Basic Consistent GNUTELLA Protocol Implementation – Design Document

Version 1.0

11/12/2012

Advanced Operating Systems

Navaneetha Krishnan

Table of Contents

[1. Objective: 3](#_Toc340532299)

[2. Processing 3](#_Toc340532300)

[2.1 Program Flow 3](#_Toc340532301)

[2.1.1 Gnutella protocol implementation 3](#_Toc340532302)

[2.1.2 Consistency implementation 4](#_Toc340532303)

[2.2 Program Description 4](#_Toc340532304)

[2.2.1 Servent / Server mode 5](#_Toc340532305)

[2.2.2 Client / User mode 6](#_Toc340532306)

[2.2.3 Coordinator mode 7](#_Toc340532307)

[2.2.4 Configuration file 7](#_Toc340532308)

[2.2.5 Master Catalog 7](#_Toc340532309)

[2.2.5 File Catalog 7](#_Toc340532310)

[2.3 Instructions to Run the Program: 8](#_Toc340532311)

[2.4 Assumptions & Decisions made: 8](#_Toc340532312)

# Objective:

Consistent Gnutella program helps to share files between peers without any need to have centralized index server & maintains consistency among the peers. If you peer need a file, it will broadcast a query message to its neighbors. Those peers will relay it to its own neighbors. If any peer has that file, it will send hit message which will reach requester by following the reverse path of query message. Then the requester directly connects with ‘HIT’ file server & retrieves the required file from it. Also consistency is maintained by two different protocols 1. Push & 2. Pop. Push protocol is strong consistency model that is, the owner of the file intimates its entire system if a file is modified. Pull protocol is polling method where peer polls origin server to check validity of the file.

# Processing

## Program Flow

The flow of the program is shown below.

### 2.1.1 Gnutella protocol implementation

QUERY MESSAGE HITQUERY MESSAGE FROM SERVENT4

**SERVERNT 2**

**SERVER**

**CO-ORDINATOR**

**SERVERNT 2**

**CLIENT/USER**

**SERVERNT 1**

**SERVER**

**CO-ORDINATOR**

**SERVERNT 1**

**CLIENT/USER**

**SERVERNT 4**

**SERVER**

**CO-ORDINATOR**

**SERVERNT 4**

**CLIENT/USER**

AFTER RECEIVING HITMESSAGE

**SERVERNT 3**

**SERVER**

**CO-ORDINATOR**

**SERVERNT 3**

**CLIENT/USER**

### 2.1.2 Consistency implementation

As mentioned earlier, the consistency is being carried out by two different protocols.

**SERVERNT 2**

**SERVER**

**CO-ORDINATOR**

**SERVERNT 2**

**CLIENT/USER**

Invalidate message for a file

(PUSH)

**SERVERNT 1**

**SERVER**

**CO-ORDINATOR**

**SERVERNT 1**

**CLIENT/USER**

POLL REPLY (PULL)

**SERVERNT 4**

**SERVER**

**CO-ORDINATOR**

**SERVERNT 4**

**CLIENT/USER**

POLL REQUEST

Checking validity of original file (PULL)

**SERVERNT 3**

**SERVER**

**CO-ORDINATOR**

**SERVERNT 3**

**CLIENT/USER**

Duplicate entry of invalidate message (PUSH)

## Program Description

The program has server, coordinator & client functionalities. The program acts as Server or Client or coordinator as per inline argument passed during execution of it.

Here the functionality of the program is described as per its mode of call.

### 2.2.1 Servent / Server mode

The Gnutella program will act as a server when it is being called in this mode. After initiating server, it will receive messages from client. As per the message type, it will call any one of the below functions.

* Query Handler
* Hit Query Handler
* Send file
* Invalidate Message Handler (PUSH)
* Poll request handling (PULL)
* Poll reply handling (PULL)

#### Query Handler:

The function query handler will be called when a query message is received from its neighboring peer. First it will check whether it has already seen that message or it is it from its own system. If not, it will store message id and upstream peer address in its associated array list.

Later the message will be relayed to its neighbors after reducing TTL value by 1 and if only if new TTL value is greater than zero. After it, it will check whether it owns that file in its local directory. If it so, it will generate hit message in the format below & send it to upstream peer.

Hitquery$msgid$TTLvalue$filename$<localIpaddress>$serverport$filelastmodtime$originidbit

#### Hitquery Handler:

This will be invoked if the server received a message with a word Hitquery. After receiving it, it will check whether it has already seen this hit query message. If not, it will check whether it is created by its own client. In that case, it will connect with its own client & passes the message to it. Otherwise it will send the message to the upstream peer from whom it received query message.

#### Send File:

This will be invoked if the request type is “obtain”. The message also contains filename which is from a client mode program of a peer. The server finds the file in its local directory & sends it to the requested peer. Along with the file, below info also sent which will be stored in file catalog. If the file is received from other peer, then the entry for the requested file in file catalog will be sent. So receiver will be having the info of origin peer.

<filename>$<version>$<originpeerid>$valid#<originpeerid>$TTR$<lastmod time>$<receiving time>

#### Invalidate Message Handler:

This message will be received while the system operates in PUSH protocol. Once the message is received, the message id will be checked whether it is received earlier. If not, it will be broadcasted to all of its neighbors. Then the filename along with version number & origin id is checked in the file catalog. If it exists, then the file is marked as invalid in the file catalog. So the invalid file will not be considered for any query request.

#### Poll Request Handling:

This request will be handled at the time of PULL protocol execution. The request would give information of filename, last modified time of that file as per requestor. The server has to check the file last modification time. If it is greater than the mentioned value, it should reply as the file is invalid. Otherwise it should reply it as valid one.

#### Poll Reply Handling:

This message will be received as a reply to the poll request raised by its own coordinator. The message contains the filename, status of the file & TTR value to renew. The file status will be updated in file catalog. If the file is valid yet, then TTR value should be renewed after status update.

### 2.2.2 Client / User mode

This mode of the program only provides an interface to give user inputs. Initially it will get the name of the file which is to be searched or retrieved. It checks whether it has its own directory. If it is not exists, then it will create a query message & sends it to all of its neighbors. The query message will be in below format.

query$msgid$TTLvalue$filename$serverport

The message id will be created in the format

<localipaddress>:<serverport><current time in milli seconds>

After sending query message to all of its neighbors, it will establish a connection with its own server on a specific port. Once it receives hitquery message from its server, it will send “obtain” message directly to the server which is holding that requested file**.** The client will save file once it starts receiving the bytes. If it does not receive hit message from its own server till the cut off time, it will close the connection & inform the user that the file is not available in the system. The info of the received file is stored in a catalog called File catalog.

### 2.2.3 Coordinator mode

The coordinator instance is the responsible for establishing consistency. The functionality of this instance will differ as per the mode of execution.

#### Push Mode:

The coordinator should get the filename from the user which needs to be updated. The file will be updated & master catalog is also be updated with new version of the requested file. After file update, an invalidate message will be constructed in the below format & send it to all of its peers.

Invalidate$msgid$filename$version$Originpeerid

#### Pull Mode:

The coordinator keeps on check on all files whether TTR value is expired or not. If the file’s TTR value is expired, then the status of the file should be updates as TTR Expired. The files whose TTR value is expired should send a request to origin peer to check whether the file is valid still or not. TTR value expiration will be found out by comparing current time with (received time+TTR).

### 2.2.4 Configuration file

The configuration file is having the IP addresses of neighbor peer along with its server port numbers. The configuration file will be read and stored in an array at the start of the program irrespective of the mode of call. But the first line of the file will intimate the system in which mode it should run (PUSH/PULL). The second line will intimate the server about the value of TTR.

### 2.2.5 Master Catalog

This is the file exists in the master directory of the peer. This catalog holds the information of filename & its version in the format <<filename>>$<<version>>

### 2.2.5 File Catalog

This file present in the received files directory. This holds information for both push protocol message & pull protocol message. It provides filename, version of the file, origin peer id, status of the file, TTR value, last modified time of the file & time of received.

## Instructions to Run the Program:

1. Initiate all the server mode programs eg***: java Servent1 servent***
   1. Remember each program (if execution is going to happen within a single system) should have separate directory & specific configuration file in it.
   2. Either the value of the ServerPORT (Static variable) or LocalIPaddress should be unique for each program.
   3. Directory path should be defined in the program statically.
2. Initiate all coordinator mode programs eg: ***java Servent1 coordinator***
   1. In push mode, the file to be modified will be taken from user.
   2. In pull mode, just initiation is enough.
   3. Start coordinators only after all servents are up.
3. Client program can be executed as ***java Servent1 user***.
   1. It will ask file name to enter. Kindly provide the filename that you wish to retrieve.
   2. Once the request is completed, it will ask “Do you wish to Continue?”, if you want to search one more file then provide ‘y’.

## Assumptions & Decisions made:

* Each peer is residing in different node.
* All peers will be having two directories. One for keeping all its master files & other one to store received files from other peers.
* The peer servers are having separate directories & path of the directories are known already which is explicitly provided in the program.
* Assumed that master files won’t be changed after first execution of the program. Since the catalogs will be having the version numbers as per it.
* The Server PORT will be statically given in the program.
* First 25 entries of the associated array lists will be cleared out once the size of the array lists reach 50.