The Development Report of Large Efficient Flexible and Trusty Files Sharing

Fengrui Cheng
1822755

Abstract

This report consists of six parts, namely Introduction, methodology, Implementation, Testing and results, Conclusion and Reference. The report begins with an introduction to project requirements, background, and analysis of Dropbox. The protocol, functions, and ideas used by the project, such as classifying the program structure into three types of Listeners, Scanner, will also be discussed. In the implementation section, some mind maps and flowcharts will be used to describe the program logic. The tests and results section will demonstrate the correctness and efficiency of the program. In the end, the whole project will be summarized and a false outlook for the future will be presented.

Contents

1 Introduction

This project needs to design a file sharing software to meet the following four requirements, large file transfer, high efficiency, flexibility, Trusty, through Python socket programming. With the development of digital technology and the increasing demand of users for fast file transfer, the development of file sharing software has been promoted[1]. In terms of file sharing, users have almost unlimited options, such as Dropbox, Box, Google Drive, Microsoft oneDrive, and Hightail, which allow users to easily share large files in the cloud and synchronize them across multiple devices[2]. Dropbox is a cloud-based productivity platform designed to help teams and individuals improve efficiency and productivity with offering a variety of benefits, reliable off-site servers, and allowing users to automatically synchronize online, even across all devices in use[3]. It also has no restrictions on the number of files to share or the users who will be granted access to the shared files[3]. By leveraging the power of cloud technology, Dropbox makes data access, storage and retrieval easier and more efficient; It is also OS independent, which means it can run on any system, be it PC, Mac, Linux, iOS or Android, further improving productivity[2]. Although Dropbox is excellent at meeting customer needs in many ways, it still has a few drawbacks to improve on. One of the drawbacks is limited search capabilities, which users with a large file database in the Dropbox cloud cannot afford. Unfortunately, Dropbox can't provide users with the latest search technology. The reason is a lack of search metadata which is information about a file, such as the date it was created[4]. In addition, Dropbox does not provide users with the opportunity to browse uploaded files because there is no basic need for searching[4]. This report will discuss how file sharing programs are implemented according to the LEFT requirements. The program is divided into four basic classes and a main class, which are the Scanner, Listener, DownLoader, and Function classes. The Scanner class is responsible for monitoring the "Share" folder to ensure that new files or files are updated to inform Peers. The Listener class listens for a particular port, and if a connection is made, creates a new thread to handle it. The DownLoader class is responsible for viewing the download list and selecting the earliest item from it, sending a download request to Peers. Function class is basic functions such as get MD5 value, compression, encryption and so on. The Main method starts the program. The file transfer is based on TCP sockets and error verification mechanism is added to the application layer to ensure smooth file transfer. By increasing the buffer size, compression, multithreading to improve the file transfer speed. The crypyto.cipher module is used to achieve secure file encryption transmission; the zlib module is used to compress large files.

2 Methodology

2.1 Protocol

The protocol uses “stateless” logic for the recipient to actively request a file, similar to HTTP.

2.1.1 For sender

The sender waits to receive a file name, finds the file size through os.path.getsize (file_name), and packs the file size, block size, and compressed flag to send to the receiver. The sender receives the block index (start location) of the file and sends binary data block by block from the block index.

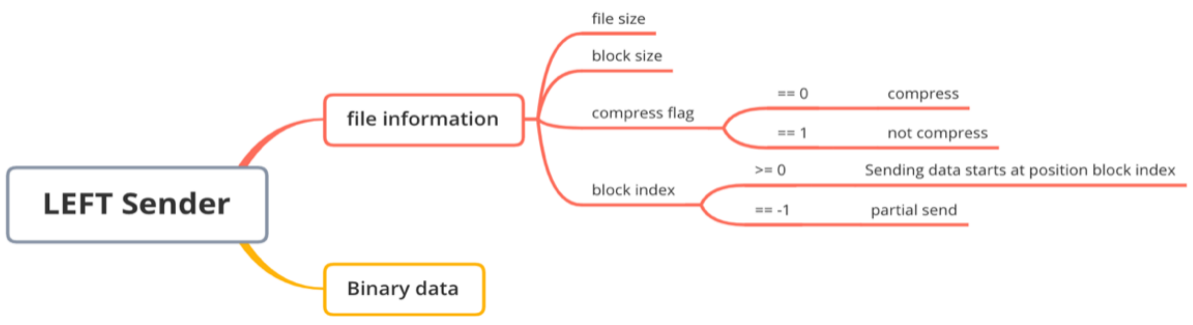


Figure 1: LEFT sender

2.1.2 For receiver

The receiver first fetches an item from the Download List containing the file name, the IP of the file owner, download Code. If download code == 0, it is a request to download a new file. If download code == 1, it is part of the update file, and only the updated file is required. If download code == 2, it is the download folder. The receiver actively requests the file to the sender via IP. The sender sends file information back to the requester, including file size, block size, and compressed identifier. The requester then sends the block index (starting position) to the sender. The sender sends the data stream file back.

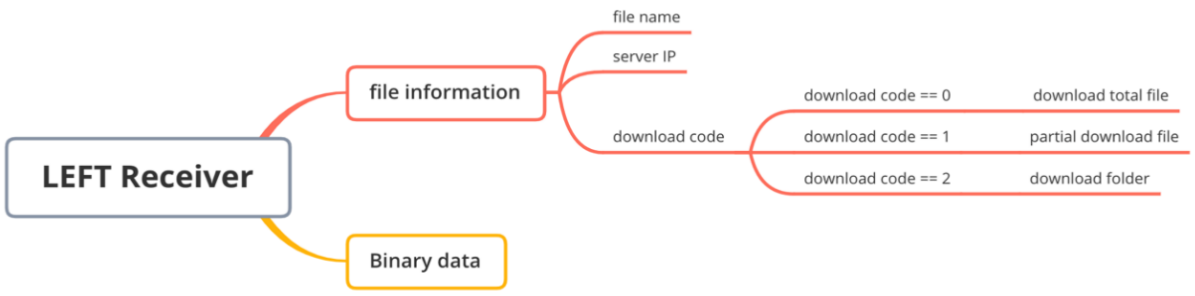


Figure 2: LEFT Receiver Xmind

2.2 Functions and Ideas

2.2.1 Scanner class

The scanner class has two main functions. Say Hello is used to notify Peers when the virtual machine starts, send the "Share" folder to Peers, and receive new file list sent by Peers at the same time. If new file is found, download.list.append(new file). The second major function is to loop scan the "Share" folder and send the file name and IP address to Peers if a new file is found.

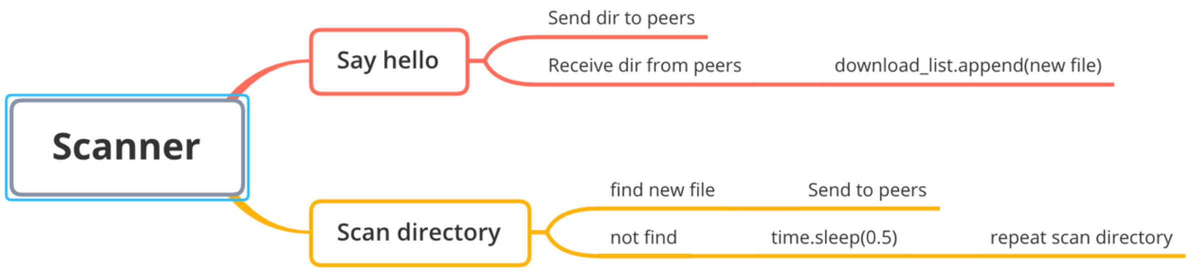


Figure 3: Scanner class Xmind

2.2.2 Listener class

The Listener class is mainly responsible for receiving requests from peers and creating a new thread to process the requests. There are three types of “operate code” that exists.

1. operate code == 0 is to receive a file sent by Peers. 1) If the file does not exist in the local "Share" folder, file information will be inserted into the download list. 2) If a file exists in the local "Share" folder but was last modified earlier than it was received, the file is updated, and the file information is added to the download list indicating that it was partially updated. 3) If a file exists in the local "share" folder and was last modified later than when the file was received, the local file is up to date across the web and does not need to be updated.

2. operate code == 1 is to send the specified file to Peers. When the thread receives a file name sent by Peers, it first sends the file information to Peers, including file size, block size and compress flag. Then the thread receives a block index, start position, from Peers. The thread reads the binary file from block index and sends it to Peers

3. operate code == 2 is to say hello to peers. Threads can monitor new Peers online. The thread will receive the file list sent by Peers. If it finds a new file, it will join the download list. At the same time, the thread will send the file list in the local “share” folder to Peers.

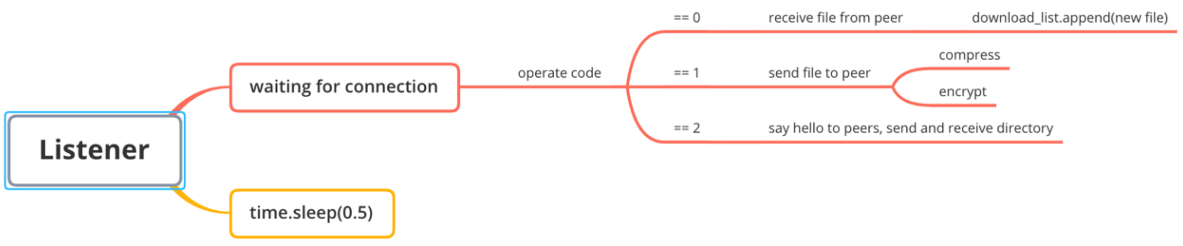


Figure 4: Listener Xmind

2.2.3 Downloader Class

The download class has four main functions. get the download list, receive file information and download file, decompress, decrypt. First, the process gets an item from the Download List containing the file name, IP, download code. The process requests a file, by file name and IP. The process receives file information first from Peers and determine whether it is compressed by judging compress Flag. compress flag == 0 will compress. Compress flag == 1 will not be compressed. Before downloading, the process will determine whether to partially update or download a new file by judging the download code. If download code == 0, then use the download function. If download code == 1, then use the partial download function.

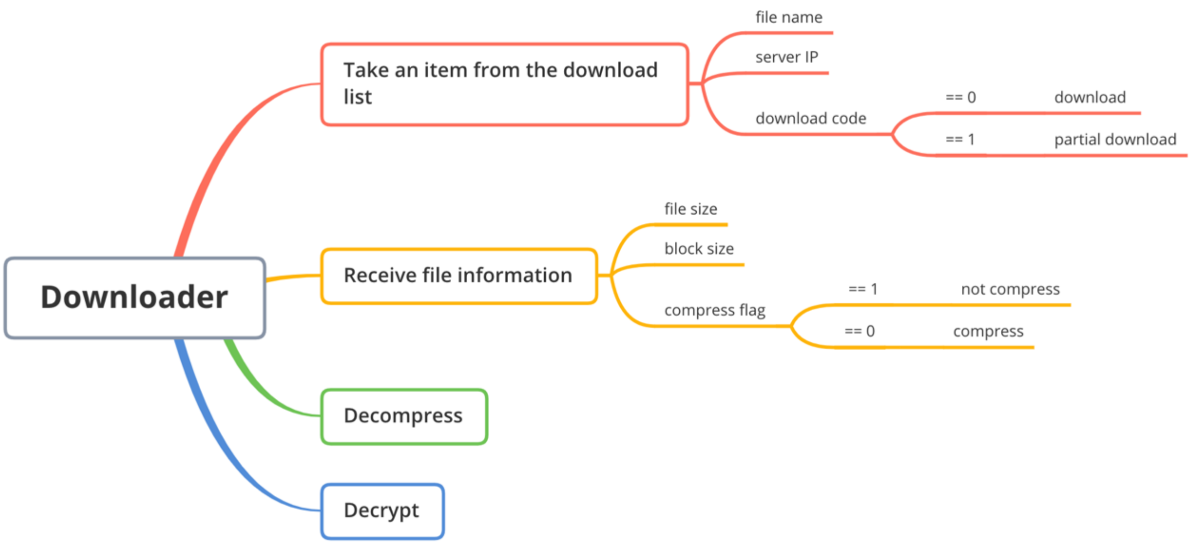


Figure 5: Downloader Xmind

3 Implementation

steps of implementation, program flow charts, programming skills (OOP, Parallel...) you used, what difficulties you met and how to solve them.

3.1 steps of implementation

1. The first step is to read the spec carefully and discover what the program needs to do

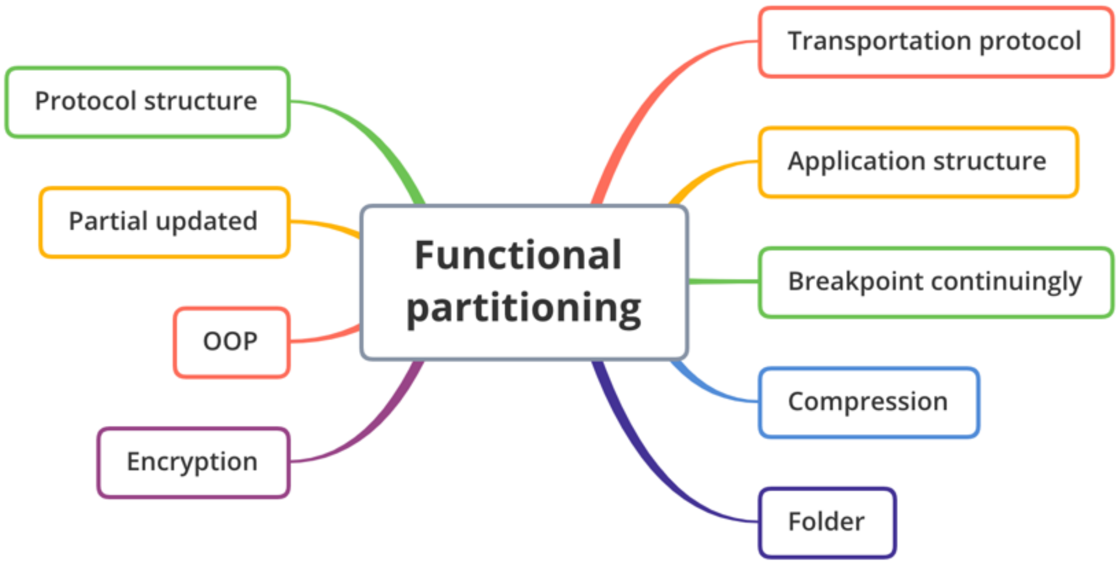


Figure 6: Functional partitioning Xmind

2. According to the first step of function division, look up information on the Internet, find the solution

   (a) Transportation protocol: this file sharing software needs to ensure the correctness of transmission while transferring large files. Meanwhile, in the LAN environment, the transmission rate of TCP is not much different from that of UDP. Therefore, the Transport Layer protocol uses TCP.

   (b) Protocol structure: the application layer protocol takes the form of table header and data, in which the table header contains file name and data segment length

   (c) Application structure: The program structure is divided into three categories: Listener, Scanner and Downloader. The Listener listens for Peers requests, the Scanner scans the "Share" folder, and the Downloader downloads and decompress files. The program structure is discussed in detail in the next section.

(d) Object Oriented Programming: There are three classes, Listener class, Scanner class and Downloader class. The Scanner class has the attributes IP, IP1, Listener_port, file_header_dic, download_list. file_header_dic is to determine if the file has been updated. The Listener class has the attributes listener port, file_header_dic, download_list, isEncrypt. isEncrypt is to determine if the file need be encrypted. The Downloader class has the attributes listener port, download list, isEncrypt.

(e) Encryption: Use the Crypto module and set the key= '0523698741254512 '. It is worth noting that the binary data must be a multiple of 32 in length, so if the data is not a multiple of 32, you need to loop in '\0' until the length is a multiple of 32.

(f) Folder: Using the TarFile module, folders can be compressed and sent directly to Peers.

(g) Compression: zlib is used for file compression.

(h) Partial updated: The Scanner class monitors md5 to see if the file is up to date. If it is not up to date, download the update section and override the write with "rb +".

(i) Breakpoint continuingly: Write the file in the download_list to a local JSON file. When the program is restarted, the program first reads the information in the JSON file and puts it in the download_list. When the specified file is downloaded, the program will first determine whether the file exists in the share file directory, and if so, determine how many more files need to be downloaded according to the file size, and send block_index (start position) to Peers.

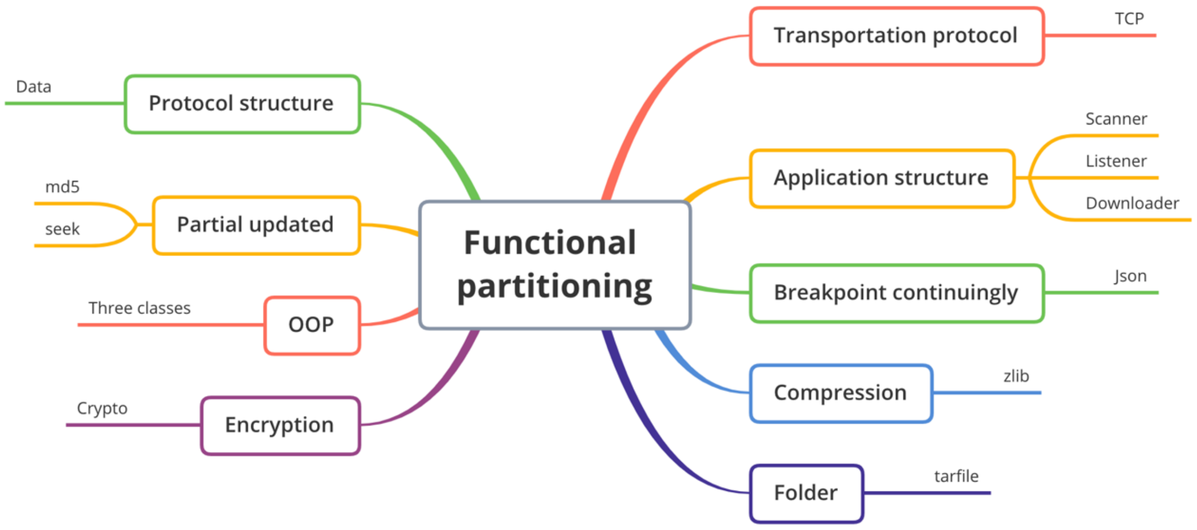


Figure 7: Implement Xmind

3.2 Program flow charts

3.2.1 Scanner class

The scanner class first Say Hello to notify Peers when the virtual machine starts, send the "Share" folder to Peers, and receive new file list sent by Peers at the same time. If new file is found, download.list.append(new file). The second part is to loop scan the "Share" folder and send the file name and IP address to Peers if a new file is found.

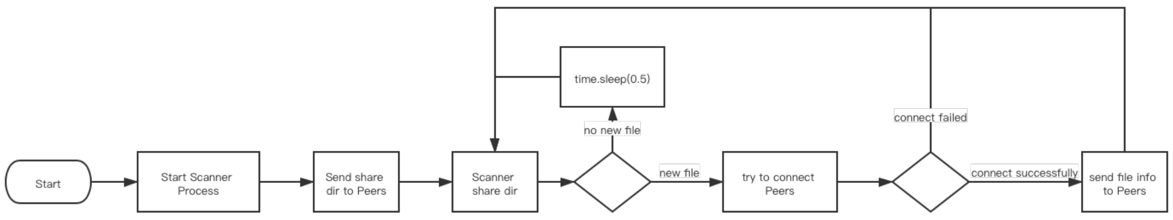


Figure 8: Scanner flow chart

3.2.2 Downloader class

First, process will read download_list.json file If there are files that have not been downloaded before. Second, the process gets an item from the Download List containing the file name, IP, download code. The process requests a file, by file name and IP. The process receives file information first from Peers and determine whether it is compressed by judging compress Flag. compress flag == 0 will compress. Compress flag == 1 will not be compressed. Before downloading, the process will determine whether to partially update or download a new file by judging the download code. If download code == 0, then use the download function. If download code == 1, then use the partial download function.

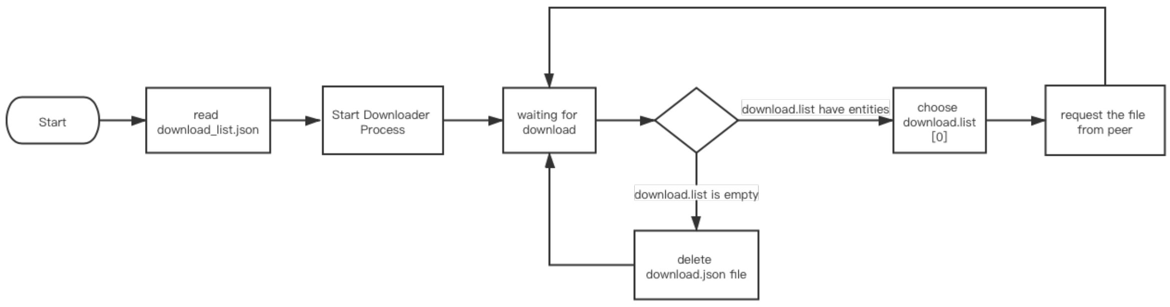


Figure 9: Downloader flow chart

3.2.3 Listener class

The Listener class is waiting for connection from peers. If find new connection, process will start a sub connection.

- operate code == 0 is to receive a file sent by Peers. 1) If the file does not exist in the local "Share" folder, file information will be inserted into the download list. 2) If a file exists in the local "Share" folder but was last modified earlier than it was received, the file is updated, and the file information is added to the download list indicating that it was partially updated. 3) If a file exists in the local "share" folder and was last modified later than when the file was received, the local file is up to date across the web and does not need to be updated.

- operate code == 1 is to send the specified file to Peers. When the thread receives a file name sent by Peers, it first sends the file information to Peers, including file size, block size and compress flag. Then the thread receives a block index, start position, from Peers. The thread reads the binary file from block index and sends it to Peers

- operate code == 2 is to say hello to peers. Threads can monitor new Peers online. The thread will receive the file list sent by Peers. If it finds a new file, it will join the download list. At the same time, the thread will send the file list in the local "share" folder to Peers.

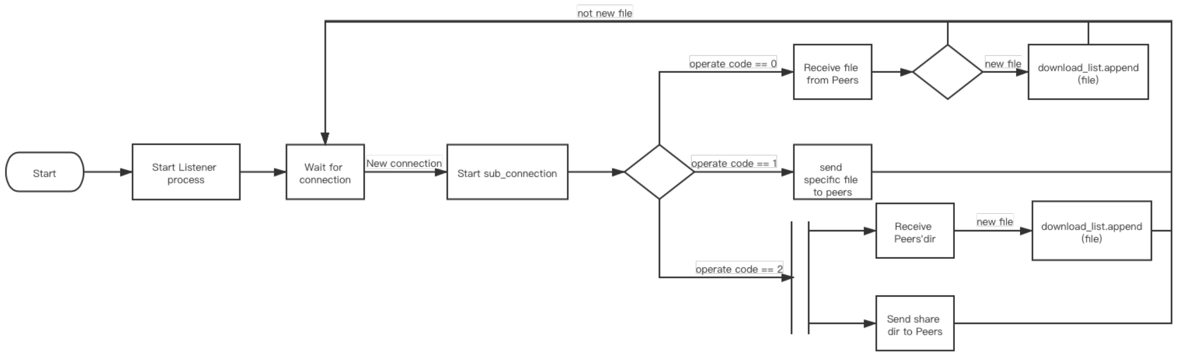


Figure 10: Listener flow chart

3.3 programming skills

- Object Oriented Programming: Listener class, Downloader class, Scanner class.
- Multithreading: multithreaded parallel download, improve the transmission rate.
- Json: Json is used to convert dictionaries, lists formatted into strings, and then into binaries for network transport.

3.4 Difficulties

TCP stick package problem: TCP sticky packets are caused by two reasons:

1. the sender needs to wait until the buffer is full before sending out;
2. the receiver does not receive packets from the buffer in a timely manner, resulting in multiple packet receptions.

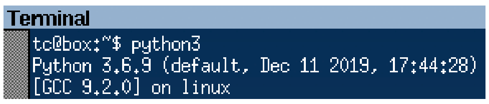
Initial Solution: the initial solution is to separate the data by encapsulating a header for the data stream. The header is sent first for each transmission, including block index and data length, and then the receiving buffer is set to the data length.

Final Solution: on the basis of encapsulating a header for the data, it is judged that the size of the data received is equal to the size of block size each time it is received. If not, the program continues to receive until a block size is filled

4 Testing and results

4.1 Testing environment

The operating system being tested is VirtualBox – Tiny Core Linux. The Python version used is 3.6.9



```
tc@box:~$ python3
Python 3.6.9 (default, Dec 11 2019, 17:44:28)
[GCC 9.2.0] on linux
```

Figure 11: terminal screenshot

4.2 Testing plan

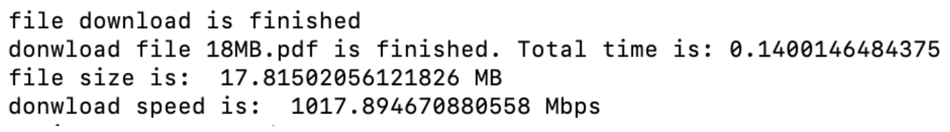
Using the modified remote run program will file to the one-time to three sets of virtual machines, virtual machine via SSH remote connection again, by typing python3 main.py - ip 192.168.56.xxx,192.168.56.xxx to start the program. Test data can be remotely transmitted to the virtual machine workplace/ CW1 /test folder using the SCP directive. Files can be soft-wired from Workplace/CW/Test to Workplace /cw1/share via ln -s.

4.3 Testing results

- download rate(Mbps) = file size(Byte) * 8 / time / 1024 / 1024 4.3.2Small file(18MB): Download;

- Small file(18MB)

  Downloading a 17.8MB pdf file takes 0.14 seconds and has a download rate of 1017.8 Mbps.

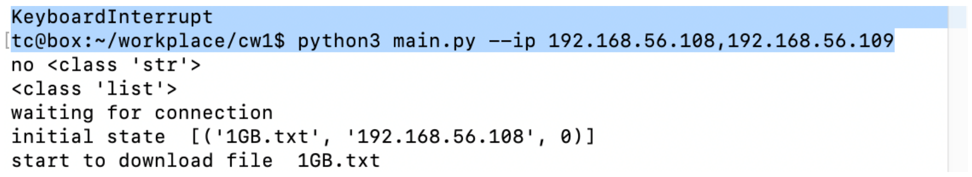


```
file download is finished
donwload file 18MB.pdf is finished. Total time is: 0.1400146484375
file size is:  17.81502056121826 MB
donwload speed is:  1017.894670880558 Mbps
```

Figure 12: 17.8MB.pdf screenshot

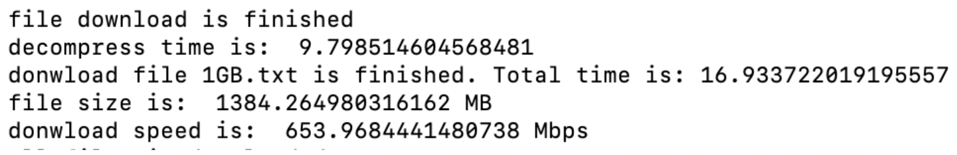
- Large file(1GB) with **breakpoint continuingly**

  Disconnect the virtual machine before reconnecting. Downloading a 1.38GB txt file takes 16.9 seconds (including decompressing time) and has a download rate of 653.9 Mbps.



```
KeyboardInterrupt
tc@box:~/workplace/cw1$ python3 main.py --ip 192.168.56.108,192.168.56.109
no <class 'str'>
<class 'list'>
waiting for connection
initial state  [('1GB.txt', '192.168.56.108', 0)]
start to download file  1GB.txt
```

Figure 13: breakpoint continuingly screenshot

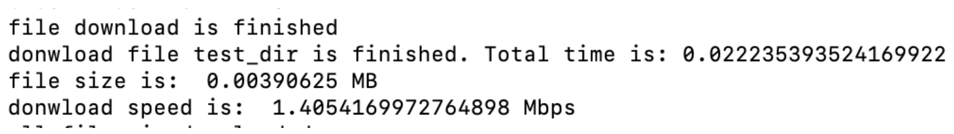


```
file download is finished
decompress time is:  9.798514604568481
donwload file 1GB.txt is finished. Total time is: 16.933722019195557
file size is:  1384.264980316162 MB
donwload speed is:  653.9684441480738 Mbps
```

Figure 14: 1GB.txt screenshot

- Folder with 50 small files

  Because the file is very small, the download time is very fast, but the file download rate is slow. Downloading a 0.0039MB folder takes 0.02 seconds (including decompressing time) and has a download rate of 1.4 Mbps.

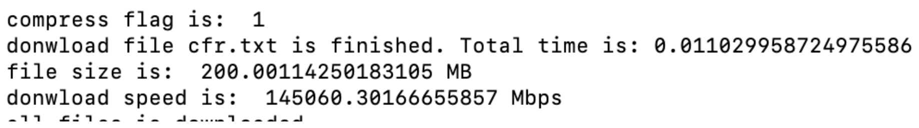


```
file download is finished
donwload file test_dir is finished. Total time is: 0.022235393524169922
file size is:  0.00390625 MB
donwload speed is:  1.4054169972764898 Mbps
```

Figure 15: folder screenshot

- File(200MB)with Partial undated

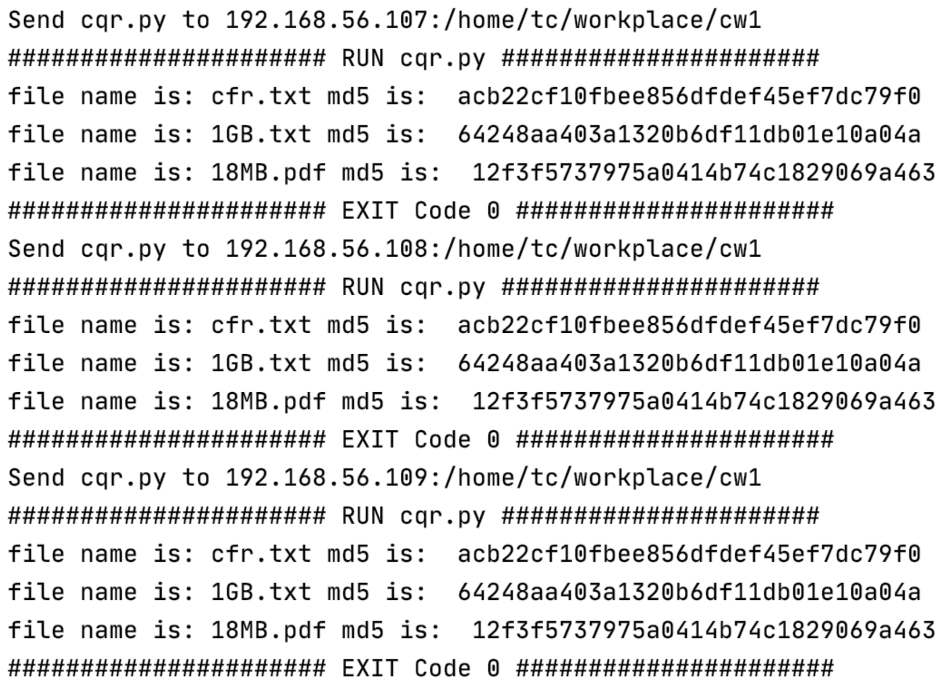
  Because only the first 0.1 percent of the file was updated, only 300KB files were retransmitted, so the download rate was very fast. The download rate in the figure is not true. The real download rate should be 300KB * 8/1024/0.01 = 234.3 Mbps



```
compress flag is:  1
donwload file cfr.txt is finished. Total time is: 0.011029958724975586
file size is:  200.00114250183105 MB
donwload speed is:  145060.30166655857 Mbps
```

Figure 16: 200MB cfr.txt screenshot

- The author's own automated script is used to monitor md5 for files in the Share file.



```
Send cqr.py to 192.168.56.107:/home/tc/workplace/cw1
##################### RUN cqr.py #####################
file name is: cfr.txt md5 is:  acb22cf10fbee856dfdef45ef7dc79f0
file name is: 1GB.txt md5 is:  64248aa403a1320b6df11db01e10a04a
file name is: 18MB.pdf md5 is:  12f3f5737975a0414b74c1829069a463
###################### EXIT Code 0 ######################
Send cqr.py to 192.168.56.108:/home/tc/workplace/cw1
##################### RUN cqr.py #####################
file name is: cfr.txt md5 is:  acb22cf10fbee856dfdef45ef7dc79f0
file name is: 1GB.txt md5 is:  64248aa403a1320b6df11db01e10a04a
file name is: 18MB.pdf md5 is:  12f3f5737975a0414b74c1829069a463
###################### EXIT Code 0 ######################
Send cqr.py to 192.168.56.109:/home/tc/workplace/cw1
##################### RUN cqr.py #####################
file name is: cfr.txt md5 is:  acb22cf10fbee856dfdef45ef7dc79f0
file name is: 1GB.txt md5 is:  64248aa403a1320b6df11db01e10a04a
file name is: 18MB.pdf md5 is:  12f3f5737975a0414b74c1829069a463
###################### EXIT Code 0 ######################
```

Figure 17: check md5 screenshot

5 Conclusion

This project realizes the file sharing program based on Python TCP socket, which can realize the functions of synchronizing files, breakpoint continuation, encryption, compression, partial update and so on. With the development of digital technology, people's demand for paperless office is increasing. Therefore, this project is designed to realize the rapid synchronization of large files within the LAN. Although this program can only implement three client synchronized files, it can be further optimized in the future to achieve true multi-party synchronized files

References

[1] M. Bergsma, "File Sharing Software Market Research Report Information -by Type (Client Server, Peer to Peer), Deployment Mode (Cloud), End user (Enterprises and Individual), Vertical (Government, Healthcare, Media and Entertainment), and Region - Global Forecast to 2023," ed, 2019, p. 101.

[2] M. James, "Top 10 file-sharing options: Dropbox, Box, Google Drive, OneDrive and more," ed: Computerworld, 2019.

[3] S. Seymour. "What is File Sharing Software? Analysis of Features, Types and Pricing." https://financesonline.com/what-is-file-sharing-software-analysis-of-features-types-and-pricing/ (accessed Dec.17, 2020).

[4] C. Schmidt, "What Is Dropbox? - 5 Intriguing Pros and Cons," ed, 2019.