CS450 - Assignment Four

Andrew Struthers

April 2023

1. If you wish to send the following data using PPP, what would the last 6 bytes before the CRC be?

   DLE B C ETX STX A ETX DLE

   - If we convert each of these pieces of information into their byte representations, we get the following:

   $$\begin{aligned} \text{DLE} &= 0\times10 \\ \text{B} &= 0\times42 \\ \text{C} &= 0\times43 \\ \text{ETX} &= 0\times03 \\ \text{STX} &= 0\times02 \\ \text{A} &= 0\times41 \\ \text{ETX} &= 0\times03 \\ \text{DLE} &= 0\times10 \end{aligned} \tag{1}$$

   So the last 6 bytes would be 0×43 0×03 0×02 0×41 0×03 0×10 or the following bit sequence: 01000011 00000011 00000010 01000001 00000011 00010000

2. What would the following data be after being encoded using HDLC?

   1101111111101101011111111011101011

   - We will encode the beginning and the end of the frame with the distinguishing sequence 01111110. We will then insert a 0 every time there are 5 consecutive 1s where the next bit is also 1. This will allow the receiver to properly decode and interpret what data belongs to the frame, and what denotes the end of the sequence. Thus, we have the following encoded message:

   01111110 1101111101110110101111110111011101011 01111110

3. Decode the following HDLC frame

   01111110 1111010111110111111010111001011111110110 01111110

- HDLC has a one byte distinguishing sequence that allows for clock synchronization before the message is received, and denotes the beginning and end of frames. That sequence is 01111110. Since HDLC also uses bit stuffing to prevent 6 consecutive 1s, any time there would be 6 consecutive 1s in the message, an additional 0 is added. To decode this, we first throw away the first 01111110 byte, and then start decoding the true message. What we have to be aware of is 5 consecutive 1s. If this happens, we will check the next bit. If it is a zero, and the next bit after that is a 1, we will remove the bit stuffed zero. If the next bit is also a 1, we will assume that there was an error in transmission (7 consecutive 1s should **never** happen). Thus, the decoded frame is the following:

  11111010111111111110111001011111110

4. Compute the 16-bit checksum for the following stream of bits

   1101 1001 0011 1010 1010 1111 0000 1101 0101 1010 1110 0100 1100 1011 0100 1101

   - First we add with carry the first 16 bits to the second 16 bits

$$\begin{aligned} \text{sum} &= 0\ 1101\ 1001\ 0011\ 1010+ \\ &= 0\ 1010\ 1111\ 0000\ 1101 \\ &= 1\ 1000\ 1000\ 0100\ 0111 \\ &= 0\ 1000\ 1000\ 0100\ 1000 \end{aligned} \tag{2}$$

   Now we take that result and add the next 16 bits:

$$\begin{aligned} \text{sum} &= 0\ 1000\ 1000\ 0100\ 1000+ \\ &= 0\ 0101\ 1010\ 1110\ 0100 \\ &= 0\ 1110\ 0011\ 0010\ 1100 \end{aligned} \tag{3}$$

   Finally, we add the last 16 bits:

$$\begin{aligned} \text{sum} &= 0\ 1110\ 0011\ 0010\ 1100+ \\ &= 0\ 1100\ 1011\ 0100\ 1101 \\ &= 1\ 1010\ 1110\ 0111\ 1001 \\ &= 0\ 1010\ 1110\ 0111\ 1010 \end{aligned} \tag{4}$$

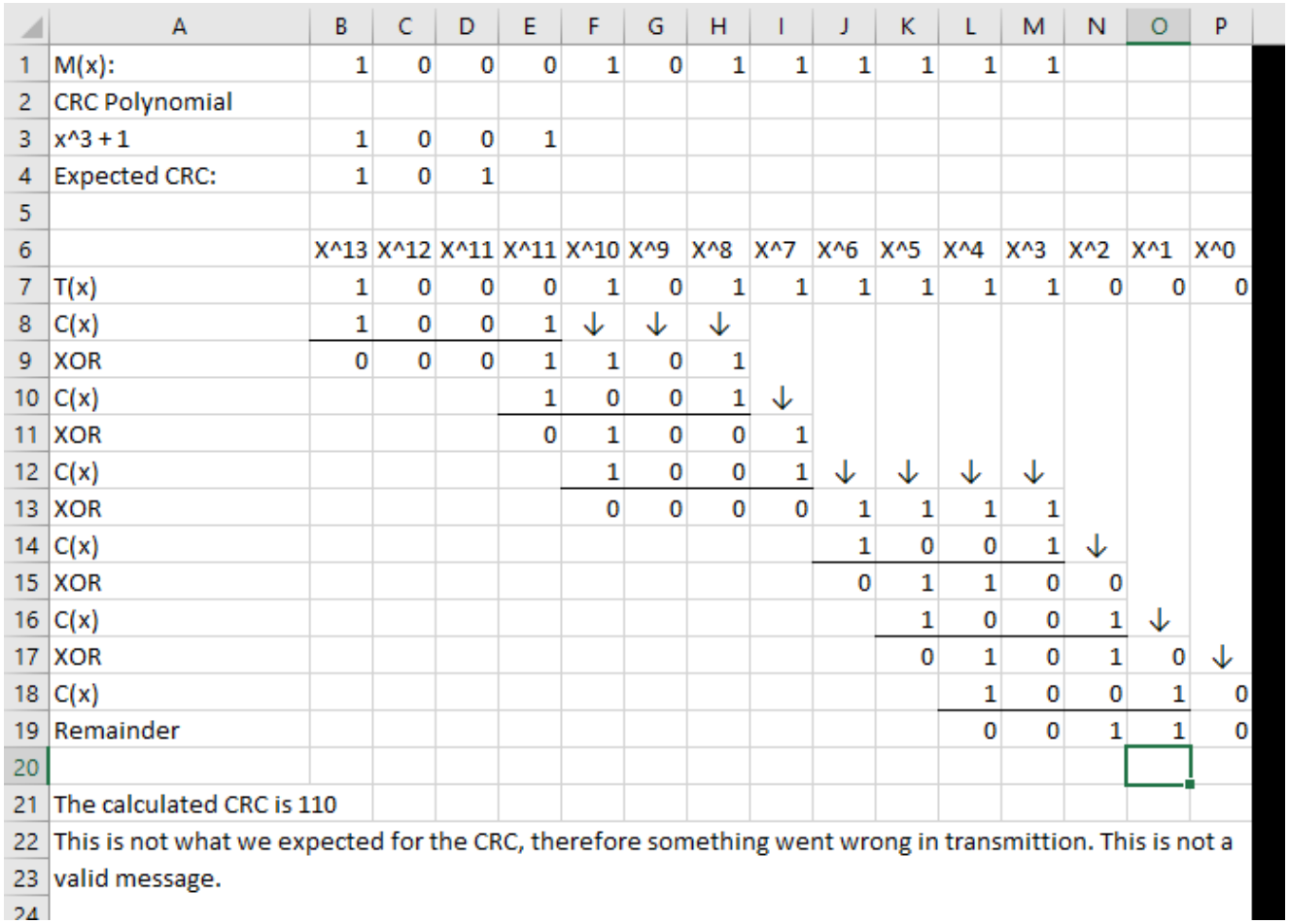
   One last step is to now negate our answer, since this is one's complement arithmetic. We will just flip all of the bits in our last sum, which gives us the result:

$$\text{sum} = 0\ 0101\ 0001\ 1000\ 0101 \tag{5}$$

   Thus, our 16-bit checksum is 0101 0001 1000 0101

5. If the following message and CRC arrived, use the $x^3 + 1$ CRC polynomial to determine if the message is valid

- Solved in Excel:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	M(x):	1	0	0	0	1	0	1	1	1	1	1	1			
2	CRC Polynomial															
3	x^3 + 1	1	0	0	1											
4	Expected CRC:	1	0	1												
5																
6		X^13	X^12	X^11	X^11	X^10	X^9	X^8	X^7	X^6	X^5	X^4	X^3	X^2	X^1	X^0
7	T(x)	1	0	0	0	1	0	1	1	1	1	1	1	0	0	0
8	C(x)	1	0	0	1	↓	↓	↓								
9	XOR	0	0	0	1	1	0	1								
10	C(x)				1	0	0	1	↓							
11	XOR				0	1	0	0	1							
12	C(x)					1	0	0	1	↓	↓	↓	↓			
13	XOR					0	0	0	0	1	1	1	1			
14	C(x)									1	0	0	1	↓		
15	XOR									0	1	1	0	0		
16	C(x)										1	0	0	1	↓	
17	XOR										0	1	0	1	0	↓
18	C(x)											1	0	0	1	0
19	Remainder											0	0	1	1	0
20																
21	The calculated CRC is 110															
22	This is not what we expected for the CRC, therefore something went wrong in transmittion. This is not a															
23	valid message.															
24																

Figure 1: CRC calculation indicating a bad transmission

6. Calculate the minimum timeout value for a stop-and-wait algorithm running over a 50-meter point-to-point copper link. Assume a minimum delay of 5 μs at the receiver

   - To figure out the minimum timeout value, we first need to calculate the RTT of this circuit. Since this is a 50m copper link, we know that the propagation delay will be: $50 \times 2.3 * 10^8 = 0.217\mu s$. The RTT for this circuit is $2 \times$ propagation delay $= 2 \times 0.217\mu s = 0.535\mu s$. The receiver adds a minimum delay of $5\mu s$, so we know the total circuit RTT is $0.435\mu s + 5\mu s = 5.435\mu s$. If the sender sends data, the absolute minimum response time the receiver can have would therefore be $5.435\mu s$, so the minimum timeout value could be set to this time. However, this is assuming absolutely perfect conditions, so it is not recommended to have such a short timeout time.

7. Draw a timeline diagram in Visio for the sliding window algorithm with SWS = RWS = 4. Assume a 2 RTT timeout and that Frame 2 is lost on its initial transmission. Show all transmissions up until an ACK is received for the successful transmission of Frame 3

   - Solved in Visio:

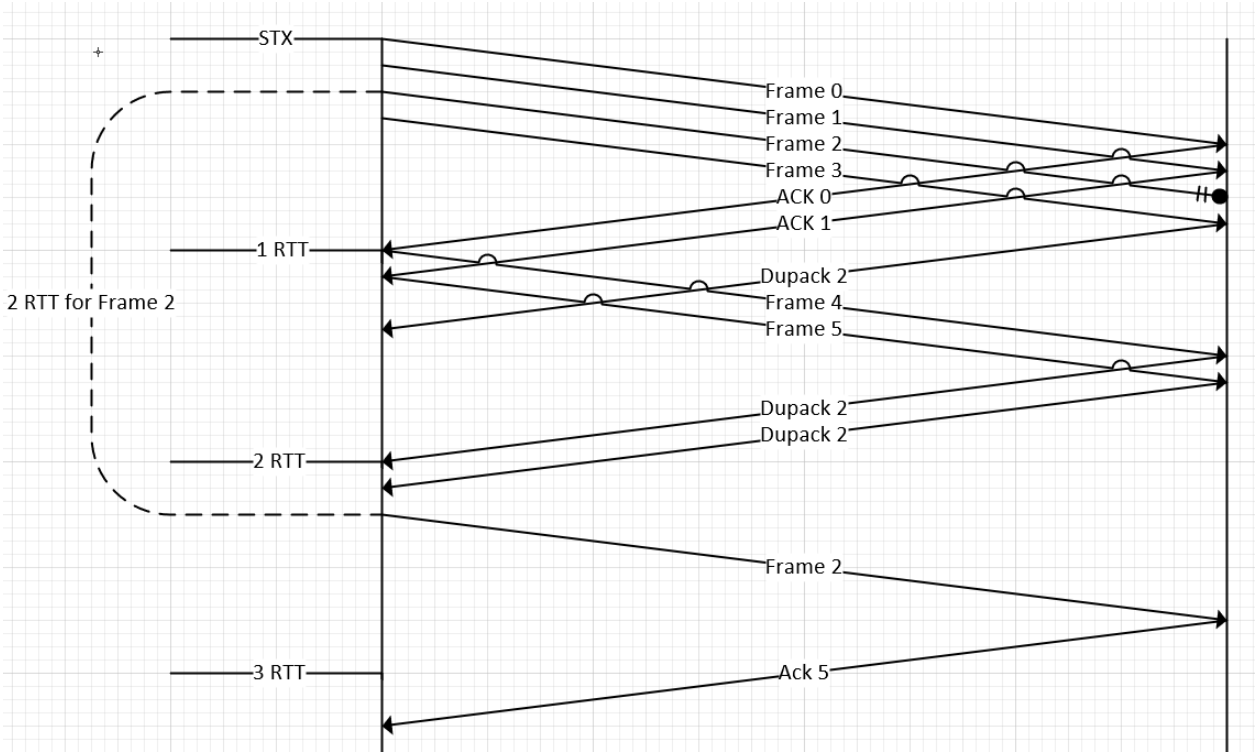


Figure 2: Sliding window protocol where frame 2 failed to send, with a 2 RTT timeout for failed transmissions

8. If a point-to-point link has a 3 RTT timeout, how much time is allotted for the possible delay on the receiving end?

   - 2 RTT is allotted for the delay on the receiving end. The sender would send a message, which would get to (or not get to) the receiver in $\frac{1}{2}$ RTT. The sender would wait its 3 RTT until re-transmission, but since the receiver was only made aware of a transmission at $\frac{1}{2}$ RTT and the sender would have to wait $\frac{1}{2}$ additional RTT to not receive the ACK before timeout, the receiver only waits 2 RTT $(3 - (\frac{1}{2} + \frac{1}{2}) = 2)$.

9. Fully explain how 802.11 achieves collision avoidance. Include the optional mechanisms described in the textbook

   - One of the problems on top of Ethernet collision avoidance is the hidden node problem. This is when we have three nodes, where node 1 and 3 are out of range of each other, but both are in range of node 2. If 1 and 3 send messages to 2, there is a collision, but because 1 and 3 are out of range of one another, they are both unaware. Only node 2 would recognize the collision, since 1 and 3 are "hidden" from each other.

     Another problem is the exposed node problem. This is where there are four nodes, 1, 2, 3, and 4. Node 1 is in range of node 2 and 3, node 4 is also in range of 2 and 3, but node 1 and 4 are out of range of one another. If 1 wants to send a message to 2, node 3 will hear that transmission and assume that everything is busy (since wireless

gets transmitted multi-directionally). Since this is the case, 3 won't transmit to or receive from node 4, even though it technically could. Node 3 could conclude that it can't transmit to 4 because of the transmission from 1 to 2 being technically in node 3's range.

To combat this, 802.11 uses CSMA/CA (collision avoidance) instead of CSMA/CD (collision detection). This works by the transmitter first checking to see if it recognizes any other transmissions in its area. If there are no other transmissions, the coast is clear and the transmitter can transmit. All is good. But, because of the hidden node problem, the receiver could still get a collision. Because of the hidden node problem, there is now a required ACK from the receiver to the transmitter. If the receiver did receive and decode the message appropriately, the receiver sends an ACK back to the transmitter and communication can occur. Additionally, there is a RTS-CTS flag (ready to send, clear to send). The transmitter can send a RTS packet to the receiver, and if the receiver is ready, it can send the CTS back. If any hidden nodes get a CTS but never get an RTS, this tells each hidden node to hold back on sending for a while, effectively preventing the hidden node problem. If two RTS frames ever collide, each transmitter will use an exponential backoff algorithm to wait until sending another RTS frame, in hopes that any collision that occurred once will not likely occur again. The RTS and CTS frames also contain information like expected transmit time, so that any other node that's either hidden or exposed will have an expectation as to when they can start sending/receiving again.

10. What type of address would a WiFi distribution system use

    - WiFi distribution systems use MAC addresses to identify access points inside of the system. Each access point inside of the distribution system also has a MAC address that can be used to communicate with each wireless client, but this MAC address isn't part of the WiFi distribution system.