جامعــة خليفــة
Khalifa University

Khalifa University of Science and Technology

Department of Electrical Engineering and Computer Science

ECCE 356 – Computer Networks

Spring 2021

Course Project: Dropbox-Like Folder Synchronization using FTP

Deadline: Thursday, May 1, 2021

Overview:

The goal the project is to extend on the knowledge you gained in your assignments with regards to the basics of network programming within the TCP/IP Internet family environment, and its socket application programming interface (API) and also a little bit of multi-threading. You will use the TCP protocol as a "delivery service", to carry your packets. The particular problem that you have to build a file transfer application with Dropbox-like synchronization property, which allows transferring files between clients and a server.

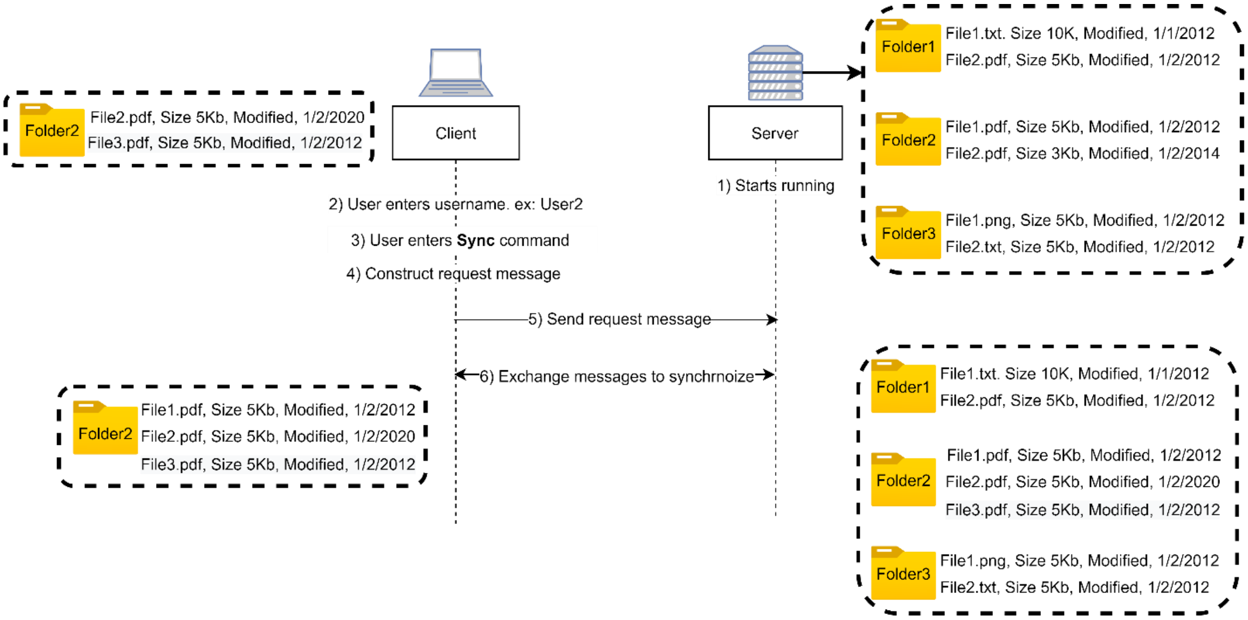
Specifications and requirements

You are required to develop a program that lets users synchronize their local directory with their remote directory on a given server. Following is a list of specifications and requirements that should be considered:

- There are pre-existing folders on the local directory and on the server.
- The local and remote folders can have different files with no subdirectories.
- Synchronization happens based on a user’s manual invocation.
- You should only exchange missing or modified files when synchronizing, don’t use putall and getall.
- The synchronization should be performed both ways to have the same list of files at the local and remote directory.

Scenario:

The following figures shows a high-level view of the expected flow of messages from the client to the server.


Folder2
File2.pdf, Size 5Kb, Modified, 1/2/2020
File3.pdf, Size 5Kb, Modified, 1/2/2012
Client
Server
Folder1
File1.txt. Size 10K, Modified, 1/1/2012
File2.pdf, Size 5Kb, Modified, 1/2/2012
Folder2
File1.pdf, Size 5Kb, Modified, 1/2/2012
File2.pdf, Size 3Kb, Modified, 1/2/2014
Folder3
File1.png, Size 5Kb, Modified, 1/2/2012
File2.txt, Size 5Kb, Modified, 1/2/2012
1) Starts running
2) User enters username. ex: User2
3) User enters **Sync** command
4) Construct request message
5) Send request message
6) Exchange messages to synchrnoize
Folder2
File1.pdf, Size 5Kb, Modified, 1/2/2012
File2.pdf, Size 5Kb, Modified, 1/2/2020
File3.pdf, Size 5Kb, Modified, 1/2/2012
Folder1
File1.txt. Size 10K, Modified, 1/1/2012
File2.pdf, Size 5Kb, Modified, 1/2/2012
Folder2
File1.pdf, Size 5Kb, Modified, 1/2/2012
File2.pdf, Size 5Kb, Modified, 1/2/2020
File3.pdf, Size 5Kb, Modified, 1/2/2012
Folder3
File1.png, Size 5Kb, Modified, 1/2/2012
File2.txt, Size 5Kb, Modified, 1/2/2012

1) **AT THE SERVER:** one partner logs at one computer, and starts one application program to act as the file transfer server (call it server.cpp). At startup, program server displays the following messages:

→ FTP Server starting at host: [ServerHostname]
waiting to be contacted...

2) **AT THE CLIENTS:** the other partner logs into the second computer, and starts the other application program to act as the first client (call it client.cpp). At start up, program client displays the following messages:

→ ftp_tcp starting on host: ClientHostname

Type name of ftp server: ServerHostname

Required Command: **sync**

Type name of folder to synchronize: folder2

Sent request to ServerHostname, waiting...

3) **AT THE SERVER:** the server receives the request, displays the identification of the requestor, the name of the folder to be transferred and the direction of transfer. The server puts out a message stating that the synchronization has started and the folder have been completely synchronized, and returns to the waiting state.

```
ClientHostname requested synchronizing folder xx.
Sending folder to ClientHostname,
Waiting...
```

4) **AT THE CLIENT:** The client puts out a message noting the arrival start of the synchronization process, and another message noting the completion of the synchronization. It then returns to ask for another transfer. Entering an ftp servername of "quit" should cause it to exit.

6) **AT THE SERVER:** You will have to click control-C to kill the server program.

Assumptions:

1. Assume the above-mentioned commands are the only applicable commands, otherwise an error should be shown to the user.
2. The server that clients connect to can change its name.
3. Assume the folders exists on the client and the server with some existing files.
4. There should be input check on the client side aside from the server validation to ensure that the input fields are valid.
5. All running codes should display its status (ex. Client print to screen, "user has been registered to the server")

Supplementary Codes:

To develop the applications required in this assignment, please use the previously developed codes in assignments 1 and 2. You can use the following link to see how dropbox synchronization works: https://www.dropboxforum.com/t5/Dropbox-files-folders/How-does-syncing-actually-work/td-p/401660

Groups

For this assignment (and for the ones to come), a team of two is permitted, and no bonus is given for working alone. Every member must explain his/her contribution to the assignment. Every group will be given 10 mins to demo and 10 mins to answer questions related to the assignment.

Deliverables

1. You are required to implement the file transfer in *both direction*s, from the Client to the Server, and from the Server to the Client. The protocol that you use to establish the transfer must accept both "get" and "put" as directions.
2. You must write your program in C/C++, and you must be prepared to demonstrate your program. Submit your assignment online to BB by the deadline. The demonstration will be performed using the submitted version.
3. You are required to submit a report indicating the features implemented and showing screenshots of the program while its running.

4. At the time of demonstration, you will be asked questions about the functioning of the program; any student of the group must be able to answer any question. Part of the marks will be assigned for demonstrating compliance with requirements at this demonstration. Marks assigned to each member of the group may be different, depending on ability to answer the questions.
5. Penalties will apply for late submission.