CS 422 (Park) **Assignment IV** **Due: Mar. 28 (Wed.), 2007**

Submission instructions: Please type your answers and submit electronic copies using `turnin` *by 5pm on the due date. You may use any number of word processing software (e.g., Framemaker, Word, LaTeX), but the final output must be in pdf or ps format that uses standard fonts (a practical test is to check if the pdf/ps file prints on a CS Department printer without problem). For experiments and programming assignments that involve output to terminal, please use* `script` *to record the output and submit the output file. Use* `gnuplot` *to plot graphs. Use* `ps2gif` *to convert a eps/ps plot to gif format (e.g., for inclusion in Word) if there is a need.*

PROBLEM 1

Read Chapters 8, 9, and 10 from Comer.

PROBLEM 2 (40 pts)

The purpose of this problem is to introduce "sniffing" raw data from the "wire" (more precisely Ethernet interface of sender/receiver PCs) on the machines in the lab. Execute the following command on sender and receiver PCs to capture Ethernet frames:

% `sudo /usr/local/etc/tcpdumpwrap-eth1 -c18 -wEX`

The command will capture 18 packets from the Ethernet LAN and save them in the file `/var/tmp/`*login-EX*, where *login* is your login ID. Generate your own traffic on the private network by doing `ping` from sender machine to receiver machine (`ping -c count receiver-IP-addr`) where `count` is the number of packets transmitted. What is the value of `count` that you need to set so that `tcpdumpwrap` captures 18 packets? Submit the log file in hexadecimal format. Also, save your `ping` terminal interaction using `script` and submit it along with the hexadecimal `tcpdumpwrap` log file.

Using the Ethernet header format(s) discussed in class, decode from the hexadecimal dump what the values for the fields in the Ethernet header are. Compare these values across the 18 captured frames. Use `/sbin/ifconfig` to compare the Ethernet addresses that you have captured with those printed out by `ifconfig`. Compare the frames captured at the transmitter with those captured at the receiver. From the value of the type/length field determine whether the Ethernet frames are DIX- or IEEE 802.3-compliant. Explain how you are able to distinguish between the two. What is the default payload size of `ping`? How long is the payload of the Ethernet frame? Are they consistent? (The default IP header size is 20 bytes.) Noting that the first four bits of an IP header indicate its version number (4 or 6), locate the version number bits in the Ethernet payload and identify the IP version running on your test machine. Noting that the last 8 bytes of a 20-byte IP header represent the 4-byte IP source address and 4-byte IP destination address, respectively, locate the IP source/destination address bits in the payload and compare their values to those output by `ifconfig`. What is the content of the payload generated by `ping`?

PROBLEM 3 (40 pts)

Reimplement the client/server network application in Problem 3, Assignment III, such that UDP (SOCK_DGRAM option in `socket`()) is used in place of TCP. Unlike TCP, UDP does not retransmit lost packets, its main function being adding port numbers and IP addresses to payload. Although UDP can be used with `write`() and `read`() systems calls, please use the UDP-specific system calls `sendto`() and `recvfrom`() in your client/server code. Keep in mind that, although unlikely (due to on average low traffic), UDP packets may go missing in the lab's Ethernet LAN. Perform the same benchmarks as in Problem 3, Assignment III, and submit the resultant output.

PROBLEM 4 (20 pts)

As a continuation of Problem 2, run `tcpdumpwrap-eth1` at the receiver-side (i.e., client-side) of the client/server network application (before running the TCP-based client/server application) so that the request and response packets are captured. In the hexadecimal dump, identify the request and response packets by their IP and Ethernet

addresses (source and destination). Show that from the captured payload, because packets are not encrypted, you can read off the content of the request and response packets (which could have carried passwords or other confidential information). Given that the first 16 bits of the TCP header represent the sender process's port number and the second 16 bits of the UDP header represent the receiver process's port number—keep in mind that the UDP header is part of the payload of the IP packet—show that the source port number of the response packet and the destination port number of request packet as seen in the hexa dump concur with the PID-to-port number mapping output by the `netstat` utility with option `--programs`.

PROBLEM 5 (40 pts)

(a) Implement the original concurrent client/server application in Problem 4, Assignment II, as a TCP-based client/server network application. Use the same test set as in Problem 4, Assignment II, and submit the client- and server-side `script` logs that show the workings of the application.

(b) Modify part (a) such that the server process does not block on `waitpid`() but calls `waitpid`() within a signal handler, call it `wait_for_child`(), that gets invoked by the kernel asynchronously when the SIGCHLD signal (indicating that a child process belonging to this process has terminated) is raised. Register your signal handler with the kernel at the beginning of the server code using the `sigaction`() system call. Do not use the `signal`() system call to do the registration. Perform the same tests as in (a).