National University of Singapore

School of Computing

CS2105 **Assignment 1** (10 Marks) Sem 2 AY21/22

Submission Deadline

23rd Mar (Wednesday) 11:59 pm. 2 marks penalty will be imposed on late submission (Late submission refers to submission or re-submission after the deadline). The submission folder will be closed on **30th Mar (Wednesday) 11:59 pm**.

Objectives

In this assignment, you will implement client and server for file transfer. After completing this assignment, you should

- be able to construct application later packets
- have a good understanding communication protocol design.

Group Work

All the work in this assignment should be done individually. However, if you find the assignment too difficult,

- you are allowed to form a group with another student
- maximum two students per group.
- Group submission is subject to **2 marks penalty** for each student.

Under no circumstances should you solve it in a group and then submit it as an individual solution. This is considered plagiarism. There will be no acceptance for excuses such as forgetting to declare as group submission. Please refer to the "Special Instructions for Group Submission" and "Plagiarism Warning" on page 2 for more details.

Grading

We will test and grade your program on the `sunfire` server. Please make sure that your program run properly on `sunfire`. Moreover, you are allowed to use libraries installed in public folders of `sunfire` (e.g. `/usr/lib`) only.

We accept submission of Python 3 (in particular, 3.7), Java, or C/C++ program, and we recommend that you use **Python 3** for your assignments. Programming languages other than Python 3, Java, and C/C++ are not allowed. For Python 3, we use the `python3` program installed in folder `/usr/local/Python-3.7/bin` on `sunfire` for grading. If you use Java or C/C++, we will compile and run your program for grading using

the default compilers on `sunfire` (`java 9.0.4` installed in `/usr/local/java/jdk/bin`, or `gcc 4.8.4` installed in `/usr/local/gcc-4.8/bin`). The grading script infers your programming language from the file extension name (.py, .java, .c). Therefore, please ensure your files have the correct extension names. For a Java program, the class name should be consistent with the source file name, and please implement the static `main()` method so that the program can be executed as a standalone process after compilation. We will **deduct 1 mark** for every type of failure to follow instructions (e.g. wrong program name).

Note that for **Java** programs, name your main class as Server/Client and hence source file name Server.java/Client.java during development. When you want to test your program using the provided script on Sunfire, or to make submission, rename your file name according to program submission and group submission section **while keeping the class name as Server/Client**. Grading script will rename your file accordingly during compilation.

We will grade your program based on its correctness only. A grading script will be used to test your program and no manual grading will be provided.

Program Submission

For individual submission, please name your two source file as `Server-<Matric number>.py` and `Client-<Matric number>.py`; and submit it to the `Assignment_2` folder of LumiNUS Files.

Here, `<Matric number>` is your matriculation number which starts with letter A. An example file name would be `Server-A0165432X.py`. If you use Java, C, or C++ to implement the web server, please use `.java`, `.c`, or `.cpp` respectively as the extension name. Note that file names are **case-sensitive** on `sunfire`.

You are not allowed to post your solutions to any publicly accessible site on the Internet.

Special Instructions for Group Submission

For group submission, please include matriculation numbers of both students in the file name, i.e. `Server-<Matric number 1>-<Matric number 2>.py`. Submit it to the same `Assignment_2_student_submission` folder. An example file name would be `Client-A0165432X-A0123456Y.py`. For each group, there should be one designated member who submits the file, to avoid problems caused by multiple branches within a group. **Do not change the designated submitter!** If the group needs to upload a new version, it should be done by the same designated submitter as well.

Plagiarism Warning

You are free to discuss this assignment with your friends. However, you should refrain from sharing your program, program fragments, or detailed algorithms with others. If

you want to solve this assignment in a group, please do so and **declare it as group work**.

We employ zero-tolerance policy against plagiarism. If a suspicious case is found, student would be asked to explain his/her code to the evaluator in face. Confirmed breach may result in zero mark for the assignment and further disciplinary action from the school.

Question & Answer

If you have any doubts on this assignment, please post your questions on Piazza forum before consulting the teaching team. However, the teaching team will NOT debug programs for students and we provide support for language-specific questions as a best-effort service. The intention of Q&A is to help clarify misconceptions or give you necessary directions.

FAQ

We will collate your questions here: link

The Unreliable Channel

We have implemented 3 simulators on the machine `137.132.92.111`, to mimic unreliable channels. Your server and client have to connect to any simulators and transfer files.

- The simulator waits for both the client and server to connect the simulator via `TCP` connection before enabling the channel.
- The architecture can be found in fig:1
- Task
  - connect to the simulator with a handshake. The handshake is similar to Assignment 1, with an extra character `C` to signify client and `S` to signify server. The handshake details can be found in fig:2
  - The simulator will send you the current waiting list number; you need to wait till you receive `0_`. We allow only 10 students to connect to a simulator at any given time.
  - Then you transfer file from the server to the client. The input file will be provided as command line parameter to the server
  - Write the MD5 hash to the file name provided to client as command line parameter
- You earn 1 mark per correct hash written in the case of `Reliable Channel`.
- You earn 2 mark per correct hash written in the case of other channels.

- Forgot to mention; the simulator really hates making friends. It will timeout and terminate in a fixed time. It is set to 100 Sec now, but will be progressively reduced based on the statistics we collect.

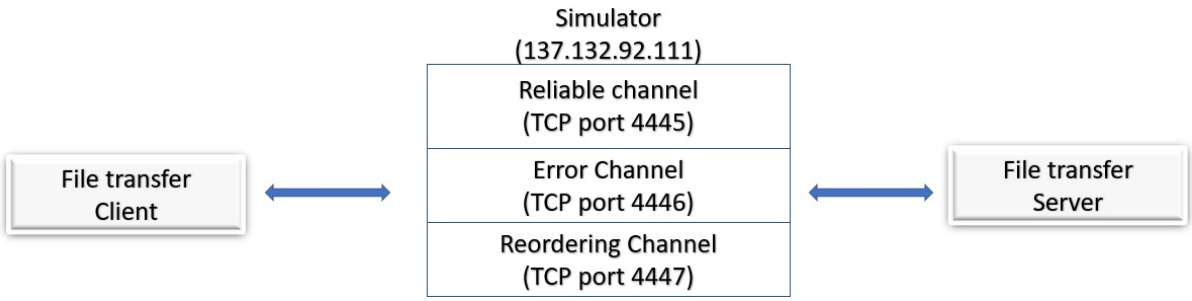

Figure 1: Architecture.

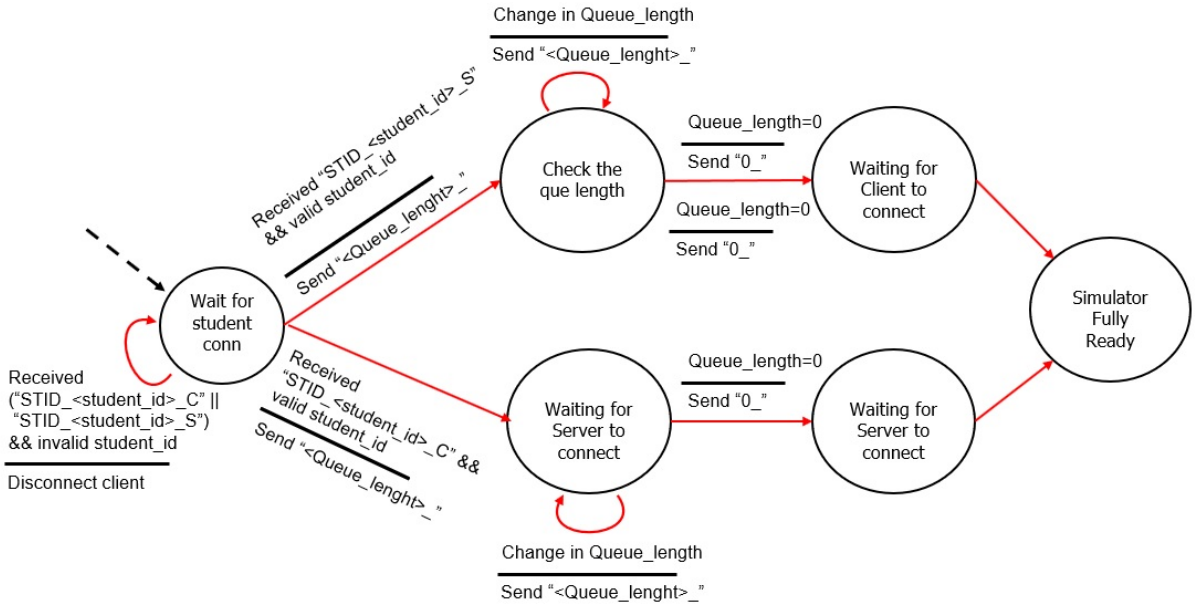

Figure 2: Handshake FSM.

Let us understand the Simulator further:

- All the simulators have the following constraints
  - The packet size from the server is fixed to 1024 B.
  - The packet size from the client is fixed to 64 B.
  - Partial packets will not be delivered
  - No packets will be dropped
- The 3 Simulators are:
  - *Reliable Channel (Mode 0)*: This channel introduces no error or reordering of packets. Fig 3
  - *Error Channel (Mode 1)*: This channel introduces errors in packets. Fig 4
  - *Reordering Channel (Mode 2)*: This channel reorders packets. Fig 5

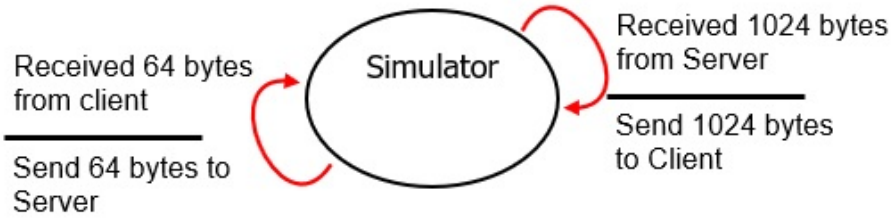

Figure 3: Reliable channel FSM.

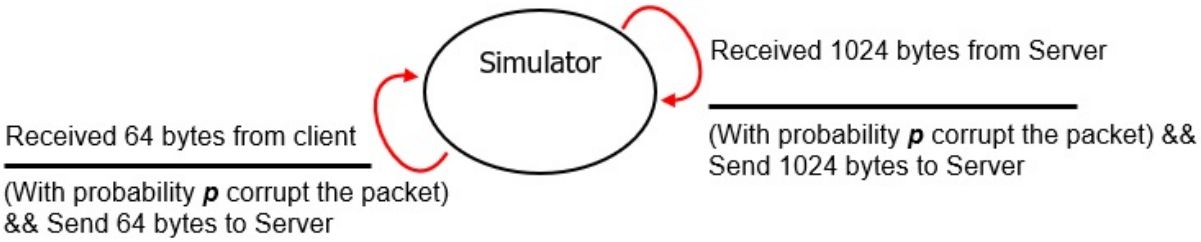

Figure 4: Error channel FSM.

Testing Your Program

To test your program, please use your SoC UNIX ID and password to log on to `sunfire` as instructed on Assignment_0.

- Since you have 2 programs, you will be running them on 2 separate terminals
- Your program should receive five **command-line argument** as the following command shows:

  `python3 Sever-A0165432X.py <student_key> <mode> <ip_address> <port_num> <input_file_name>`

  `python3 Client-A0165432X.py <student_key> <mode> <ip_address> <port_num> <output_file_name>`

  - `<student_key>`: The 6 digit key is the same as the one used in Assignment_1.
  - `<mode>`: Is the integer input corresponding to the simulator we are connecting to
  - `<ip_address>`: The IP address of the machine on which the simulator is running.
  - `<port_num>`: The port number of the `TCP` socket corresponding to the simulator.
  - `<input_file_name>`: The file to be transferred
  - `<output_file_name>`: The file to which the hash is to be written
- During the evaluation, we will change the input files, the IP address and the port number; hence your code should accept all the above mentioned command line parameters.
- Note that your program should not read from stdin. Your program can print anything to stdout or stderr, and our test script will silently ignore them.
- We also release a set of grading scripts to you under the **test** folder.

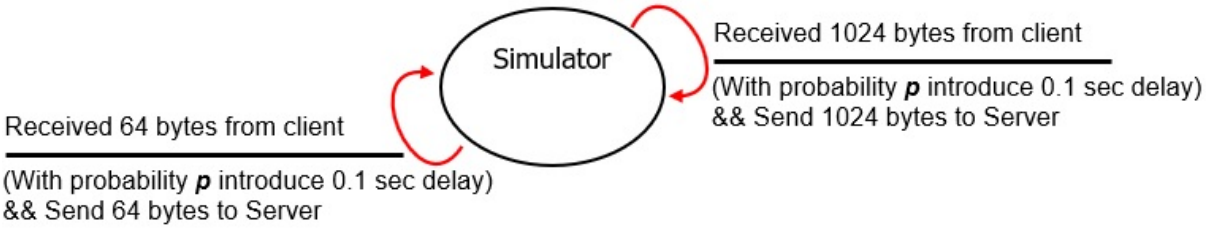

Figure 5: Reorder channel FSM.

- To use the grading script, please upload your program along with the `test` folder given in the package to `sunfire`. Ensure that your program and the `test` folder are in the same directory. You can run the following command for help:

  ```
  bash test/FileTransfer.sh -h
  ```

- To run the test script for server, kindly use the `-s` option

All of you will be connection to a single server, hence start the assignment early to avoid "congestion" during last few days.