CS 421 – Computer Networks
Programming Assignment I
Application Level Routing

Due: 16 November 2018

I. Introduction

In this programming assignment, you are asked to implement a program in Java. The program is a simple routing program which will route the incoming message from a client to other servers. The program which sends the original data is called the **client**. The servers to which the client wants to send the messages are called **operators**. However, the client does not have any information about where these operators are. Hence, we need a routing program called **router** which routes the message to the operators in series that will be programmed by you.

The client uses a custom application layer protocol; so, be sure to follow the specifications of the protocol. We will provide you with the tools to test your routing program. Design specifications for this assignment do not require any multi-threading. The assignment should be done using the Java programming language.

Constraints For this programming assignment, you are not allowed to use any third party HTTP client libraries, or the HTTP specific core or non-core APIs supplied with the JDK including but not limited to `java.net.HttpURLConnection`. You **MUST NOT** use the wildcard import keyword (e.g., "`import.net.*`;") and instead explicitly import all necessary classes you use.

Your project grade will be penalized by 50% of the points you receive if you use one of these libraries.

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

When preparing your project please mind the formatting as this project 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 us for a manual grading. Problems caused by Windows/Linux incompatibility will have no penalty. However, errors caused by incorrectly naming the project files and folder structure will cause you to lose points.

II. Design Specifications

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

```
java Router <Addr> <OpKey1> <OpAddr1> … <OpKeyN> <OpAddrN>
```

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

- `<Addr>` [Required] The IP address and port number to which your routing program will bind. You should use the same value for starting up the testing code. The format of this argument will be like `127.0.0.1:9999`
- `<OpKeyi>` [Required] The key of operator `i`
- `<OpAddri>` [Required] The IP address and port number of operator `i`

  Your program should be able to take an indefinite number `N` of `OpKey` and `OpAddr` argument pairs for `i` = `1` to `N`, where `N` ≥ `1`.

e.g.,

```
java Router 127.0.0.1:9999 A 127.0.0.1:10000 B 127.0.0.1: 10001 C
127.0.0.1:10002
```

In the above example, the router is bound to `127.0.0.1`, i.e., localhost, on port `9999`, and then it knows that operators `A`, `B` and `C` are bound to ports `10000`, `10001` and `10002` with the same IPs respectively.

When a user enters the above command, your program will start to listen for incoming connections at `127.0.0.1:9999`. Now you can start the test program to test your routing program.

The custom application layer protocol defined for this assignment is rather simple as it is intended to teach you the basics of how application layer protocols work.

The client sends the message to your router in the following format:

`DATA:<8-digit-number> OPS: <key1>,<key2>,…,<keyN>`

and similarly expects the answer in the following format:

`DATA:<8-digit-number>`

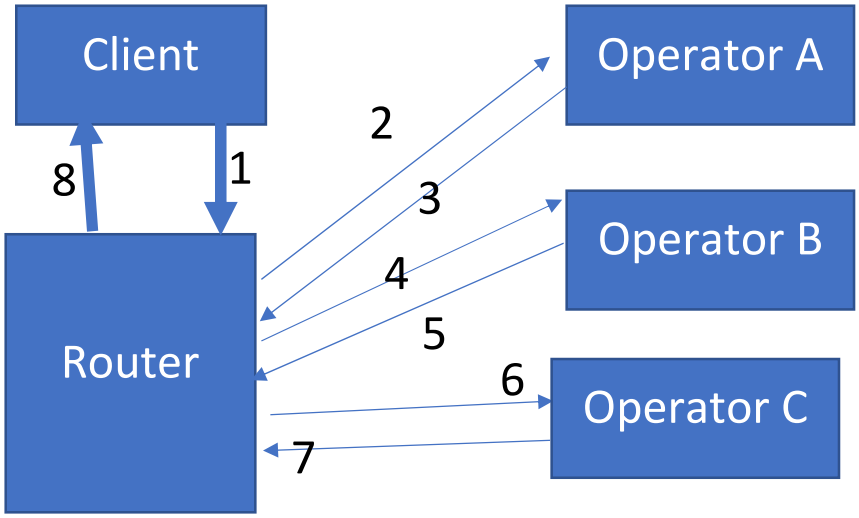
The aforementioned format indicates that the client wants to send data, which consists of 8 digits, to `N` operators in the specified order. One must not assume that the included keys are unique. For example, the following messages are all valid:

`DATA:00000001 OPS: A,B,C`

`DATA:00003231 OPS: op1,op2,op3,op3,op2`

`DATA:00003231 OPS: op1,foo,bar,bar,foo`

In the first message, your router should send the data to operator `A` and wait for its response. Then it should send the received response to operator `B` and then finally to `C`. Then it should send the final output, i.e., the message that arrives from `C`, to the client. The below diagram depicts the expected behavior for the first message:



The **operators** in this assignment carry out simple arithmetic operations like multiplication and addition. **Please note that we will test with operators containing different constants for operations; so, your router program should connect to the indicated operators.**

The testing program is written in **Python** (to avoid decompiling to Java source code for use in the assignment); it will automatically start the client and operators, and will send a test message.

Make sure that both "`Client.py`" and "`Operator.py`" are in the same directory and use the following command structure to test your router:

```
python TestClientandOperators.py <routerAddr>
```

The router will generate a random message and will try to send it to your router program. After that it will switch to listening from the same port. **You should start the test program after you run your router program.**

For this assignment the testing program will launch the client and operators at localhost (i.e., `127.0.0.1`). See the table below of all operators and their ports.

Type	Key	Operation	IP	Port
`Client`	–	-	`127.0.0.1`	`9999`
`Operator`	`A`	Multiply with 17	`127.0.0.1`	`10000`
`Operator`	`B`	Add 13	`127.0.0.1`	`10001`
`Operator`	`C`	Multiply with 11	`127.0.0.1`	`10002`
`Operator`	`D`	Subtract 7	`127.0.0.1`	`10003`
`Operator`	`E`	Multiply with 13	`127.0.0.1`	`10004`

These are the default ports on which the program will run. If you have port conflicts with some other program, you can start the tester program with

```
python TestClientandOperators.py <routerAddr> <port1> <port2>
<port3> <port4> <port5>
```

to manually set the ports of the operators.

For example, when you test the program, the testing program chooses `22` as the data and `A,B,A,D,D,E` as the sequence of routers to visit for the operation, the program will generate the following application layer message:

```
DATA:00000022 OPS: A,B,A,D,D,E
```

For this example, the output message sent from the router after going through all operations should be

$$\Big(\big((22 \times 17) + 13\big) \times 17 - 7 - 7\Big) \times 13 = 85345$$

and the message sent to the client from your router should be

`DATA: 00085345.`

III. Implementation Specifications

Use TCP connections for communicating with the server and the operators.

Assumptions and Hints

- Please contact your assistant if you have any doubt about the assignment.
- You can modify the source code of the client and the operator for experimental purposes. However, do not forget that your projects will be evaluated based on the version we provide.
- We have tested that these programs work with the discussed Java-Python combination.
- Do not forget that many of the socket exceptions you will get is probably because your program failed to close sockets from its previous instance. In that case, you can manually shut down those ports by waiting them to timeout, restarting the machine, etc.

Submission rules

You need to apply all of 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 `yarkin.cetin[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_2018FALL _PA1`], and include your name and student ID. For example, the subject line must be

[CS421_2018FALL _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_2018FALL _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_2018FALL _PA1] part**. The file must be a `.zip` file, not a `.rar` file, or any other compressed file.
- All of 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 class files or other executables,
  - Any third-party library archives (i.e. `jar` files),
  - Any text files **except the report in `pdf` format**,
  - 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 Regulation', items 7.j, 8.l and 8.m.