CS 422 (Park) **Assignment IV** **Due: Oct. 27 (Wed.), 1999**

PROBLEM 1 (20 pts)

(a) What are the drawbacks of the current 32-bit IP address structure? What optimizations/refinements have been carried out on the 32 bits to make better use of the total address space? Assuming there are 60 million hosts on the Internet needing, say, one IP address each (some are, of course, multihomed), is the 32-bit address space, in principle, sufficient? What if you consider the class A, B, ... classification?

(b) Assume you have 128 bits to use to identify or assign names to hosts (i.e., interfaces) on the IP internetwork. With a suitable routing mechanism, every host should be efficiently found and without ambiguity. Propose your own design, how it is to function (i.e., routing mechanism at hosts connected to two or more LANs), and justify why you think it is a "good" design. First, it has to be correct design, and second, it should be efficient and "nice" in other senses that you will explain.

PROBLEM 2 (15 pts)

Assume our goal is to turn a CSMA/CD Ethernet into a token ring. That is, while preserving Ethernet's existing MAC, we want to build a system—in software—that emulates a token ring-based access mechanism with its associated deterministic properties. How would you go about doing that? Give a detailed sketch of a design and explain how it is able to function as a token ring.

Assuming that you are able to keep your software overhead to a negligible level, under what circumstances, if any, would your system perform better than the pure CSMA/CD Ethernet with respect to throughput?

PROBLEM 3 (40 pts)

As a continuation of Problem 4, Assignment III, extend the client/server application such that client and server processes are allowed to run on different hosts and communication is achieved via socket calls using UDP/IP. Thus the FIFO IPC structure needs to be replaced by the appropriate networking calls. Note that UDP is a connectionless (i.e., stateless), unreliable transport protocol. Test your application using the two laptops as hosts where the server and client processes are run, respectively. Assume the server listens on a well-known port, say 10101. The client should accept three command-line arguments

% `client.bin` *IP-address timeout command*

where *IP-address* is in dotted decimal form. To prevent indefinite waiting from occuring when packets are lost or the server is down, the client, upon making a request to the server, should set an alarm to go off in *timeout* msec as in the extension carried out in the previous assignment. When SIGALRM is caught, the client should resend the request with the entire procedure repeating at most 5 times.

On your host machine (one of the laptops), open two windows, one for each separate host, running the server on one machine and the client on the other. Test the system by executing the client with the requested commands `date`, `who am i`, `ps`, `ps -l`, `ps -l -a`, `ls`, `ls -a`, `ls -a -l -i`, `terve` (e.g., % `client.bin` *IP-address* 500 `ps -l`). In addition to the code and script files, hand in a screen dump which shows the contents of the two windows.

PROBLEM 4 (25 pts)

Use the `snoopwrap` utility for capturing 3 packets that pass through the wire. The command to be executed is as follows:

% `/usr/local/etc/sudo /usr/local/etc/snoopwrap-elxl0 -c3 -o /var/tmp/loginid`

To generate network traffic, use `ping` or `telnet` on the island network (`snoopwrap` works only on the private LAN). Write a program which will read the log file and print the values of the various fields, in readable format, for the Ethernet frame and the IP datagram headers, respectively. For example, *preamble* = , *src-address* = , ... , *length/type* = for Ethernet, and *version* = , *header-length* = , *TOS* = (the bit pattern), ... , *dest-address* = for IP. As a case in point, the IP source and destination addresses should be printed in dotted decimal format and the corresponding Ethernet addresses should be printed in hexadecimals separated by colons (i.e., `ff:ff:ff:ff:ff:ff`). The type-of-service field and flags field in the IP header should be output as binary strings so as to discern their bit patterns.

Verify these values of MAC and IP addresses using the `ifconfig` command against their true addresses. For each of the fields (except for the *preamble* field in the Ethernet frame header), give an interpretation of the value found. E.g., the *version number* in the IP header might read 4 as in IPv4. Do the same for the *TTL, TOS* and all other fields. Do any of the values in the fields across the three network packets change? Explain. Submit the hard copy of your output, and use `turnin` for your code.