Advanced Operating Systems
CS-550

Programming Assignment 3

Maintaining File Consistency in Gnutella-Style P2P System

Group Members:

Kunal Dhande
A20369862

kdhande@hawk.iit.edu

Mohit Kulpe
A20372614

mkulpe@hawk.iit.edu

Content Table

1. Introduction

1.1. Problem Statement

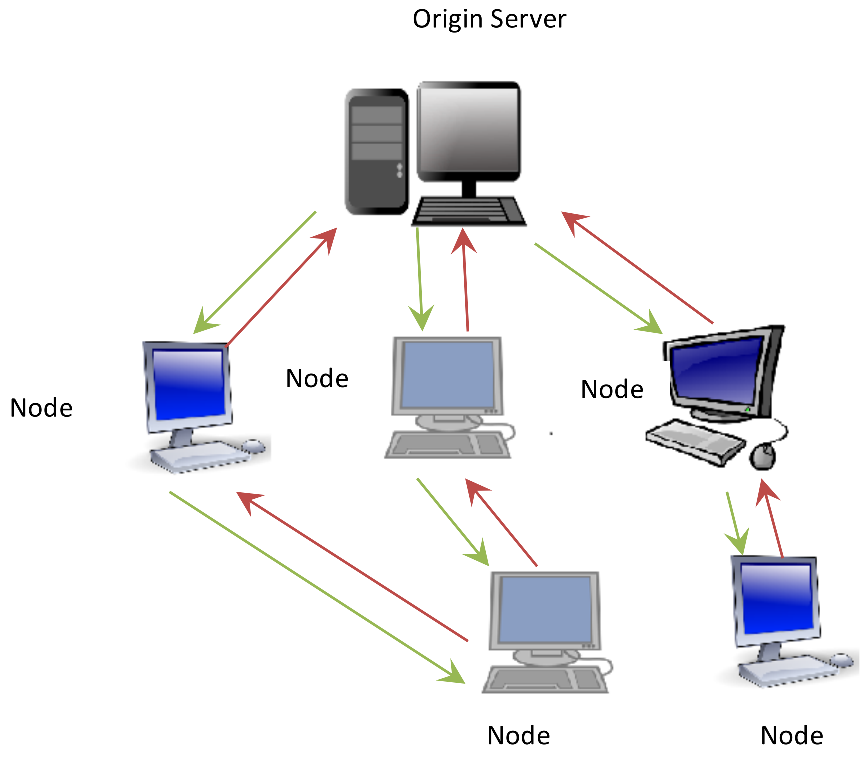
The goal of this project is to add consistency mechanisms to a Gnutella-style P2P file sharing system. While audio files that are typically shared using today's P2P systems rarely change after creation, in the future we can expect to use P2P systems to share other types of files that do change after creation. The objective of the assignment is to ensure that all copies of a file in the P2P system consistent with one another.

1.2. System Introduction

This document provides a detailed design of the Distributed File Sharing System. It provides an overview of how different components in the system interact with each other. In this Peer to Peer File Sharing system searching is done in a distributed manner. As a Client user passes a query on other end as a Server it accepts query and searches in local database responds to that respected query. In addition to this we have implemented Consistency mechanism to check whether the cached versions of objects are expired or not. This can be achieved either by using *push or pull*.

2. Architecture

A decentralized peer-to-peer system, consisting of hosts connected to one another and running software that implements the Gnutella protocol. The connection of individual nodes forms a network of computers exchanging request & response in form of queries & response to queries. In addition to this it can validate the object present in cache by passing PUSH & PULL messages.



Note:

→: PUSH (Invalidate message)

→: PULL

Figure 1

2.1 Working of system

1) Whenever the master copy of the file is modified, the origin server broadcasts an PUSH ("Invalidate") message for the file.

2) Upon receiving an "Invalidate" message, each peer checks if it has a copy of the object and if so discards it.

3) Further, it propagates the “Invalidate” message to all its neighboring peers. Also, broadcast that "Invalidate" to multiple other nodes in network as shown in figure.

4) In the pull approach each peer can initiate PULL message to the origin server to see if a cached object is valid or not.

5) The origin server upon receiving PULL message will check the version number of the master copy and respond accordingly. If the master copy is newer, the origin server sends a "file out of date" message back and the peer then discards its cached copy and notifies the user.

3. Implementation

3.1 Origin Server (PUSH)

Whenever the master copy of the file is modified, the origin server broadcasts an PUSH ("Invalidate") message for the file. Upon receiving an "Invalidate" message, each peer checks if it has a copy of the object and if so discards it.

3.2 Node (PULL)

In the pull approach, each peer can initiate PULL message to the origin server to see if a cached object is valid or not. The origin server upon receiving PULL message will check the version number of the master copy and respond accordingly. If the master copy is newer, the origin server sends a "file out of date" message back and the peer then discards its cached copy and notifies the user.

.

4. Future Enhancements

- We should pass full name of file which should be searched/download, hence our future enhancement will be to apply search algorithms to find the partial matches.

- Network efficiency and scalability can be improved by adding some nodes in network with high bandwidth.

5. Assumptions and Constraints

- If the node which has file is closed, then the other peer which wants to download that file will not be able to connect and download the file.

- Requester should type exact file name that should be searched else the query response.

- A node in a network with low bandwidth will not perform efficiently as compare to other nodes with high bandwidth, hence will degrade overall performance.