Cpr E 489: Computer Networking and Data Communications
Lab Experiment #2: TCP Sockets Programming
(100 points in total)

Objective

To use TCP sockets to write client and server programs to implement the `ruptime` UNIX command.

Pre-Lab

Investigate approaches on how to capture the output from `ruptime` within your server program. Include two of these approaches in your lab report.

Lab Expectations

Work through the lab and let the TA know if you have any questions. **Demonstrate your program to the TA after you have completed it**. After the lab, write up a lab report. Be sure to

1) summarize what you learned in a few paragraphs. (**20 points**)
2) include your **two approaches** for capturing uptime output from the pre-lab. (**10 points**)
3) include your **well-commented** code and demo your code to the TA
   a) correctly implement the TCP socket. (**50 points**)
   b) correctly parse the output of the uptime command. (**20 points**)
4) specify the effort levels of each group member (totaling to 100%)

Your lab report is due at the beginning of the next lab.

Problem Description

From the `ruptime` man page on a UNIX system https://linux.die.net/man/1/ruptime:

> `ruptime` gives a status line like uptime for each machine on the local network; these are formed from packets broadcast by each host on the network once a minute.

In this lab experiment, you are required to implement a simple version of the `ruptime` UNIX command by **writing two programs: a client** and **a server.**

- The server program (called `ruptimeServer`) will be demonstrated on two machines (the localhost and another computer in the lab) by typing this command on each of the two machines (this means the `ruptimeServer` program must be copied to both computers and run from each of them)

  ```
  $ ruptimeServer
  ```

- The client (called `ruptimeClient`) will be run from one or both of the machines with two IP addresses as parameters (the localhost's IP and the server's IP) and will connect to the copy of your server running on those machines. Use the command

  ```
  $ ruptimeClient 127.0.0.1 192.168.254.X
  ```

  (where X is dependent upon the machine that is not the localhost in this case).

- When the server is contacted by the client, it will execute the UNIX shell command uptime and send the information returned by uptime to the client.

- The client will display a server's address followed by the uptime information received from that server. For example, if one of the machines contacted was 192.168.254.2, the output from `ruptimeClient` for that machine could be:

```
192.168.254.2: 10:47am up 27 day(s), 50 mins, 1 user, load average: 0.18, 0.26, 0.20
```

- After printing the uptime information for the two servers, the client then quits. The servers remain open and wait for further connections.

Procedure

- Write the two programs **ruptimeServer.c** and **ruptimeClient.c** in C under Linux. Make sure the code is well commented, and don't forget to do error-checking.
- The two programs communicate using TCP sockets.
- The `ruptimeServer` program should listen on a TCP port with a number chosen randomly between 1024 and 65535 (see the notes below for hints on how to choose the port number). This parameter can be passed to the server program through an argument if you wish.
- Test your programs by running the `ruptimeClient` and inspecting the output. Repeat several times in order to make sure that your programs are working properly.
- **Demonstrate your programs to the lab TA.**
- **Submit a copy of ruptimeServer.c and ruptimeClient.c to the TA with your lab report.**

Notes

- The localhost has the IP address of 127.0.0.1.
- In order to avoid contention and confusion, use a server port number that is equal to the rightmost five digits in your student number. For example, if your Student ID number is 123-45-6789, then the port number should be 56789. If this number is less than 1023, greater than 65535, or equal to any reserved port number (see the /etc/services file) you may use any random port number.
- If there is any missing information, you may make any reasonable assumptions, but clearly state these assumptions in your solution.

How to Write and Run Your Networking Programs

All programs will be written in C, under the Linux environment.

You should follow these steps:

- Edit your program using any of the editors available under UNIX (e.g., `pico`, `vi`, `emacs`, etc.).
- Compile your program using `gcc`.
- For programs using sockets, use the following command to compile:

  ```
  gcc –o file file.c
  ```

  where file.c is your code, and file is the required executable file. Note that you can link to other libraries as needed, such as the math library using -lm. Also, please note that under other versions of UNIX besides Linux (e.g., Solaris), you need the "`-lsocket`" and "`-lnsl`" options for socket programs. (This shouldn't be an issue with the computers in Coover 2061.)
- You may run your program by typing the full path to your compiled executable

  ```
  ./file
  ```