UCLA CS 118 Computer Networks

Fall 2020 Midterm Exam, Long Portion, Allocated time, 1 hour

Directions: Write your name on the exam and on every page you submit. Write something for every question. Students who do not write something for everything lose out over students who write down wild guesses. You will get some points if you attempt a solution but nothing for a blank sheet. Write something down, even wild guesses. Problems take long to read but can be answered concisely. Unless you have an accomodation, you must submit in 1 hour after you start.

Question	Maximum	Score
1	20	
2	20	
3	20	
Total		

Q1, Intersymbol Interference : This is a small variation on your HW 1 question on Intersymbol Interference. Figure 1 shows the response of a wire to an input bit of width 1 μs. The response to both slow and fast bits is the same. The response is an approximation to the sinc function we studied in class. Unlike HW 1, this approximation is a trapezoid and not a traingle. It rises to 2V in 0.25μs, stays constant for 0.5s falls to 0V after 0.25μs. It then falls to -1V at 1.25μs, climbs to 0V at 1.5s and remains zero. Assume that a 1 is encoded as a 2V signal, and a 0 is encoded as 0V. Assume that the output to a 0V input of any bit width is also 0V for all time. If at any sampling instant, the receiver measures a voltage of 1V or more, it assumes it has received a 1; else it assumes it has received a 0. The sender is going to send just 3 bits 101.

1. **Slow Bits** (6 points) First, assume the sender sends bits at the slow rate of once every 1μs. Assume the sender sends its 3 bits at times 0, 1, and 2 us respectively. The sampling instants the receiver uses are 0.5, 1.5,and 2.5 μs respectively for 3 bits. At any sampling instant, the receiver measures the output voltage as the sum of the voltage values of the output of the 3 bits. Write down the measured outputs. What bits does the receiver output?

2. **Fast Bits** (6 points) Following the same instructions as in Part 1 but now the sender now sends its 3 bits at times 0, 0.5, and 1μs respectively where the bit-width is now 0.5μs as opposed to 1μs as in part 1. The sampling instants the receiver uses are 0.25, 0.75,and 1.25μs respectively for 3 bits. Write down the measured outputs. What bits does the receiver output?

3. **Supersonic Bits** (6 points) Following the same instructions as in part 1 but, the sender now sends its 3 bits at times 0, 0.25, and 0.5μs respectively where the bit-width is now 0.25μs as opposed to 1μs as in part 1. The sampling instants the receiver uses are 0.125, 0.375,and 0.625μs respectively for 3 bits. Write down the measured outputs. What bits does the receiver output?

4. **Sensitivity to Shape** (2 points) Based on your knowledge of the sinc function, the HW 1 problem, and this problem, what do these results suggest about the "shape" of the output wave required to transmit at the Nyquist limit of 2B.

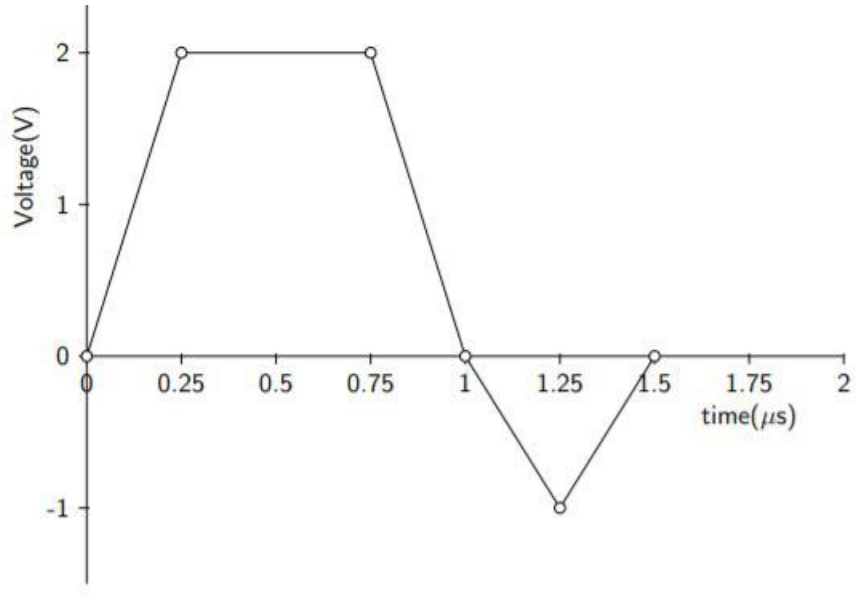

Figure 1: Wire response to sender's bit 1 (2V)

2, 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 occurence 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

4. Window Size and Modulus for Go-back-n, 20 points: The figure below shows two examples of a Go-back-n protocol working with a window size of 4, and a modulus of 4. Recall that if the modulus is 4, all sequence numbers and acks are incremented mod 4. We want to show that the modulus is too small. In both cases, the sender starts by sending the first 4 packets it is allowed to in its current window (D(0) through D(3)) without waiting for an ack.

- **a, 4 points:** In Case 1 shown on the left, the Ack to the first data packet D(0) is received. On receipt of A(1), the window slides and the sender sends a different D(0) from the one it sent earlier. What should the receiver ideally do in this case. Specify how the receiver number changes if it does, whether the data packet should be accepted, and what ack number should be sent back.
- **b, 4 points:** In Case 2 shown on the right, the Ack to the first data packet D(0) is lost. After a timeout occurs, the sender retransmits the old D(0). What should the receiver do in this case. Specify how the receiver number changes if it does, whether the data packet should be accepted, and what ack number should be sent back.
- **c, 5 points:** Argue from these 2 cases, that the modulus is too small to work correctly.
- **d, 5 points:** Using the ideas in this example, argue briefly why the modulus should be at least $w + 1$, for a general window size of w. Try to make your argyument general that works for any w not just 4.
- **e, 2 points:** For the special case of window size $w = 1$ explain why this makes sense in terms of what you know of the Alternating Bit protocol.