Lecture 10
Plan: Hamming coding (3.2.1)

Review: We talked about framing (how to divide bit streams into frames (byte count, character stuffing, bit stuffing) and error detection (difference bet detection and correction, CRC)

1. Error correction code:
Reason: repeated resending the whole frame may not be good (such as wireless networks). then we hope that the redundancy may help us correct the transmission errors.
In general, for a frame of m bits, send an additional r bits, making an m+r =n bits frame. Most of time, not all 2^n codes are valid because the r bits are constructed structurally.

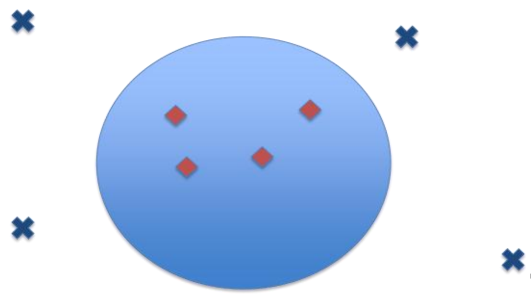
Q:/ What is the general requirement differences between detection and correction.
A:/
Define: Hamming distance
If we have two codes, the hamming distance between these two codes is the number of bit positions in which two codes differ.
e.g. 10001001 and 10110001. Their distance is 3. We can get this by using exclusive-OR and count the # of 1's we got

If we have a set of codes, then the hamming distance of the set is the minimal hamming distance of any two codes in the set.

Q:/ What's the significance?
A:/ If two code words are a hamming distance *d* apart, then it will require at least *d* single bit errors to convert one into the other.

To detect d errors, we need to construct a set of codes whose hamming distance is *d*+1.
Q:/ Why?
A:/ this ensures that d errors won't change a valid code to another valid code.

To correct d errors, we need codes with hamming distance of 2d+1.
Q:/ why?
A:/ This ensures that even after d errors, the error message is still closest to the original valid code.



The requirement of 2d+1 will make sure x are far away from each other.

Conclusion: For error correction, more redundancy is needed.
Hamming code: To correct 1 bit, we need $(m+r+1)<2^r$ (lower limit on the number of bits needed to check a single error in a m-bit word)

Q:/ why?
A:/
m: length of original message, 2^m different possible messages
r: length of my check bits
n: length of my final message =m+r

For each of the original message, there is a corresponding n bit long final message. ➔ there are n possible 1 bit error messages correspond to each original message (flip each bit) ➔ there are (n+1) patterns for an original message

$$(n+1)2^m < total\ available(2^n)$$

$$so$$

$$(m+r+1) <= 2^r$$

e.g.
if m=4, then r=3 is ok. ➔Simpliest form of hamming code: 7-4 hamming code (a whole algebra theory behind it)
4 data bits and 3 parity bits (to correct an error, we need codes that are 3 hamming distance apart)
We will use even parity in this example (the # of 1's is even)

The parity bits are placed at position 1 2, and 4 from left to the right.
parity bit 1 tell the parity of the bits whose position number in binary has a 1 in the 1's digit (odd numbers: 1, 3, 5)
parity bit 2 tell the parity of the bits whose position number in binary has a 1 in the 2's digit (2, 3, 6, 7)
similary for parity bit 4 (4, 5, 6, 7)

For example, the hamming code of the following codes are

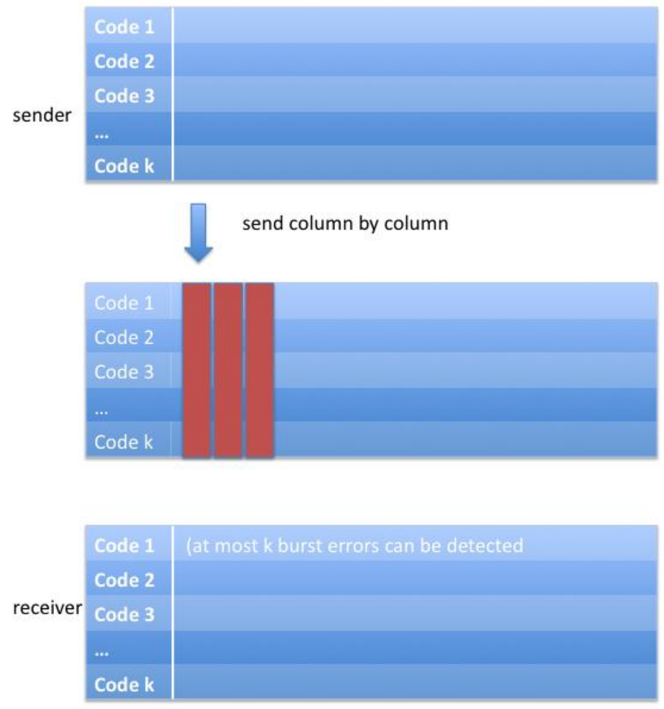
0000$\rightarrow\underline{0}\underline{0}0\underline{0}000$
0001$\rightarrow$ $\underline{1}\underline{1}0\underline{1}001$
0010$\rightarrow\underline{0}\underline{1}0\underline{1}010$
0011$\rightarrow\underline{1}\underline{0}0\underline{0}011$
etc

The hamming distance of this code is 3. Furthermore, any code with a 1-bit error will be at a Hamming distance of 1 from one valid code, but a greater distance from any other valid code.

Eg. Suppose that I receive the word 1111000.
Q:/ Is it correct? If not, can I correct any single bit error?
A:/ check bit 1: bits of 1, 3, 5, 7. OK
check bit 2: bits of 2,3, 6, 7 OK
check bit 4: bits 4, 5,6, 7 not OK.

It tells that bit 4 is wrong. The original code should be 1000

Q:/ How to correct burst errors?
A:/ use a block to correct it.



Use kr redundancy to make block km data bits immune to single burst error of length k.