Design Document

1. Introduction

This program is about a Napster-style peer-to-peer (P2P) file sharing system, mainly including two parts, a central indexing server, which indexes the file names of all of the peers that register with it, and a peer that play a role of both a client and a server.

In this program, java programming language and socket programming and multithreads methods were used to implement this simple P2P system.

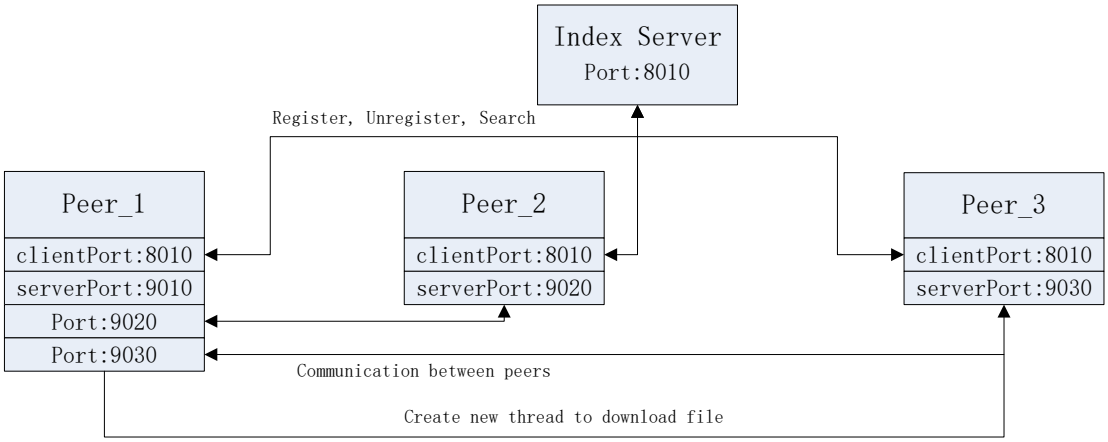
2. Program Design

This program design is divided into 2 parts, index server and peers. TCP socket and multithread are mainly used in this program. The detail design principles and implemented functions are showed as follows.

2.1 Overall Program

The index server uses server port to connect with peers. All peers use the same port to communicate with index server. In the following figure, we take the server port 8010 as an example. Each peer uses this port to register or unregister file on the index server. Peers' requests of search are also through this port connect to the index server.

Each peer uses its own server port to accept other peers' request messages. We set server port of peer_1, peer_2 and peer_3 to 9010, 9020 and 9030 separately as an example.



2.2 Index Server

A central indexing server includes two functions. One is to be invoked by a peer to register all its files with the indexing server. Another is response to peer's search requests.

2.2.1 Registry

Peer uses client port to connect with index server, sending and receiving messages. When peer needs to register a file, it sends one register message to index server, including register command and file name. The server receives register message to add this file into server file list.

On the index server, we just use array list to store files' information instead of using other complicated data structures, such as data base for not too much files. And the file information contains file name and peer ID, so that we know which peer this file came from and we will not confuse about the same file.

2.2.2 Search

Peer sends search message to the index server, including search command and file name. The index server calls the search function on server side to find the file information and return this information back to the peer. The complicated search algorithm is not used.

2.2.3 Accept multiple client requests at the same time

The serverThread class is created to deal with multiple client requests at the same time. When one peer send request message to the index server, the server side creates one thread to deal with the request message.

Apart from that, the registry, unregister and search functions are also implemented by using thread method to guarantee several peers call the same function at the same time.

2.3 Peer

A peer is both a client and a server.

As a client:

The first, it can register a local file on the index server for other peers to search and download.

The second, it can search files from the index server.

The third, if the index server has that file, the peer can choose whether to download it or not. If it chooses to download this file, it will set up a connection to the peer, where this file is located in and then download it.

As a server:

It can accept connection requests from other peers for downloading request and provide downloading for other peers.

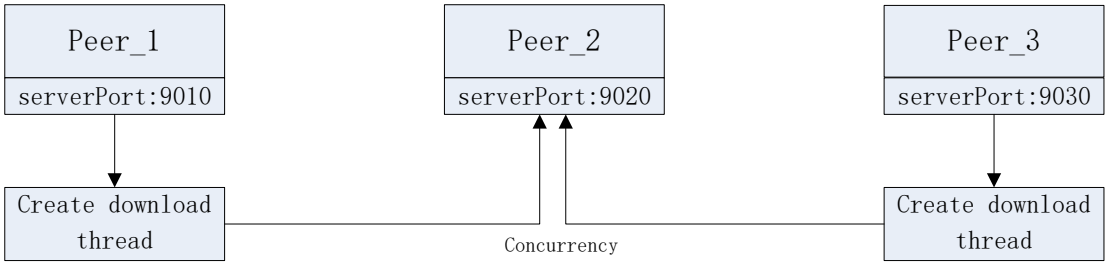
2.3.1 Automatic Update Mechanism

In this part, we use a Timer class to create a schedule that executes every 100ms. In each execution, it scans the specified folder to watch the change of files in it. According to each scan and compare with the register list in the local peer, if one file

registered on the server is removed, it will call the unregister function to unregister this file from the index server automatically.

2.3.2 Obtain

In order to implement multiple peer to download file from the same user at the same time, multithread is used to deal with this problem. When peer_1 want to download file from peer_2, it send a message to peer_2, telling him that it will setup a server socket using another port for downloading. Peer_2 receive this message and then set up connection by using this specified port. Similarly, it has the same procedure to make connect with other peers.



3. Improvements and Extensions

In the first, we implement a download function to obtain a file from one peer. To be more efficient, we can divide one file into several parts, and obtain each part from one peer. In another word, implement the download function to get file from several peers simultaneously.

In the second, another way to implement a multithread mechanism is using thread pool for efficient scheduling of multithread, which can improve the efficiency to allow more users to connect to the index server at the same time.