0. Before starting this program

This program need a configuration file except for the source code of this program. The file ‘configuration.xml’ contains server ID, server IP, server Port, and TTR (Basically the IP address are set to localhost). Also, there should be the shared directory to transfer or download from other neighbors. The shared directory contains master copies and download directory. The download directory contains downloaded files from other peers. The shared directory path that is located in ‘Gnutella.java’ should be modified.

1. Main driver - (Gnutella.java)

Firstly, this main driver reads the network topology and store the information of servers into running program so that the program can be ready to participate in the file sharing network. When we start this program, we can put the argument that is used to decide the consistency policy of the network. The first is push-based and the second is pull-based. By deciding the consistency policy, this main driver will run the proper client thread and server thread. Secondly, there is a String variable ‘sharedDirectoryPath’ in this driver that stores the path of the shared directory. When this program starts, user is supposed to enter the server id. After entering the server id, the shared directory’s name become ‘sharedDrectoryPath+ServerID’ (for example, when the server id is 1 then shared directory is ~/incoming1). Then, this driver gets the neighbor numbers by console input to connect to the neighbor of this servent. Now all the requirement for starting file sharing is set. Finally, this driver starts the proper (according to the consistency policy) client thread and server thread at the same time so that this program can obtain a file and react to other messages concurrently.

2. SharedDirectoryMonitor

The shared directory monitor detects any kind of event on the shared directory. To detect the events any time, it was implemented by thread and WatchService class. When the monitor detects creation of a file, it adds the file into master copies list. When the monitor detects deletion of a file, it removes the file from master copies list. Finally, when the monitor detects modification of file, the monitor increases the file version by 1 and set the last modified value newly.

3. Push-Based Client

The push-based client is implemented with single thread for getting user input and requesting to other servents concurrently. Even though the server is running, user can request to search or obtain file in the network without any interrupt. Once this client thread starts, it prints the menu on the console. After selecting a menu, it prints the menu iteratively. With this menu, the user can search a file, obtain the file or exit this program. When searching is selected, ‘Query’ method is executed to make a new query for searching the file. After making an initial query, this client passes the query to server thread to broadcast the query into entire network (result of searching will be described at server design). When obtaining is selected, this client part prints the list of servent that are holding the file. Then user can select the servent to download. After that, this client make a new message for downloading consisted of ‘obtain, filename, requestorServentIP, requestorServentPort’. When the server gets this message it starts upload threads that connect to this server. Additionally, when a shared directory monitor detects any modification of master file, this client makes an invalidation message and broadcast it to network.

4. Pull-Based Client

Like push-based client above, the pull-based client also perform search, obtain, and exit. Also, this pull-based client can refresh the out dated file that are figured out by the original server that holds master copy. The most important thing for this pull-based client is that this pull-based client sends a pull message when a specific downloaded file’s TTR is expired. Then the pull-based server will gets the response whether there was modification of the master copy.

5. Push-Based Server

Server is also implemented with thread but for the multiple connection between neighbors, threads is created when the connection is established from any neighbors. So, this server waits connection requests from single or multiple neighbors and whenever connection is established, this server makes new thread. This server has an inner thread that is called ServerProcessor. When this server’s serversocket gets connection, it passes the accepted socket to the server processor then the server processor processes every request. There are 3 types of messages query(broadcast), queryhit, and download request. Each message is consisted of information of specific server information that are proper to operations (such as query, query hit, obtaining).

Query Message :
SeqNo/IP(where the query made)/Message Sender(where the message was sent)/filename/TTL

Queryhit Message :
Sequence No/IP(that has the file)/List of Sender(for traceback)/IP(where the query made)/ servent id of current sender/filename/TTL/hit/last modified time/master copy bit

Obtain Message :
obtain, filename, serverip, serverport

When the server gets query message, the server looks up the local file list. If file exists, the server makes new query hit message and trace back to the original servent. If file not exists, the IP address of the servent(hop) is added to the sender list and broadcasted until the file is found or TTL is expired. And when the server gets query hit message, the server compares the last item of sender list and current server information. When the information is same, the original sender recognizes that it is the message that indicates specific servent is hoilding the requested file. When the server gets obtain message then the server starts file uploader with the requestor server information.

6. Pull-Based Server

The pull-based server is almost same with push-based server. However, this pull-based server gets request from other peers asking the validity of file and response as well. When this server gets pull request from other pull-based server, this server checks whether there was modification of master file. And responses to the requestor accordingly. Whenever any other peers requests obtain a file, this server immediately send the file information including last-modified time of the file.

7. TTR-Checker (Pull-Based only)

This TTR-Checker is executed right after a file downloaded from another peer. By using TTR value gotten by original as a response message of obtain, the TTR Checker will automatically ask new information to original server whether the master copy was modified or not.

8. Data Structure - (MessageID.java ServerInfo.java)
ServerInfo stores information of server information that are given in the configuration file. When the program need to get information about network or other servers, the program will access to the ServerInfo to get specific information. MessageId store information of messages like sequence number, original requestor, ipAddress and port number when they are required.

9. File transfer - (FileDownloader.java FileUploader.java)
File downloader and file uploader were implemented with thread so that multiple request and response are done concurrently. So, the requests or responses don't need to be queued or stored to be processed later.

10. XML Parser
XML Parser gets network information from 'configuration.xml' so that the driver can use it.

11. Improvement and extensions.
If the network is much huge, ping-pong must be implemented so that peers really can distribute the burden of tasks and figure out the original sever and the requester server. Also, there can be a secure encryption function in this program so that any other person can't get the information of this network. Or, if a file is huge, it will be better to make a client to get a specific file from multiple servers.