Programming Assignment 2

MAINTAINING FILE CONSISTENCY IN YOUR GNUTELLA-STYLE P2P SYSTEM

CS550 – Advanced Operating System

March 26, 2018

NIKITA V. JADHAV

A20401223

1. Introduction

The P2P system is dissimilar from the older Client/Server Models where the files would reside on one Central Server and all the transfers would happen only between the Central Server and the Clients. In P2P File Sharing Application, the File transfer can occur between the individual Nodes/Peers. In this project, we were required to add consistency mechanisms to a Gnutella-style P2P file sharing system. If the file is modified or changed, same change has to be reflected everywhere. Instances of P2P File sharing applications are Napster, Gnutella, Free net etc.

2. Description

The Project is based on a Hybrid P2P model which involves a decentralized peer to peer file sharing system. Each peer should be both a server and a client. As a client, it provides interfaces through which users can issue queries and view search results. As a server, it accepts queries from other peers, checks for matches against its local data set, and responds with corresponding results. In addition, since there's no central indexing server, search is done in a distributed manner. Each peer maintains a list of peers as its neighbor.

Whenever a query request comes in, the peer will broadcast the query to all its neighbors in addition to searching its local storage. In this model all file transfers made between peers are always done directly through RMI (Remote Method Invocation) that is made between the peer sharing the file and the peer requesting for it. Peer acts as a server as well as a client.

3. Purpose

To Design and study the internals of Maintaining File Consistency in Your Gnutella-Style P2P System along with acquaintance to the concepts of RMI, Processes, Threads and events.

4. Requirements

a. Hardware requirements:

Multiple systems to simulate the Peers or can use Multiple Virtual Machines to the Peers.

b. Software requirements:

- Java Development Kit (JDK), JRE (Java Runtime Environment) and Netbeans (IDE).
- Apache Ant- apache-ant-1.10.1

Programming Language: Java

5. Design Overview

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

Following are the 2 methods to achieve consistency in the Gnutella style P2P system:-

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 peers 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. The advantage would be 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.

6. Design

- The P2P File sharing system is designed keeping in mind the P2P architecture and its underlying protocols. The entire design is implemented using Java, Sockets and Threads. As a client, it provides interfaces through which users can issue queries and view search results. As a server, it accepts queries from other peers, checks for matches against its local data set, and responds with corresponding results.
- In addition, since there's no central indexing server, search is done in a distributed manner. Each peer maintains a list of peers as its neighbor. The origin server will "broadcast" an invalidation message out whenever there's a modification to a file. This message requires no reply from the recipients.
- The push approach, whenever the master copy of the file is modified, the origin server broadcasts an "Invalidate" message for the file. The invalidate message propagate exactly like a "query" message. Upon receiving an "Invalidate" message, each peer checks if it has a copy of the object (and if so, discards it).
- Further, it propagates the “Invalidate” message to all its neighboring peers. In this manner, the invalidate message propagates through the system and invalidates all cached versions of the object.

7. Implementation

The Implementation is done using RMI for communication between the decentralized peers Each Peer acts as both client and server in which server registers all the files, query the file and queryHit in case of the file found.

The various Functional Modules are:

PeerClient:

- . acceptFile
- findFile(String filename)

- download file
- public void setup()

PeerServer:

- boolean sendFile(PeerClientIF peerClient, String filename)
- void query(String msgID, String cIP, String cPortNo, long timeToLive, String filename, String cPeerName)
- void queryhit(String msgID, long timeToLive, String filename, String hitPeerIP, String hitPeerPN, String hitPeerName)
- void invalidate(String msgID, String pIP, String pPN, String originID, String fname, int vn, String pname)
- void updateFileListCtr(String dirType)
- void updateFileListDel(String dirType)
- void updateFileListMod(String dirType)
- int pullValidation(String fname, int versionNum)

Peer as the Client:

A) acceptFile:

This function is used to create the file and write to the file using FileOutputStream

B) findFile(String filename):

It searches the entire network for the peers that contains the file intended for download and then lists the peers that was returned and prompts user to choose which of the peers to download from

c) Download file:
downloads chosen file directly from peer.

Peer as the Server

A) Send File:
task: sends a requested file to the requesting peer by converting the file to message stream using 'FileInputStream'

B) query method:

task: spuns two threads. one to search itself for the file and the other to subsequently query all neighboring peers

C) QueryHit:

Each neighbor looks up the specified file using a local index and responds with a queryhit message in the event of a hit. The queryhit message is propagated back to the original sender by following the reverse path of the query. Regardless of a hit or a miss, the peer also forwards the query to all of its neighbors.

D) Validation:

The origin server will "broadcast" an invalidation message out whenever there's a modification to a file. This message requires no reply from the recipients.

8. Advantages and Disadvantages:

Advantages

- Fully decentralized
- Search cost distributed
- Processing per node permits powerful search semantics
- File Consistent among peers.

Disadvantages

- Search scope may be quite large
- Search time may be quite long
- High overhead, and nodes come and go often

9. Tradeoffs

- Dimensional array is used instead of ArrayList or hashmap.
- Files of large size say few 100MB's cannot be transmitted/downloaded.

10. Possible Improvements/Extensions

- To implement a dynamic file watcher for the directory, for instance, the file is checked based for the last update time through user defined functions and the operations are performed on request
- Support for Large files over a few 100 MB's
- Performance improvement by using different data structures like HashMap.

11. References:

a. http://ieeexplore.ieee.org/document/1210292
b. http://www.vogella.com/tutorials/ApacheAnt/article.html