

Assignment 6

This assignment is for your practice, you do not need to submit anything. However, knowledge about the above will be tested in the lab test/viva.

Reference: Unix Network Programming Vol. 2 by Richard Stevens

You can also search and study from the net, there is more than enough material.

The program `server.c` is basically the same sample TCP concurrent server we gave earlier, but with a `sleep(10)` put in before the send so that we can test the client side code properly. You should start the server, and then immediately start the client (before 10 seconds definitely). You can increase the delay in the server as you wish.

There are 2 sample client programs `client_signal.c` and `client_sockopt.c` for signal driven I/O and getting and setting socket options. All of them can run with the same server program `server.c`.

There are also two sample programs showing raw socket programming: `myping.c` which is a simple ping client that creates the ICMP header explicitly, and `pinger.c`, which is a ping program that creates both ICMP and IP headers explicitly. Make sure you understand both codes and run them (as root). Then modify `myping.c` to do the following:

1. Measure and report the roundtrip delay for each packet.

Then modify `pinger.c` to write a simple traceroute program (to be discussed in class).

Signal-driven I/O

In case of signal-driven I/O, the socket is made enabled for receiving a SIGIO signal, which is sent when certain conditions happen (incl. receive of data). The handler can then do the I/O, the main program can continue to do its own work.

The program `client_signal.c` is a simple client side program showing signal-driven I/O. Understand the code, look up the man pages of the function calls you see. Run it against the server and make sure you understand the output. Try answering the question at the end of the sample program.

Getting and setting socket options

There are several options, whose values can be obtained using the `getsockopt()` call and set using the `setsockopt()` call. The program `client_sockopt.c` shows the use of these two functions with respect to two of the options, the send buffer size and the receive buffer size associated with a socket. First look up the man pages of the two calls, and then understand the program carefully. Then run the program against the server. You will see

the output is not as expected always. Try the variations suggested at the end and try to form an idea as to what Linux TCP code is doing underneath.