Communication Errors

Introduction
Implementations of memory, storage, and communication are not perfect. Therefore, it is possible that when I store a number on a hard drive or send it over a network, a slightly different number is actually stored/received. There are techniques to deal with this.

Definition An *error detection (correction) code* is a method of encoding data in so that if a certain number of errors has occurred during transmission, it can be detected (corrected).

Definition The *hamming distance* between two bit strings is the number of bits where they differ.

Example The hamming distance between 10010011 and 11100001 is 4 since these bit patterns differ in bits 2, 3, 4, and 7.

10010011
11100001

Parity Check Code

The simplest method of detecting errors is by using a *parity bit*. To use a parity bit, one simply appends one additional bit (to the beginning or end—it doesn't matter which as long as you are consistent) to a bit string so that the number of 1s in the string is odd (or even). To detect an error you add the number of '1' bits. If it is not odd (even) you know an error has occurred.

Example The bit string 10010011 is encoded as 100100111 using an odd parity check code, and the bit string 11110001 is encoded as 111100010 using an odd parity check code.

Question

- How many errors can this method detect? What happens if more errors occur?
- How many errors can this method correct?

Repetition Code

A *repetition code* encodes a bit string by appending k copies of the bit string. This can be done by either copying the first bit k times, followed by the next bit k times, etc., or copying the entire message k times.

Example If I use a repetition code with $k = 3$, I would encode the bit string 0010 as 000000111000. Alternatively, it could be encoded as 001000100010. For our purposes we will use the first method that repeats each bit.

Questions

- How can I determine what string was sent?
- What values of k make sense to use?
- For a given value of k, how many errors can be detected?
- For a given value of k, how many errors can be corrected?

Hamming Code

We can use a *Hamming Code* to both detect and correct errors. The table below gives a Hamming Code that encodes 4 bits into 7 bits. We won't give the details of how this is computed, but notice that each pair of code words has a Hamming distance of at least 3. Therefore if at most one bit is flipped, we can uniquely decode the word.

We can use the *parity check matrix (PCM)* to determine which bit has an error (again, we won't go into detail about why this works).

1. Bitwise AND each row of the PCM with the string.
2. Determine the parity of each of the bit strings from step 1.
3. We now have a 3-bit string that gives us information about the error:
   a. If we get 000, there is no error.
   b. If we get anything else, the column of the PCM that matches the result is the column with the error.

Parity Check Matrix

```
1001011
0101101
0010111
```

A Hamming Code		
Letter	String	Code word
0	0000	0000000
1	0001	1110001
2	0010	1010010
3	0011	0100011
4	0100	0110100
5	0101	1000101
6	0110	1100110
7	0111	0010111
8	1000	1101000
9	1001	0011001
A	1010	0111010
B	1011	1001011
C	1100	1011100
D	1101	0101101
E	1110	0001110
F	1111	1111111

Example
Code word: **0001100**

1001011 AND 0001100	→	0001000	→	**1**
0101101 AND 0001100	→	0001100	→	**0**
0010111 AND 0001100	→	0000100	→	**1**

```
                    ↓
0001100 ← 1001011
  ↓       0101101
0001110   0010111
  ↓
1110→E
```

Example Decode the string 1101011 1101010 0110101 using the parity check matrix.
Example Encode the hexadecimal string ***FAB5*** using the Hamming Code above.
Example Encode "Meh." in ASCII. Then encode the result using each of the three methods above.

Question Compare the three encoding methods. Give advantages/disadvantages of each.