CAN201 Introduction to Networking

Coursework 1

Large Efficient Flexible and Trusty (LEFT) Files Sharing

Contribution to Overall Marks	**45%**
Submission Deadline	**Monday 7th Dec. 2020, 0:01**
Type	**Individual coursework**
Learning Outcome	**[A] [B] [C] [D] [E]**

How the work should be submitted?

- **SOFT COPY ONLY!**
- You must submit your work through Learning Mall.

Specification

File sharing is a commonly used network-based application in our daily life. There are a lot of good platforms that provided such services, such as Dropbox, Google Drive, Baidu NetDisk, iCloud, and XJTLU BOX. You can install their apps on your PC or mobile phone to share and synchronize your files.

This project aims at using Python Socket network programming to design and implement Large Efficient Flexible and Trusty (LEFT) Files Sharing. From the name of this coursework, you may obtain the **requirements** of this coursework:

Large:

- Format: any format, including hidden files and folders
- Size: single file is up to1GB

Efficient:

- Fast: the faster, the better
- Automatic: the changed files can be synchronized automatically
- Partial update: the partially changed files can be updated partially
- Compression: compression can be used to reduce the total size

Flexible:

- The IP addresses should be set as an argument
- Resume from interruption

Trusty:

- No Error for any files
- Error recovery without retransmission
- Data transmission security

You can decide the following items:

Application layer protocol:

- You can design your own protocol. But HTTP is not allowed to be implemented.

Transport layer protocol:

- TCP; or UDP; or Mixed.

Architecture:

- C/S; or P2P; or Mixed.

Port number:

- Any port between 20000 to 30000 can be used;
- You can use one or more ports.

What should be submitted:

Codes:

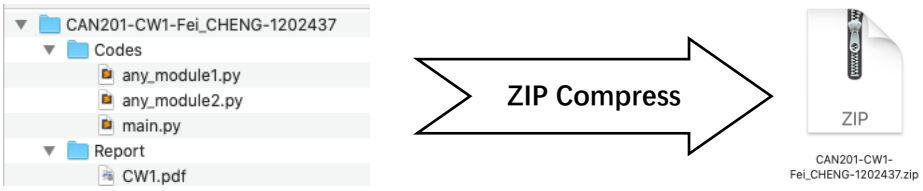
- Python 3;
- Your application can be implemented using multiple Python scripts;
- But there is only one application. There is no difference between a "Client" and a "Server".

A development report:

- A cover page with your full name and student ID;
- 3~ 8 pages, single column, 1.5x line space, 2.54cm margins, Serif font[1], font size:12pt;
- PDF format, LaTeX is recommended;
- Including:
  - Abstract
  - Introduction: project requirement (do not copy from this document), background, literature review (try to find research papers, development reports or testing report of similar apps), and introduce what you did in this coursework;
  - Methodology: proposed protocols (using FSM or mind map), proposed functions and ideas;
  - Implementation: steps of implementation, program flow charts, programming skills (OOP, Parallel…) you used, what difficulties you met and how to solve them;
  - Testing and results: testing environment, testing plan and testing results (screenshot, tables or curves for showing performance)
  - Conclusion: what you did and why you did it. Future plan and so on.
  - Reference [IEEE format]

Meanwhile, you have to follow the compulsory requirement (no tolerance[2]):

- The file structure of your submission:



- Only ZIP file is allowed to submit.

[1] Eg. Times New Roman, Modern No. 20 or Cambria.
[2] It means that if you do not follow the compulsory requirement, your work will be marked as zero.

- The starting Python script file should be named as "main.py";
- Run command: `python3 main.py --ip <ipv4 addresses>` [3]

Allowed Python modules:

- General modules: *os, sys, socket, struct, hashlib, math, tqdm, numpy, threading, multiprocessing, gzip, pycryptodome, zlib, zipfile, time …*[4]
- Any modules for encryption
- Any modules for compression

Marking Criteria

Report (40%)

Marking Criteria	Item	Mark
Structure (5%)	Structure	5%
Contents (30%)	Abstract	2%
	Introduction	4%
	Methodology	6%
	Implementation	7%
	Testing and results	8%
	Conclusion and reference	3%
Format and language (5%)	Report style and format	3%
	Language	2%

Marking scheme:

1. Structure (5%)
   - Well organized structure: 5%
   - Reasonable structure: 3% ~ 4%
   - Disordered structure: 0% ~ 2%
2. Contents (30%)
   2.1. Abstract (2%)
   - Appropriate abstract (2%)
   - No abstract (0%)

   2.2. Introduction (4%)
   - Excellent (4%)
   - Lack of necessary parts (1%-3%)
   - No introduction (0%)

   2.3. Methodology (6%)
   - Excellent methodology: sufficient and accurate figures and text description (6%)
   - Reasonable methodology: clear figures and text description (3%-5%)
   - Incomplete methodology (1%-2%)
   - No methodology (0%)

   2.4. Implementation (7%)
   - Excellent implementation: sufficient steps of the implementation with proper figures or charts (7%)

[3] Eg. python3 main.py --ip 192.168.10.1,192.168.10.2

[4] If you think some python module is really required, please let me know before week 9. I will add such module to the white list. Any modules in the virtual machine provided for this CW is allowed to use.

- Reasonable methodology: clear steps of the implementation with figures or charts (4%-6%)
- Incomplete implementation (1%-3%)
- No implementation (0%)

2.5. Testing and results (8%)
- Excellent: sufficient testing plan for different cases, sufficient results to show the performance with proper analysis (7%-8%)
- Acceptable: clear testing plan, clear results to show the performance with analysis (3%-6%)
- Lack of testing plan, results or analysis (1%-2%)
- No testing and results (0%)

2.6. Conclusion and reference (3%)
- Excellent conclusion and reference with the correct format (2%-3%)
- Acceptable conclusion and reference (1%)
- No conclusion or incorrect reference (0%)

3. Format and language (5%)

3.1. Report style and format (3%)
- Beautiful and clear typography: 3%
- Acceptable typography: 2%
- Bad typography: 0% ~ 1%

3.2. Language (2%)
- Accurate and concise language: 2%
- Unclear and confusing language: 0% ~ 1%

- ***Code (60%)***

Code running environment:

- It will be published to you at the first day of week 7;
- You have to test your app using the provided virtual machine without any modification;
- The app that cannot be executed properly in this virtual machine will be marked as zero.

Code testing steps:

1. Your app (you have only one app with single python code file or multiple python codes) will be copied to 3 virtual machines: VMA, VMB and VMC, with the same settings;
2. For each virtual machine, your app will be started as:
   `python3 main.py --ip 192.168.xxx.xxx,192.168.xxx.xxx` [5]
3. The VMA will be started and it will be running[6]. It should run without any error (RUN_A).
4. I will add *File 1* around 10MB to the "share" folder in the current working directory[7] of VMA.
5. The VMB will be started without error (RUN_B). Your app in VMB will get the *File 1* and put it in the "share" folder in the current working directory[8]. I will check the md5 of *File 1* (MD5_1B) record the time consuming (TC_1B).
6. The VMC will be started without error (RUN_C). Your app in VMC will get the *File 1*. I will check the md5 of *File 1* (MD5_1C) record the time consuming (TC_1C).
7. I will add *File 2* around 1GB and a folder (any name) with 50 small files to the "share" folder in the current working directory of VMB.

[5] These are other hosts' IP addresses. xxx is from 0~254.
[6] The code will run until being killed by Control-C. A "while True:" loop can be used for this.
[7] Generally, it is the folder where your app is started. Pleased do not change it in your codes.
[8] Actually, all the files for sending and receiving should be located in the "share" folder of the current working dire

8. *File 2* and the folder with 50 small files should start to be synchronized to VMA and VMC.
9. After 2 seconds of step 7, the app on VMA will be killed by me.
10. All the files and the folder will be synchronized on VMC. I will check the md5 of *File 2* (MD5_2C) and md5 of every file in the folder (MD5_FC) and record the time consuming of *File 2* (TC_2C) and the folder (TC_FC).
11. I will restart the app on VMA. All the files and the folder will be synchronized on VMC. I will check the md5 of *File 2* (MD5_2A) and md5 of every file in the folder (MD5_FA) and record the time consuming of *File 2* (TC_2A) and the folder (TC_FA).
12. I will add *File 3* around 200MB with random binary data to the "share" folder in the current working directory of VMC.
13. *File 3* will be synchronized on VMA and VMB.
14. I will replace around 0.1% continuous area[9] with other random binary data for *File 3* on VMC.
15. The updated *File 3* will be partially synchronized on VMA and VMB. I will check the md5 of *File 3* on VMA and VMB (MD5_3A, MD5_3B) and record the time consuming of *File* 3 on VMA and VMB (TC_3A, TC_3B).
16. Compression will not be tested directly. But the *File 2* will be easy to compress to a very small size. If you implement the compression, you will get benefit from TC_2A and TC_2C.
17. Data security will not be tested directly. I will check your code. As encryption will waste time, please use "`--encryption yes`" argument to switch on the encryption function. (ENC)

Marking Criteria	**Item for testing**	**Mark**
Phase 1 (6%)	RUN_A	2%
	RUN_B	2%
	RUN_C	2%
If you cannot get 6% of Phase 1, your app will not go to the testing of the next phase.		
Phase 2 (14%)	MD5_1B	2%
	TC_1B: Top 10%: 5% Top 10%-30%:4% Top 40%-60%:3% Top 80%-100%:2% > 40 second: 0%	5%
	MD5_1C	2%
	TC_1C: Top 10%: 5% Top 10%-30%:4% Top 40%-60%:3% Top 80%-100%:2% > 20 second: 0%	5%
If you cannot get >=8% of Phase 2, your app will not go to the testing of the next phase.		
Phase 3 (20%)	MD5_2C	2%
	TC_2C: Top 10%: 5% Top 10%-30%:4% Top 40%-60%:3%	5%

[9] To make the situation easier, the change will start from the first bytes of the file.

	Top 80%-100%:2% > 5 min: 0%	
	MD5_FC	2%
	TC_FC: < 1min: 1% Or: 0%	1%
	MD5_2A	2%
	TC_2A: Top 10%: 5% Top 10%-30%:4% Top 40%-60%:3% Top 80%-100%:2% > 5 min: 0%	5%
	MD5_FA	2%
	TC_FA < 1min: 1% Or: 0%	1%
If you cannot get >=12% of Phase 3, your app will not go to the testing of the next phase.		
Phase 4 (10%)	MD5_3A	2%
	MD5_3B	2%
	TC_3A: < 5 second: 3% 5~10 second: 2% 15~20 second: 1% > 20 second: 0%	3%
	TC_3B: < 5 second: 3% 5~10 second: 2% 15~20 second: 1% > 20 second: 0%	3%
Phase 5 (5%)	ENC	5%
Coding style (5%)	Comments / readability	5%