Computer Science 118, Homework 2

Michael Wu
UID: 404751542

October 30th, 2018

Problem 1

a) Consider the frame containing the bit sequence 01. After adding flags to this bit sequence, we will obtain the frame 11111111 01 11111111. But when the receiver tries to decode this message, it will think that the last 1 is a part of the ending flag. It decodes the bit sequence 0, which is different from what was sent. So this choice of a flag would not work.

b) A stuffing rule that would work would be to add a 0 after every seven consecutive 1's, and add a 0 at the end of the encoded data. The flag could not occur in the encoded data because any group of eight 1's would have a 0 stuffed somewhere in between. The flag could not occur across a stuffed bit boundary because the stuffed bits are only zeros, and the flag contains no zeros. Finally, a false flag could not be created at the end of the encoded data because the encoded data will have a 0 stuffed at the end. So this stuffing rule would be correct. In the worse case, this rule would result in an efficiency of 50%, which happens if our data frame only contains one bit. Then one stuffed bit will be added, resulting in 1 in 2 bits being discarded when the receiver decodes the received bits.

c) This would work. The flag could not appear in the encoded data, since 00111100 would be turned into 001111100. The flag could not appear across a stuffed bit boundary because a stuffed bit will always occur within a string of at least five 1's, which does not appear in the flag. A false flag could not appear at the end of the encoded data because the only suffix that forms a

false flag with the prefix of the final 5flag would be 001111. After stuffing, this becomes 0011111, which cannot make a false flag with a prefix of the final flag.

d) This would not work. Consider the frame containing the bit sequence 1001111. After adding flags to this bit sequence and stuffing, we will obtain the frame 00111100 1001111 00111100. But when the receiver tries to decode this message, it will think that the last six values in the encoded data are a part of the final flag. Thus after decoding the bit sequence, the receiver thinks that the data that was sent was a 1. This is different from what was sent, so this stuffing scheme would not work.

Problem 2

a) The message will be shifted left by 5 bits, and the remainder when dividing will be 11001. This results in the message 10011001.

b) The resulting decoded message will be equal to the original message with the lowest order bit flipped. This is because the original message is shifted by $r-1$ bits to the left, where r is the length of the CRC generator. So the highest bit of the CRC generator, which must be 1, will flip the lowest bit of the original message and leave the rest unchanged. This error will not be detected since the modified message will still divide by the CRC generator. In the previous example, the resulting decoded message would be 101.

c) Yes this generator detects all one bit errors. This is because no one bit error will result in a modified message that is divisible by the CRC generator, as the CRC generator is a polynomial with 4 terms and a one bit error will be a polynomial with one term.

d) Yes this generator detects all odd bit errors. This is because the generator is equal to $(x^4+1)(x+1)$, and an odd number of bit errors would produce a polynomial that is not divisible by $x+1$. This can be seen by substituting $x=1$ into the polynomials, the odd bit error will result in an odd number of terms and thus an odd sum, while the expression $x+1$ would result in an even number. Since odd numbers are not divisible by even numbers, this generator detects all odd bit errors.

e) There are only four, $(x^3+1)(x^5+x^4+x+1)$, $(x^3+x+1)(x^5+x^4+x+1)$, $(x^3+x^2+1)(x^5+x^4+x+1)$, and $(x^3+x^2+x+1)(x^5+x^4+x+1)$. This is the case because the only multiples of the CRC generator that work must include x^3 as the highest order polynomial, since any higher would result in an error with a degree greater than 8. Also the only multiples of the CRC generator that work must include 1, since the burst error includes 1 and the only way to achieve this is to do 1×1. So the only four multiples that work are x^3+1, x^3+x+1, x^3+x^2+1, and x^3+x^2+x+1.

f) There are 2^4 polynomials with a highest power x^5 and a lowest power 1. However, in context it seems like this question relates to the previous question. In that case there are four multiples of the CRC generator that create an undetected burst error of length 9.

g) Consider a burst error of length n. There are 2^{n-2} different such burst errors, since the highest and lowest order terms are fixed. The multiples of the CRC generator that lead to an undetected error must be a polynomial of order $n-1-k$. Since the lowest order term must be a 1, there are 2^{n-2-k} different such polynomials. Therefore the probability of an undetected burst error of length n is

$$\frac{2^{n-2-k}}{2^{n-2}} = 2^{-k}$$

Problem 3

a) Yes, consider if there are no numbers on the data frames and an ack gets lost. Then duplication could occur because the sender retransmits and the receiver accepts twice.

b) No, the receiver does not need to number ack frames. Since the protocol is synchronous, the sender will know which ack corresponds to which sent frame. Thus the number of the ack can be deduced from the number of the data frame.

c) The sender simply issues a restart data frame. If it does not receive an ack on time, it knows that the ack has not arrived so it should resend the

restart until it receives an ack. At this point the sender and receiver are synchronized.

Problem 4

a) The status packet is needed to ensure that the receiver is correctly receiving the data. If the receiver never sent a nak, then the sender would assume that the receiver is working correctly. But it may just be that the receiver is not receiving any packets at all, so it never knows to send a nak. Periodically sending a status signal lets the sender tell the difference, since a nak would indicate that the receiver is working correctly and not dropping all the packets.

b) This property guarantees that any data packet given to the sender will be sent because the status timer will periodically cause a nak signal to be sent. Based on the number included with the nak, the sender knows which packets to retry, until all are send.

c) The timer must start and keep running when there is data available to be transmitted. When the sender is idle, the status timer can be stopped.

d) The latency would be the time it takes to send the first D that fails, send the next packet, receive a nak, retry D, wait for the status timer to send a status signal, then receive a nak that indicates D was sent. Since the maximum time between two naks is the time between two status signals, assuming all naks are received, the maximum latency would be the period of time between two status signals.