CSCI 473: Network Programming
Exam #2 -- April 29

Write your answers on your own paper. This exam is to be turned in by 12:10 PM. For each question, I have given, in addition to the number of points, a suggested number of minutes that you should allocate to the question.

Problem 1. (40 points) -- [25 minutes, 11:00 AM to 11:25 AM]

Suppose you are given 1000 IBM PC's each with a 100 megabyte hard disk and an Ethernet interface card along with seven miles of Ethernet cable. You've been asked to connect these machines so that all pairs of machines can communicate using TCP/IP. You've also been given a software implementation of TCP/IP for IBM PC's.

Subproblem 1A.
First, describe your network layout. Outline your strategy for assigning Ethernet numbers and IP numbers to the machines.

Subproblem 1B.
In your network layout, how many machines may transmit UDP packets simultaneously?

Subproblem 1C.
How many machines may have open TCP connections simultaneously?

Now suppose you are given three *brouters* in addition to the all the PC's. A brouter is special device that can serve as either an Ethernet bridge or an IP router.

Subproblems 1D, 1E, and 1F.
Answer subproblems 1A, 1B, and 1C using your new resources.

Problem 2. (10 points) -- [5 minutes, 11:25 AM to 11:30 AM]

Larry Sna says that Unix socket programming is a pain, because you have to know the machine and port numbers of remote processes. Write a paragraph or two to educate him!

Problem 3. (35 points) -- [20 minutes, 11:30 AM to 11:50 AM]

Here's a useful server-client application for you. Five philosophers are sitting at Sun workstations. Their machines will be the clients. A sixth machine will be the server. Every minute throughout the day, the server will randomly choose two philosophers and send a thought provoking question to those two. The philosopher who replies first wins that round. The philosopher who wins the most rounds during the day is declared the sagest.

Describe how the Unix system calls and socket interface can be used to implement this application. Don't go into a lot of detail (*no C code allowed!*). Just say enough to convince me that you could program the sagest philosopher contest. By the way, keep in mind that some philosophers might attempt to cheat by typing in a answer before they see the question.

Problem 4. (15 points) -- [10 minutes, 11:50 AM to 12:00 NOON]

The pseudo-terminal interface in Unix provides a mechanism for ``fooling'' a process into thinking it is connected to a physical terminal. In standard Unix distributions, all master pseudo-terminals are protected so that any user can read or write to them; however, the kernel does not allow a master pseudo-terminal to be opened more than once simultaneously. For example, if two processes attempt to open **/dev/ptyp5** at nearly the same time, the first will succeed and the second will fail due to an ``I/O error.''

Subproblem 4A:
It's a general rule that no Unix file or device should be universally writable, yet an exception is made for master pseudo-terminals. Why is it necessary to leave the master pseudo-terminals so unprotected?

Subproblem 4B:
What mischief could you perform if you were able to open and write to master pseudo-terminals being used by other processes?