CS 536 (Park) **Assignment II** **Due: Oct. 18 (Thu.), 2001**

PROBLEM 1 (80 pts)

Suppose you are hired as the CTO of a large company with tens of thousands of employees, and offices worldwide. Your company has its own global network, where every local area network is comprised of PCs connected to Ethernet switches. The Ethernet switches, in turn, are connected to IP routers. Assume the owner of the company has instructed you to get rid of all IP routers in your network (perhaps a not so brilliant idea), while still enabling communication between any two PCs connected to Ethernet switches. Since you are guaranteed a $100M bonus upon successful completion of the project, you decide to go along.

Assume your Ethernet switch has an internal look-up table that is used to determine which output port an incoming Ethernet frame should take to reach its destination. Thus, the look-up table is a 2-column table where the first column contains 6-byte destination addresses, and the second column contains port numbers. If an incoming frame finds its output port busy—i.e., being used by some other Ethernet frame—then it is discarded, and a jam signal is sent on the input port where the frame arrived to activate the sender's CSMA/CD protocol. Thus, if the sender is a PC with an Ethernet card, then it will trigger CSMA/CD's backoff mechanism. If an Ethernet switch receives a jam signal on any of its input ports, the Ethernet switch will shut down due to a programming bug.

Exploiting the fact that every Ethernet card has a unique 6-byte address, describe an architecture comprised of PCs and Ethernet switches only, that enables communication between all PCs in the corporate network. Explain why your architecture works, how routing is done, and describe what happens to an Ethernet frame as it travels from the source PC to the final destination PC. Indicate what, if any, special entries must be placed in the Ethernet's look-up table. Your design must assure application transparency; that is, all application processes must be able to run and communicate using IP as is. (Note that ARP is the protocol that translates IP addresses into Ethernet addresses; you can assume it works perfectly without wondering how.)

What are the pros/cons of your design vis-à-vis that of the IP router based system it is replacing (try to find all the positive aspects)? What impact, if any, would Ethernet switches directly communicating with each other using CSMA/CD (i.e., not shutting down) have on your final design?

PROBLEM 2 (70 pts)

This is a continuation of Problem 3 (b), Assignment I. Modify the client/server code so that the server and client run on different IP hosts, the server listens on a well-known port (e.g., 11111 or you can choose your own), the client, call it `rrun`, accepts two command-line arguments consisting of an IP address and UNIX command that is to be remotely executed, e.g.,

% `rrun` xinu11.cs.purdue.edu finger

and submits a request to the server, call it `rrund`, running on the IP host, which prints on its terminal the time of day, the sender's IP address, and the command received, and executes the requested command using `execvp`(). The client receives the output from the server and displays it on its terminal. Note that the UNIX command could have its own command-line arguments, for example,

% `rrun` xinu11.cs.purdue.edu ls -l -a

Your implementation should preserve the concurrent server structure of the existing client/server example, while carrying out network communication using TCP (socket with SOCK_STREAM option).

Test your client/server application by running the server on three separate machines, then running your client on a fourth machine with UNIX commands `hostname`, `date`, `whoami`, and `boiler`. Explain what happens when you run the last command. Do a screen dump of what appears on your client's terminal, and submit a hardcopy along with the electronic submission of your code using `turnin`.

What happens when you run

% `rrun` xinu$A$.cs.purdue.edu `rrun` xinu$B$.cs.purdue.edu `rrun` xinu$A$.cs.purdue.edu hostname

where xinu$B$.cs.purdue.edu is the client Xinu machine and xinu$A$.cs.purdue.edu is the server machine of your own choosing? Explain your findings.

PROBLEM 3 (70 pts)

Design an application layer "router" that performs a tunneling service, call it `tunnel_route`, as follows. It listens on a well-known port (e.g., 22222 or you can choose your own) for UDP packets. When it is asynchronously notified of the arrival of one or more packets via the SIGPOLL signal, its SIGPOLL signal handler dequeues a UDP packet, inspects the 3rd, 4th, 5th and 6th bytes of the payload, interpretes them as a destination IP address, formats a UDP packet with the new destination address, the old/original source IP address, and a modified payload where the first two bytes remain the same, the 3rd, 4th, 5th and 6th bytes are deleted, and the rest of the payload are shifted to the front so that they start from the 3rd byte position. The packet is sent off, and a message is printed on the terminal containing the time of day, sequence number of the packet—the first two bytes of the payload are interpreted as a sequence number—and the source address and new destination IP address. Before returning from the signal handler, it should check that there are no further queued UDP packets. If there are, they should be processed as before.

The sender, call it, `snake`, will take several command line arguments

% `snake` *timer volume IP-addr-1 IP-addr-2* $\cdots$ *IP-addr-n*

where *timer* is a time interval denoted in milliseconds, *volume* denotes the number of packets to send (maximum of 64000), and *IP-addr-i* is a sequence of IP addresses to be written into the UDP payload (a maximum of 40, i.e., $n \leq 40$). The sender uses SIGALRM to wake up every *timer* milliseconds, at which point it formats a UDP packet with its source IP address and *IP-addr-1* as the destination IP address, and the well-know port number. The first two bytes of the payload will contain a counter value (starting at 0, and incremented after every send operation), followed by 4-byte blocks containing *IP-addr-2*, *IP-addr-3*, and so forth. The sender will terminate after *volume* number of packets have been sent.

The receiver, call it `target`, will listen for UDP packets on the well-known port, and upon receipt of a UDP packet print the time of day, sequence number (first 2 bytes of payload), source IP address and destination IP address.

Perform benchmark runs with 5 IP addresses (choose 5 Xinu machines), for *volume* 25, and *timer* values in the range 1000, 300, 50, 10, 5, 1. For each *timer* run, do a screen dump of all 7 machines and submit their hardcopies. As *timer* decreases, what do you observe? Explain your findings.