

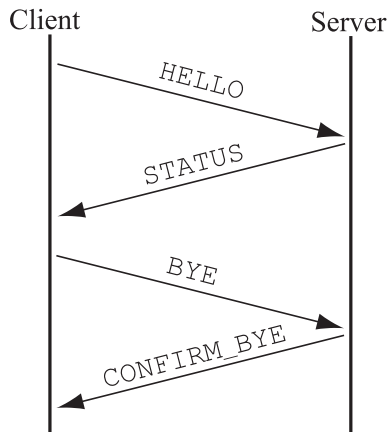
This project is due at 11:59pm on September 17th, 2009.

1 Description

This assignment is intended to familiarize you with writing simple network code. You will implement a client program which communicates with a server using `sockets`. We will give you a sketch of the client program; you only need to fill in the socket-specific bits.

2 Protocol

The server runs on the machine `SERVER_HOSTNAME` and listens for requests on a TCP socket bound to port `SERVER_PORT`. Both constants are defined in the header file provided for you. This exercise has 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 protocol outline is given in the figure above. 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 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:

Project adapted, with permission, from project by Dr. Bobby Bhattacharjee at UMD.

- **HELLO** (From the client to the server: 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 (`cs4700fall12009`). 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.

- **Last Name**

The last field is your last name. Please do **NOT** put spaces in your last name, even if it contain spaces.

An example **HELLO** message might look like this:

```
cs4700fall12009 HELLO Huang
```

- **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.

An example **STATUS** message might be:

```
cs4700fall12009 STATUS 356 478 2.5.2.2:39293
```

- **BYE** (Client → Server)

The **BYE** message has 4 fields in the following order:

- **Magic String**

The 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).

- **Comment**

Optional string field you may send to the server. You may put any visible ASCII characters in the comment, as long as the length of the entire message does not exceed `MAX_STR_SIZE`.

An example BYE message would be:

```
cs4700fall12009 BYE 246 See you later!
```

- **CONFIRM BYE (Server → Client)**

The `CONFIRM_BYE` message has 2 fields in the following order:

- **Magic String**

The same as above.

- **Message Type**

Must be set to `CONFIRM_BYE`.

An example `CONFIRM_BYE` message would be:

```
cs4700fall12009 CONFIRM_BYE
```

3 Your client program

The command line syntax for the client is given below. The client program takes command line argument of your last name. The hostname and port specifications are optional. If included, they should override the default definition of `SERVER_HOSTNAME` and `SERVER_PORT` in `common.h`. The syntax for launching your program is therefore:

```
client [<hostname>[ <port>]] <last name>
```

You should develop your client program on the CCIS Linux machines, as these have the necessary compiler and library support. You are welcome to use your own Linux machines, but you are responsible for getting your code working, and your code *must* work when graded on the CCIS Linux machines. If you do not have a CCIS account, you should get one ASAP in order to complete the project.

4 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. You will be building on these programs for subsequent stages of the term project so it is in your own best interest to make them maintainable.

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 code must be `-Wall` clean on `gcc`. Do not ask the TA for help on (or post to the forum) code⁴ that is not `-Wall` clean unless getting rid of the warning is what the problem is in the first place.

5 Submitting your project

You should submit your project by emailing the instructor at `amislove@ccs.neu.edu`. You should submit only your `client.c` file as an attachment to the email. Please have the subject line for the email be `[CS4700] [Project 0 Submission]` followed by your last name and the last name of your other group member(s).