XI’AN JIAOTONG-LIVERPOOL UNIVERSITY

西交利物浦大学

REMOTE OPEN BOOK EXAM ANSWER SUBMISSION COVER SHEET

|  |  |  |
| --- | --- | --- |
| Name | (Surname)  Ren | (Given Name)  Qinxin |
| Student ID Number | page1image518146561824098 | |
| Programme | DMT | |
| Module Title | network | |
| Module Code | CAN201 | |
| Module Examiner | Cheng Fei | |

By uploading or submitting the answers of this Remote Open Book Exam , I certify the following:

* I will act fairly to my classmates and teachers by completing all of my academic work with integrity. This means that I will respect the standards and instructions set by the Module Leader and the University, be responsible for the consequences of my choices, honestly represent my knowledge and abilities, and be a community member that others can trust to do the right thing even when no one is watching. I will always put learning before grades, and integrity before performance.
* I have read and understood the definitions of collusion, copying, plagiarism, and dishonest use of data as outlined in the Academic Integrity Policy, and cheating behaviors in the Regulations for the Conduct of Examinations of Xi’an

Jiaotong-Liverpool University.

* This work is produced all on my own and can effectively represent my own

knowledge and abilities.

I understand collusion, plagiarism, dishonest use of data, submission of procured work, submission of work produced and/or contributed by others are serious academic misconducts. By uploading or submitting the answers with this statement, I acknowledge that I will be subject to disciplinary action if I am found to have committed such acts.

Signature .............qinxin.ren............................. Date ......*.........2020.12.22...........................*...............

1. Abstract:

File synchronization has been widely used in people’s daily life. There have been various applications like dropbox, iCloud which implement file sharing function and greatly convenient people’s life. To have a better understanding of principles of socket, I work very hard trying to implement a simple project. This project aims at using Python Socket network programming to design and implement Large Efficient Flexible and Trusty File Sharing.

In this project, python socket and multi-processing and threading are used as the basic technology. Apart from this, protocols of transform files between peers are also well designed. Three process which individually achieves listening, file scanning and file downloading are called in main function. There are also lots of threadings inside of each process to discover and download the required file. As a result, it achieves the function that synchronize files between 3 virtual machines. Large files can be sent safely as well as folders.

These results suggest that file sharing can be realized on different virtual machines in one physical machine by using tcp. Moreover, transition between different computers can be realized in the further working.

1. Introduction:

File sharing refers to the active sharing of your computer files over a network (Internet or small network). Generally, file sharing uses P2P mode, and the files themselves are stored on the user's personal computer. Most people who participate in file sharing also download shared files provided by other users. Sometimes the two actions go hand in hand. File synchronization and sharing refers to sharing files with others in the same time. In my project, I use C/S mode to achieve the file sharing and synchronization, for there are only 3 machines. The application scene is that when 3 machines belong to one physical machine are online, once one machine detect that there is a new file or folder, it will send a message to its peers. If the peer finds out this file doesn’t exist in current directory, it requests from one who has to download this file. Also, when one detects that the file is update, others will also request this new file.

My project is initially based on tcp socket. Considering the fact that it is a reliable data transfer and all data are sent as a stream, I use json to encrypt data and the data length together, the receiver each time will receive a given length data, in case of stick bags.

Most importantly, because each virtual machine is both a server and a client, so I put them in two processes, then they can work individually and share some parameters in between. In order to implement different functions such as scanning files or requesting files, I use threadings in process to handle these functions at the same time.

* 1. Literature review:

Taking Dropbox, the most widely used application nowadays, which has made significant strides forward in recent years for example. It is user-facing, thus user can back up anything they like including a folder to server and get it everywhere. For safety aspect, it also encrypted data to keep it safe from prying eyes. Data transferred using the service are protected by SSL encryption while stored files are guarded by AES-256 protection [1]. There are also some limitations for example the file location. At times, the lack of specific tabs can make it very hard to locate older versions of files or deleted copies.

1. Methodology:

3.1 Proposed protocol and flow chart:

图示

描述已自动生成

3.2 Proposed function:

There are lots of functions defined in my project, based on the flow chart given above. Functionally, I aim to realize file synchronization between 3 virtual machines. Specifically speaking, there are a scanner to discover new files or updated files, a server to inform his peers that he has accepted a new file and a client to check whether it needs it and then request for the download.

1. Steps of implementation

4.1 Determine the object

Firstly, I use OOP to divide objects. There are classes server, the one who detects new file and provide downloading; serverMessage, the one to encrypted server’s message into json form; client, the one who checks whether he needs this file and request for downloading; clientMessage, the one to encrypted client’s message into json form; main, the one which call server and client at the same time.

4.2 Determine the behavior and attributes of objects

After creating these classes, I determine each object own behavior and unique attributes during this process. Taking the most important 2 classes for example: there are 3 parameters needed when being called. Host and port are used for binding a server, encryption is used to switch on the encrypt function; client has same 3 parameters to switch on the function when being called.

Inside of server, firstly it should first wait until a client is accepted. Then it switches on the scanning method to detect whether there’s a new file or any file whose last modify time has changed. Then it sends message to client by using snedFile method and wait for client’s request. When receive the request, server firstly analysis the operation code. When operation code is 1, it will send file block automatically. Otherwise it will return to the previous block. Encryption is implemented when file block is made.

Inside of client, it initially tries to connect a server. When successfully connect, it begins to wait until server send a new file information. When a new file information is received, it firstly checks whether has had this file. If not has, request for the downloading directly. Otherwise, request more information from the server. Checks the size firstly to see that if this file has been completely received, then check the md5 to see whether the file has been correctly received. Any of these conditions is not satisfied will start to request download from server.

4.3 Improvement of coding

To have a more clear and well-structured code, I instantiate these classes and write them in the same form. Finally, I carried out the realization of this project.

4.4 Test and debug

Frankly speaking, I encountered countless bugs, I work very hard to try to solve them while some are still there. Those will be explained detailly later.

1. Testing and result

When virtual machine A receives a new file, the server opens the thread to scan the new file, and sends the information of the new file to the client. The client of other two virtual machines compares the new file with the local one, and requests the download if it is different. If is different, then request for download.

Suppose B disconnects and reconnects. Each connection sends information to the client, and then a new file is sent separately.

Suppose C receives a modify file, it will send to its peers and inform them to download the first 0.1% file.

When a folder or a big file is detected, the server will automatically zip this file. When a zip file is received by the client, it will unzip this file. After unzip, this zip file will be delete automatically.

Every file is firstly downloaded with a suffix ‘.temp’, after downloading, this name will be corrected to the original one. When the server is scanning the files, it won’t send a file with ‘.temp’ suffix.

1. Conclusion

This project is really a tough task. After desperately debugging, I finally carry out the implementation. Overall speaking, my code is not very complex but all functions are achieved. It can automatically detect new file and send it to peers, also receive file at the same time. Compression is used for large file to increase the efficiency and encryption is also achieved. However, there are also a lot of shortcomings.

The speed of sending file is not very high. If enough time is given, I would try threading when send and receive file. However, the control of threading is too difficult to achieve in such a limited time. Moreover, the check function also needs improvement, for sometimes it will send a same file twice.

Finally, this project is achieved, I really learned a lot about python, socket programming and OOP. It’s really a significant task for me. Although my ability is limited, it is greatly improved after this project.

7. Reference

[1] “Dropbox: The Pros And Cons Of Cloud Storage”, https://leedrozak.com/pros-cons-dropbox.