Xi'an Jiaotong-Liverpool University
西交利物浦大学

# Course Work Submission

| | | |
|---|---|---|
| Name | Shen | Wang |
| ID Number | 1824260 | |
| Programme | Information and Computing Science | |
| Module Title | Introduction to Networking | |
| Module Code | CAN201 | |
| Assignment Title | Coursework 1: Large Efficient Flexible and Trusty (LEFT) Files Sharing | |
| Submission Deadline | Monday, 21 December 2020, 00:01 | |
| Lecturer Responsible | Fei Cheng | |

I certify that:

- I have read and understood the University's definitions of COLLUSION and PLAGIARISM (available in the Student Handbook of Xi'an Jiaotong-Liverpool University).

With reference to these definitions, I certify that:

- I have not colluded with any other student in the preparation and production of this work;
- this document has been written solely by me and in my own words except where I have clearly indicated and acknowledged that I have quoted or used figures from published or unpublished sources (including the web);
- where appropriate, I have provided an honest statement of the contributions made to my work by other people including technical and other support staff.

I understand that unauthorized collusion and the incorporation of material from other works without acknowledgement (plagiarism) are serious disciplinary offences.

Signature: Shen Wang Date: 2020/12/22

| For Academic Office use: | Date Received | Days Late | Penalty |
|---|---|---|---|
| | | | |

# An Introduction to CAN201 Coursework 1: Large Efficient Flexible and Trusty (LEFT) Files Sharing

## Abstract

File synchronization is one of the most widely used methods to share files and data. There are plenty of popular file syncing applications, such as Dropbox, Baidu Cloud and iCloud, which are used every day. This report is about the Python application of CAN201 coursework 1: large efficient flexible and trusty file sharing. The report will introduce the methodology and implementation to achieve file synchronization. Furthermore, some tests and the result will be provided to help evaluate this synchronization more visually.

## Contents

# 1. Introduction

The aim of this coursework is to create a Python application that implements large files synchronization efficiently flexibly and trustily. The files to be synchronized is requested to be in any format, which includes hidden files and hidden folders. Simultaneously, the maximum file size can be up to 1GB. When a file is changed, the file shared to another position should be changed as well. In addition, breakpoint resume method contributes to the efficiency. Ultimately, the file transmission should be security without any error.

The running background of this application is limited. Codes should be in Python 3 and run in the provided Linux virtual machine. The application will run on three virtual machines with the same configuration to test the file synchronization.

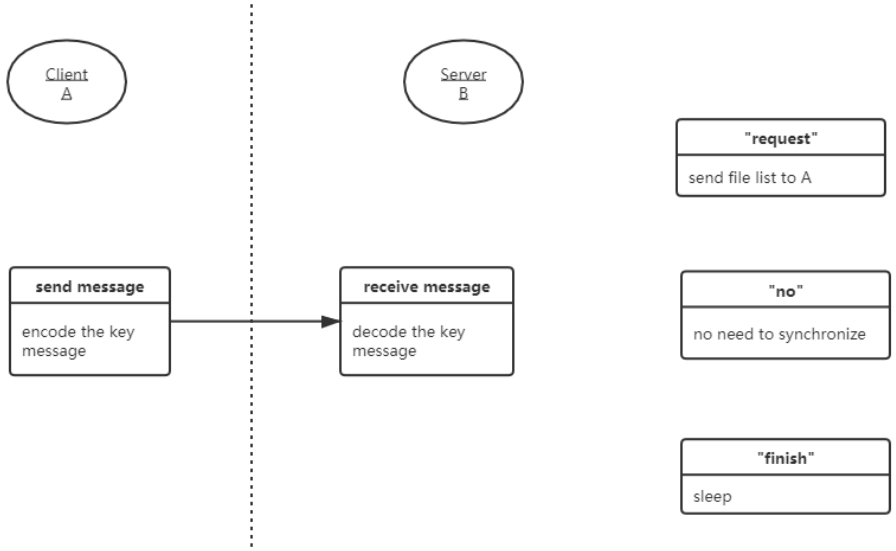
It is useful to analyse an existing case to create a file sharing application. As an example, Dropbox is a practical and popular file syncing software. It is easy to know that Dropbox has little free space and few functions. However, now there are more than 100 million users using Dropbox. That is because Dropbox has distinct advantages. Firstly, the synchronization speed of Dropbox is tremendous, for example, a 1 GB file with Dropbox's client can be uploaded to its network hard disk instantly. Ivanov [1] thought that BBRv2 congestion control used by Dropbox make a contribution to this high speed. Secondly, when modifying parts of a large file, Dropbox can find the changed parts and follow the changes on the server side, which make synchronization faster. Thirdly, Dropbox is cross-platform, it can work on Windows, MAC, Linux or other mobile phone operating systems [2], which make it comparatively user friendly.

For this coursework, I create a Python application using multithreading and a TCP based transport protocol. This application can synchronize files through three virtual machines. If a file is changed in one virtual machine, others will also be changed.

# 2. Methodology

I use the Client / Server architecture and socket based on TCP transport protocol to realize the file and data communication between virtual machine A and virtual machine B. In fact, there is no differences between a client and a server. Both of them means a virtual machine. How these clients and servers work will be introduced detailedly in part 3 implementation.

Here is a diagram to help understand part of my code structure:




The client can send key message, and the server will do different things according to the received key message. The client also sends the file request, then the received server sends file size to help client prepare to download file. After that, the server and client begin uploading and downloading. The file transmission part can be seen in the diagram at part 3 implementation.

It is worth mentioning that I write a function with argparse to read parameters input on the virtual machine terminal. Moreover, I build a recursive get_file function. Input the file path, the function will traverse the files and folders and store files in a list.

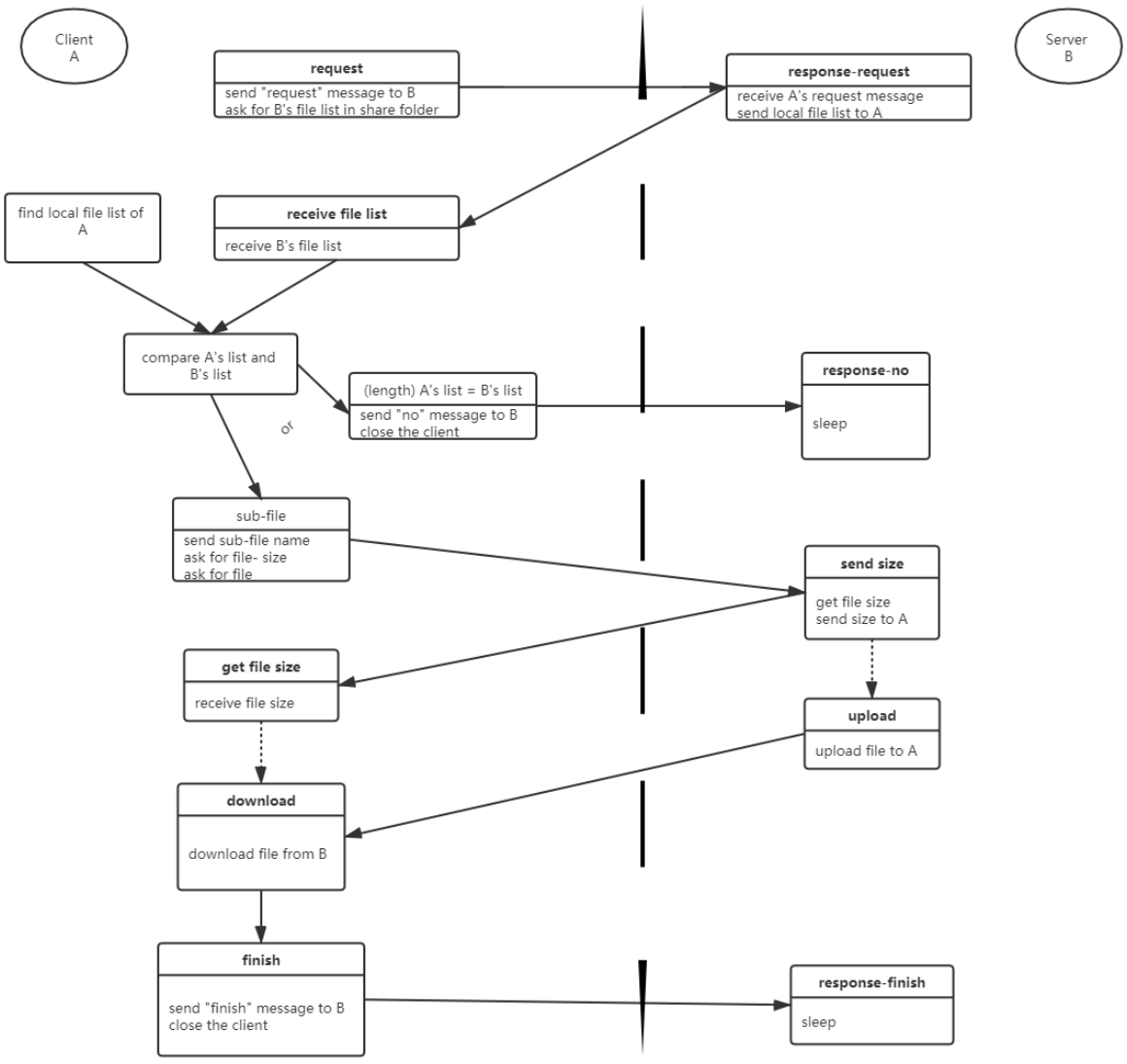
# 3. Implementation

As mentioned above, there are no differences between a “client” and a “server”. The client and server in the same host are run in different threads. However, they can be seen as two objects. When the application start, the server will run to listen to other clients. If a client connection is accepted, the socket and ip of the client will be stored in a list. After a client is connected, this information will be removed from the list to prevent duplication with the next client socket. It will start communication in a thread.

As for the client, it will connect other two servers with two threads: thread B and thread C. If thread B is not alive, thread C will start; if thread C is not alive, thread B will start. This allows

client to connect to two servers alternatively.

Here is a diagram to show how a client and a server communicate:



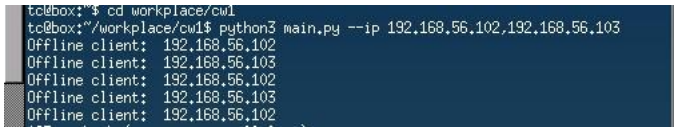

When coding, I met plenty of problems. For example, the data structure and operations of list [3], how to read and write file and how to use json [4], how to control flow with “while True”, “break”, “continue” [5] and “while True: try…except” [6]. I also need to find the functions in Python modules, such as os.path [7] and so on.

# 4. Testing and results

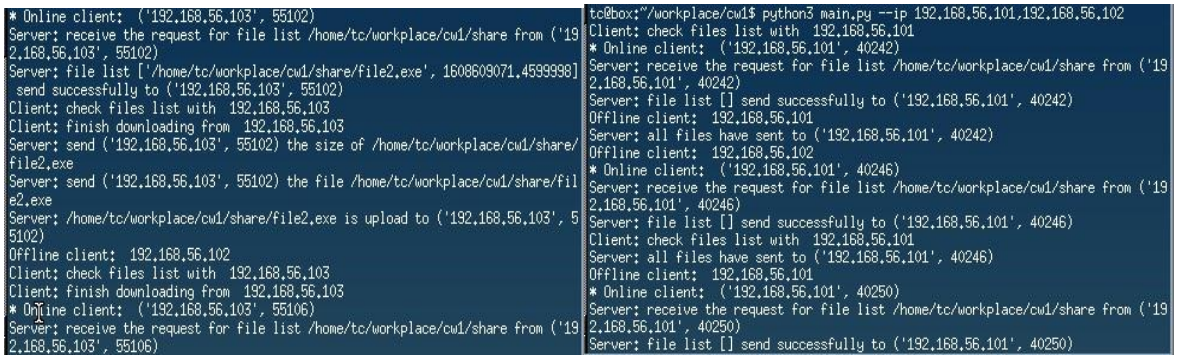
Run the Python application in 3 virtual machines.

Start A and run. Put file into the “share” folder. Then run B.

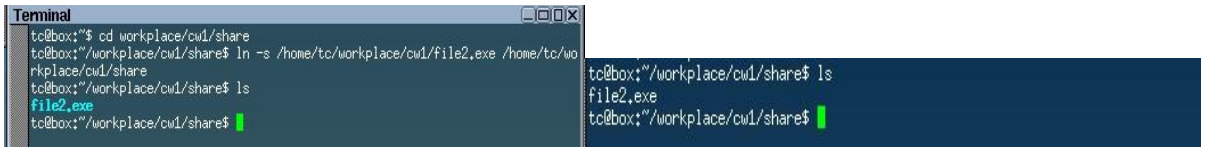
Only A is running:




A run then B run ( A B ):




File synchronization ( A B ):




File transmission successed.

When C run, it also synchronize file successfully.

# 5. Conclusion

In this file sharing application programming, I use threads to increase the transmission speed. The program can read the input parameters and make a virtual machine to communicate with another. Two virtual machines can send file to each other. By improving the data transfer method and simplify the flow, the speed of file synchronization will increase and the application can have more functions.

# References:


[1] A. Ivanov.” Evaluating BBRv2 on the Dropbox Edge Network.” arxiv.org. https://arxiv.org/abs/2008.07699 (accessed Aug.18, 2020).

[2] dropbox.com. https://www.dropbox.com/

[3] “5. Data Structures.” docs.python.org. https://docs.python.org/3/tutorial/datastructures.html#more-on-lists

[4] “7. Input and Output.” docs.python.org. https://docs.python.org/3/tutorial/inputoutput.html#reading-and-writing-files

[5] “4. More Control Flow Tools.” docs.python.org. https://docs.python.org/3/tutorial/controlflow.html

[6] “8. Errors and Exceptions.” docs.python.org. https://docs.python.org/3/tutorial/errors.html#exceptions

[7] X.Wu. “Python OS.Path module: commonly used methods’ details.” cnblogs.com. https://www.cnblogs.com/wuxie1989/p/5623435.html (accessed at Jun.28, 2016).