P2P Application Specification

Introduction

This assignment is asking to implement a part of the peer-to-peer (P2P) protocol Circular DHT. A primary requirement for a P2P application is that the peers are form a connected network at all time.

It is asking to implement one P2P application which has the functions of requesting files and gracefully departing the P2P circle. In this assignment, there not be real files to be requested, but only figure out the location of the files. Moreover, A complication that this P2P network must be able to deal with is that peers can leave the network suddenly (because there maybe a crash), then the remaining peers must try to keep a connected network without these peers.

It is asked to implement this using socket programming of both TCP and UDP protocols.

Design of The System

The system is divided into 6 classes, which are PeerInfo.java, cdht_ex.java, UDPClient.java, UDPServer.java, TCPClient.java and TCPServer.java. UDPClient and UDPServer are used to continuously ping successors to make sure the connection of the established P2P network. TCPClient and TCPServer are used to deal with user requests such as file request and quit. PeerInfo is a javaBean class which contains static arguments in order to share peer record in the hole system. The cdht_ex class is the entrance of this system.

The location is allocated to a file which be formed as follows:

1.The file name wxyz(4-digits) is mapped to the integer N = w*10^3+x*10^2+y*10+z. Then the integer will be hashed by the hash function N/256.

2.The location of a file in a P2P network depends on the hash of file. The rule is, for a file whose hash is n, the file will be stored in the peer that is the closest successor of n. For example, if the hash value of the file is 3, it should be stored in peer 4 (if there is a peer 4 in the P2P network).

Classes

PeerInfo.java

This class is a javaBean class which contains 6 static arguments which are five integers and one boolean. The integers are used to store the peer chain record of one peer. The boolean is used as the condition of the while loops in UDPClient, UDPServer, TCPClient and TCPServer classes.

cdht_ex.java

This class take the 3 arguments(peer, successor1, successor2) passed by the user's typing and store they in the PeerInfo javaBean. Then it starts the hole P2P system by starting UDPServer, UDPClient, TCPServer and TCPClient threads.

UDPClient.java

This class extends the Thread class and is responsible for pinging the two successors of a peer. The pinging message use the UDP protocol.

This class will continuously ping the two successors of a peer in every 10 seconds. This is done by setting the condition of a while loop to be always true unless this peer is quitted by users, and making the thread sleep for 10 seconds at the end of the loop.

The format of the pinging messages is as "X:Y:pN", the first argument before the first ":" indicates the current peer is peer X, the second argument after the first ":" points out this is the Yth pinging message that will be send, and the last argument after the second ":" is used to tell that this peer will be the Nth predecessor of its successor.

For every time when starting the pinging, a timer will be started. The timer is used to monitor the pinging processes, if there is no response back from the successors after timeout. This function will catch a SocketTimeout exception. This mechanism is used to find out a losing package. Furthermore, this method is combined with two arguments "seq" and "acne", which used to check the status of the successors. "seq" and "ackN" means sequence number and acknowledged number respectively. When there catches a SocketTimeout exception, the number of unacknowledged pinging messages will be calculated by use seq minus acne. If the number is equal to 3, then it can be claimed that a successor is no longer alive. Then the UDPClient will be started to request another successor to ask informations to update its successors list.

UDPServer.java

This class extends Thread class and is responsible for continuously giving response of pinging messages received from predecessors of a peer and keeping records of the predecessors. This is also done by a while loop which condition always be true unless the peer is quitted by users.

Once getting the request message, the three arguments which mentioned above in the introduction of UDPClient.java will be retrieved. The "X" which is predecessor's number will be stored in the PeerInfo javaBeans according to the "pN" which means Nth predecessor of the peer and "Y" which is the sequence number of pinging messages will be sent back as a response.

TCPClient.java

This class is designed to continuously get the commands from users and execute these commands. There are two type of commands which are "request xxxx(4-digits file's name)” and “quit”. After gaining the commands from users, the format will be check first. If it doesn’t match the format mentioned above, TCPClient will ask users to type again.

For the request command, the file name contains in the command will be retrieved first. Then a file requests message as “request:xxxx(4-digits file’s name):peerN” will be formed and send to the peer’s successor1.

After receiving the quit command, a quit notice as “quit:peerN:successor1N:successor2N” will be formed and send to the two predecessors of the peer. The message means that peerN will quit and its successors are successor1N and successor2N. If the two predecessors receives the message and successfully update their records, they will send response message back. Then the peer can stop all the running servers and clients.

TCPServer.java

This class is designed to response for the request which is sent by the client. In this assignment, there are three kind of requests to deal with: request files, quit, and forceQuit; one response to handle: has file.

After receiving a request to request a file, the location of this file will be checked based on the rules which explained above. If the file is being confirmed that it is stored in the current peer, the peer will send a response message to peer who requests the file. If the file is being confirmed that it is not stored in the current peer, the TCPServer will forward the request to its successor1 to continue search.

When receiving a “has file” response from other peer, the server knows the location of the file, and the hole file request flow is done.

If the server received a quit request, it will first get the information of the departing peer from the request which id sent by that peer. Then updates its own record of peers. Finally, the server will send a response to the departing peer.

If the server received a forceQuit request, it will send the number of the first successor back.

Further Improvements

The way how to handle exceptions could be improved further. Instead of painted the java exception informations on the user-side interface, I could could paint it on the log and just paint massages to users such as “Sorry, the server cannot be found.”. This will need some plug-ins such as log4.