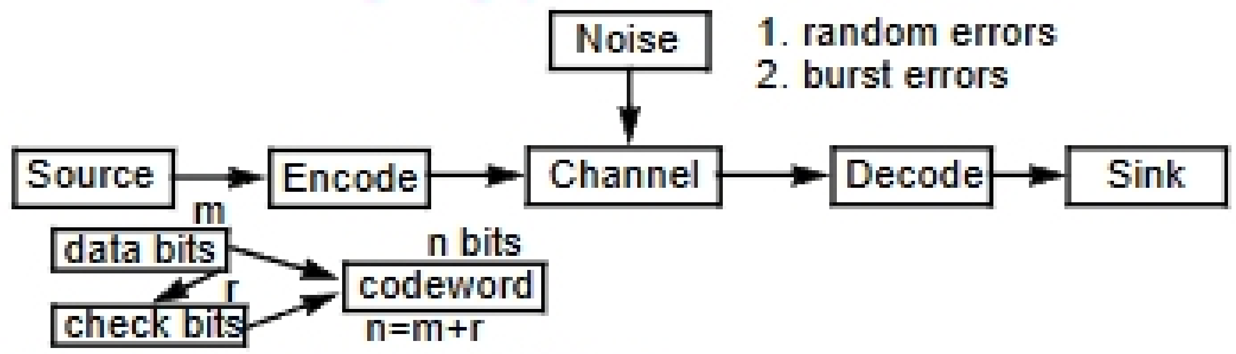
Error Control Techniques

Model of a conventional signaling system



Two categories of error control techniques

1. ARQ (automatic-repeat-request)
   buffering, error-detection-codes, acknowledgment channel, retransmission.
2. FEC (forward error control)
   error-correction codes (put enough redundancy information for correction).

FEC is inferior to ARQ except when

1. an acknowledgment channel is not available or expensive, or, even dangerous! (e.g. deep space comm. such as Voyage II)
2. the small fraction of correctable error patterns has almost all the probability weight.

Arithmetic Checksum

Error detection at the higher layer is usually done by ordinary arithmetic operations. This is simpler in software but somewhat less effective than a CRC.

Standard technique is to view packet as sequence of k numbers of n bits each, say $x_1, x_2, ..., x_k$.

Checksum is then the n bit number $x_1+x_2+...+x_k$ using ordinary arithmetic with no carry.

Alternatively, checksum might be 2n bits; first n bits is (sum) $x_1+x_2+...+x_k$ and second n bits is (sum of sum) $x_1+2x_2+3x_3+...+kx_k$.

Example: In TCP, n=16, checksum is 16 bits and one's complement of the sum.
In ISBN, the data are radix 10 digits, checksum is radix 11 digit (with 10 represented as X) and is (sum of sum of all digits)/11.

Weighted code used in ISBN number for Error Detection

Our textbook has a ISBN number 0-13-162959-X.

To check that this number is a proper ISBN number we proceed as follows:

Number	SUM	SUM of SUM
0	0	0
1	1	1
3	4	5
1	5	10
6	11	21
2	13	34
9	22	56
5	27	83
9	36	119
10=X	46	165=11x15

165 mod 11 =0 → a correct ISBN number

Coding Theory

"Coding and Information Theory", by Richard Hamming, Prentice-Hall.

A code consists of the rule/algorithm for computing check bits from data bits and for generating codewords from data bits and check bits.

The coding algorithm defines the legal or illegal codewords.

For fixed length codes,

For x, y ∈ the set of codewords, *Hamming distance*, Hd(x,y) is the no. of 1's in d, and $d=x \oplus y$.

The *Hamming distance of a code*, C, is Hd(C)=min{z | z = Hd(x,y) where x, y are code-words of C, and x≠y}.

To detect d (single) errors, we need a code C with Hd(C)=d+1.

To correct d errors, we need a code C with Hd(C)=2d+1.

Exercise: Prove the Hd(odd parity code) = 2.

For single error correcting code, where m(r) is the no. of data(check) bits,

How many check bits is required? Prove that r must satisfy $(m+r+1) \le 2^r$.

- Each of 2^m msgs has n illegal codes at Hamming distance 1 from it.
- ⇒ Each of 2^m msgs requires n+1 bit patterns from 2^n bit patterns.
- ⇒ $(n+1)2^m \le 2^n \Rightarrow (m+r+1)2^m \le 2^{m+r} \Rightarrow (m+r+1) \le 2^r$.
- For m=8, $8+r+1 \le 2^r$, ⇒ r=4.

Hamming's Single Error Correcting Code (SECC)

- It can correct single bit error. Invented by Richard Hamming.
- Codewords are encoded with check bits interleaved with data bits in the following order with check bits C_0 C_1 C_2, C_3, ..., C_n in position 1, 2, 8,..., 2^n
  3 5 6 7 9 10 11 Data bit Positions
  $C_o C_1 D_1 C_2 D_2 D_3 D_4 C_3 D_5 D_6 D_7 D_8 D_9 D_{10} D_{11} D_{12} D_{13} D_{14} D_{15}$

To generate the codeword,

- First line up the data bits on postions 3,5,6,7,9,...
- For each bit with value 1, use its bit positions in the codeword to generate binary parity bit pattern contributing to the check bits.
  e.g. if D1 is 1, then since its position in the codeword is 3, it contributes binary parity pattern of 0011. if D is 1, since its codeword position is 7, it contributes binary parity pattern of 0111.
  - The binary bit pattern corresponds to check bit $C_n...C_2C_1C_0$. Therefore 0011 pattern implies the contribution of 1 to the parity bit computation of check bits C_0 and C_1.
  - Reverse the binary bit pattern and align them beneath check bits, since in the codeword, the parity bits are lined up with $C_0C_1C_2C_3$... order.

- For each check bit, counting the parity bit contributing to it by all data bits and generate the check bit using the even parity bit.
  In the following example, C_0 receives 1 parity bit contributed by D_1 and another by D_4. With even number of parity bit contribution, we generate bit 0 for C_0. For C_2, only D_4 contribute one parity bit, therefore, we generate bit 1 for C_2 to make the total number of bit 1 even.
- After check bits are generated, the codeword is sent to the receiver.

Decoding SECC

- After receiving the codeword, the receiver will set aside those check bits and recompute the check bits only based on the data bits.
- If a check bit computed by the receiver is different from the received check bit, then mark the check position with X.
- For each error check bit position, add the value $2^{(\text{check bit position})}$ to a variable error_location. For example, in the following example, we have C_0, C_1, and C_2 error. We add error_location=$2^0+2^1+2^2=1+2+4=7$. This indicates that bit position 7 in the codeword is wrong. Let us correct it by complementing its value.