CODICI RILEVATORI

CRC

La semplice **rilevazione** degli errori è più efficiente; il metodo usato comunemente è il codice **CRC** o codice a ridondanza ciclica. (Ciclic Redundancy Check).

- II bit di dati da inviare vengono considerati come **coefficienti** (di valore 0 o 1) dei termini di un polinomio ***M(x)***.
  Se i bit sono **m** essi rappresentano i coefficienti di un polinomio di termini da x^{m-1} a ***x°,*** in cui sono presenti i termini che corrispondono a bit di valore 1 e sono assenti quelli che corrispondono a bit di valore 0.

  Es.: stringa da inviare è 11001 (in questo caso m=5),

  il polinomio che la rappresenta è $M(x)= x^4 + x^3 + 1$ di grado 4

- Viene inoltre utilizzato un polinomio ***G(x),*** chiamato **polinomio generatore** (di grado r e composto da r+1 bit), noto al mittente e al ricevente, in cui **i bit di ordine più alto** e **più basso** devono essere a **1**.

  Es.: polinomio generatore $G(x)= x^3 + 1$ di grado 3 che equivale a: 1001 (4 bit)

- Sia *M(x)* la stringa di bit da inviare (di *m* bit) e *G(x)* il polinomio generatore (di r+1 bit): **il grado di *M(x)* deve essere maggiore di quello di *G(x),* cioè *m > r*.**

- Il metodo si basa sul calcolo di una **checksum** che dipende da *M(x)* e *G(x),* da aggiungere in fondo a *M(x)* in modo che la stringa ottenuta sia divisibile per *G(x).*

- II mittente invia un frame composto dai bit di dati *(M(x))* e dalla checksum.

Per **calcolare la checksum** di un pacchetto *M(x)* di *m* bit si procede così:

- **si appendono *r* bit 0** (dove r è il grado di *G(x)*) a destra di *M(x)* in modo da ottenere il polinomio ***P(x)***$= x^r M(x)$ di m + r bit;
- **si divide** il polinomio ottenuto *P(x)* per *G(x);*
- **si sottrae il resto** (che è sempre di r bit o meno) da *P(x);* il risultato è il frame completo di checksum da trasmettere; **questo frame è divisibile per *G(x)*** *(resto 0).*

Per verificare la corretta ricezione basta che il ricevente divida il frame per *G(x);* se la divisione dà un resto vuol dire che si è verificato un errore.

Le operazioni sui polinomi vengono eseguite in *aritmetica modulo 2*: *non ci sono riporti per l'addizione o prestiti per la sottrazione*; addizione e sottrazione sono identiche all'*or esclusivo.* La divisione viene eseguita come se fosse binaria ma la sottrazione è eseguita in modulo 2 (in pratica un divisore è contenuto nel dividendo se il dividendo ha tanti bit quanti il divisore, contando le cifre dal primo 1 a sinistra).

Questo metodo non è infallibile; possono verificarsi degli errori nel frame inviato, che danno comunque 0 come risultato della divisione del frame per *G(x),* e che quindi non vengono rilevati.

La scelta del polinomio *G(x)* è importante per la bontà del metodo.

Alcuni polinomi che sono diventati standard internazionali sono:

CRC-8 ATM(HEC) $= X^8+X^2+X+1$
CRC-12 $= X^{12}+X^{11}+X^3+X^2+X^1+1$
CRC-16 $= X^{16}+X^{15}+X^2+1$
CRC-32 (Ethernet FDDI) $= X^{32}+ X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+ X^{10}+X^{11}+X^8+ X^7+ X^5+ X^4+ X^2+ X+ 1$
CRC-CCITT $= X^{16}+X^{12}+X^5+1$

Esempio

Sia ***M(x)*** = 010110100. $x^7 + x^5 + x^4 + x^2$ di grado 8 (il termine x^8 ha coefficiente 0)
Sia ***G(x)*** = x^4+x^2+1 di grado 4 cioè 10101 polinomio generatore
Il grado del polinomio generatore è 4 (lunghezza – 1) quindi *P(x)* è 0101101000000

```
0101101000000:10101 = 10011011
----111
    1110
    11100
    -10010
     --1110
       11100
       -10010
        --111    resto
```

0101101000000 - 111 → 0101101000111 **sequenza che viene inviata**

Il destinatario deve dividere questo polinomio ricevuto per *G(x)* e ottenere resto 0 (se il resto non è 0 si è verificato un errore).

```
0101101000111:10101 = 10011011
----111
    1110
    11100
    -10010
     --1111
       11111
       -10101
        -----
```

CODICI CORRETTORI

CODICE DI HAMMING

Il codice di Hamming è un codice che permette di aggiungere un certo numero di bit ai bit di dati in modo da comporre parole con **distanza 3** in grado di rilevare e correggere errori su un singolo bit. Il numero di bit da aggiungere aumenta all'aumentare del numero dei bit di dati.

I bit aggiunti sono **bit di parità** calcolati su sottoinsiemi di bit della parola di codice; numerando a partire da 1 a sinistra i bit che compongono la parola di codice, i bit di parità vengono inseriti nelle posizioni che sono potenze di 2 (1, 2, 4, 8,16 ...); gli altri bit sono i bit di dati. Ogni bit di parità viene calcolato su un sottoinsieme di bit; ogni bit di dati può essere incluso in diversi sottoinsiemi e influire su diversi bit di parità. Per sapere su quali bit di parità influisce il bit di dati *k* basta riscrivere *k* come somma di potenze di 2 (per esempio 11 =1 +2 + 8); ogni bit di dati è controllato da tutti e soli i bit di parità che appartengono alla sua espansione (il bit 11 è controllato dai bit 1, 2 e 8).

In ricezione, per ogni parola di codice vengono ricalcolati i bit di parità per ogni posizione *k* (con k= 1, 2, 4, 8,16...). Se la parità non è corretta viene aggiunto *k a* un contatore inizializzato a 0; al termine il valore del contatore indica la posizione del bit errato (se non sono corretti i bit di parità 1, 2 e 8 il bit errato è quello in posizione 11).

Esempio:

```
                    (1 2 3 4) posizioni
Partendo dalla sequenza:  0 1 1 0
```

si calcolano e si inseriscono i *bit di parità* nelle posizioni 1, 2 e 4:

```
(1 2 3 4 5 6 7) posizioni
 _ _ 0 _ 1 1 0
```

quindi *i bit di dati* occupano le posizioni 3, 5, 6 e 7.

```
(1 2 3 4 5 6 7) posizioni
 _ _ 0 _ 1 1 0
     1   1   1
     2     2 2
           4 4 4
```

Il bit 3 influenza i bit di parità 1 e 2 (3=1+2). il bit 5 influenza i bit di parità 1 e 4 (5=1+4). il bit 6 influenza i bit di parità 2 e 4 (6=2+4). il bit 7 influenza i bit di parità 1, 2 e 4 (7=1+2+4).

Il bit di parità 1 è calcolato sui bit di dati 3, 5 e 7 (0 1 0) e quindi vale 1. Il bit di parità 2 è calcolato sui bit di dati 3, 6 e 7 (0 1 0) e quindi vale 1. Il bit di parità 4 è calcolato sui bit di dati 5, 6 e 7 (1 1 0) e quindi vale 0.

Perciò si ottiene la sequenza: 1 1 0 0 1 1 0

(1 2 3 4 5 6 7) posizioni

Se un errore modifica la sequenza in: 1 1 0 0 1 **0** 0

ricalcolando i bit di parità si ottiene:
bit di parità 1 calcolato sui bit di dati 3, 5 e 7 (0 1 0): 1
bit di parità 2 calcolato sui bit di dati 3, 6 e 7 (0 0 0): 0
bit di parità 4 calcolato sui bit di dati 5, 6 e 7 (1 0 0): 1

Confrontando i bit della sequenza con quelli calcolati viene incrementato il contatore k:
il bit di parità 1 è uguale: k=0;
il bit di parità 2 è diverso: k=2 (+2);
il bit di parità 4 è diverso: k=6 (+4).

Pertanto si può stabilire che c'è un bit errato nella posizione 6 e lo si può correggere riottenendo la sequenza: 1 1 0 0 1 1 0.

Con questa codifica, se ***m*** **sono i bit che costituiscono l'informazione**, il numero ***r*** **di bit da aggiungere** (ridondanti) si calcola come:

il minimo ***r*** tale che $2^r >= m + r + 1$