CS 421 – Computer Networks

Programming Assignment I

Image Labeler

Due: Saturday Nov. 9, 2019 at 11:59PM

1) Introduction

In this programming assignment, you are asked to implement a program in **Java**. You should code a client that asks for 3 consecutive images from the server and label them accordingly after the authentication. The client should then send the label results to the server to get confirmation. The server program written in Python 3 is provided for you to test your program.

The goal of the assignment is to make you familiar with the application layer and TCP sockets. You must implement your program using the **Java Socket API of the JDK**. If you have any doubts about what to use or not to use, please contact your teaching assistant.

When preparing your project please keep in mind that your projects will be auto-graded by a computer program. Any problems in the formatting can create problems in the grading; while you will not get a zero from your project, you may need to have an appointment with your teaching assistant for a manual grading. Errors caused by incorrectly naming the project files and folder structure will cause you to lose points.

2) Specifications

"Image Labeler" uses a custom application level protocol which is built on top of TCP to communicate with the client. There will be 4 possible labels for the images where the image dataset is given to you in the HW document. These photos should be in the server side, and clients should request these photos which will be sent 3 at a time and then download them to their side. The sequence of the sent images is important since label check on the server will look for the correct permutation so label messages should be in accordance. After downloading the images to the client side, your client program should label all 3 images and send them in one message to the server, and then wait for the

confirmation from the server. In this homework, you need to ask for these images at least 4 times in the client program lifecycle and it will be graded in that way. The flowchart below can help you in this process.

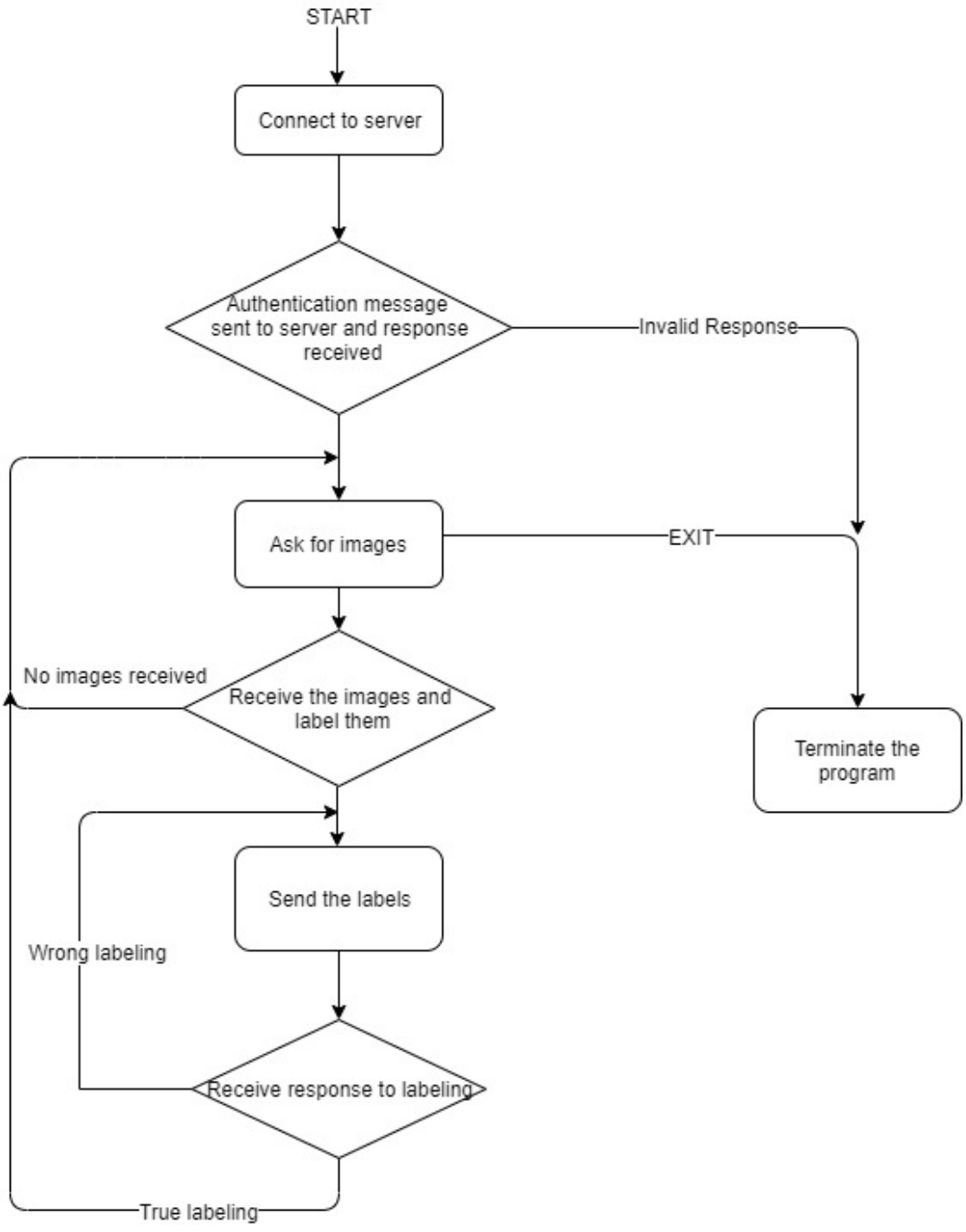

Figure 1 Client process flowchart

3) Connection Formats

i) Commands

For acquiring the program specifications, you will need to use specific commands to communicate with the server. Commands should be strings encoded in **US-ASCII**. The format is below:

<MessageType><Space><Arguments><CR><LF>

- <MessageType> is the name of the command. For all possible commands and their explanations, see Figure 2.
- <Space> is a single space character. Can be omitted if <Argument> is empty.
- <Argument> is the argument specific to the command. Can be empty if no argument is needed for the given command. See Figure 2 for more information.
- <CR><LF> is a carriage return character followed by a line feed character, i.e., "\r\n".

MessageType	Arguments	Definition	Example
USER	<username>	Send the username for authentication	USER bilkentstu\r\n
PASS	<password>	Send the password for authentication	PASS cs421f2019\r\n
IGET	-	Request 3 consecutive image packages (one IGET command is enough for you to get 3 images, don't send 3 IGET commands consecutively)	IGET\r\n
ILBL	<label1>,<label2>, <label3>	Return the labels of the images in one message	ILBL cat,dog,bear\r\n
EXIT	-	Terminates the program	EXIT\r\n

Figure 2 List of commands with examples

ii) Responses

The response messages sent by the server are also encoded in **US-ASCII**. Responses have the following format:

```
<Code><Reason><CR><LF>
```

- <Code> is either OK (success) or INVALID (failure), for the sake of simplicity. You should check the <Reason> for the reason of the failure if <Code> is INVALID.
- <Reason> is the response message. It is always empty when <Code> is OK.
- <CR><LF> is a carriage return character followed by a line feed character, i.e., "\r\n".

For "IGET" command, the response is 3 consecutive packages with the following format:

```
<Code><Size><ImageData>
```

- <Code> is "ISND" for this response to make it different than casual responses.
- <Size> is the size of the image **in bytes** so that end point of the data stream can be known for different images. Size of this part is 3 bytes regardless of the size of image. Maximum size of an image can be 131.072 kilobytes.
- <ImageData> is the image data sent as **bytes** through the socket.

4) Image Analysis

After receiving the images, it is up to you how to analyze the images. However, it is possible to manually label the images yourselves after you save them to the disk and view them so you don't have to code extra to label them. No points will be deducted for labelling the data yourselves.

5) Running the server program

The server program we provide is written in **Python 3.7**. You are also provided the images in a folder, which is required for the server program to work. You must put the image folder in the **same directory** as Server.py and start the server program **before** running your own program using the following command:

```
python Server.py <Addr> <ControlPort>
```

where "< >" denotes command-line arguments. These command-line arguments are:

- <Addr> The IP address of the server. Since you will be running both your program and the server program on your own machine you should use 127.0.0.1 or localhost for this argument.
- <ControlPort> The control port to which the server will bind. Your program should connect to the server from this port to send the control commands.

Example:

```
C:\Users\user\Desktop\CS421_2019FALL_PA1>python Server.py 127.0.0.1 60000
```

The command above starts the server with IP 127.0.0.1, i.e., localhost, which uses port 60000 for the control commands.

6) Running the ImageLabeler

Your program must be **a console application** (no graphical user interface, GUI, is allowed) and should be named as ImageLabeler.java (i.e., the name of the class that includes the main method should be ImageLabeler). Your program should run with the command

```
java ImageLabeler <Addr> <ControlPort>
```

where "< >" denotes command-line arguments. These arguments must be the same as the arguments for the server program, which are explained above.

Example:

```
C:\Users\user\Desktop\CS421_2019FALL_PA1>java ImageLabeler 127.0.0.1 60000
```

In this example, the program connects to the server with IP 127.0.0.1, i.e., localhost, on port 60000. **Please note that you must run your program after you start the server program**.

7) Final Remarks

- **Please contact your teaching assistant Mehmet Sakaoğlu at mehmet.sakaoglu[at]bilkent.edu.tr if you have any doubts about the assignment.**
- **Do not forget to check the response message after sending each command to see if your code is working properly** and correct it if it is not. Note that the server cannot detect all the errors that you make; therefore, you must test a significant number of times in order to detect and correct all your errors.
- You can modify the source code of the server for experimental purposes. However, do not forget that your projects will be evaluated based on the version we provide.
- You might receive some socket exceptions if your program fails to close sockets from its previous instance. In that case, you can manually shut down those ports by waiting for them to timeout, restarting the machine, etc.
- Remember that all the commands must be constructed as strings and encoded with US-ASCII encoding.
- Use big-endian format if it is necessary to use.
- **Put the folder containing the images under the same directory with the server and client codes.**

8) Submission rules

You need to apply all the following rules in your submission. You will lose points if you do not obey the submission rules below or your program does not run as described in the assignment above.

- The assignment should be submitted as an e-mail attachment sent to **mehmet.sakaoglu[at]bilkent.edu.tr.** Any other methods (Disk/CD/DVD) of submission will not be accepted.
- The subject of the e-mail should start with [CS421_2019FALL_PA1], and include your name and student ID. For example, the subject line must be [CS421_2019FALL_PA1]AliVelioglu20141222 if your name and ID are Ali Velioglu and 20141222, respectively. If you are submitting an assignment done by two students, the subject line should include the names and IDs of both group members. The subject of the e-mail should be [CS421_2019FALL_PA1]AliVelioglu20141222AyseFatmaoglu20255666 if group members are Ali Velioglu and Ayse Fatmaoglu with IDs 20141222 and 20255666, respectively.
- All the files must be submitted in a zip file whose name is the same as the subject line except the [CS421_2019FALL_PA1] part. The file must be a .zip file, not a .rar file, or any other compressed file.
- All the files must be in the root of the zip file; directory structures are not allowed. Please note that this also disallows organizing your code into Java packages. The archive should not contain any file other than the source code(s) with .java extension. The archive should not contain these:
    - Any class files or other executables,
    - Any third-party library archives (i.e., jar files),
    - Project files used by IDEs (e.g., JCreator, JBuilder, SunOne, Eclipse, Idea or NetBeans, etc.). You may, and are encouraged to, use these programs while developing, but the end result must be a clean, IDE-independent program.
- The standard rules for plagiarism and academic honesty apply; if in doubt refer to the 'Student Disciplinary Rules and Regulations', items 7.j, 8.l and 8.m.