Exercise 3 — Error Detection + Retransmission

Due: Friday, Feb. 27, 1998

This assignment is a combination of problems about error detecting codes (in particular the CRC) and retransmission protocols used at the data link layer.

To prepare for the discussion of retransmission protocols, you should finish reading sections 5.1 through 5.3 of Shay.

I plan to do more analysis of the behavior of these protocols in the presence of errors than you will find in the text. To prepare you for this discussion, I will distribute and ask you to read a handout that discusses some of the basics of probability theory.

1. In class, we looked at the behavior of the a one-bit CRC based on the generator polynomial $x+1$. One bit isn't enough if you are really serious about error detection. So, for this problem I'd like you to think about the options if you decide to use a two bit CRC!

   If you use a two bit CRC there are four possible generator polynomials: x^2, x^2+1, x^2+x, and x^2+x+1. Of these four, which do you feel would result in the code with the best error detection properties? Justify your answer by briefly discussing the sorts of errors each polynomial's CRC code would or would not detect. There may not be one "right" answer to this question. Each code may have different strengths and weaknesses that aren't quite comparable. Therefore, your justification may prove more important than exactly which code you select as "best".

   For the code you do select as "best", please describe directly how each of the two check bits added relate to the bits of the message sent. That is, in the case of $x+1$ we discovered that the check bit was simply a parity bit for all the bits of the message. Are there similar, simple relationships between the two bits produced by the code you have selected and the message bits?

2. Certain variations in network protocols seem certain to improve performance. For example, the technique of "piggybacking" acknowledgements seems certain to improve performance by reducing the number of overhead bits transmitted.

   Shay introduces piggybacking in his discussion of Go-Back-N. The technique, however, could be applied even to the simple "stop and wait" protocol. In stop and wait, acknowledgement packets contain 1-bit worth of content. If this bit is one, the acknowledgement gives the sender permission ot move on to the next frame. Otherwise, the sender will resend the most recently sent frame.

   If this one bit of information is sent in its own packet, it will be accompanied by some sort of preamble, parity bits, addresses and who knows what else. Basically, the ratio of overhead to actual packet content will be rather high. The alternative of adding one extra bit to the format used for data packets so that these acknowledgements could be piggybacked onto data messages seems attractive.

   The following problem is taken from "Data Networks" by Bertsekas and Gallagher.

   > Consider a stop-and-wait system with two way traffic between nodes A and B. Assume that data frames are all of the same length and require D seconds for transmission. Acknowledgement frames require R seconds for transmission and there is a propagation delay P on the link. Assume that A and B both have an unending sequence of packets to send, assume that no transmission errors occur, and assume that the time-out interval at which a node resends a previously transmitted packet is very large. (Note (from Tom): Shay does not even include the notion of a timeout interval in his version of stop and wait. Obviously, Bertsekas and Gallagher's description of stop and wait was

a bit different. This assumption, however, means that timeouts will never occur so this difference in presentation is irrelevant.) Finally, assume that each node sends new data packets and acknowledgements as fast as possible, subject to the rules of the stop-and-wait protocol.

(a) Show that the rate at which packets are transmitted in each direction is $(D+R+2P)^{-1}$. Show that this is true whether or not the starting time between A and B is synchronized.

(b) Assume, in addition, that whenever a node has both an acknowledgement and a data packet to send, the acknowledgement is piggybacked onto the data frame (which still requires D seconds). Assume that node B does not start transmitting data until the instant it receives the first frame from A (i.e. B's first frame contains a piggybacked ACK). Show that all subsequent frames in each direction are data frames with a piggybacked ACK. Show that the rate in each direction is now $(2D+2P)^{-1}$. Note that if $D > R$ this rate is less than that in part (a).

In case you didn't notice, the conclusion of this problems seems to say that, contrary to intuition, piggybacking of ACK's reduces throughput.

3. Do problem number 7 at the end of Chapter 5 (really this time).