GetMyMusic

Authors: Tianchang Yang, Wallace He and Nicholas Wan

Description:

GetMyMusic is a file/music synchronization application that enables multiple machines to ensure that they have the same files in their music directory.

Structure:

We put the summary of songs into the data.dat files which contains song names and coreresponding SHAs. One database file corresponds to one music directory and both server and client get its local music information from database file.

Server:

server would wait for the client to respond, in order to handle the multi-Clients, we used Pthread function.

After server recieved the messages from Client, we will parse the message inside of our helper function: handle the client.

Based on the message, there are three different ways to handle the client:LIST, PULL and PUSH.

LIST:

LIST message contains nothing but the message type and length field (which is zero). After received LIST message, server would retrieve all the songs from the database and find song names and SHA's from database and store in listResponse packet to send back to client.

PULL:

PULL message contains type field, length field (2 bytes), and SHA. After received the Pull message, server would get the song name corresponding to SHA, get the actual song file from current directory and create response message which would contain the songs inside and send back to client

PUSH:

PUSH is different than PULL, PUSH message contains type field, length field (2 bytes), song name, and song file. It can see as clients push their song to server and store inside of database of server. The purpose of this design is to check whether or not two files, even if named differently, contain the same content. If the content is the same, the SHA would be same and therefore when file is added into the database,it would be saw as duplicate.

LEAV:

After receive this message, server would cut down the connection between that client but not shut down the server.

Client:

Based on user's input, client would have four part of the "functions": LIST, DIFF, SYNC and LEAVE.

LIST:

In the LIST part, we just send the LIST message (as described above) to the server and receive listResponse from the server, then it would print out every song from listResponse and read another command from user.

DIFF:

In the DIFF part, we will send a LIST message to the server and get all the song name from the server and then compare the songs in client by compareSongsToClient or compareSongsToServer.

SYNC:

In this part, client would send LIST message to server and receive listResponse from server. By using compareSongsToClient or compareSongsToServer, it would upload all songs in client but not in server to the server by sending the PUSH. Then it would pull all the song from server to client.

LEAV:

shut down the connection to server.