Homework 2

Due on Oct 23

Directions: Start early! We will pick 1 to grade out of 4 but you **must** hand in Problems 1,2, and 3 for completion credit. Problems 4 and 5 are only for exam practice; do not hand in.

1. **PPP Byte Stuffing versus HDLC Bit Stuffing**: While HDLC used bit stuffing, a much more commonly used protocol today is called PPP and uses byte stuffing. PPP uses the same flag F that HDLC does (01111110). To prevent the occurrence of the flag in the data, PPP uses an escape byte E (01111101) and a mask byte M (00100000). When a flag byte F is encountered in the data, the sender replaces it with two bytes: first, the escape byte E followed by a second byte, $ExOR(F, M)$. ExOR (A, B) denotes the Exclusive OR of A and B. Similarly, if the data contains an escape byte E, then the sender replaces it with two bytes: E followed by ExOR(E, M). As an example if the data is 01111110 00010001 01111101 the stuffed output data is 01111101 01011110 00010001 01111101 01011101

(7 points) Suppose the sender has the following data to send: 01111101 01111101 01111110 01111110. Show the resulting frame after stuffing and adding flags.

(3 points) Why is byte stuffing easier for software implementations than bit stuffing?

(5 points) In HDLC, assuming all bit patterns are equally likely, there is roughly a 1 in 32 chance that the 5-bit pattern 11111 occurs in random data. This leads to roughly 3% overhead for bit stuffing in random data. Assuming that user data is random and that each byte value is equally likely, what is the chance that byte stuffing will be done in a PPP frame.?

(5 points) What is the *worst* case overhead for HDLC? In other words, pick a sequence of data bits that causes HDLC to add the most stuffed bits and find the overhead. Define overhead as stuffed bits divided by total bits What is the worst case overhead for PPP? In each case, give the data that causes the worst case and also the worst case overhead

-

2**, CRCs Polynomial View**: Consider the 4-bit CRC generator $x^4 + 1$.

- Consider the message 1001. Calculate the CRC for this message using the 4th degree polynomial generator above (3 points)
- The simplest way to create an undetected error is to add an error polynomial equal to generator to the message + CRC. What is the resulting message that is received? (2 points)
- Does this generator detect all 1-bit errors? Why? (Hint: review arguments in Notes) (1 points)
- Does this generator detect all odd bit errors (1 point, see Notes)
- How many undetected burst errors (starting at Offset 0) can there be of burst length 10? To help you do this, note that a burst error of length 10 is a polynomial that starts with x^9 and ends with 1 with optional terms for the remaining powers between 8 and 1. For an undetected error, the resulting burst error must be divisible by the generator $x^4 + 1$. Instead of seeing which bursts are divisible by the generator, find how many bursts of length 11 are multiples of the generator? For example, if we multiply $x^4 + 1$ with $x^5 + 1$ we get a length 10 burst.

1.

Given that the only way to get x^9 as the highest power and 1 as the lowest power is to multiply by polynomials whose highest power is x^5 and lowest power is 1, write down all such polynomials. How many such polynomials are there? (8 points)

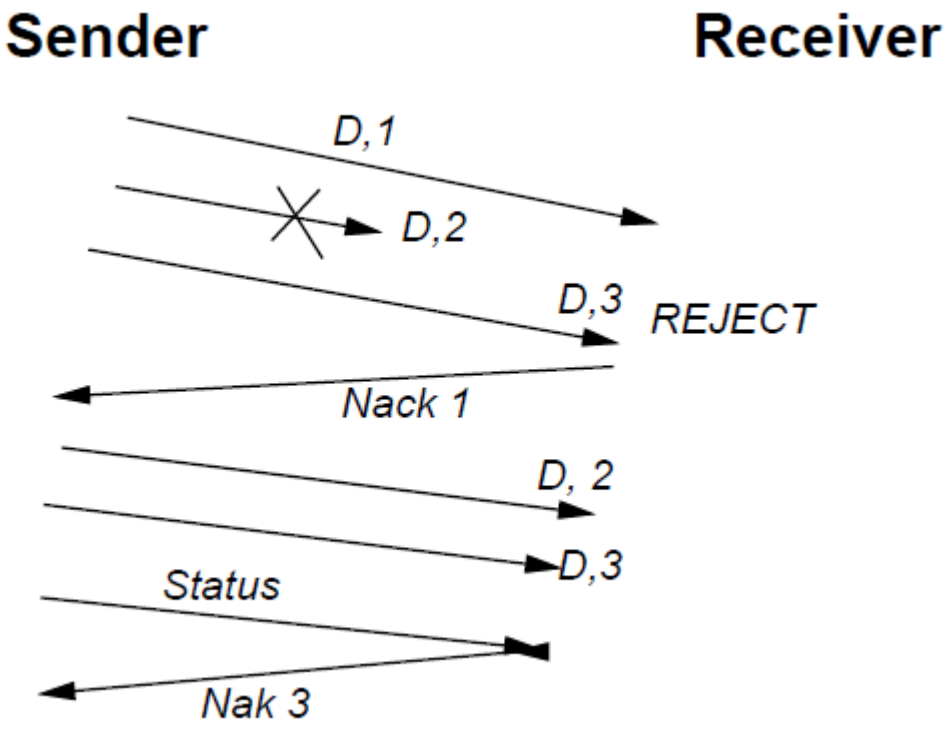
- Based on the last two answers, can you argue briefly why the probability of a detecting a burst of arbitrary length is $1/2^k$ where k is the degree of the CRC (5 in this case). Hint: consider the ratio of the number of undetected bursts to the total number of possible bursts of any given length. What does this tell you about the power of CRC 32? (5 points)

-

3. **Error Recovery:** Peter Protocol has been consulting with an Internet Service Provider and finds that they use an unusual error recovery protocol shown in the Figure below (next page). The protocol is very similar to Go-back-N with numbered data packets; the key difference is that the sender does not normally send ACKs. The receiver sends a message called a NACK only if it detects an error in the received sequence or if it receives a so-called STATUS packet. In the figure below, the sender sends off the first three data packets. The second one is lost; thus when the receiver gets the third packet it detects an error and sends a NAK which contains the highest number the receiver has received in sequence. When NAK 1 gets to the sender, the sender retransmits data packets 2 and 3. Periodically, based on a timer, the sender transmits a STATUS packet. The receiver always replies to a STATUS packet using a NAK.

   - Why is the STATUS packet needed? What can go wrong if the sender does not send STATUS packets? (3 points)
   - A STATUS packet is sent when a STATUS timer expires. The sender maintains the following property: ``While there remains unacknowledged data, the STATUS timer is running." Why does this property guarantee that any data packet given to the sender will evntually reach the receiver (as long as the link delivers most packets without errors)? (2 points)
   - Under what conditions must the timer be stopped and started so as to maintain the property (5 points)
   - Consider sending a single data packet D that is lost. After that no packets get lost. What is the worst-case latency before the receiver receives D and the sender knows the receiver has got D. (10 points)

Figure for Problem 4



1.

4. **(Exam Practice Only)** Data Link Protocols on Synchronous Links: So far in all our Data Link protocols we have assumed the links to be asynchronous in that the delay of a frame or ack could be arbitrary. Now we consider the case that the time taken for a message or ack is 0.5 time units. Further senders send frames only at integer times like 0,1,2. When a receiver gets an error-free frame (sent at time n) at time n + 0.5, the receiver sends an ack back that arrives (if successful) just before time n + 1. Suppose we use the standard alternating bit protocol except that the sender also waits to send at integer times.

   - Does the sender need to number the data frames? If your answer is yes give a counterexample to show what goes wrong when it does not. (10 points)
   - Does the receiver need to number the ack frames? If your answer is yes give a counterexample to show what goes wrong when it does not. (10 points)
   - Describe a simple protocol for the sender to initialize the receiver state after a crash? (5 points)

- **5. (Exam Practice only, do not hand in) HDLC Framing**: The HDLC protocol uses a flag 01111110 at the start and end of frames. In order to prevent data bits from being confused with flags, the sender stuffs a zero after every 5 consecutive ones in the data. We want to understand further that not all flags work but perhaps some others do work so the HDLC flag is not the only possible one.

  - Consider the flag 11111111 and a similar stuffing rule to HDLC (stuff a 0 after 5 consecutive 1's). Show a counterexample to show this does not work. (5 points)
  - Consider the flag 11111111. Find a stuffing rule that works and argue that it is correct. What is the worst case efficiency of this rule? (Recall HDLC had a worst case efficiency of 1 in 5 bits, or 20%) (5 points)
  - Hugh Hopeful has invented another new flag for HDLC (Hopeful Data Link Control) protocol. He uses the flag 00111100. In order to prevent data bits from being confused with flags, the sender stuffs a one after receiving 001111. Does this work? Justify your answer with a short proof or counterexample. (8 points)
  - To reduce the overhead, Hugh tries to stuff a 1 after receiving 0011110. Will this work? Justify your answer with a short proof or counterexample. (7 points)