CS 740, Fall 1995
Homework Assignment 4

Assigned: Oct. 31
Due: Thurs., Nov. 9, 1:00PM

1. Policy

You may work in a group of up to 3 people in solving the problems for this assignment. However, you must prepare your own writeup, and you must identify your collaborators in your writeup.

2. Logistics

Any clarifications and revisions to the assignment will be posted on Web page `assigns.html` in the class WWW directory.

For this assignment, you will want to retrieve the files in the directory

`/afs/cs.cmu.edu/academic/class/15740-f95/public/asst/asst4`

You only need to hand in a hard copy document for this assignment. Formatted text is preferred to hand written. If some question arises during grading regarding the correctness of your code, you may be asked to submit an electronic version.

3. Introduction

In this assignment you will write several short programs for transmitting messages between workstations using the UDP and TCP protocols. The goal will be to learn the basics of Unix network programming, and to assess the speed at which data can be transfered across an Ethernet, or across a series of Ethernets.

To help you get started, we have written four programs, `udpserver.c`, `udpclient.c`, `tcpserver.c`, and `tcpclient.c`. These programs work as follows. First execute one of the server programs, either `udpserver` or `tcpserver`. The server will then wait for a message from the corresponding client program, either `udpclient` or `tcpclient`. Next, execute the client program from either the same workstation that the server is running on or from another workstation. When you enter the client program, it will first ask you for the internet address of the server. It will then ask you to type in the message that you want to send to the server. If all goes well, the message will print out on the server's screen, and both the client and server programs will terminate. (Note: the message ends at the first space, tab, or carriage return.)

You can learn more about the UDP and TCP protocols, and about Unix network programming, from the following two books,

1. D. E. Comer, *Internetworking with TCP/IP Vol I: Principles, Protocols, and Architecture*, Prentice Hall, Englewood Cliffs, NJ, 1991.

2. W. R. Stevens, *UNIX Network Programming*, Prentice Hall, Englewood Cliffs, NJ, 1990.

Both are well written. You might be able to complete this assignment without opening either book. If, however, you have questions about the network programming functions used by the four programs, turn to Chapter 6 of the book by Stevens.

4. Your task

Your task is to modify the four programs so that you can transmit large amounts of data between the client and server. You should also add instrumentation to the programs so that you can measure both the speed and accuracy with which the data is transferred. You may find the files `cycle-time.c` and `cycle-time.h`, which you used in previous assignments, useful for measuring time.

You are to perform a series of experiments. For both the UDP and TCP protocols, you are to measure the rate and acurracy at which data can be transferred

1. when the client and server are running on the same workstation,
2. when the client and server are running on different workstations, but on the same Ethernet,
3. and when the client and server are running on different workstations on different Ethernets.

For each of these experiments, you should try sending 1 byte of data, 1000 bytes of data, and 1,000,000 bytes of data.

Because your experiments might interfere with the ability of other users to carry out their normal work, you should not attempt to transfer large amounts of data until you are reasonably certain that your programs are working correctly. Also, please conduct your experiments when the system is not being heavily used.

Try to make your programs transfer data is quickly as possible.

5. The UDP and TCP protocols

The User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) belong to the TCP/IP suite of protocols. (IP stands for Internet Protocol.)

UDP is a lower level protocol than TCP. The UDP protocol allows one workstation to send messages to another with little overhead. On the Unix operating system, the receiver first calls the `socket` and `bind` functions to set up a socket on which messages may be received. It then calls the `recvfrom` function to wait for a message to arrive. The sender also uses the `socket` and `bind` functions to set up a socket. It then calls the `sendto` function to send a message to the receiver. UDP makes no guarantee that the message will be successfully delivered, and no acknowledgement is sent back to the sender. (Messages should not be dropped, however, without good cause.)

Unlike UDP, TCP guarantees correct, in order, delivery of all data. TCP is a connection-oriented protocol. To set up a TCP connection, the server first calls the `socket` and `bind` functions to set up a socket, then calls the `listen` and `accept` functions to listen for, and accept, any requests

for connections from clients. To initiate a connection, the client first uses the `socket` function to set up a socket, and then uses the `connect` function to request a connection to the server. Once a connection has been established, the server and client may exchange data using the `read` and `write` functions. The TCP protocol uses acknowledgements, buffering, and reordering to ensure that all data is received, and in the right order. The `read` and `write` functions take care of the details of implementing TCP for you.

You should be able to figure out how to use these functions by examining the code and the comments in the four programs that we have provided. You can find more information about these functions in Chapter 6 of the book by Stevens.

6. Hints

To debug your program, start with the server and client running on the same workstation.

The TCP protocol is supposed to be error free, i.e., all of the data should arrive, and in the right order. If you are detecting the loss or corruption of data when using the TCP protocol, something may be wrong with your program.

Use the TCP server with the TCP client, and the UDP server with the UDP client.

In order to detect the loss, corruption, or reordering of data, have the client send data with a regular pattern to the server. For these experiments, it's o.k. for the server to know the pattern in advance.

To improve the speed of your programs, transfer data in large blocks, not one byte at a time.

Watch out for the difference between network byte order and host byte order. For TCP/IP and other network protcols such as XNS and SNA, network byte order is big endian, i.e., in a 16 or 32-bit word, the byte with the lowest address is the high order byte. On some host machines, including Intel 80x86 machines and DEC VAX machines, the byte ordering is little endian. Some unix functions return results in host order (e.g, `gethostid`) while others expect their arguments to be in network order (e.g., `bind`). You'll see that we have performed conversions between network order and host order in several places.

7. What to hand in

Hand in your four programs, along with a writeup of your experiments. For each experiment, indicate the rate at which the data was transferred, and the fraction of data that was successfully delivered. For the experiments in which messages are sent between different workstations, indicate the number of repeaters, bridges, and gateways through which the data passes. (You will probably have to consult a network map to find this information. You are not expected to deduce it from your experiments.)