CS550 Advanced Operating System
Programming Assignment 2

Shan Huang A20281384
Fan Zhang A20280966
Oct 19, 2014

A Simple Gnutella-style P2P File Sharing System

Overview

This document describes the overall design and implementation of file consistency mechanisms of the Gnutella-style Peer to Peer file sharing system. The whole system is implemented in JAVA. And the technologies that involved in the system are Object-Oriented method, TCP socket, and multi-threads programming. The system can work on any operating system, which Java environment is provided, and can also be set up on one or multiple machines.

Whole Design

This system contains multiple peer programs. Each peer is both client and a server. As a client it can send queries to neighbors to search or download a file. As a server it can respond the queries of searching and provide the upload function. Peers are connected through two topologies, Star and 2D-mesh, which determines the neighbors of a peer.

We provide a client user interface which binds to local peer for users to input commands, and for each peer, it has a monitor for users to check the processing status of a peer program. In our program, we can have multiple user client running on different machines just by starting the user interface of each local peer.

The major responsibilities of a peer as a client is searching a file. One user input a searching file commands, this command be received by a thread, and the system will generate a query and broadcast it to all neighbors of this peer. The neighbors will look into their own local file folders and return a result: hit or miss. Then peers in neighbors will broadcast the query to their neighbors. In the end, all peers will return a hit or miss result. In order to prevent the query is forwarded forever, we limited the number of it. The figure1 shows the process of searching a file. The major responsibilities of a peer as a server is uploading a file, and send it to the peer which initiate the download function. In our design, we use ObjectStream method to send a message and we make it read several bytes at the same time for sending a file, the message for downloading includes tempbyte[][] and byteread[],which are all encapsulated into message class. By using this method, the file sending function is not limited to the type of ".txt". The figure2 shows the process of downloading a file.

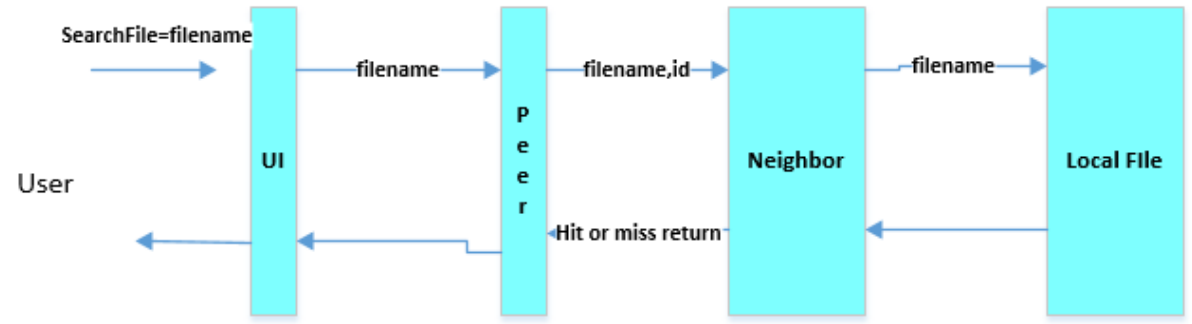

Figure 1. Search File Process

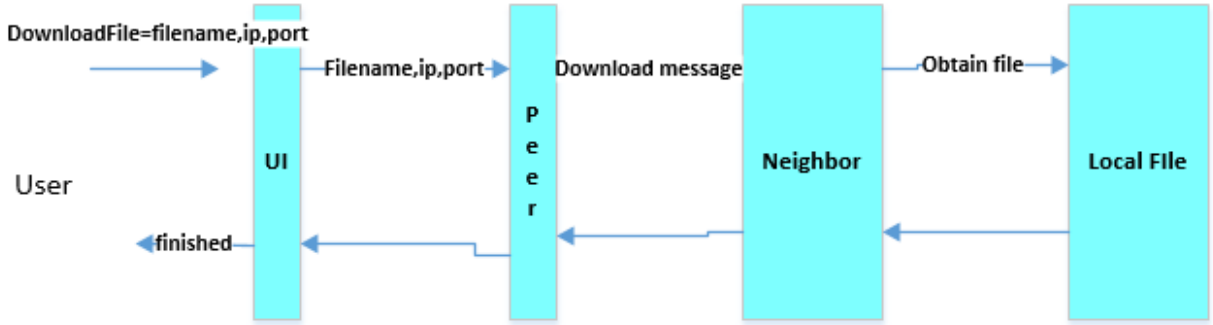

Figure 2. Download File Process

Design Issues

1. **How the query message is designed?**

The message type for each file searching query is:

Message (TTL,sourceid,filename,message id, RequestFunction);

The message ID is generated by peerID and system time. Each message ID is unique.

messageid = System.nanoTime()+"_"+peer.getid();

TTL vale is set to 7 according to our topology, once a query has been replied, it will decreased by 1.

The sourceid is added to message in order to broadcast. In our design, peer will not broadcast a query to its source peer.

The response message from a peer is: Message (TTL,sourceid,filename,message id, RequestFunction, peerID and peerPort);If a peer has the requested file, it will return the peer ID and the port number. If not, it will return nothing.

2. **How to prevent the queries from unlimited forwarding?**

To prevent the queries from unlimited forwarding, we put a time to live (TTL) value in system. Each query has a TTL value which is initialized to 7. Once a query was replied with result, the value would be decreased by 1. If the TTL value is reduced to 0, the message query will not be forwarded to neighbors.

3. **How to maintain queries ?**

The mechanism to maintain queries in our design is adding a message list (ArrayList<String>) which will store the message ID of each query message. Once the query message is forwarded to a peer, it will check the message list first, if it's already in the list, other message which is not in the list will be processed first.

Topologies

There are two kinds of topologies. Star and 2D mesh.

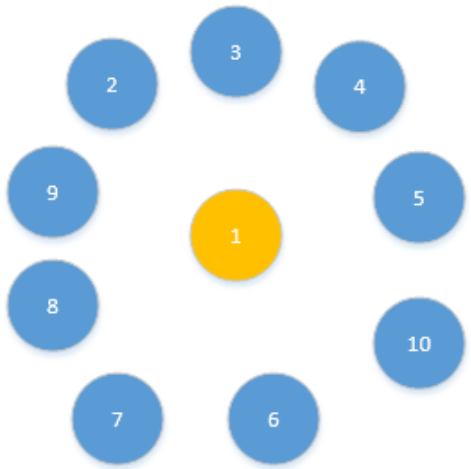

Figure 3. Star Topology

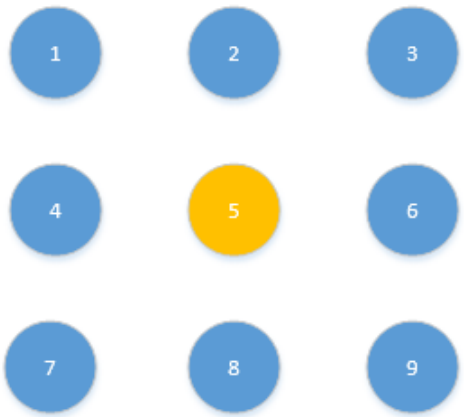

Figure 4. 2D mesh Topology

Design tradeoffs

This system was implemented in a test-driven development method. Because of the time limitation, we did not focus too much on error detections, and small details for user's convenient. We did not provide GUI either, but all the functions are implemented and tested successfully.

Improvements&Extensions

Compared to Napster P2P System, we have made these improvements.

1) User can generate numerous peers by setting the configuration files, and create different topologies.
2) The system can be run on different machines under same LAN.

The possible improvement of the system is to provide a more efficient algorithm of TTL. Message is sent repeatedly in topologies, when the system becomes very large, it will create wasted message and consume the bandwidth.

Contribution of each member

Shan Huang:
Design the framework of the system, Write code for major functions. Test and Debug the program, write documents.
Fan Zhang:
Test and Debug the program. Write code for part of functions, write documents.