EECS 325/425: Computer Networks
Project #4
Due: April 27, 3:00pm

This project involves designing an application layer protocol and writing a sockets-based client and server that implements this protocol. In particular:

- Your programs must use only system calls and calls found in `libc`. I.e., no libraries that abstract the details of sockets away.
- You will develop both a client—called `proj4`—and a server—called `proj4d`.
- You must design an application protocol to accomplish some task. Your protocol must consist of at least five commands / operations.
- Your client can be designed such that each invocation performs only a single operation. Your server process must continue to accept and process requests indefinitely. Your server may deal with clients serially (i.e., one at a time), but ideally would handle multiple requests in parallel.
- The server must print each communication received from the client and each communication sent by the server so that the direction is clear. E.g.,

  ```
  C -> S: HELLO
  S -> C: READY
  ```

- The hostnames and port numbers used by your programs should not be hard-coded into the binaries, but rather should be command line options. I.e., the client should accept a hostname and a port number to connect to on the command line and the server should take the port it listens on from the command line. The specifics of the format of the command line options is up to you to design (and document as noted below).
- Your application may utilize UDP or TCP (or both if needed). Choose the most appropriate transport protocol for your task.
- You must write a document (submitted in PDF) that contains: (*i*) an over-arching description of your application in high-level terms, (*ii*) usage instructions for both your client and server, (*iii*) a description of the application protocol you developed and implemented (both the syntax and the semantics of all operations) and (*iv*) justification for using a particular transport protocol. Please write clearly as your explanation of your protocol will be a non-trivial portion of your grade.

Graduate Student addendum:

- In addition to the basic five commands / operations your protocol must do "something else". Some examples: additional non-trivial commands (not just one more like the first five), leveraging more than one connection to complete a task, interfacing with something external—another network service, a disk, some other peripheral, etc.—or offering enhancements that aid performance in some dimension. The "something else" must be documented in your writeup.
- Undergraduates may do the addendum for extra credit.
  - Note: No extra credit will be given for late submissions.
  - Note: No extra credit will be given unless the main portion of the project is substantially completed.

Notes:

1. Your submission must be a gzip-ed tar file called "CaseID-proj4.tar.gz" that contains all code and a Makefile that by default produces *two executables* called "`proj5`" and "`proj5d`" (for the client and server, respectively). If your protocol requires additional files in some format, describe that format in your writeup and include samples with your submission. Do not include multiple versions of your program in your submission. Finally, do not include any directories in the tarball.

2. Any notes you wish to include with your submission should be put in the included writeup.

3. Every source file must contain a header comment that includes (*i*) your name, (*ii*) your Case network ID, (*iii*) the filename, (*iv*) the date created and (*v*) a brief description of the code contained in the file.

4. Do not use spaces in file/directory names.

5. Remember to use good programming practices—including meaningfully naming variables/constants, using modularity, checking for errors, using consistent style, avoiding repetitive code, leveraging reasonable data structures, etc.

6. Remember that your project must build and run on the class server or it will not be graded.

7. Hint: Errors will be thrown at your project. E.g., if you design a command-line directive "-a X" for the client, you can expect me to try to run it without the argument ("X") and/or with two arguments ("X Y").

8. *WHEN STUCK, ASK QUESTIONS!*
   When asking question via email, please include all your source code and a Makefile.