CS550 Programming Assignment 3 (PA#3)

Maintaining File Consistency in Your hierarchical Gnutella-Style P2P System

Design Document

Dated: 2-11-2018

Authors:

Sukhada Pande

Contents

1. INTRODUCTION

Purpose

The main purpose of this document is describe the design and implementation of an hierarchical Gnutella style peer to peer file sharing system using Java RMI.

Scope

The scope of this project is peer will be registering to one super-peer which is like an indexing server, peers can search for a file in its connected network of super-peer, if any of the files get modified we use a push and pull mechanism to propagate the file validity and maintain its consistency.

2. Goals and Assumptions:

Goals

The goal of this project is it will allow you to register peer to the super-peer ,and let you search in its respective network topology and maintain consistency of files , if they are modified by the origin peers.

Assumptions

- The peer knows the file name it wants to search for.
- Peer specifies the file name with extension when it wants to download/search/register/delete a file.
- Each Peer has its own directory in which it stores all its files and also downloads desired files from other peers.
- The peer knows the port of super-peer it is connected to.
- The super-peer configuration property based , thus the port numbers in the property files and that of super-peer are same.
- We know the file names that we want to modify.
- The time to refer is set constant now.

3. SYSTEM ARCHITECTURE

Architectural Design

This project has two main components: 1. Super-peer 2. Peer

Super-Peers:

It is like an indexing server , which registers all the peers ,and maintains the list of files these register peers have in their directory. It sends in the query request to it neighboring super peers to search for files in their directory.

Peer:

Peers are the leaf-node , which register themselves with the super-peer and request for a file they are searching for to download. These behave as peer as well as servers. They behave as peer while using the search functionality and servers when it lets another peer download from it. It also modifies the files in its file directory and broadcast the same in push approach and pulls information of the modified files in pull approach.

Communication System

For communication between the peers and server we will be using Remote Method Invocation i.e. Java RMI.

It basically allows to invoke methods or functions on an object that exists on different machine or different address space or remote machines. It basically has a server and client programs.

A server creates some remote object and waits for client to access it .The clients obtains a remote reference to the remote objects on the server. It lets the client and server communicate with each other.

We are using RMI its is object oriented which lets you use it easily. It is particularly easy to write and code , and it implements threads internally, so we do not have to create threads for launching multiple peers.

Also, it lets application to register remote objects using rmiregistry so that other application can access them and invoke methods.

We are using Gnutella style file transfer in which the peers send request to it neighboring peers , in this every peer is server and vise-versa. The peers sends a query to other peers and the peer where the file find it sends back query hit.

Flow Diagram

Diagram showing communication between 4 super-peer and their 3 peers.

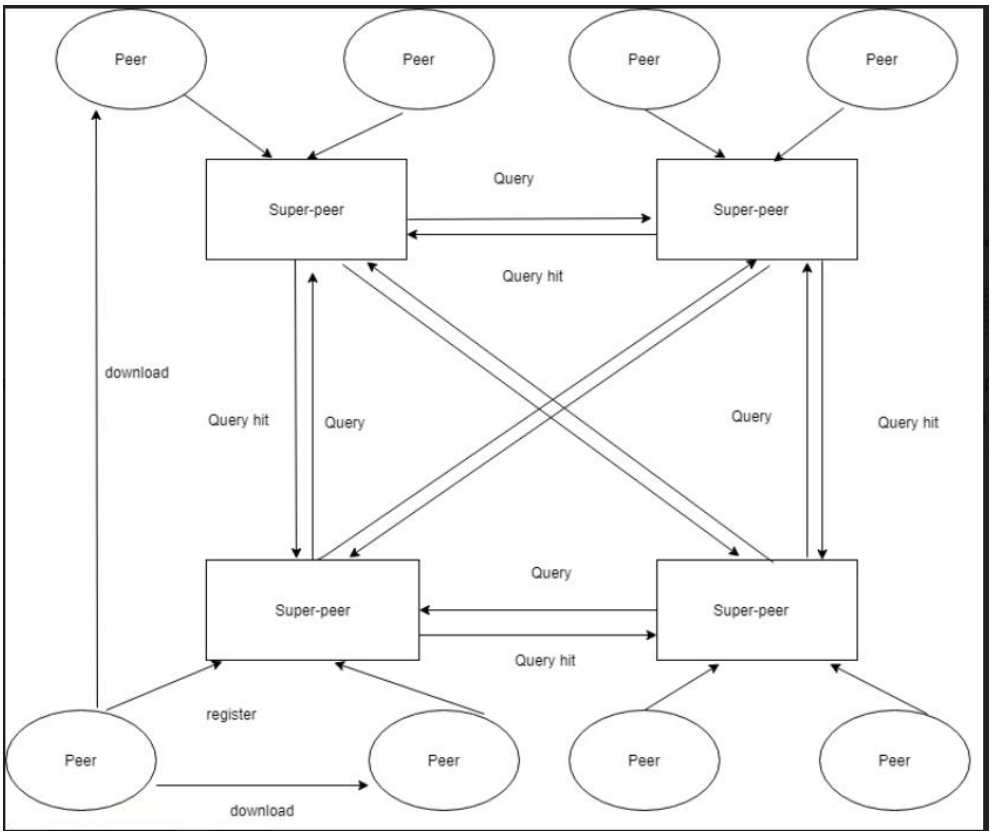
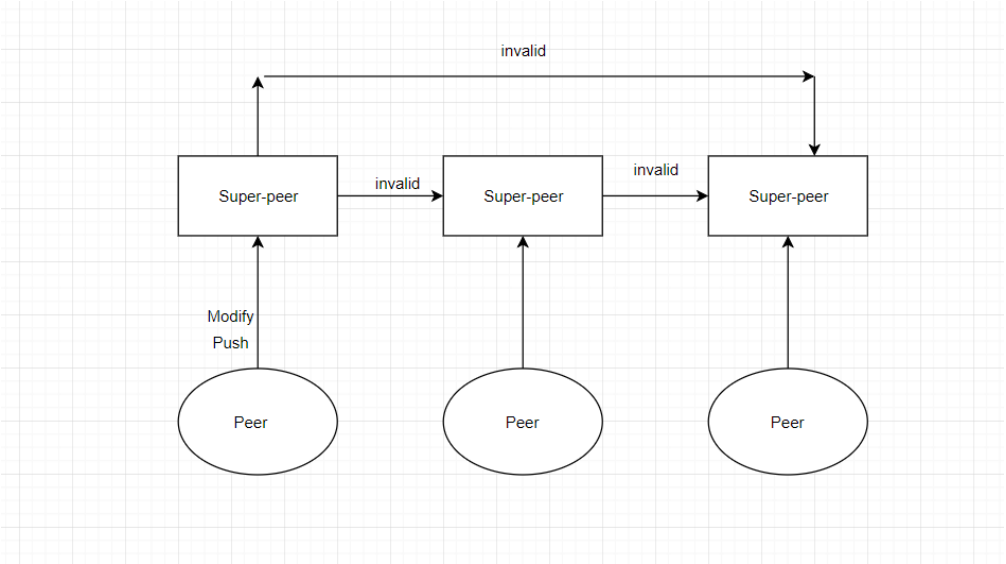


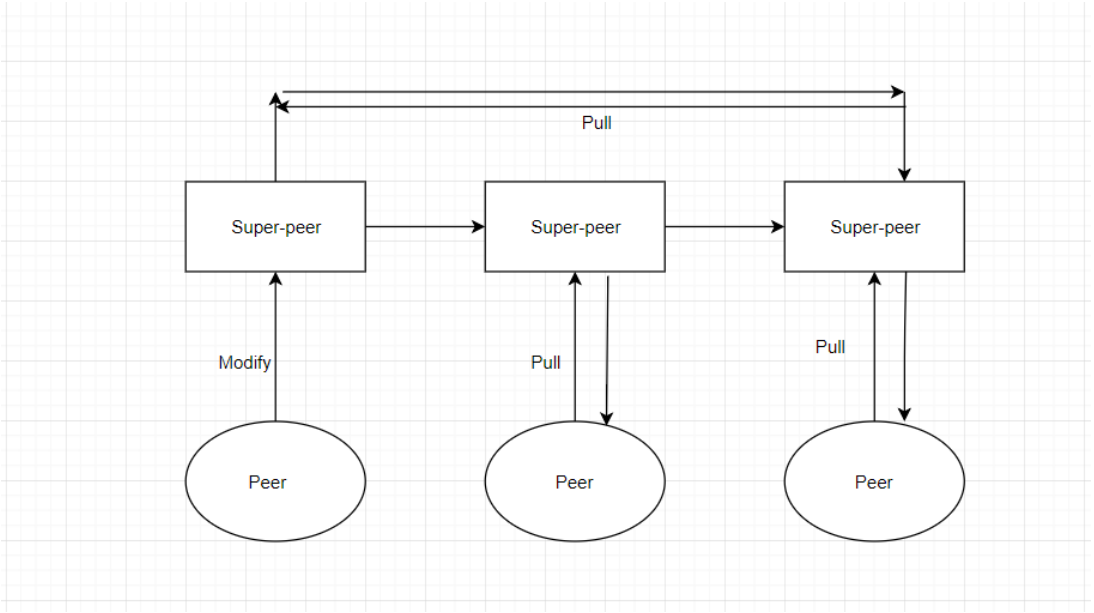
Diagram showing super- peer to super-peer and peer-peer communication when searching for a file and downloading a file from respective.

Push:



Peer sends modified info to all the other peers.

Pull:



If file modified scheduler pulls info from the original server if the file is modified.

4. DATA DESIGN

Data Description

Here we are using Arraylist and HashMap to basically store the peer registration file information. The PeerList is of PeerDetail which saves the file information and thus lets the peer search through the registered files.

It also uses this list to check for any duplicate information.

Throughout the project we are using Arraylist as it easily stores information in its instance.

5. COMPONENT DESIGN

Server Side Classes

1. ServerInterface

This interface extends the Remote interface and has abstract methods to registerPeer, SearchforFile, DeleteFile, which depict various server functionalities.

The Remote interface serves to identify interfaces whose methods may be invoked from a non-local virtual machine. Any object that is a remote object must directly or indirectly implement this interface. Only those methods specified in a "remote interface", an interface that extends java.rmi.Remote are available remotely.
(Ref : https://docs.oracle.com/javase/7/docs/api/java/rmi/Remote.html).

2. ServerImpl :

This class implements the methods of ServerInterface.

- While registering a peer, it iterates over the list of peers and checks if peer information is unique and then adds it to the list of peers as well as adds its files to the list of FileInfo objects.

- While searching for a file that peer has requested, it iterates over the list of checks if they are available in the local server directory.

- If file isn't available it searches for it neighbors using getneighbouringSuperpeers , and sends query message to neighbouring to find if files are available in there peers.

- In push mechanism the peer send invalidate message to it connected supe-peer and then this super-peer sends the message to it connected peer through invalidate method.

- In pull mechanism we use sendpullrequest to fetch the version of downloaded files in every peers version.

- The checkDownloadedList method checks if the time-to-refresh has expired in order to mark it TTR expired.

3. Server: This is the entry point to Server application.

   It allows the user to enter the port on which the server need to be run. Creates a registry and binds the server object to particular name, which is used by client for remote access. We have to provide the port to which the server is binded.

Peer Classes

1. Peer Interface: It implements remote interface and has an abstract method called retrieve to be remotely accessed by other peers to download a file and queryhit method is response from super-peers if any of the neighbors find the file in its registered peers.

2. Peer Impl: It implements Peer Interface.

- When a peer requests a file, it retrieves the file from its directory, converts to the byte array and sends it to peer.

The queryhit method get the reply from super-peer if the file is found in, this methods get the port and leaf-node ,messageid.

fileoutOfOutInfo informs if the file is out of date and lets you download it.

3. Peer: This is the entry point to the Peer application. It asks for peer Id, port num and directory for registering the peer and also creates a registry for the peer and binds it to the port number for remote peers to retrieve file from it.

It also has methods to do intermediate processing by using Hash maps, File Input stream, output stream for searching the files from server, connecting to a peer, downloading from a peer to its current directory.

Schedulers:

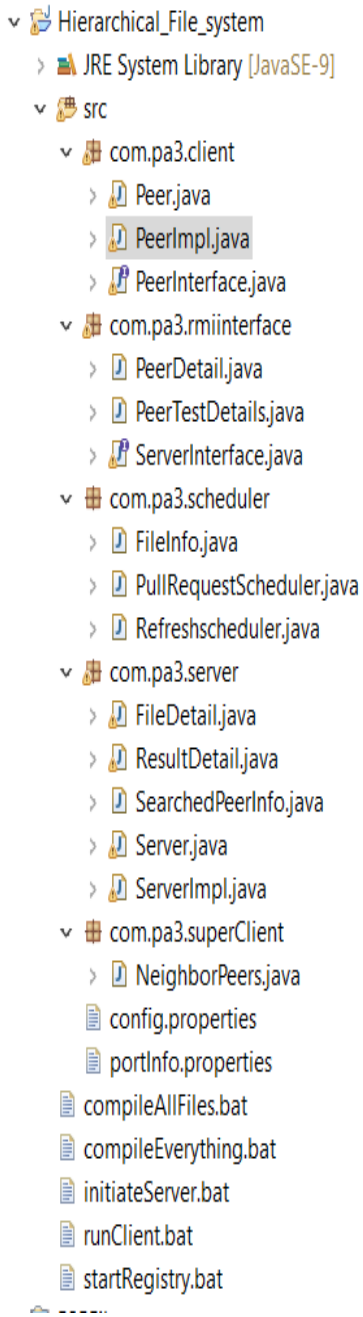
We have two schedulers which we trigger so that the functionality works without being dependent on the peers and server.

1. Refreshscheduler : It marks all the files whose Time-to-refresh has expired as TTR expired.

2. PullRequestScheduler : this is Pull request where the super-peer checks the version of the downloaded file list and discards it it has a cached copy.

6. Package Structure

The structure in which components are placed is shown in the following figure:



8. Further Improvements

- Error handling mechanism can be improved , as to make the system more robust.
- The code can be made more generic.