Problem Set 1 Solutions

Problem 1: Text Problem 2.10

We provide a proof by counterexample. Suppose the first codeword and the string differ in $x_1 < d/2$ bits, and the string and a second codeword differ in $x_2 \leq d/2$ bits. Then the first codeword and the second differ in at most $x_1 + x_2 < d$ bits. However, since the minimum distance of the code is d, this cannot be the case.

From this, we can conclude that if fewer than $d/2$ bit errors occur in a codeword, the resulting noise-corrupted string must be a distance of greater than $d/2$ away from any other codeword. Therefore, the minimum-distance decoder rule will correctly choose the original codeword.

Problem 2: Text Problem 2.11

We assume that the first 3 bits are data, and the last 4 bits the parity bits. Notice that the data bits in the codewords given are 100, 010, and 001. Since only a single data bit is non-zero in each codeword, we can simply look at the parity bits for each codeword to see which data bits are being checked by each parity bit.

In the first codeword, only the first data bit is non-zero. Since the parity bits are 1011, the first, third, and fourth parity bits must check the first bit. Similarly, for the second codeword, since only the second data bit is non-zero, the first, second, and fourth parity bits must check the second bit. Finally, from the last codeword, the first, second, and third parity bits must check the third data bit.

In summary,

$$
\begin{aligned}
C_1 &= s_1 + s_2 + s_3 \\
C_2 &= \quad + s_2 + s_3 \\
C_3 &= s_1 + \quad + s_3 \\
C_4 &= s_1 + s_2 +
\end{aligned}
$$

Problem 3: Text Problem 2.33

The stuffed sequence becomes:

11011010**0**010**0**10**0**10**0**11101**0** **0101**

The destuffing rule is to decode (destuff) the string bit by bit starting at the beginning. A given 0 bit is then deleted from the string if the preceding three decoded bits are 010. The flag is detected when a 1 is preceded by the three decoded bits 010 and the most recently decoded bit was not deleted. The above is a general rule for detecting any type of flag sequence, rather than just 0101; for this special case, it is sufficient to look for the substring 0101 in the received string. The reason for the simplification is that if an insertion occurs within the flag, it has to occur by simply a repetition of the first flag bit.

Destuffing the second sequence gives:

11010x010x10x110 **0101**

Problem 4: Text Problem 2.39

a) Because the actual amount of the transmitted bits is $\left\lceil \frac{M}{K_{\max}} \right\rceil K_{\max}$, it is given by

$$TC = (K_{\max} + V)(j-1) + (K_{\max} + V)\left\lceil \frac{M}{K_{\max}} \right\rceil$$

b) The approximation is given by

$$E\{TC\} \approx (K_{\max} + V)\left(j - 1 + \frac{E\{M\}}{K_{\max}} + \frac{1}{2} \right)$$

The above function $E\{TC\}$ is a strictly convex function with respect to K, and hence it is minimized at a stationary point which is unique. Taking derivative of $E\{TC\}$ with respect to K, and equating the derivative to zero yield the stationary point, which is given by

$$K_{\max}{}^{*} = \sqrt{\frac{E\{M\}V}{j - 1/2}}$$

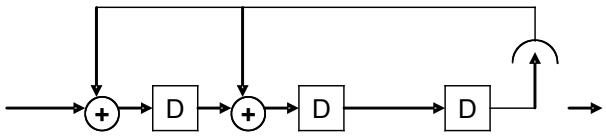
c) When $j = 1$, TC in (2.42) is given by $TC = M + \left\lceil \frac{M}{K_{\max}} \right\rceil V$. In this case, any $K_{\max} \geq M$ minimizes TC, and so does $K_{\max} = \infty$. In contrast, TC in a) becomes $TC = (K_{\max} + V)\left\lceil \frac{M}{K_{\max}} \right\rceil$ for $j = 1$. It is obvious that this TC is minimized when $K_{\max} = M$. In this case, we have $TC = M + V$, implying that it is always better to transmit the message as a single packet. Hence, the finite $K_{\max}$ in b) is reasonable. It is however based on the approximation $E\left\{ \left\lceil \frac{M}{K_{\max}} \right\rceil \right\} \approx E\left\{ \frac{M}{K_{\max}} \right\} + \frac{1}{2}$, which is poor for large $K_{\max}$ and hence $K_{\max}{}^{*}$ in b) is inaccurate.

Problem 5: CRC codes

a) For G = 1011 and M = 10110, we will have 3 CRC bits (since the generator is length 4). We therefore append 3 zero bits to M and divide by G to obtain the CRC:

```
1011 | 1011000
       1011
       ----
       0000
```

Since there is no remainder, the CRC bits are 000, and the message is T = 10110000. The shift register implementation is:



b) We perform the division to obtain:

```
          1011
       ---------
1001 | 1010111
       1001
       ---------
         1111
         1001
       ---------
          1101
          1001
       ---------
           100
```

Since the remainder is non-zero, we know an error occurred.

c) Again, the division yields:

```
          1011
       ---------
1001 | 1010011
       1001
       ---------
         1101
         1001
       ---------
          1001
          1001
       ---------
           000
```

This time, since the remainder is zero, no error was detected. An error could still have occurred if it mapped the original codeword to a new (valid) codeword.

Problem 6

a) We have

$D_P = 0.25$, (note 0.5 is the round-trip delay)

$D_{TP} = D_{TA} = 10000 / 1{,}000{,}000 = 0.01$, and $S = D_{TP} + 2D_P + D_{TA} = 0.5 + 2(0.01) = 0.52$

The probability that either the packet or its acknowledgement is lost is given by:

$q = 1 - (1-p)^2 = 0.19$

From the class notes, we have that the expected transmission time of a packet in this system is given by:

$$E[X] = S + TO*q(1-q)$$

Assuming that TO = S We have, $E[X] = 0.642$

$$\text{Efficiency} = D_{TP} / E[X] = 0.01 / 0.642 = 1.56\%$$

b) Efficiency is given by,

$E = D_{TP} / E[X] = D_{TP}/S(1+ q(1-q)) = D_{TP}/((0.5+2\ D_{TP})*(1.235))$

For efficiency of 20% , we solve for D_{TP} to obtain, $D_{TP} = 0.244$ or packet length = 244000 bits.

Problem 7

a) In order for a pair of errors to go undetected, they must occur in the same position in the block. For example, the following error pattern could go undetected

00010000
00010000

Because the two errors in the same bit position would generate the same sum bit as if the bits were not in error. However, due to the carry-out, if the original bits are (0,0) or (1,1) the error would still be detected because of an error in the carry bit. Note that the bits (1,1) would generate a carry and if they both incur errors, then the resulting (0,0) would not generate the same carry and hence the error would be detected.

b) For a block length of K, a pair of errors would align on the same position with probability 1/K. Since only half of the errors in the same bit positions would be undetected (due to the carry), the fraction of two bit errors that would be undetected is 1/(2K).