Programming Project 4
CS5352 Advanced Operating Systems Design
Spring 2017

Implementation

This project is about implementation of Gnutella-style peer-to-peer (P2P) file sharing system, where signal peer act as both client and server with performing the file search within the specified network without using the Indexing server. I have implemented using java socket programming and multithreading concepts. When peer is started its respected server thread is started which runs in the background, given config file is read to find its peer and starts the client thread for each neighbor and when the neighboring peer gets online connection is established this connection is used to search file among the network.

Server thread is used to search file within its local directory and store the result into the list once the local search is performed it reads the config file to start the client thread to all its neighbors. time-to-live (TTL) is attached to the message along with the filename need to be searched. TTL is used for preventing query messages from being forwarded infinitely many times. Client thread is used to send the file to search to servers which are connected to it. Server Download thread performs the downloading of file from the peer which is selected by the user.

Same server and client threads are used for broadcasting the file modification message to all the neighboring peers which has that modified file.

How it Works

Main.java

Main.java class take 2 arguments (Config file name, Peer ID, Peer's shared directory). Server thread is created with this peer id and which will run in background poll's for a client thread with the same port number which is specified in config file for this peer.

Peer start the client thread for the neighbors by reading the config file and waits for the server thread with the same respective port number. Once all the client thread created list will be defined which contains list of peers and their shared files.

User will be provided with two option one to search file and download and to Broadcast file modification message. In the file download option user enter the file to be downloaded and the client thread sends message to all the neighboring peers a search message with filename and TTL. Neighboring search its local shared directory for the requested file and forwards the request to its neighbors and request will forwarded to neighbors till TTL value reaches 0. All the neighbor returns its peers id if it contains the requested file. List of peers containing the file will be displayed to the user.

User select the peer from which file need to be downloaded. Once peer id is selected server thread is invoked and connections is established with the current peers thread as a client. Then the requested file will be downloaded as stream of bytes from server to client.

In the other flow that is broadcasting file modification follow I have used the push based approach. In this user is asked for a file name to broadcast the message same server and client threads are used here message stating the file modified will be pushed to all the neighboring peers which has the file.

Trade-offs and Improvements

Socket programming makes this Gnutella-style peer-to-peer (P2P) model platform independent, client and server would form connection regardless of the platform in which they are implemented. But in usage when a client connects to the server till that time server continuously polls for request from a client leading low performance. Also, the InputStream of client continuously polls until it receives a data from client. Also, since the threads are extensively used hence it induces a delay when there are thousands of peers connected in a topology.

Improvements:

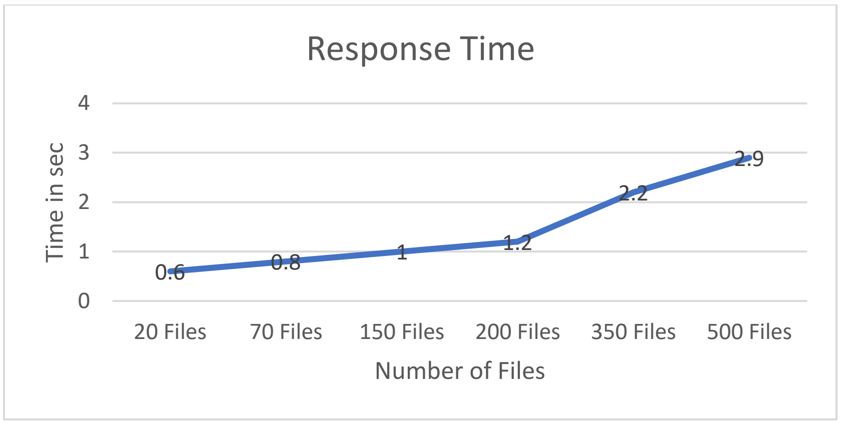
•Performance of the system can be improved by using Avro or protobuf for serializing and transmitting t the data in place of native java serializable

• This Project can be improved by provide User interface

Performance

Response Time

Number of Files	Time in sec
20 Files	0.6
70 Files	0.8
150 Files	1
200 Files	1.2
350 Files	2.2
500 Files	2.9


Response Time
Time in sec
4
3
2
1
0
0.6
0.8
1
1.2
2.2
2.9
20 Files
70 Files
150 Files
200 Files
350 Files
500 Files
Number of Files