COMP 4320 Introduction to Computer Networks
Homework Assignment 2 Liam Maher
Due on Sunday, July 7 on Canvas (upload your answer sheet)

Instruction: Every student should finish the following questions independently. Give justification for the results (i.e., show the calculation process) to receive credits. Scan your answer sheet and upload it to Canvas.

1. Show the 4B/5B encoding, and the resulting NRZI signal, for the following bit sequence:

1110 0101 0000 0011

Soln:

4-bit data symbol	5-bit code
1110	11100
0101	01011
0000	11110
0011	10101

4B/5B encoding => 11100 01011 11110 10101

Resulting NRZI signal:

Initial state: low
1: Low -> high : 1
1: High -> low : 0
1: low -> high : 1
0: high : 1
0: high : 1
0: high : 1
1: high -> low : 0
0: low : 0
1: low -> high : 1
1: high -> low : 0
1: low -> high :1
1: high-> low :0
1: low -> high :1
1: high -> low : 0
0: low :0
1: low -> high : 1
0: high : 1

1: high -> low : 0
0: low : 0
1: low -> high : 1

NRZI signal: 10111 10010 10100 11001

2. Assuming a framing protocol that uses bit stuffing, show the bit sequence transmitted over the link when the frame contains the following bit sequence:

110101111101011111101011111110

Mark the stuffed bits.

Soln: (Stuffed bits marked with highlight)
110101111100101111101010111110110

3. Show that two-dimensional parity allows detection of all 3-bit errors. Give an example of a 4bit error that would not be detected by two-dimensional parity, as illustrated in Figure 2.14 in the Peterson's book. What is the general set of circumstances under which 4-bit errors will be undetected? Show that two-dimensional parity provides the receiver enough information to correct any 1-bit error (assuming the receiver knows only 1 bit is bad), but not any 2-bit error.
Soln:
Figure 2.14 (blue columns are parity bits)

0	1	0	1	0	0	1	1
1	1	0	1	0	0	1	0
1	0	1	1	1	1	0	1
0	0	0	1	1	1	0	1
0	1	1	0	1	0	0	1
1	0	1	1	1	1	1	0
1	1	1	1	0	1	1	0

3-bit errors:

1	1	0	1	0	0	1	1 / 0
1	0	0	1	0	0	1	0 / 1
1	0	0	1	1	1	0	1 / 0
0	0	0	1	1	1	0	1
0	1	1	0	1	0	0	1
1	0	1	1	1	1	1	0
1 / 0	1 / 0	1 / 0	1	0	1	1	0 / 1

1	1	0	1	0	0	1	1 / 0
0	1	0	1	0	0	1	0 / 1
0	0	1	1	1	1	0	1 / 0
0	0	0	1	1	1	0	1
0	1	1	0	1	0	0	1
1	0	1	1	1	1	1	0
1 / 0	1	1	1	0	1	1	0 / 1

1	1	0	1	0	0	1	1 / 0
0	1	0	1	0	0	1	0 / 1
1	1	1	1	1	1	0	1 / 0
0	0	0	1	1	1	0	1
0	1	1	0	1	0	0	1
1	0	1	1	1	1	1	0
1	1 / 0	1	1	0	1	1	0 / 1

Red boxes are bits that are flipped. The slashes indicate the correct parity / new parity after bit error. In a two-dimensional parity, all 3-bit errors would always result in at least one row and column having the incorrect parity. Because each of the columns and rows has its own parity bit, it will be able to detect the 3-bit error because a 3 bit error will cause at least one row and column parity mismatch. If all of the error bits were in the same row/column, this would also cause the parity to be off, which would lead to it being detected. Furthermore, if 2 of the error bits were in the same row/column, and 1 was in a different row/column, the parities of those rows and columns would also be incorrect, which would lead to it being detected.

4-bit error that would not be detected:

1	0	0	1	0	0	1	1
0	0	0	1	0	0	1	0
1	0	1	1	1	1	0	1
0	0	0	1	1	1	0	1
0	1	1	0	1	0	0	1
1	0	1	1	1	1	1	0
1	1	1	1	0	1	1	0

As we can see from the red boxes, with the switched bits, the 4 bit error here is undetected as the totals for the parity bits are still right, even though for of the bits were flipped.

General set of circumstances where 4-bit error would go undetected:
4-bit errors will go undetected if they happen in a way that preserves the parity of each row and column. This generally occurs where 2 bits are flipped in 2 different rows and columns, resulting in the preservation of the parity calculations even with the 4 bit errors, such as we saw in the example above.

2-dim parity provides receiver enough info to correct any 1 bit error, but not any 2 bit error:
2 Dimensional parity provides the receiver with enough info to correct a 1 bit error because for all 1 bit errors, as show in the example below, the corresponding row and column parity are now incorrect. This points to the position of the 1-bit error.

0	1	0	1	0	0	1	1
1	0	0	1	0	0	1	0 / 1
1	0	1	1	1	1	0	1
0	0	0	1	1	1	0	1
0	1	1	0	1	0	0	1
1	0	1	1	1	1	1	0
1	1 / 0	1	1	0	1	1	0 / 1

However, for a 2-bit error, if both of the bits are in the same row/column (example 1 below) it will still change the parity but not show where the error is occuring. If both bits are in a different row and column, it will again detect errors due to the incorrect parities, but it will not be able to pinpoint there exact location. Multiple combinations of bit flips can cause this to happen that is why it can not determine exact location.

1	1	0	1	0	0	1	1 / 0
0	1	0	1	0	0	1	0 / 1
1	0	1	1	1	1	0	1
0	0	0	1	1	1	0	1
0	1	1	0	1	0	0	1
1	0	1	1	1	1	1	0
1	1	1	1	0	1	1	0

1	1	0	1	0	0	1	1 / 0
1	0	0	1	0	0	1	0 / 1
1	0	1	1	1	1	0	1
0	0	0	1	1	1	0	1
0	1	1	0	1	0	0	1
1	0	1	1	1	1	1	0
1 / 0	1 / 0	1	1	0	1	1	0

4. Suppose we want to transmit the message 11100011 and protect it from errors using the CRC polynomial $x^3 + 1$.

(a) Use polynomial long division to determine the message that should be transmitted.
M = 11100011
C(x) = generator = x^3+1 = 1001
T(x) = 11100011 | 000 = 11100011000 (3 bits added)

```
     ___________
1001)11100011000
     1001
      1110
      1001
       1110
       1001
        1111
        1001
         1101
         1001
          1000
          1001
           0010
           0000
            0100 this is remainder
```

R(x) = 100
P(x) = 11100011 | 100 = 11100011100

(b) Suppose the leftmost bit of the message is inverted due to noise on the transmission link. What is the result of the receiver's CRC calculation? How does the receiver know that an error has occurred?

M = 01100011100 (100 added from previous calc, with leftmost bit flipped)
C(x) = generator = x^3+1 = 1001

```
     ___________
1001)01100011100
     0000
      1100
      1001
       1010
       1001
        0111
        0000
         1111
         1001
          1101
          1001
```

```
1000
1001
 0010
 0000
  0010 => Remainder != 0
```
Since the remainder is not 0, the server knows that there is an error in the data.