Computer Networks Assignment 1

March 4, 2017

Deadline: 11:55 PM, 20th March Allowed languages: C, C++, Python

1 Aim

To create an Application Level program to keep two separate directories synced, similar to Dropbox, using sockets. Program should keep an index of files in the current directory, and support hashes to verify files.

2 Features

- The system should have 2 clients (acting as servers simultaneously) listening to the communication channel for requests and waiting to share files (avoiding collisions) using an application layer protocol.
- Each client has the ability to do the following:
  - Know the files present on each others machines in the designated shared folders.
  - Download files from the other shared folder.
- The system should periodically check for any changes made to the shared folders.
- File transfer should incorporate MD5 checksum to handle file transfer errors.
- The history of requests made by either clients should be logged at each of the clients respectively.

3 Part I (10 marks)

Create a command line interface similar to a shell, to perform certain operations.

3.1 Commands

- `index <flag> [args]...`
  - Requests the display of the shared files on the connected system.
  - The flag variable can be:
    - longlist: Flag would mean that client wants to know the entire listing of the shared directory including 'name', 'size', 'timestamp' and 'type' of the files.
      Ex: `prompt> index longlist`
      Output: Should include 'name', 'size', 'timestamp' and 'type' of all the files in the other directory.
    - shortlist: This flag would mean that the client only wants to know the names of files between a specific set of timestamps. (Any valid format for timestamps)
      Ex: `prompt> index shortlist <starttimestamp> <endtimestamp>`
      Output: Should include 'name', 'size', 'timestamp' and 'type' of the files between the start and end time stamps.
    - regex: This flag means that the output should be the listing of all files which match the regex.
      Ex: `prompt> index regex \.txt$`
      Output: Details of all files that end with .txt should be displayed.
- `hash <flag> [args]...`
  - This command indicates that the client wants to check if any of the files on the other end have been changed. The flag variable can take two values, verify and checkall
    - verify: flag should check for the specific file name provided as command line argument and return its 'checksum' and 'last modified' timestamp.
      Ex: `promt> hash verify <filename>`
      Output: checksum and last modified timestamp of the input file.
    - checkall: flag should check perform what 'verify' does for all the files in the shared folder. The files of shared folders.
      Ex: `prompt> hash checkall`
      Output: filename, checksum and last modified timestamp of all the files in the shared directory.
- `download <flag> [args]...`
  - As the name suggests, would be used to download files from the shared folder of connected user to our shared folder. The flag variable can take the value of TCP or UDP depending on the user's request. If a socket is not available, it should be created and both clients must use this socket for file transfer.
    Ex: `prompt> download TCP <filename>`
    Output: Should display the filename, filesize, last modified timestamp and the MD5 hash of the requested file. Should also download the file and verify it in case of UDP.

4 Part II (10 marks)

Use the functions created for the commands in the above part to create an automatic file sharing service. The program should periodically check for any updates in the remote folder using the hash values and download the remote file, if it is more recent.

5 Bonus (2.5 marks each)

1. The programs should not only sync files, but also directories.
2. The program should preserve file permissions on transfer.

6 Clarifications

1. This assignment will take time to complete. Start early.
2. The languages permitted for this assignment are C, Python, C++.
3. All error scenarios must be gracefully handled (Programs crashing during testing will be penalised).
4. The sending and receiving part of the code can be in different files.
5. The shared directories can be on the same or different computers, as per your wish.
6. Using servers like pyftplib for Python is obviously not allowed. Sockets must be used.
7. Zipping the entire shared directory and sending it is not allowed.
8. Use moodle for general doubts discussion and clarification.
9. There will be manual evaluation of the assignment.
10. Plagiarism in any form shall not be tolerated (MOSS will be used) and a straight F grade for the course will be given.

7 Submission format

Put all your codes in a single folder and compress this to <roll number>.tar and upload it to moodle. Failing to follow the submission format would be penalised.