CSC 424 Computer networks
ECE 534/634 Communication Networks

Project 1: Simple Client

The purpose of this project is to familiarize you with socket programming by implementing a simple client program. This client communicates with a server on a remote machine by following a simple protocol. You need to implement only the client but not the server.

1. Protocol

The server runs on the machine SERVER_ADDR and listens for requests on a TCP socket bound to port SERVER_PORT. Both constants are defined in the header file (common.h) provided for you. There are four types of messages: HELLO, STATUS, BYE and CONFIRM_BYE. Each message is an ASCII string consisting of multiple fields separated by spaces (0x20) and terminated with a line feed (0x0A). The maximum length of each message is MAX_STR_SIZE, which is also defined in the header file given to you.

The client initiates the protocol by sending a HELLO message to the server. The server replies with a STATUS message. The client then sends a BYE message, and the server terminates the connection by sending a CONFIRM BYE message. A connection is successful if and only if all of these messages are correctly sent and received. Since we are using TCP (reliable bytestreams) for communication in this assignment, you do not have to worry about lost messages etc.; you only need to ensure that all messages are sent correctly (and that you receive and parse messages correctly).

The details of each message are as follows:

- HELLO (Client → Server)
  The HELLO message has 3 fields EXACTLY in the following order
  - **Magic String**: It MUST set to be MAGIC_STRING which is a constant defined in the header file ("COMPUTER_NETS2020"). If you send a message which does not contain this magic string, the message will be ignored.
  - **Message Type**: The type string MUST be HELLO to indicate a message type HELLO. The server is case-sensitive.
  - **CID**: The last field is your CID (only the 8 digits without "c").

  Example HELLO message: COMPUTER_NETS2020 HELLO 12345678

- STATUS (Server → Client)
  The STATUS message has 5 fields in the following order:
  - **Magic String**: Same as above.
  - **Message Type**: Must be set to STATUS.
  - **Cookie1**: An integer randomly generated by the server (represented in ASCII). The range is between 1 and 1000.

- **Cookie2**: Another integer randomly generated by the server (represented in ASCII). The range is between 1 and 1000.
- **IP Address and Port number**: A string of the form a.b.c.d:e, representing the IP address and port number of the client.

Example STATUS message: COMPUTER_NETS2020 STATUS 314 159 10.33.12.31:3333

- BYE (Client → Server)
  The BYE message has 3 fields in the following order:
  - **Magic String**: Same as above.
  - **Message Type**: Must be set to BYE.
  - **Sum of Cookies**: An integer set to the sum of the two cookies that are received from the server (represented in ASCII).

Example BYE message: COMPUTER_NETS2020 BYE 473

- CONFIRM_BYE (Server → Client)
  The CONFIRM_BYE message has 3 fields in the following order:
  - **Magic String**: Same as above
  - **Message Type**: Must be set to CONFIRM_BYE.
  - **Secret String**: A string that contains a secret message. You will get the secret string only if the sum of Cookies that you send to the server is correct. The secret string contains a number computed using a secret function based on your cid so it is different for different students.

Example BYE_CONFIRM message: COMPUTER_NETS2020 CONFIRM_BYE Secret(3)!

2. Your client program

It is recommended that you develop your client program on a Linux machine (or a Linux virtual machine) that has the necessary compiler (gcc) and library support (a common Linux distribution such as Ubuntu will do). A tutorial on how to compile and run a C program is provided in the handouts. However, you are allowed to use any programming language (e.g., Java, Python, etc.)

The command line syntax for the client is:

./client <cid>

The client program takes command line argument of your CID.

3. Requirements

You may test your client code with our server as many times as you like. Your client should conform to the protocol described above, or otherwise the server will terminate the connection silently.

Your client program must verify the validity of messages by checking the magic string and message type fields in STATUS and CONFIRM BYE messages. If a received message is not as expected, such as an incorrect magic string or wrong message type, you must assert an error and terminate your program. You should be strict; if the returned message does not exactly conform to the specification above, you should assert an error. Remember that network-facing code should be written defensively.

Your client program should print on screen every message that you send and received. Each message starts with a new line. An example of what you will see on the screen is

```
fakepath> ./client 12345678
COMPUTER_NETS2020 HELLO 123456
COMPUTER_NETS2020 STATUS 314 159 10.33.12.42:3333
COMPUTER_NETS2020  BYE 473
COMPUTER_NETS2020 CONFIRM_BYE Secret(3)!
```

The numbers are randomly generated at the server and recorded. Do not make up your own numbers.

A client_template.c file is also included in the handouts. It provides a skeleton for building the client program. You are free to use this template to complete the codes or choose a different programming language.

4. Submitting your project

You should submit your project through the Blackboard system. Submit two files:

1. The client.c file (or the source file of other programming languages)
2. ONE screen shot (in .jpg or .png format) of a successful run

The files must be uploaded as a single .rar or .zip archived file. Name your archived file as "FirstName_LastName_Project1.rar" or "FirstName_LastName_Project1.zip".