CptS/EE 455 – Computer Networks
Spring 2002
Project 1 – Basic Socket Programming
Assigned: Monday, January 28, 2002
**Due Date: February 11, Monday, in class**
**10 Point Penalty per Day Late**
**NO SUBMISSIONS AFTER February 13**

## Project Description

The purpose of this project is for you to learn the basics of the TCP/IP socket-programming interface and to experience some of the issues that arise in programming network applications.

You will be modifying implementations of a TCP echo client and server in accordance with the specification given below.

This is an individual project: each student must do his or her own programming. You are encouraged to talk to other students about the project, either to ask for help or to offer assistance, but **you may NOT share code, either in machine-readable form or by showing your code to another student.**

The code for this project is to be written in the C language and is to run on the Linux machines in the SNIF Lab, ETRL-301. (The remaining projects for this class will not carry this restriction, but for this first project it is important to encounter low-level details that would be hidden in some other environments.)

## Project Specification

I have provided the source code for the TCP echo client and server that we will discuss in class. The provided server merely echoes messages sent by a client. The provided client sends a single message to a server and then exits. You will be changing both of these behaviors.

- The server will prefix its response to the client with two binary numbers corresponding to the fields `tv_sec` and `tv_usec` of a `struct timeval` as filled in by a call to `gettimeofday()` made just before the server sends its response. The binary numbers are to be in network byte order.
- The client must take the **name** of the server computer as a command line argument instead of taking the **IP address** of the server.
- Instead of sending a single message to the server, the client will **repeatedly** read a line from its standard input and send it to the server then read the response. Upon receiving the response the client will use the two numbers at the beginning of the response to calculate an estimate of how long it took for the message to reach the server and it will calculate how long the "round trip" took. It will print these two numbers on a line by themselves, and then on a new line will print the echoed characters from the server. It will then go on to read the next line of input. The client may be exited with CTRL-C or by end-of-file on standard input.

- Note: use a call to `gettimeofday()` just before the `send()` in the client and another just after the `receive()` as the basis for the calculation of the round trip time (RTT). Your code must be prepared for two counter-intuitive cases: (1) the estimated time for the trip from the client to the server may be negative(!); (2) the estimated time for the trip from the client to the server may be larger than the total round trip time. You should be prepared to explain why these situations can occur.

## Lab Environment – ETRL 301

ETRL 301, the SNIF lab, is the lab provided for this course. The lab contains more than 15 machines running Linux. The lab is shared with several other courses this semester, so you will be well-advised to start on the project early. To gain entry to ETRL-301 you will need a door access code, which you can get from the EECS Systems Staff in Sloan 358.

You log into the NIF lab machines with your EECS account name and password (again, if you don't have an EECS account, obtain one from the Systems Staff).

In addition to the machines in ETRL 301, a set of equivalent machines is available for remote access. You will need a Secure Shell (ssh) client in order to use them. These machines are `nif-c{4,26,27,28,29,30,31,32,34,35,36,37}`. Apart from the fact that you can't sit in front of them, they are otherwise the same as the machines in the lab.

In addition to your EECS home directory, you will have access to a class-specific directory in the lab for turning in your project. This is `/net/cs455/$USER`. You may use space in this directory as you wish, but specific subdirectories are required for turning in your work. See **What To Submit**, below.

Lab etiquette: *please* help keep the lab tidy, both in the physical space and on the computers. For the physical space, this mostly means throwing away your trash – don't leave papers lying about: take them with you or throw them away. For the computers this means don't leave processes running when you are not working on them. It is very easy to make the mistake of leaving a server process running when you are finished testing your client. Please make sure you kill them. It is *particularly* rude to leave a process running that is using lots of memory or CPU time. Form a habit of using the `ps` command to find your processes and the `kill` command to get rid of them.

## Starting files

Files to get you started are located in `/net/cs455/assignments/proj1/project1.tar`. Copy this file to your working directory and run

```
% tar xvf project1.tar
```

to get to the files. A `Makefile` is provided to compile and link the programs.

## Test session

You must test your client and server and turn in the results of that testing. The following shows what that test my look like. In what follows, you type something at each "Line to be echoed: " prompt.

```
nif-c16% ./TCPEchoServer 23456 &

nif-c17% ./TCPEchoClient nif-c16 23456
Line to be echoed: GO COUGS!
0.0012 0.0024
GO COUGS!
Line to be echoed: WHAT'S HAPPENING HERE?
-2.0345 0.0030
WHAT'S HAPPENING HERE?
Line to be echoed: ANOTHER ODD CASE
1.0752 0.0020
ANOTHER ODD CASE
```

The port number used by your server must not be used by another student or a well-known service. If you get a "port in use" error you'll need to pick a different port number. I suggest using the last 4 digits of your phone number + 2000 as the port number. Except for those of you who are roommates that should avoid most problems. If you have repeated conflicts with another student's program please get together with them and decide who will use what port.

Use the Unix `script` command to capture the interactive session in which you test your programs. (Read the `man` page for `script`).

## Submitting Your Work

When you have completed the assignment put the following files in `/net/cs455/$USER/project1 (files submitted after class time on Monday, Feb. 11 will be considered` **`late.`**

- Source for the client
- Source for the server
- `Makefile`
  Typing the command `make` in the directory must generate all the required executables
- The file produced by `script` during the session in which you do your final program testing.
- `README`
  A file containing brief instructions on how to compile, run and test your program
- `COMMENTS`

A file describing your experience with the project. What surprises did you encounter? Did something not work the way you expected? In this file you must do your best to explain why your code must be ready for the two counter-intuitive situations mentioned in the project specification. Did either of these occur during your testing? Finally, please tell me how many hours you spent on this project.

ALSO: hand in a printout of the source, script output, Makefile, README and COMMENTS during class on Feb. 11th. Please make sure that your source code indenting has survived the printing process: often the interpretation of TAB characters is different in printing and viewing. The difference can render the printout quite difficult to interpret.

## Grading

The TAs will test your programs on the SNIF-lab machines using the directions you provide in your `README` file. Grades will consider the following components:

- Client and server modifications performed correctly. 70 points
- `COMMENTS` file: the file should **thoughtfully** address the questions mentioned above and your experience with this project. 20 points
- `README` and `script` output: these files must be present and correct. 10 points
- If your code does not compile and run (e.g. compilation errors, core dump, etc.) the maximum credit for the project will be 50 points.

## Learning on your own

There are several system and library calls in both the client and server that you will not be modifying for this assignment. *Nevertheless*, I expect that you will study all of them in order to learn the role of each and every parameter: what kind of values can that parameter take on and what is the effect of different parameter values on the behavior of the system call. All system calls and library functions have `man` pages. Read them for at least first-order understanding.