CS 536 (Park) **Assignment VII** **Due: Oct. 24 (Fri.), 1997**

PROBLEM 1

Read Sections 6.1, 6.2, and 6.5 from P & D.

PROBLEM 2

Implement a "greedy" file transport protocol called `ftp-greedy` which achieves fast (or not-so-fast), reliable file transport on top of UDP using a form of selective ARQ. Given a file F of size $|F|$ bytes to be transferred, `ftp-greedy` segments F into $\lceil |F|/M \rceil$ blocks of size $M < |F|$ (except for possibly the last block which can be of size less than M), sending each block encapsulated in an UDP packet. Before encapulation, each block is assigned a header where a sequence number i identifying the block is enscribed. The sequence number is represented using a single byte, thus giving a range of 0–255. If more than 255 blocks are needed, i mod 255 is used to achieve wrap-around. `ftp-greedy`, on the sender side, is to be used as

% `ftp-greedy` *filename IP-address*

where *filename* is the name of the file to be transferred and *IP-address* is the IP address in dotted decimal form. On the receiver side, `ftp-greedy` is executed without command-line arguments. This puts `ftp-greedy` in "receive mode." When the transfer of a file is complete, `ftp-greedy` (on the receiver side) prints out a message to *stdout* saying that the file in question has been (reliably) received after which it terminates. For example,

% `ftp-greedy`

filename received (X *bytes*)

%

might be the interaction seen on the terminal. For the receiver to know the filename in which to store the received file, the sender, in block 0, sends the filename. To indicate that no more blocks are to come (i.e., the last block i has been sent), the sender sends an empty block with sequence number $i + 1$ mod 255 at the very end.

Two parameters (e.g., hardcoded using #define) influence the sender's behavior, M and S. M is the block size and S is the sender's interpacket waiting time (in msec). That is, S measures the number of milliseconds the sender waits between successive sending of packets (an event triggered by SIGALRM). The smaller S, the more "greedier" `ftp-greedy`.

The receiver uses negative ACKs to achieve reliable communication using retransmission. As soon as the receiver sees a "jump," i.e., a hole in its received packet sequence,

$$\cdots i-2,\ i-1,\ i,\ i+2$$

it transmits a NACK packet to the sender demanding that packet $i+1$ be retransmitted. At the same time, it starts a timer to expire after T msec. If the missing packet is received prior to the timer expiration, the timer is cancelled. If not, a NACK is sent again. Set T initially to 3 msec; each time a NACK is sent again because of timer expiration, increment T by 2 msec. Each time a retransmitted packet arrives before timer expiration, decrement T by 1 msec. T should never be allowed to go below 3 msec or above 15 msec.

The receiver can maintain a buffer size of K packets to hold packets arriving while the missing $i + 1$ packet is being processed. You can maintain the buffer anyway you want, but it can never exceed the K packet bound.

Test your application by creating a $|F| = 100$ KB dummy file containing the ASCII characters '0', '1', '2', ..., '9', '0', '1', ... Measure the file transmission completion time (wall time) at the receiver from the moment that the first packet is received to the moment that the last packet is received. The last packet may be marked by sending a termination packet with empty payload. Perform the measurements for $S = 10, 30, 50, 70, 100, 200$ msec with $M = 1000$ B. Draw the performance curve (completion time) as a function of S. For the S corresponding to the smallest completion time, call it S^*, test your application for $M = 500, 1500, 2500$ B. Draw the performance curve as a function of M. Give a 1-page written interpretation of your results. Hand in the source code, scripts, and your summarized results.

PROBLEM 3

Consider the network shown in Figure 1 where hosts A and B are sending data to host C via gateway (or router) G. Assume the transport layer of the protocol stack at A, B engages in fragmentation whereby its PDU (protocol data unit) or packet of size X is split into $m = X/Y$ network layer PDUs of size Y where $X > Y$ and Y divides X. No further fragmentation is carried out in the layers below.

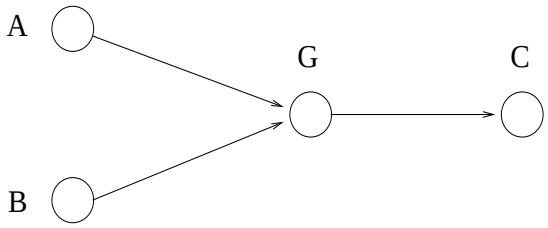

Figure 1: 4-node network system.

Assume the transport protocols at A, B are driven by applications that generate transport layer PDUs with probability p per time unit v; i.e., the PDUs at A, B form a Bernoulli process. If at time t (say at A) a packet is generated, then fragmentation has the effect of generating $m = X/Y$ network layer PDUs that are sent out from A at times $t + \delta, t + 2\delta, \ldots, t + m\delta$ where $\delta = v/m$.

The gateway G has zero buffer capacity and can serve only a single network layer packet per $\delta = v/m$ time unit. Hence, if two packets from A and B simultaneously arrive, one of them must be dropped. Assume A's packet is dropped with probability $\varepsilon < 0.5$ at such instances. That is, A traffic stream has "higher priority." C drops the whole transport layer PDU if even a single network layer PDU fragment is missing. Packets are not retransmitted.

What are the transport layer packet loss rates for A and B? Show your derivations. Plot your packet loss rate solutions as a function of m (for fixed X). Show the graphs for A, B on the same plot. What (if any) assumptions are you making? How does it change if we assume a buffer capacity of 1?