threads. The thread creation is set as "dynamic" so that the Indexing server could manage its threads based on the demand.

2) Peer:

- Peer is implemented as a parent class while *Server* and *Client* as its sub-ordinates.
- The system has a total of 8 peers, each of which are separate binaries.
- Peer object has all the required methods to identify its directory, record the contents, register with the Indexing server and alter its record with every downloads.
- Every peer has 2 RCF interfaces, one for communicating with the Indexing server (as saw above) and the other for its server to transfer (download) files to the requesting peer.
  Following is Peer-server's RPC method:
    obtain_file() – A simple function that implements the file transfer operation. To serve concurrent client requests, the server can spawn multiple threads which is again "dynamic" as with Indexing server.
- The file search and download operations are handled by the peer's client implementation. Basically, the client is what the user interacts with while the whole system is running.
- The files for each peer is available at **data folder**, under the names as given by their respective endpoints. Each one has 1500 files each of 4KB wiith some random character data.

Operation:

- The Indexing Server begins by finding its configuration values from the **serv_common.conf** file. It finds the values based on its location co-ordinates (Latitude, Longitude). It will then expose its interface and wait for the peer/s to join.
- Peer too uses it's location co-ordinates to find the nearby server and registers with that.
- If the peer can't find the file in the nearby server it will search active servers (using the same config file) to get the file.
- Peer registers their contents by calling *register_peer()* RPC procedure. During this operation, the Indexing server indexes the files.
- On the Peer side, the client will look up the files using *lookup_file()* RPC procedure to get the peers sources (seeds) containing the file.
- Client will connect with the peer (file) server and download the requested file using *obtain_file()* RPC procedure.
- After every file downloaded, the peer re-registers its contents with the Indexing server so that it could become the source for the newly downloaded file.
- Concurrent requests are handled on both the sides (Indexing server and Peer serves) using threads and mutex locks.

Scope:

- The system is designed to handle the following issues:
    1) When the Indexing server identifies there exists duplicate indices for the files.
    2) When a download fails from the requested server. In this case, next peer, if exists for the file, will be requested.
    3) When a client requests a file that it already has in its directory.

- The system is now implemented with 8 Indexing servers and 8 peers. The Indexing servers could be removed and the file transfer can be realized by mutual communication between the peers. Also, the number of peers interacting with the system can be increased.