

---

**Iowa State University**  
**Department of Electrical and Computer Engineering**  
**CPR E 4890: Computer Networking and Data Communications**  
**Lab Experiment #2: TCP Socket Programming**  
**(Total Points: 100)**

## Objective

- To understand the basic concepts of TCP socket programming
- To learn how to create a TCP socket and establish a connection between a client and a server
- To learn how to send and receive data using a TCP socket

## Pre-Lab

- Read SocketProgramming.pdf
- Investigate approaches on how to capture the output of command-line commands (e.g. uptime) 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** in your .zip code and **demo** your code to the TA.
  - a) correct implementation of the TCP socket. **(50 points)**
  - b) correctly parse the output of the `uptime` command. **(20 points)**

Reports are due **before** the start of the next lab section.

## Demonstration Policy

This lab will require you to **demo** your code to the TA. Review the lab manual carefully to ensure you have completed all the requirements of the demonstration. Labs may be demonstrated:

- At any time during the lab section the demonstration was assigned
- During TA office hours
- During the first **15 minutes** of the following lab (everyone will have a chance to demo **once**)

## Problem Description

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.**

From the `ruptime` man page on a UNIX system:

`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.

- You are required to execute and demonstrate the server program (called `ruptimeServer`) on one of the Coover 2061 machines by typing

```
$ ./ruptimeServer 192.168.254.X
```

- You are required to execute and demonstrate the client (called `ruptimeClient`) from one of the Coover 2061 machines (you may use the machine that your `ruptimeServer` is running on) and will

---

connect to the copy of your server running on this machine. Use the command:

```
$ ./ruptimeClient 192.168.254.Y
```

Note that X and Y above are dependent upon the machine. Since this lab will be performed individually, you may use 127.0.0.1 to execute your programs on the localhost.

- When the server is contacted by the client, the server will execute the UNIX shell command `uptime` and send the system 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 server, the client then quits. The server remains open and waits for further connections.

## Procedure

- Write the two programs **ruptimeServer.c** and **ruptimeClient.c** in C under Linux using your favorite text editor (pico, vi, emacs, etc.) Make sure the code is well commented, and don't forget to do error-checking.
- The two programs communicate using TCP sockets.
- Compile your programs using `gcc`.
  - Example usage: `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`.
  - You may run your program by typing the full path to your compiled executable. Example usage: `./file arguments`
- The `ruptimeServer` program should listen on a TCP port with a number chosen between 1024 and 65535. 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.
- At a minimum, your server should display uptime with hours, minutes, and seconds.
- If there is any missing information, you may make any reasonable assumptions, but clearly state these assumptions in your solution.