Design Document
PA2

1. Introduction
This program is to implement a Decentralized file sharing system based on previous assignment of Centralized file sharing system. Most of the architecture is reused from previous assignment and changes are made where necessary. It mainly including two parts, first is indexing servers that keep track connected peers and files registered by them and second is peers that register files and request files from other peer.
This Decentralized file sharing system was developed in C++ programming language using concepts of multithreading, multiprocessing and socket programming.

2. Program Design
This program design is divided into 2 main parts, a indexing servers and multiple peers.
The detail design principles and implemented functions are showed as follows.

2.1 Overall Program
A server uses server port to accept connections from peers. All peers connected to the same server use the same port to communicate with that server. Other peers can be connected to other servers as required.
Following figure explains structure of system, two servers are listening on ports 20000 and 30000. Peer1 and Peer2 are connected to Server with Port 20000 while Peer3 and Peer4 are connected to Server with Port 30000. Each peer uses tits server to register a file . Peers' register and lookup requests are handled through their respective servers.
Each peer uses its own server port to accept other peers' request messages. As an example in the image we have set server port of peer1, peer2, peer3 and peer4 to 9010, 9020,9030 and 9040 respectively.

2.2 Indexing Servers
The indexing server listens for incoming connections on its port. As soon as a connection is made to server by a peer, a child process is created on server that caters to the requests from that peer. The parent process keeps listening for more incoming connections. Each new connected peer is serviced by a new process.

2.2.1 Registering a file
To handle a register request from peer, peer needs to send the port number at which it is listening to incoming requests and the filename to register. Server stores this port number and filename data as a pair on an external file on separate line. This is easier to implement than interprocess communication pipes.

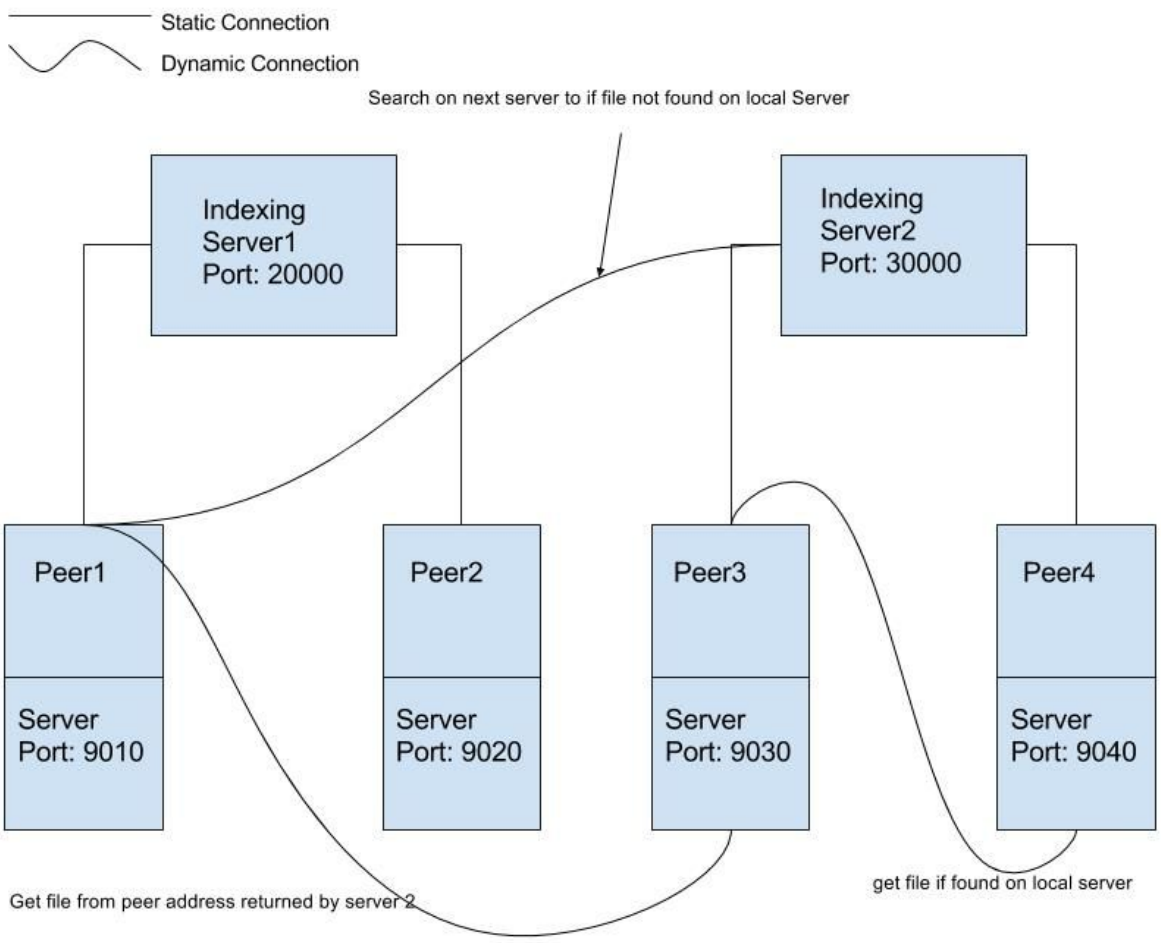

Figure1 : System Structure

2.2.2 Accept multiple client requests at the same time

The central indexing server listens for incoming connections on its port. As soon as a connection is made to server by a peer, a child process is created on server that caters to the requests from that peer. The parent process keeps listening for more incoming connections. Each new connected peer is serviced by a new process. So server can handle requests from multiple clients independently and simultaneously.

2.3 Peer

As soon as a peer is created, a thread is generated that acts as server. Operation of this thread is independent from peer's operation as client.

A peer acts as client and server simultaneously.

As a client, it can register a local file on the central indexing server for other peers to search and download. It can search files from the index server and choose to initiate the download.

As a server, it listens for incoming connection requests, and provides requested file once the connection is made.

2.3.1 Lookup

The peer sends the name of file it is looking for to the local server. Local server looks through the external peer index file for an exact match of that required file.If file is found server sends list of all the peers at which said file is available.
If file is not found,Peer connects to the external server and searches file on them one ny one until file is found or last server is searched.

2.3.2 Getfile

Once a peer chooses another peer to get file from, a connection is made to target peer's listening socket. Requesting peer sends name of file to be downloaded, the peer that has the file sends its size and both peers initiate data transfer for that file.Downloaded file can be observed in that peer's directory.

3. Future Scope

Since the register index is maintained in an external file, we can implement locking and unlocking of that file and buffer mechanisms to temporary store register requests so that the program does not misbehave when multiple register requests are made.
We can also implement auto registering of all the files available with peer to indexing server with the help of time header file.
More use of threads to improve performance.