Distributed File Sharing

Design Document

Message Passing between Server and Client for Registration

When client enters register <srv ip> <port> then client creates a socket and connects to server. If server accepts the connection the **REGISTER OK** message is sent to client, there by client conforms that registration is successful.

Immediately after sending **REGISTER OK** message server sends list of available peers using **UPDATE <list>** message.

Message Passing between two Clients for connection

A client creates a socket and try to connect to other peer immediately after user enters connect <ip> <port>. If connection is successful the other peer sends **CONNECT OK** message indicating successful connection. In case of failure the other peer sends **CONNECT FAIL <REASON>.**

Message Passing For PUT request

When client requests a to upload file to other connected peer using PUT <Connection ID> <File Name> then client sends **PUT <FILENAME> <SIZE>** and the other sends **PUT OK** end starts preparing to download the file. In case of failure like no disk space or accessing files outside my directory then the other peer returns **PUT FAIL** indicating failure.

Message Passing For GET request

When client requests a to upload file to other connected peer using GET <Connection ID> <File Name> then client sends **GET <FILENAME> <SIZE>** and the other sends **GET OK** end starts preparing to download the file. In case of failure like file does not exist then the other peer returns **GET FAIL** indicating failure.

Message Passing For SYNC request

When a SYNC command is entered on a client, then a message **SYNC** is sent to server. Then the server gives write access to a client by sending **SYNC OK** message and read access to other clients by sending **SYNC READ** message. Upon write completion the client sends **SYNC FIN** to server, which in turn gives write access to other clients until every one is done.

Implementation

Each server and client start listening on a port as soon as they are started. Then eventhandler() function calls specific functions based on user commands, incoming connections and messages from existing clients.

eventhandler() function runs an infinite loop listening to data on STDIN, listen socket and existing sockets using select function.

When there is a user command then commandShell() is invoked, where data from command line is read and parsed for a specific action.

When there is data on listening socket then newConnectionHandler() function is invoked.