National University of Singapore
School of Computing

CS2105 **Assignment 1** Semester 2 AY15/16

Submission Deadline

22 February 2016 (Monday), 6pm sharp. 2 points penalty will be imposed on late submission (Late submission refers to submission or re-submission after the deadline for whatever reason).

Objectives

In this assignment, you are required to transfer a file over UDP protocol. This programming assignment is worth 5 marks and shall be completed **individually**.

Testing Your Programs

You are free to write your programs on any platform/IDE that you are familiar with. However, you are responsible to ensure that your programs run properly on **`sunfire`** **server** because **we will test and grade your programs on `sunfire`.**

Program Submission

Please submit your programs to **`CodeCrunch`**: https://codecrunch.comp.nus.edu.sg.

You may submit multiple programs to **`CodeCrunch`** simultaneously by pressing the <Ctrl> key when choosing multiple programs to upload. Note that **`CodeCrunch`** is just used for program submission and no test case has been mounted on it. You may ignore the feedback from CodeCrunch regarding the quality of your programs (e.g., can improve).

Grading

Your programs will be graded according to their correctness using a grading script:

- [1 point] Programs are compilable on **`sunfire`**, program execution follows specified **`Java`** commands (see sections below) and terminates properly.
- [4 points] Programs can successfully transfer a file from sender to receiver. The received file should have identical content as the sent file (use **`cmp`** command to check, as stated in Assignment 0 Exercise 4). Your program should work for both text and binary files, and for both small and large files (a few MBs). Grading script doesn't care what messages your programs print on the screen. It just checks if the generated file is identical to the sent file.

FileSender Class

In this assignment, you are required to write a **`FileSender`** program and a **`FileReceiver`** program that implement a file transfer application over UDP protocol.

The **`FileSender`** program is basically a file uploader that opens a given file and sends its content to a **`FileReceiver`** through UDP protocol.

To run **`FileSender`** on **`sunfire`**, type command:

```
java FileSender <path/filename> <rcvHostName> <rcvPort> <rcvFileName>
```

For example:

```
java FileSender ../test/cny.mp3 localhost 9000 gxfc.mp3
```

sends the file **`cny.mp3`** from directory **`../test`** to a receiver running in the same host at port 9000. Receiver will store the file as **`gxfc.mp3`** in the same directory as the receiver program.

You may assume that during testing, (1) your sender program will be supplied with the correct path and filename, and (2) filename won't exceed 100 bytes long. No input validation is needed.

(Note: Windows system uses a different file separator '\', e.g., **`..\test\cny.mp3`**)

FileReceiver Class

The **`FileReceiver`** program receives a file from **`FileSender`** and saves it in the same directory as **`FileReceiver`** program, with a filename given by the **`FileSender`**.

To run **`FileReceiver`** on **`sunfire`**, type command:

```
java FileReceiver <port>
```

For example:

```
java FileReceiver 9000
```

listens on port 9000 and dumps the bytes received into a file whose name is given by sender.

Note that receiver program (**`FileReceiver`**) should be launched before sender program (**`FileSender`**) starts sending data.

Though UDP transmission is unreliable, in this assignment, you may assume that underlying transmission channel is perfect and all data will be received in good order. There is no need to implement any reliable data transfer feature. When testing, you should always use "**`localhost`**" as the **`<rcvHostName>`** value so that data is transmitted within local host (thus indeed no network transmission error).

You will transmit data over an unreliable channel in the next assignment where you will enhance your programs accordingly.

Skeleton Programs

Two skeleton Java programs, **`FileSender.java`** and **`FileReceiver.java`**, are provided. You are suggested to modify them for this assignment though you may feel free to develop your programs from scratch. If you choose not to use given skeletons, please make sure your own programs have the same filenames and class names as in given skeletons.

The sender and receiver program should exit elegantly (e.g., no exception, no infinite loop) when file transmission finishes. Both programs should receive user input from command-line argument, i.e., no more user input once programs start execution.

Passing Filename from Sender to Receiver

In this assignment, receiver will store the received file in a name specified by sender. To pass a filename from sender to receiver program, you have a few options. For example, sender may send the filename as the first packet. Alternatively, sender may embed the filename in every packet (as illustrated in Figure 1 below). Your tutorial tutor may discuss with you the pros of cons of these two approaches.

You may add other header fields as appropriate.

The discussions above are deliberately made sketchy so that you have a lot of latitude in deciding your designs.

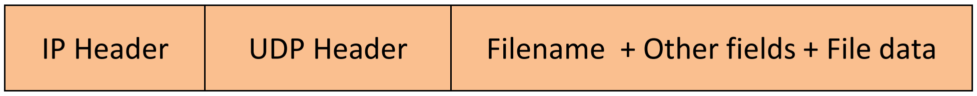

Figure 1: User created application header

Note that each packet **`FileSender`** sends should contain at most 1000 bytes of application data, inclusive of application headers and file data.

Plagiarism Warning

You are free to discuss this assignment with your friends. But, ultimately, you should write your own code. We employ zero-tolerance policy against plagiarism and will do random check among student submissions. 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 this assignment and further disciplinary action from the school.

Question & Answer

If you have any doubts on this assignment, please post your question on IVLE forum or consult the teaching team.