Communication Networks
File Sharing Protocol

Download, Upload, Indexed searching.

—

By Unni Krishnan R Nair (20171405) and Abhilash Varma (20161155)

Aim :

- To create a file sharing protocol similar to Dropbox with support for download and upload for files and indexed searching.

- The system should have 2 clients (acting as servers simultaneously) listening to the communication channel for requests and waiting to share files

- Each client has the ability to do the following :
    - Know the files present on each others machines in the designated shared folders.
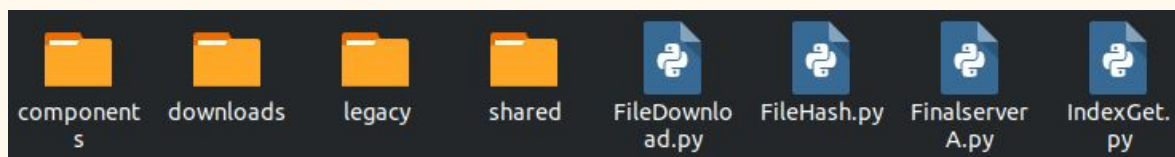   - Download files from this shared folder

Setup for emulating 2 pcs on lan
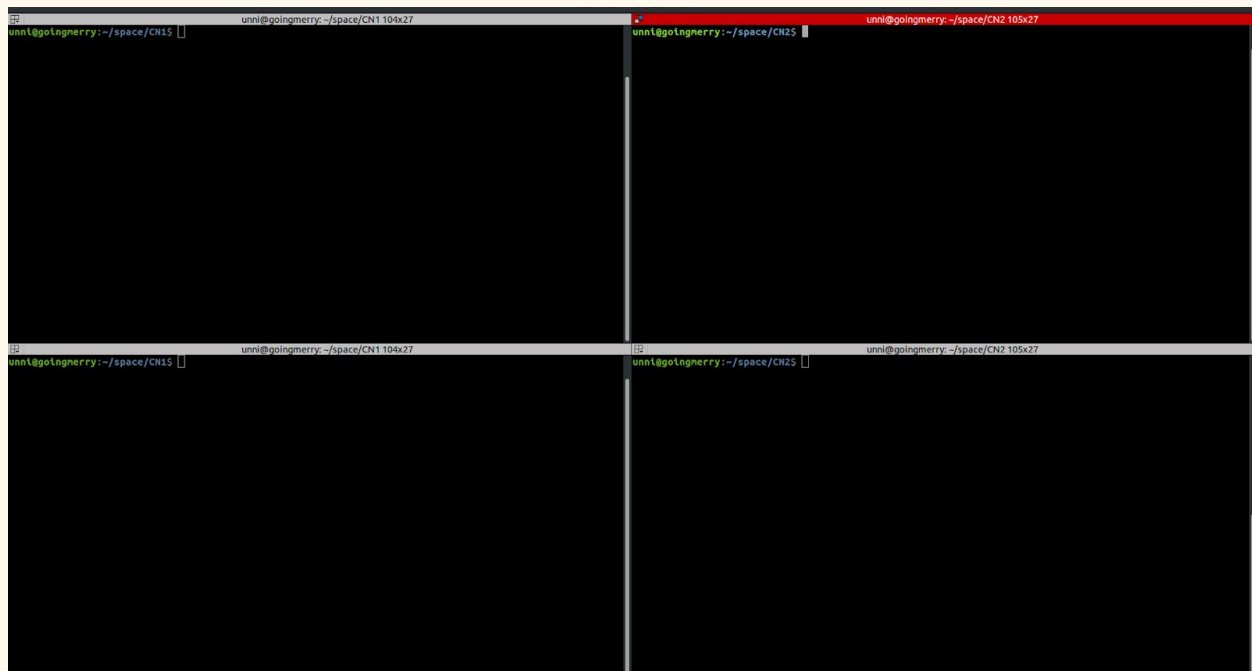
The left 2 panels are from pc1 and the right 2 panels are from pc2.

The idea is to have the always on servers for both pcs to be on the top most panels and execute commands on the bottom ones.

Directory Structure per PC

- components
- downloads
- legacy
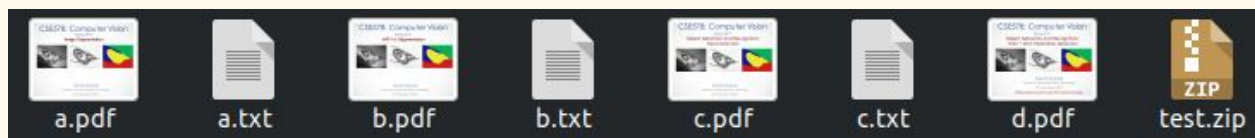- shared
- command executables

Downloads store the downloaded files.

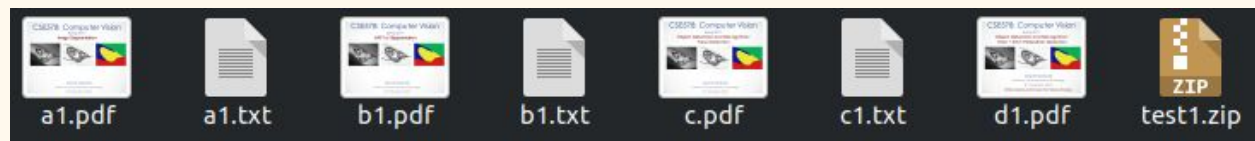Legacy contains some old basic code used to build the final system. Exists for reference.

Shared

Shared store the shared files. The files on pc1 are a.pdf b.pdf a.txt …. The files on pc2 are a1.pdf b1.pdf a1.txt …..

PC1

PC2

Components

Components store hashtables and the history for both the machines.
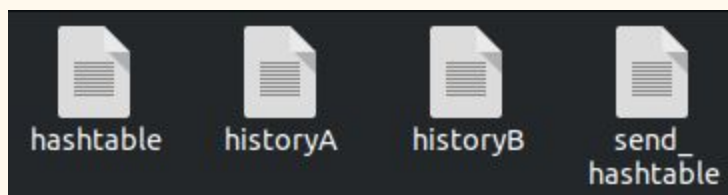
Hashtable = own shared folders files hashtable

send_Hashtable = other shared folders files hashtable

historyA = list of requests made by indexing by A

historyB = list of requests made by indexing by B

Executables

These are used as the commands for downloading indexing and starting of the server.

IndexGet flag <args>

Can request the display of the shared files on the connected system. The history of requests made by either clients is maintained at each of the clients respectively.The flag variable can be shortlist , longlist .

" Shortlist" Flag would mean the the client only wants to know the names of files between a specific set of timestamps. The sample query:

- $> IndexGet shortlist <starttimestamp><endtimestamp>
- Output includes 'name' , 'size' ,'timestamp' and 'type' of the files between the start and end timestamps.
- Return only *.txt , *.pdf files between specified time stamps with this type of

  query $> IndexGet shortlist <starttimestamp><endtimestamp> *.txt or *.pdf

Note: Timestamp should be in epoch a helper script is given to convert time in date time to epochs.

" Longlist" flag would mean that client wants to know the entire listing of the shared folder/directory including 'name', 'size' , 'timestamp' and 'type' of the files .

- $>IndexGet longlist
- Output : similar to above , but with complete listing.
- Return longlist for only *.txt file contained a word "Programmer" in it.

FileHash 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.
    - $>FileHash verify <filename>
    - Output : checksum and lastmodified timestamp of the input file.

Checkall : flag should check perform what 'verify' does for all the files in the shared folder.
(Hint: this command can be used for the periodic check of changes in the files of shared folders)

- ○ $> FileHash checkall
- ○ Output : filename , checksum and last modified timestamp of all the files in the shared directory.

FileDownload flag <args> :

This would be used to download files from the shared folder of connected user to our shared folder.

- The flag variable can take the value TCP or UDP depending on the users request.
- If a socket is not available , it should be created and both clients must use this socket for the file transfer.
  - ○ $>FileDownload <filename>
  - ○ Output : contains the filename , filesize , lastmodified timestamp and the MD5hash of the requested file.

The protocol:

The protocol designed depends on the natural integrity of small file transfers and message transfers via tcp. All hashtables are maintained at each pc for the shared folders and we simply send these hashtables along with the files to check for file integrity. Also the same hashtables can be used for the hashcheck function and the indexing functions required to be coded.

The server

There is a server on each pc listening on the following ports

server b

- 60001 tcp
- 60002 udp

server a

- 60003 tcp
- 60004 udp

Default transfer method for all files and messages is TCP.

The server understands the following messages:

Send_hashtable, filename UDP, filename TCP, historyX

(where X is the history of the indexing command)

```python
if command == 'send_hashtable':
        generatehash()
        sendfile('./components','hashtable',conn)
elif command.split(' ')[0].strip() == 'UDP':
        sendfile_udp('./shared',command.split(' ')[1].strip())
elif command.split(' ')[0].strip() == 'TCP':
        sendfile('./shared',command.split(' ')[1].strip(),conn)
elif command == 'historyB':
        sendfile('./components','historyB',conn)
else:
        sendfile('./shared',command,conn)
conn.close()
```

- send _hashtable sends a hashtable to the client of all files in the shared folder.
- filename UDP sends file via UDP.
- filename TCP sends file via TCP.
- historyB sends all the commands used by the indexing command on server B.
- We use md5 hashing for all the hashes in this assignment.
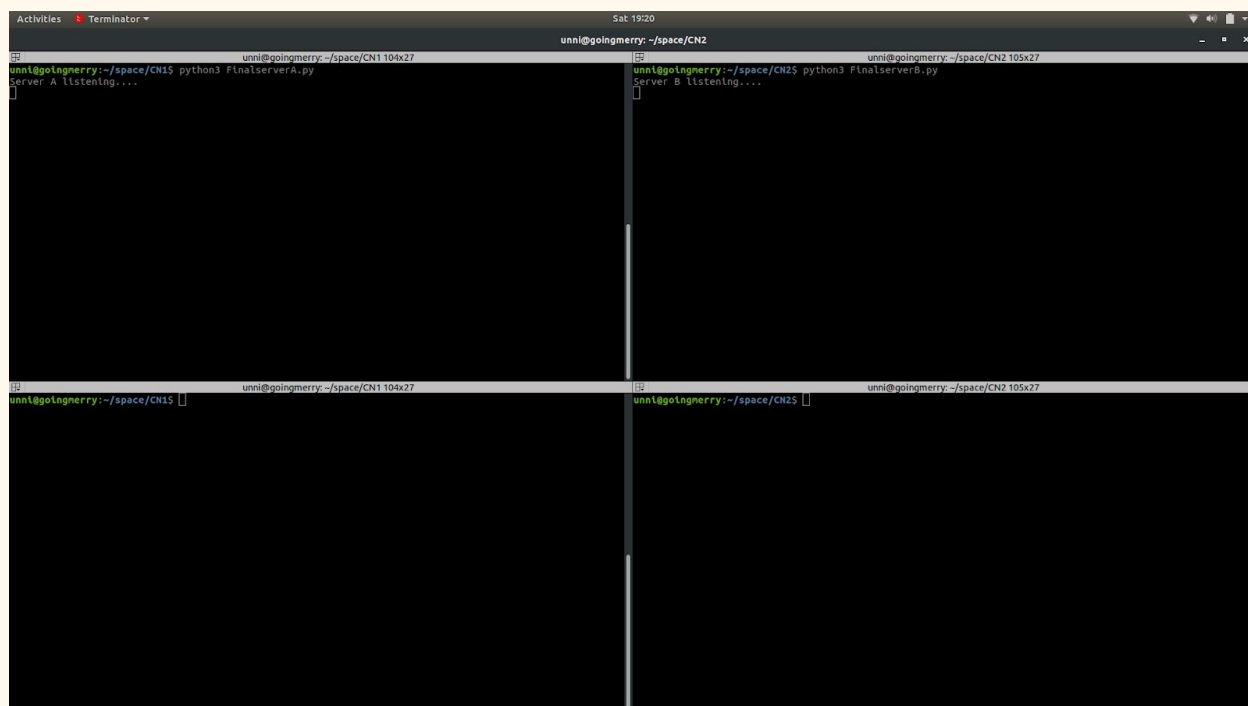
The client:

Can get the hash tables from the server by sending the required messages (command) and then simply parse the hashtable for indexing and the file integrity checks.

The file transfer methods available are TCP and UDP they use the defined ports above for file transfers.
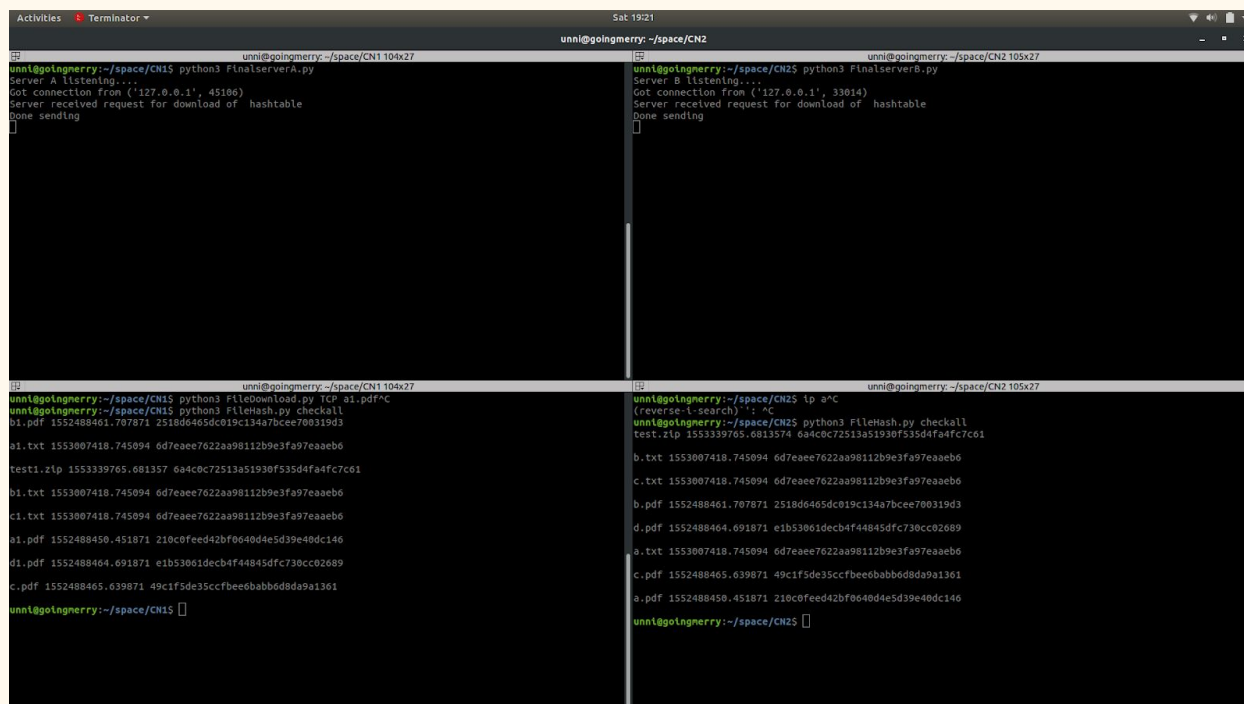
The Results

Starting the severs

Updating hashtables via the Filehash checkall command.

Downloading the file a1.pdf from serverB and a.pdf from serverA

Verification of the file hashes via the FileHash verify filename command

Indexing with shortlist, *.pdf

Indexing with longlist, *.pdf

Downloading the file a1.pdf from serverB via UDP and it failing integrity check

Downloading the file a1.pdf from serverB via UDP and it being received successfully

Note: Removed the shared and downloaded files due to size constraints in moodle.