

Design Document for Programming Assignment#1

Central Index Server

Central Index Server indexes the contents of the entire peer that register with it. It keeps list of file names present in the directory of the registered peer and provides the functionality to search for file name present to other registered peer of server. If the file name is found in any connected peer it will response peer with the peer IP address and port no that have those files.

Central Index Server API:

Central index server is written by keeping in mind of scalability. To make server more scalable I have created an API for the server:

1. This API can be exposed to any application for accessing required functionality of server without worry of internal functionality.
2. We can change our server implementation any time on need basis without changing the API that is for client side.
3. As API is written for generic use so any developer can change the internal functionality without worry or hampering the outer server functionality.

Managing Peer Connections

Central index server application accepts connection at any given port given there is no process running on that port by default server accepts connection at port no 9000. Every time any application makes connection to server listening port. CISServer instance acting as producer put that connection in LinkedBlockingQueue and pulled by ConnectionHandler instance acting as consumer from the queue. ConnectionHandler then gives that connection to ConnectionManager instance. ConnectionManager instance takes a Runnable instance of PeerConnection object for every connection made to the server.

Every connection is handled by a new instance of PeerConnection, which is Runnable.

Adding a Peer Connection to Server

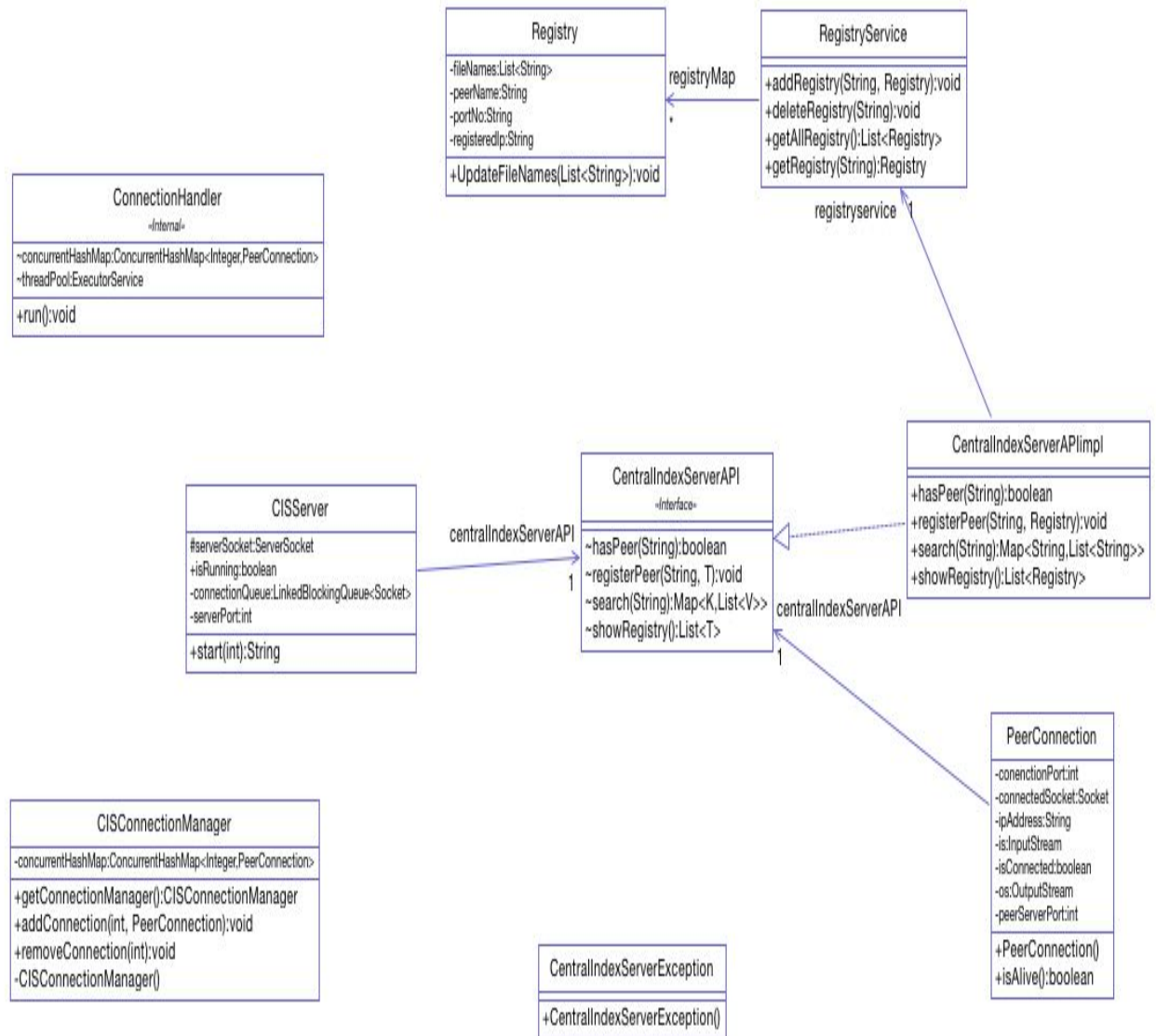
Once the connection is added to ConnectionManager. PeerConnection object send message to connected peer connection for creating a registry in server by providing Peer Name, Peer File transfer port and Peer File Names.

Serializable objects for data transfer

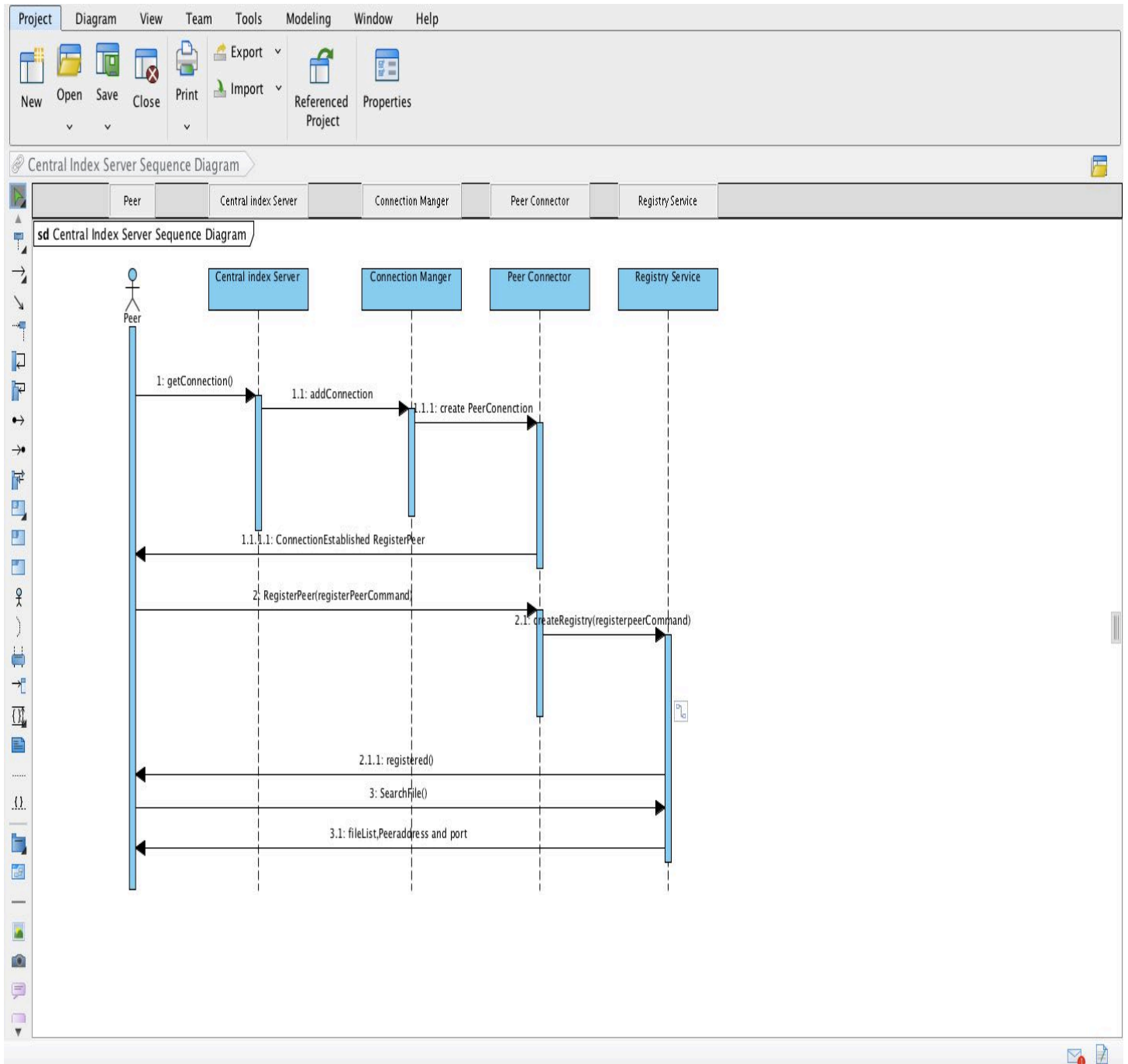
Central index server only accepts the Command object that travel through network because Commands object as Serializable and makes the connection between peer and client more secure as any client that does not have correct command object cannot communicate with server or register with server.

If the peer connection sends the specific command for registering of peer then central index server creates a new registry for that peer and add that connection to registry service as well as allow that client to search for file names from central index server using command objects that are understood by Central index server. Otherwise the connection is disconnected from server and removed from connectionManager.

Class Diagram of Central Index Server



Sequence Diagram Of Central Index Server



Peer Application

Peer Application is designed in such a way that it act as client for getting file names from central index server as well as work as server to other peer for downloading data. It's a multithreaded application that has one thread acting as client for central index server, another thread working as server for accepting the connection from other peers to facilitate file download functionality as well as there is another scheduler thread working ever 5 min to check the change in files in directory of peer and updating it to central index server

Peer Application acting as client

1. Making Connection with Server: When we start the peer application it creates a connection to central index server using the default IP address 127.0.0.1 and port no 9000. It can be configured to other IP Address and port no
2. Registering to Application to Server: Once Peer Application gets response form central index server it ask the client to register the peer to Server by giving the peer a name and providing a directory from where peer application can share it's available file list as well download the available file form other peers.
3. File Look up in server: After the peer is registered then application ask to search any file in server and server give response with matching filename as well as location of peer IP address and port no.
4. On background there is a thread running every 5 mins for lookup of change in directory files. If there is a change in files it will send server command to update the peer files.

Peer Application working as File download Server

Every peer client has an additional thread working as server that accepts request from other peer and allows those connected peer to download the file from it directory. Port no of every peer application can be configured using properties files. Peer file download server instance can handle multiple peers those acting as client to the file download server.

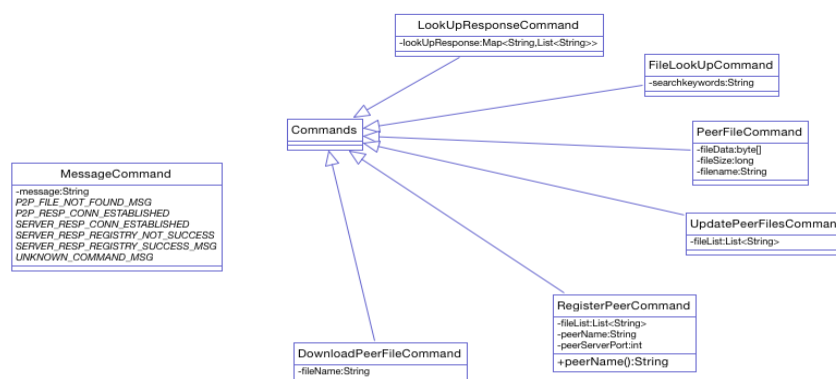
Communication Between Central Index Server and Peer

To communicate between server and client we used serialized object and Enum, which is also by default serialized in java. Parent of the serialized object is named as "Command" Object and all other other serialized object child of this Command.

Benefits of Using Serialized object for communication

- Every Communication between Central index server and peer is done by using Command object of Message Enum.
- Command Object makes the application more secure as anonymous peer does not having the correct commad object will not be able to send request to server.

Class Diagram of Command Objects



Language and Data structure Used

Programming Language: Java

Build Tool: Apache Ant

Network Connection: Socket

Connection Management: Using Java Concurrency package

Data Structure:

1. `LinkedBlockingQueue`: For Peer connection management
2. `ConcurrentHashMap`: For keeping connection of Peer. where key is connectedport and value is `PeerConnection` Object)
3. `HashMap`: For keeping connection successful connection of peer. Where key is combination of IP address of Peer and File transfer port and value is a custom Object that keeps record for every successfully added peer and its file list.
4. `Registry` Object: keeps information about connected peer and its file list