CAN201 Introduction to Networking

Networking Project (Resit)

Large Efficient Flexible and Trusty (LEFT) Files Sharing

Contribution to Overall Marks	**100%**
Submission Deadline	**5th Aug. 2022 23:59**
Type	**Individual coursework**
Learning Outcome	**ALL**

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 excellent apps that provided such services, such as Dropbox, Google Drive, Baidu NetDisk, iCloud, and XJTLU BOX.

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

Large:

- Format: files with any format and folders (excluding hidden files and folders)
- Size: single file is up to 500MB (50 MB will be tested)

Efficient:

- Fast: the faster, the better
- Automatic: the new files/folders and changed files/folders can be synchronized automatically

Flexible:

- The IP addresses of peers should be set as an argument (only two peers)
- Resume from interruption

Trusty:

- No Error for any files

You can decide the following items:

Application layer protocol:

- You should design your own protocol. Any existing mature application protocols, such as HTTP and FTP are not allowed to be used.

Transport layer protocol:

- TCP;

Performance statistics

- Display the network performance statistics, such as average bit per second.

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.

Security risk

- Implement security protocols and/or identify security risks in the report.

What should be submitted:

Codes:

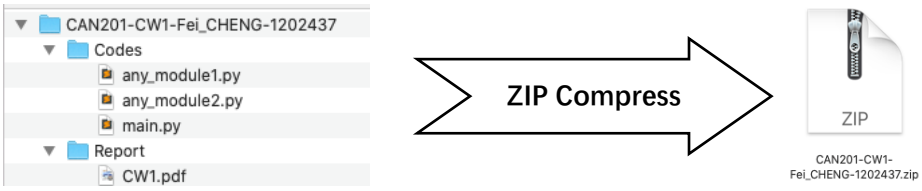
- >= Python 3.6;
- Your application can be implemented using multiple Python scripts;
- But there is only one application. There is no difference between a "Client" and a "Server".

A development report:

- A cover page with your full name (pinyin for Chinese student; name on your passport for international student) 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 reports 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 kinds of 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 <peer's ipv4 address>` [3]

Allowed Python modules:

- *os, sys, shutil, socket, struct, hashlib, math, tqdm, numpy, threading, multiprocessing, gzip, zlib, zipfile, time, argphase, json*

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

[3] Eg. python3 main.py --ip 192.168.1.100

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 testing steps:

1. Your app (you have only one app with single python code file or multiple python codes) will be copied to a folder of 2 machines: **PC_A**, **PC**_B;
2. Folder structure:
   |---current_working_directory
   |---main.py
   |---*other_py_files*.py
   |---share
   |---*files_or_folders_for_sharing*
3. For each machine, your app will be started as:
   `python3 main.py --ip 192.168.xxx.xxx` [4]
4. The app on PC_A will be executed, and it will be running for long time[5]. It should run without any error (**RUN_A**).
5. ***File_1*** (10MB) will be added to the "share" folder in the current working directory[6] of PC_A.
6. The app on PC_B will be executed, and it will be running for long time. It should run without any error (**RUN_B**).
7. Your app in PC_B will get the ***File_1*** and put it in the "share" folder in the current working directory[7]. The timeout is 10s.
8. The md5 of ***File_1*** (**MD5_1B**) will be checked and the transmission time (**TC_1B**) will be recorded.
9. ***File_2*** around 500MB and a ***folder*** (any name) with 50 small files (1 KB) will be added to

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

the "share" folder in the current working directory of PC_B.

10. ***File_2*** and the ***folder*** with 50 small files should start to be synchronized to PC_A. The timeout is 60s.
11. After 0.5 seconds of step 8, the app on **PC_A** will be killed (interrupted, normally some files will not be received by PC_A).
12. The app on PC_A will be executed again. All the files and the folder will be synchronized to PC_A.
13. The md5 of ***File_2*** (**MD5_2A**) and md5 of every file in the folder (MD5_FA) will be checked and record the time consuming of ***File_2*** (**TC_2A**) and the folder (**TC_FA**) will be recorded.
14. ***File_2*** on PC_A will be randomly updated (at least 10% of the bytes will be changed).
15. The updated ***File_2*** will be synchronized to PC_B. The timeout is 50s.
16. The md5 of ***File_2*** on PC_B (MD5_2B) and the time consuming (TC_2B) will be recorded.

Marking Criteria	**Item for testing**	**Mark**
Phase 1 (10%)	RUN_A	5%
	RUN_B	5%
If you cannot get 10% of Phase 1, your app will not go to the testing of the next phase.		
Phase 2 (15%) (timeout 10s)	MD5_1B	10%
	TC_1B: Top 10%: 5% Top 10%-30%: 4% Top 40%-60%: 3% Top 80%-100%: 2% > 10 second: 0%	5%
If you cannot get >=10% of Phase 2, your app will not go to the testing of the next phase.		
Phase 3 (20%) (timeout 60s)	MD5_2A	5%
	TC_2A + TC_FA: Top 10%: 10% Top 10%-30%:8% Top 40%-60%:6% Top 80%-100%:3% > 60 second: 0%	10%
	MD5_FA	5%
If you cannot get >=10% of Phase 3, your app will not go to the testing of the next phase.		
Phase 4 (10%) (timeout 50s)	MD5_2B	5%
	TC_2B: Top 10%: 5% Top 10%-30%: 4% Top 40%-60%: 3% Top 80%-100%: 2% > 50 second: 0%	5%
Coding style (5%)	Comments / readability	5%

<u>Code testing environment:</u>

- The online testing environment will be published to you at the last day of week 9;
- The testing environment only provide the testing result of tags: ***RUN_A, RUN_B and***

MD5_1B.