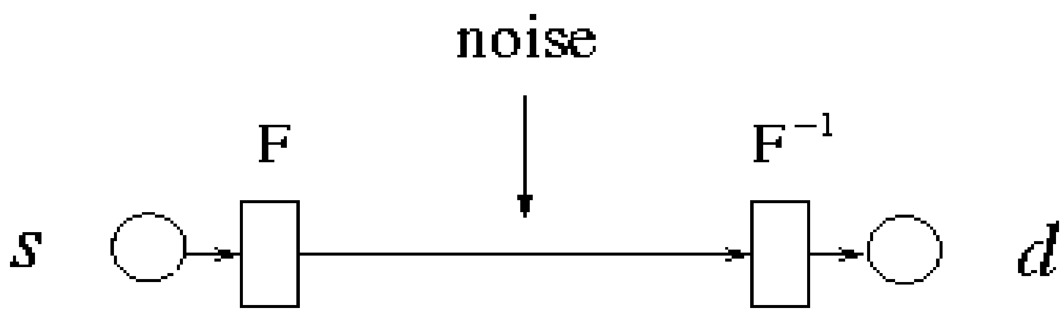
Error Detection and Correction

$\longrightarrow$ recall: reliable transmission over noisy channel



Key problem:

- sender wishes to send a; transmits code word w_a
- receiver receives w
- due to noise, w may, or may not, be equal to w_a

$\longrightarrow$ would like to detect error has occurred

$\longrightarrow$ would like to correct error

Error detection problem:

- determine if w is a valid code word

  $\longrightarrow$ i.e., for some symbol $c \in \Sigma$, $F(c) = w$

- e.g., parity bit in ASCII transmission

  $\longrightarrow$ odd or even parity

  $\longrightarrow$ limitation?

Error correction problem:

- even if $w \neq w_a$, recover symbol a from scrambled w

  $\longrightarrow$ correction is tougher than detection

- how to correct single errors for ASCII transmission?

  $\longrightarrow$ e.g., assume 21 bits available

  $\longrightarrow$ what about 14 bits?

Conceptual approach to detection & correction:

Error detection:

- valid/legal code word set $S = \{w_a : a \in \Sigma\}$
- can detect k-bit errors if

  $\rightarrow$ corrupted w does not belong to S

  $\rightarrow$ for all k-bit error patterns

  $\longrightarrow$ flipped code word cannot impersonate as valid

What kind of S can satisfy these properties?

$\longrightarrow$ e.g., ASCII with 1-bit, 2-bit, . . ., k-bit flips

$\longrightarrow$ intuition?

Key idea:

$\longrightarrow$ valid code words should not look alike

$\longrightarrow$ well-separatedness

$\longrightarrow$ "distance" between two binary strings?

Error correction:

- suppose w_a has turned into w under k-bit errors
- for all $b \in \Sigma$, calculate $d(w_b, w)$

  $\rightarrow$ use Hamming distance

  $\rightarrow$ e.g., $d(1011, 1101) = 2$
- pick $c \in \Sigma$ with smallest $d(w_c, w)$ as answer

Ex.: $0 \mapsto 000$ and $1 \mapsto 111$

$\longrightarrow$ want to send 0, hence send 000

$\longrightarrow$ 010 arrives: $d(010, 000) = 1$ & $d(010, 111) = 2$

$\longrightarrow$ conclude 000 was corrupted into 010

$\longrightarrow$ original data bit: 0

Obviously not fool-proof . . .

$\longrightarrow$ the larger k, the more distant the code words

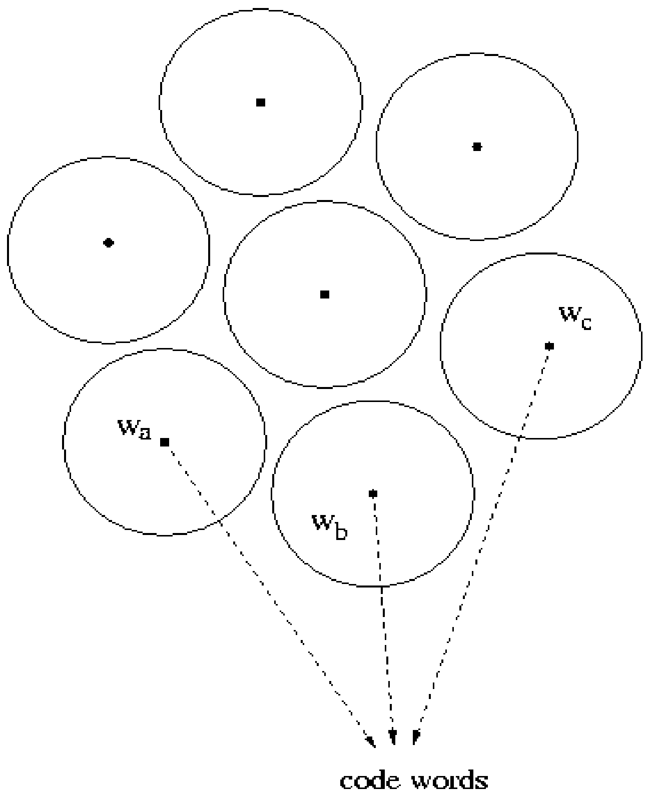
$\longrightarrow$ need a roomier playing area

$\longrightarrow$ imbed valid/legal code words

Pictorially: "ball" of radius r centered at w_a

$\longrightarrow$ $B_r(w_a) = \{w : d(w_a, w) \leq r\}$

$\longrightarrow$ well-separated code word set S layout



If k bit flips, sufficient conditions for error detection and correction in terms of $d(w_a, w_b)$ for all $a, b \in \Sigma$?

Network protocol context: different approach to detection vs. correction

$\longrightarrow$ error detection: use checksum and CRC codes

$\longrightarrow$ error correction: use retransmission

$\longrightarrow$ humans?

$\longrightarrow$ can also use FEC; for real-time data

Internet checksum: group message into 16-bit words; calculate their sum in one's complement; append "checksum" to message.

$\longrightarrow$ problem?

Cyclic redundancy check (CRC): polynomial arithmetic over finite field.

View n-bit string $a_{n-1}a_{n-2}\cdots a_0$ as a polynomial of degree $n-1$:

$$M(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_1 x + a_0.$$

Ex.: 1011 is interpreted as

$$1 \cdot x^3 + 0 \cdot x^2 + 1 \cdot x^1 + 1 \cdot x^0 = x^3 + x + 1$$

$\longrightarrow$ $M(x)$: data or message to be sent

Some facts about polynomial arithmetic:

- how do we add/subtract polynomials

  $\rightarrow$ component-wise addition/subtraction

  $\rightarrow$ "mod 2" when binary coefficients

- how do we multiply/divide polynomials?

Goal: detect multiple bit flips

Set-up: fix some generator polynomial $G(x)$ of degree k.

$\longrightarrow$ $G(x)$ "generates" (i.e., divides) code words

$\longrightarrow$ like prime number

$\longrightarrow$ choice of $G(x)$ important

Encode: Two steps

(1) Let $R(x)$ be the remainder of $x^k M(x)/G(x)$.

$\longrightarrow$ note: $x^k M(x)$ is k-bit left shift operation

$\longrightarrow$ like adding redundancy (k extra bits)

$\longrightarrow$ total length: $n + k$

$\longrightarrow$ e.g., Ethernet

(2) Set $T(x) = x^k M(x) - R(x)$.

$\longrightarrow$ $T(x)$ is the code word

$\longrightarrow$ why subtract $R(x)$?

Transmit: $T(x)$

Noise:

$\longrightarrow$ $T(x) + E(x)$ arrives at receiver

$\longrightarrow$ $E(x)$ represents the bit flips

$\longrightarrow$ degree of $E(x)$?

$\longrightarrow$ $M(x) = a$, $T(x) = w_a$, $T(x) + E(x) = w$

Decode: i.e., detect bit flip

- if $E(x) = 0$ then $(T(x)+E(x))/G(x)$: remainder $= 0$

  $\rightarrow$ no errors

- if $E(x) \neq 0$ then $(T(x)+E(x))/G(x)$: remainder $\neq 0$

  $\rightarrow$ error has occured

Is the decision rule sufficient?

Choice of $G(x)$ depends on allowed noise vector (i.e., polynomial) $E(x)$

Single bit flip:

- we have $E(x) = x^i$, $0 \le i \le n+k-1$ (i.e., a single error at position i)
- if $G(x)$ contains at least two terms, $G(x)$ will not divide $E(x)$: $G(x) = x^k + 1$

Two bit flips:

- $E(x) = x^i + x^j$ $(i > j)$

  $\longrightarrow$ write $E(x) = x^j(x^{i-j} + 1)$
- assuming x does not divide $G(x)$, it is sufficient that $G(x)$ not divide $x^{i-j} + 1$
- fact: $G(x) = x^{15} + x^{14} + 1$ will not divide $x^r + 1$ for $r < 32768$

  $\longrightarrow$ pretty long messages: meaning of r?

Burst (i.e., consecutive) errors

$\longrightarrow$ additional analysis

Ex.: commonly used CRC generator polynomials

- CRC-32: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

  $\rightarrow$ e.g., FDDI, Ethernet, WLAN

  $\rightarrow$ also used in compression

- CRC-CCITT: $x^{16} + x^{12} + x^5 + 1$ (HDLC)
- CRC-8: $x^8 + x^2 + x + 1$ (ATM)

$\longrightarrow$ guaranteed: single, double, k-burst errors

$\longrightarrow$ typically: other error patterns