Programming Assignment 1:

Develop an Application Level file sharing protocol with support for upload/download and indexed searching.

Specifications

Problem Setting: Two network clients listening for requests waiting to share files.
Approach to be used: Design an application layer protocol like FTP/HTTP to enable this

Goals of the assignment: To enable two clients to:
-Know list of files on each other's machines in designate shared folders -Upload a file to each other
-Download a file to each other
-Periodically check for changes in the shared folder
-Application level error checking using MD5 checksum
-Enable both TCP and UDP based transport of files as per client requests

Specifications:
The file sharing protocol should implement the following features:

1. An "IndexGet" request which can request different styles of the same index of the shared folder on the other client as listed below. The history of requests made by either clients should be maintained at each of the clients respectively.

a. A "ShortList" request indicating that the requesting client wants to know only the names files chosen from the time-stamps specified by the requesting client, i.e., the client only wishes to learn about a few files.

E.g., Sample request: IndexGet<space>ShortList<space>starting-time-stamp<space>ending-time-stamp<CRLF/EOL>

The response should include the "names", "sizes", "last modified" time-stamp and "type" of files (if available)

b. A "LongList" request indicating that the requesting client wants to know the entire listing of the directory of the shared folder including the"names", "sizes", "last modified timestamp" and "type"(if available)

E.g., is similar to above with necessary changes.

c. A "RegEx" request indicating that the requesting client wants to know the list of files that contain the regular expression pattern in their file names. The response should

include all the file names which "contain" the regular expression pattern in their names, the sizes of the files and "type" (if available).

E.g., IndexGet<space>RegEx<space>"*mp3"<CRLF/EOL>

2. A "FileHash" request indicates that the client to enable the client to check if any file's content has changed. Two types of "FileHash" should be supported:

a. A "Verify" request which gives the name of the file that the client wants the hash for. The response should contain the MD5 hash of the file and the name of the file and last modified time stamp.

E.g., FileHash<space>Verify<space>Name-of-file<CRLF/EOL>

b.A "Check All" request which is used to periodically check for modifications in the file. The response should include the hashes of all the files, their names and the last modified time stamp.

E.g., FileHash<space>CheckAll<CRLF/EOL>

c. A "FileDownload" request, which includes the name of the file that the client wants to download. The response should include the File, the file name, the file size, the MD5 hash and the time-stamp when the file was last modified. The "file size" parameter should be used by the requesting client to allocate memory and receive the file in the allocated memory.

E.g., FileDownload<space>Name-of-file<CRLF/EOL>

d. A "FileUpload" request, which includes the name and size of the file that the client wants to upload.The other client can either end a"FileUploadDeny"or "FileUploadAllow" response.The other client should use the file size parameter to allocate memory as done in "FileUpload" request. The client should upload the file,its md5 hash and the time-stamp if it receives a"FileUploadAllow" response. If it gets a "FileUpload Deny" then the client should go back to listening for other requests.

E.g., FileUpload<space>Name-of-file<CRLF/EOL>

f. A "FileUpload" or "FileDownload" requests can be serviced through a UDP or TCP based connection. If such a socket is not available, it should be created and both clients should use that socket to exchange the file.

Important Notes
1. Programming language is C

2. All error scenarios should be handled graceful. If a program crashes during testing, it will be penalized.

3. Programs that don't compile will automatically be given 0.

4. We have given only examples of the "Request" messages. You must design appropriate "Response" messages with proper headers. You can look at the HTTP response header design for this purpose and design the response header for your protocol.

5. You don't have to use threads for this project.
6. Plagiarism in any form will not be tolerated.
7. Project can be done in groups of at most 2, only one submission is required.

Deadline: 28th Feb.