MAINTAINING FILE CONSISTENCY IN GNUTELLA STYLE P2P SYSTEM

Submitted by

Kashyap Vengalathur Srikanth A20325618

Jai Vijai Baskar Sathyanarayanan A20309537

# INTRODUCTION

In this project, we were required to add consistency mechanisms to a Gnutella-style P2P file sharing system.

# DESIGN

We have used Java programming language to implement the Gnutella system and Remote Method Invocation (RMI) to achieve consistency.

The file search done is purely distributed. Every peer has information about its own files that are present at that particular peer. It also maintains a list that has information about its neighbouring peers. Every file has a master copy which is stored at the origin server. This origin server is considered to be the owner of the file.

So when a peer searches for a file, it sends the query to all the neighbouring peers it is connected to. It also checks for that searched file within its list of files. The result is returned with all the peers that contain the searched file. The querying peer then selects which source to download from.

The returned file will actually be consistent across all the peers. If at one peer, the file is not updated, that peer will not return its instance of the file. Instead it will pretend to not have that file at all.

Consistency in this Gnutella style P2P system is achieved by the following two methods:-

1. Push approach
2. Pull approach

## Push Approach

At every peer, there are two directories, one that has the files that the peer owns and another to keep the downloaded files. Whenever a file is changed at its master copy location that peer broadcasts an INVALID message to all the other peers. This renders all the copies of the file invalid, and these do not appear when searched. Every peer stores the version number, origin server id and consistency state of each file.

**Advantage**

If a file is changed, the master server has to push the update to all the peers.

## Pull Approach

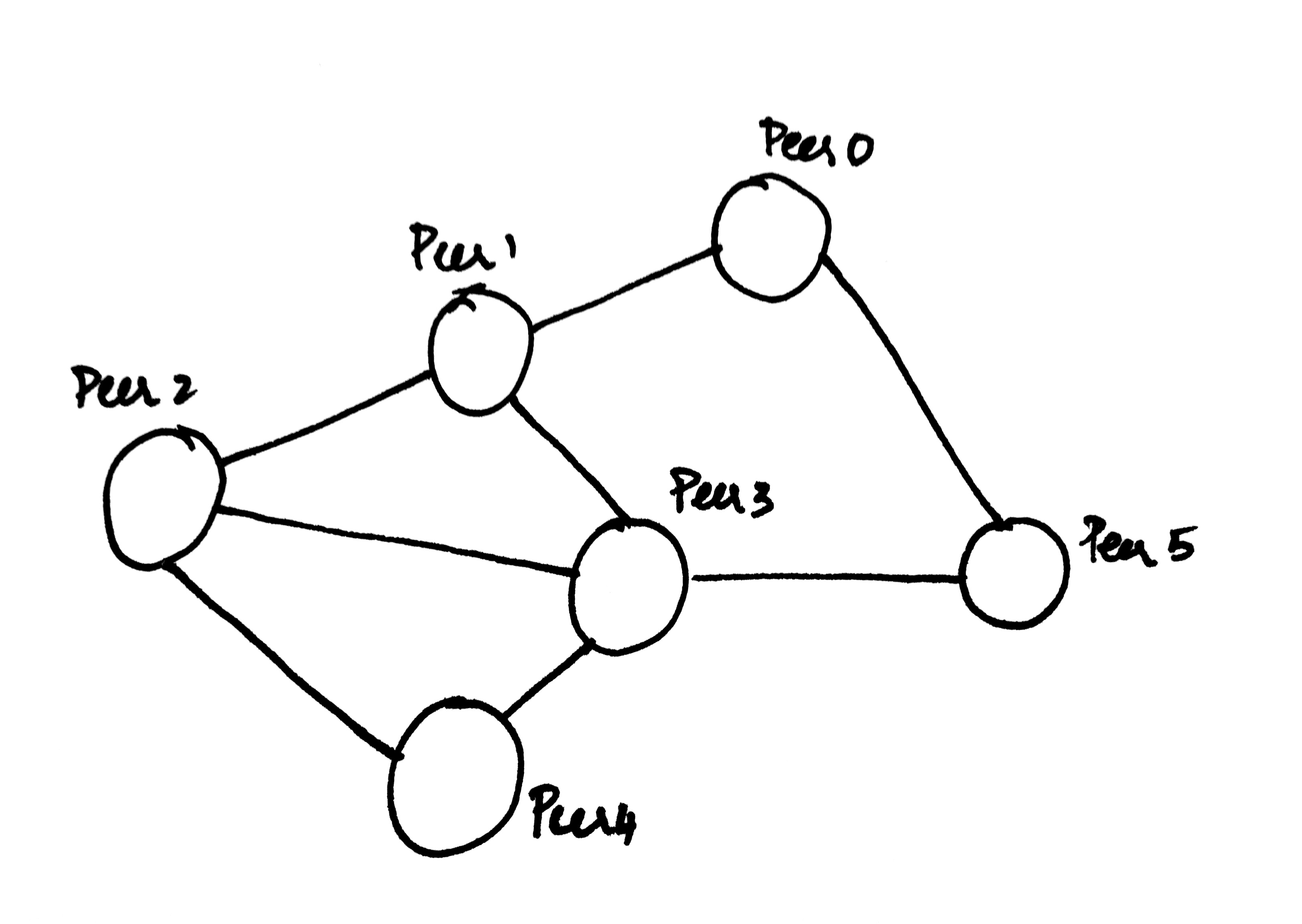
The TTR (time-to-refresh) value is set to be a constant. This is the value used for every file a particular peer owns. Any modifications made to the file will be updated within this constant period of time.

When a peer requests for a file, all the other servers which contain the latest version of the file return values. The file is checked versions based on the last modified time of the file. The server id, TTR, and the last modified time are sent with the request. The client then chooses the location from where the file has to be downloaded. And it is then copied to the requesting peer.

**Advantages**

If a change occurs, it is updated only when a file is searched.

Our design is implemented as follows



Our designed system will work only when all the peers are up and running.

The entire system is implemented in the following parts:-

# **Peer**

Every time the peers are initialized, the config file is read and the peer names, port id and ip address are set. This is setup depending on the connection type the user wishes to arrange the peers.

## Search\_files()

This class contains methods that searches within a peer whether a file is present or not.

## P2pfns()

This class initializes the peer names and establishes connection between neighbouring peers and

## Peer\_main()

The function works for push and pull pull approach in the following manner

### Pull approach

The Gnutella type mesh architecture is constructed and each peer has the information about its preceding peer and its future peers. In this approach, the file that is modified in one server traverses to the corresponding servers and issues a message that the file has been changed and gives out a message that the file has been modified.

The file changed in the root directory is displayed to the other servers when it is defined by the user.

## Pull approach

In the pull approach, the files changed in the root folder is scanned every two minutes and if the files are changed, the invalidation query is sent to every other peer server that has this file.

A separate thread is spawned to watch for files that are updated. This approach uses thread to track the changes of the specified folder.

Methods in this class obtain the ip address of the file and return its path to the client. The client is then given options to choose from being, connecting to that peer server, getting portid and name, sending the files to the server, selecting the file to download, and committing changes to the directory.

It then does the following functions, based on the option chosen by the client:

1. Connection is established to the folder address
2. The port name and the peer id of the peer is obtained.
3. The files that are available on the peer node is enlisted registered to their local index using Hashmap.
4. A new peer server (node) is created and connection is established to the existing peer.
5. The file that is modified in the root directory is sent to all the other peers that are connected in the network.

## Filedownload()

This implements the download or file transfer functionality in the system. It simply inputs the destination and the source directory address. When the situation that a file that has to be transferred from one peer to another arises, it reads the file, creates a new file at the destination and writes onto it. The data transfer is done using byte array. The download process is taken care of by a thread which is created by the peer node.

## Run()

This function measures the time taken for a peer to request and download a file, i.e., sent from one peer to another through the network connection.

## p2pinterface()

This class implements function that call objects remotely using RMI. It establishes the connection to the folder path and also downloads the file to the desired node.

# PERFORMANCE RESULTS

|  |  |  |
| --- | --- | --- |
| **Approach** | **No. of Peers** | **Time taken for search to complete (seconds)** |
| Push | 4 | 20.927 |
| Push | 5 | 22.076 |
| Push | 6 | 25.812 |
| Pull | 3 | 18.039 |
| Pull | 4 | 33.666 |

# POSSIBLE IMPROVEMENTS

1. To make the system read the data from config file has to be implemented.
2. To implement a dynamic filewatcher for the directory, in the current scenario, the file is checked based for the last update time through user defined functions and the operations are performed on request.