CS 536 (Park) **Assignment III** **Due: Oct. 12 (Tue.), 2004**

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).*

PROBLEM 1 (30 pts)

Read Sections 2.6 and 2.7 from P & D. Solve Problems 6, 13, 15, and 19 from Chapter 1, and Problems 19 and 25 from Chapter 2.

PROBLEM 2 (30 pts)

In this problem, you will monitor/log Ethernet frames in the Xinu lab by running a sniffing program in promiscuous mode as super-user. You will then inspect the captured frames and determine their structure and content. Execute the following command on a machine in the Xinu lab which captures Ethernet frames from the LAN:

```
% /usr/local/etc/sudo /usr/local/etc/snoopwrap-elxl0 -c10 -o /var/tmp/login-ID
```

The command will capture 10 packets and save them in the file `/var/tmp/login-ID` where `login-ID` is your login ID. After starting the sniffing program, generate your own traffic by running `scp` between two Xinu machines, distinct from the monitoring machine, that transfers a file of 2600 bytes containing the periodically repeating ASCII character sequence

$$0123456789012345678901234567890123456789\cdots$$

Submit the log file in hexadecimal format. Inspect the Ethernet headers of the 10 frames and determine the values of the header fields. What format, DIX or IEEE 802.3, do the captured frames follow? Which of the captured frames are part of the `scp` protocol that you ran? (Use `ifconfig` to check the Ethernet addresses of the `scp` sender/receiver machines.) Which of the captured frames carry the actual data (2600 bytes)? Explain the steps in your deductive "detective work." Is it possible to read off the data bytes $01234567890123456789\cdots$ from the captured payloads? 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 system. 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`. Pick the smallest Ethernet frame and check if the CRC-32 polynomial divides the CRC value.

PROBLEM 3 (40 pts)

(a) Calculate $E[\text{good}]$ in the simplified CSMA/CD utilization analysis. Then calculate the utilization

$$\varrho = E[\text{good}]/(E[\text{good}] + E[\text{bad}])$$

while keeping the time units in slots (not seconds) so that frame size (F), bandwidth (B), and wire length (L) do not come into play. Assuming the probability of grabbing a slot is set to $p = 1/k$ and the number of users k is made "large" (i.e., $k \to \infty$), what is the resulting utilization? What if k is kept finite, say, 10 or 20?

(b) In the simplified CSMA/CD analysis, the probability that a slot is "usefully used" is $\eta = kp(1-p)^{k-1}$. Show that η is maximized at $p = 1/k$. Using `gnuplot`, draw η as a function of p ($0 \le p \le 1$) for $k = 10$ and 20 (in the same plot). Given that η is the probability that a time slot is usefully used, is η not the same as utilization ϱ (i.e., fraction of total bandwidth attained)? Explain.

(c) Suppose there is a way in Ethernet LANs to determine how many hosts k, at any given time, have frames

to send, and p-persistence is instituted in CSMA/CD with $p = 1/k$ so that even though there is data to be sent transmission is attempted only with probability p. Note that when time is slotted, the slot size is fixed, and a frame fits into one slot, CS (carrier sense) is useless since a new grabbing competition is started among the k hosts at the beginning of every slot. In such a baseband multiple access channel, is exponential backoff upon collision still desirable? The answer is no but you need to provide the reasoning.

(d) As a continuation of (c), suppose that one of the k hosts transmits high priority traffic and needs to be given precedence over the other $k-1$ hosts. The low priority $k-1$ hosts are to continue sharing slots equitably amongst themselves. Propose a contention-based MAC that is a modified version of the protocol in (c) such that the high priority host gets 50% of the slots. The protocol must remain decentralized but you may assume that k is known. Discuss the pros/cons of your contention-based protocol compared to a contention-free TDM protocol where slots are allocated to the k users a priori.

PROBLEM 4 (50 pts)

This is a continuation of Problem 5(a) in Assignment II. UDP is a minimalist transport layer protocol running on top of IP that uses 16-bit port numbers to identify the target process at a target IP host. IP addresses, which are 32 bits long, identify a specific NIC on a host (or any other networked device). In the Xinu lab, this means that a 3-level addressing is at play where UDP uses port numbers of identify processes on a Solaris host, IP uses 4-byte addresses to identify the NICs on a host (if multi-homed, the host may have two or more IP addresses, one per NIC), and Ethernet uses 6-byte MAC addresses to identify the hardware address of the NIC. Note that for communication within a LAN, IP addresses, strictly speaking, are superfluous. Since we will not be writing networking applications that directly interface with Ethernet at the driver level—most OSes do not provide a system call interface to directly access link layer protocols in the protocol stack (we will consider dynamically loadable kernel modules and hooks when discussing implementations of IP)—we will, whether we need it or not, use IP when building network applications as a software engineering abstraction, at least for the moment. With that background in mind, modify the FIFO IPC based client/server application such that UDP sockets are used in place of FIFOs. Sockets, as with FIFOs, are one of the 7 file types in UNIX, and as such obey the usual rules of creation (to get a socket/file descriptor), initialization, access, and closure. In our case, we will use `sendto`() and `recvfrom`() system calls to transmit and receive one autonomous packet at a time whose payload, unless too large, will be carried in a single IP packet that is carried in a single Ethernet frame. As neither IP nor UDP perform ARQ, UDP packets may be lost due to buffer overflow above the link layer while in transit. Therefore your UDP implementation of the client/server application should perform ARQ using stop-and-wait with a retransmission timer whose default value is set to 500msec. Since, without artificially injected congestion, it is unlikely that UDP packets will be dropped in the Xinu lab's Ethernet LAN, the modified server should, upon receiving a request, with probability 0.9 ignore the request. Thus, on average, the 10th retransmission of a client request will result in a response. When a response arrives before the timer expires, the timer should be silenced. Perform the same benchmark test as in Assignment II, but now with `date`, `ls`, `ps`, and `hostname` executed on different hosts. There are a total of 5 hosts inclusive the server. Record the interaction/output on the five terminals using `script`. Try to execute the 4 client processes "simultaneously" as in Assignment II.