CSCI 4210 — Operating Systems
Lecture Exercise 4 (document version 1.0)
Network Programming and UDP

- This lecture exercise is due by 11:59PM ET on Monday, August 16, 2021
- This lecture exercise consists of practice problems and problems to be handed in for a grade; graded problems are to be done individually, so **do not share your work on graded problems with anyone else**
- For all lecture exercise problems, take the time to work through the corresponding video lecture(s) to practice, learn, and master the material; while the problems posed here are usually not exceedingly difficult, they are important to understand before attempting to solve the more extensive homeworks in this course
- You **must** use C for this assignment, and all submitted code **must** successfully compile via `gcc` with no warning messages when the `-Wall` (i.e., warn all) compiler option is used; we will also use `-Werror`, which will treat all warnings as critical errors
- All submitted code **must** successfully compile and run on Submitty, which uses Ubuntu v18.04.5 LTS and `gcc` version 7.5.0 (`Ubuntu 7.5.0-3ubuntu1~18.04`)

Practice problems

Work through the practice problems below, but do not submit solutions to these problems. Feel free to post questions, comments, and answers in our Discussion Forum.

1. Why are protocols so important to any form of network communication?
2. Why does the OSI Reference Model used a layered stack approach? What are the benefits of using such an architecture?
3. In `udp-server.c`, what would happen if a datagram with 16,384 bytes of data was received? More specifically, what would the server output to the terminal?

   Also, what would the data field of the response datagram sent by the `sendto()` system call contain? Show the bytes exactly.

   And finally, describe what data is lost, if any.
4. What class network is `usps.com`? As a hint, use the `host` or `ping` (or another) terminal command to help determine this.

Graded problems

Complete the problem below and submit via Submitty for a grade. Please do not post or share any answers; all work on this problem is to be your own.

1. Copy the given `udp-server.c` code to a separate `lecex4-q1.c` file.

   Modify this program as follows:

   (a) Add support for an optional command-line argument (i.e., `argv[1]`) that indicates the specific port number to bind the UDP socket to. If `argv[1]` is present and would be a valid port number, use this value as the port number in the `bind()` call. Be sure to perform data marshalling here.

   (b) Set `MAXBUFFER` to 512.

   (c) Change the application protocol to count the number of `'G'` characters in the received data. Send this 4-byte `int` value in the response datagram. Be sure to perform data marshalling such that this value is correctly interpreted by the remote endpoint.

Additional hint

In your server code, output to `stdout` is block-buffered when run on Submitty. To ensure your server produces output that you can view, add `setvbuf()` as the first line of your `main()` code as follows:

```
setvbuf( stdout, NULL, _IONBF, 0 );
```

Note that your server output is **not** graded via Submitty, but this might help you debug any problems you face.