Codici per il rilevamento degli errori

I codici correttori di errore sono usati raramente (ad esempio in presenza di trasmissioni simplex, nelle quali non è possibile inviare al mittente una richiesta di ritrasmissione), perché in generale è più efficiente limitarsi a rilevare gli errori e ritrasmettere saltuariamente i dati piuttosto che impiegare un codice (più dispendioso in termini di ridondanza) per la correzione degli errori.

Infatti, ad esempio, supponiamo di avere:

- canale con errori isolati e probabilità di errore uguale a 10^{-6} per bit;
- blocchi dati di 1.000 bit.

Per correggere errori singoli su un blocco di 1.000 bit, ci vogliono 10 bit, per cui un Megabit richiede 10.000 check bit.

Viceversa, per rilevare l'errore in un blocco, basta un bit (con parity code). Ora, con 10^{-6} di tasso d'errore, solo un blocco su 1.000 è sbagliato e quindi deve essere ritrasmesso. Di conseguenza, per ogni Megabit si devono rispedire 1.001 bit (un blocco più il parity bit).

Dunque, l'overhead totale su un Megabit è:

- 1.000 bit per parity bit su 1.000 blocchi,
- 1.001 bit per il blocco ritrasmesso,
- per un totale di 2.000 bit contro i 10.000 del caso precedente.

L'uso del parity bit può servire (con un meccanismo analogo a quello visto per la correzione di burst di k errori) per rilevare burst di errori di lunghezza <= k. La differenza è che non si usano r check bit per ogni codeword, ma uno solo.

Esiste però un altro metodo che nella pratica viene usato quasi sempre, il ***Cyclic Redundancy Code*** (***CRC***), noto anche come ***polynomial code***. I polynomial code sono basati sull'idea di considerare le stringhe di bit come rappresentazioni di polinomi a coefficenti 0 e 1 (un numero ad m bit corrisponde ad un polinomio di grado m-1).

Ad sempio, la stringa di bit 1101 corrisponde al polinomio $x^3 + x^2 + x^0$.

L'aritmetica polinomiale è fatta modulo 2, secondo le regole della teoria algebrica dei campi. In particolare:

- addizione e sottrazione sono equivalenti all'or esclusivo (non c'è riporto o prestito);
- la divisione è come in binario, calcolata attraverso la sottrazione modulo 2.

Il mittente ed il destinatario si mettono d'accordo su un polinomio generatore G(x), che deve avere il bit più significativo e quello meno significativo entrambi uguali ad 1. Supponiamo che G(x) abbia r bit.

Il frame M(x), del quale si vuole calcolare il checksum, dev'essere più lungo di G(x). Supponiamo che abbia m bit, con m > r.

L'idea è di appendere in coda al frame un checksum tale che il polinomio corrispondente (che ha grado m + r -1) sia divisibile per G(x).

Quando il ricevitore riceve il frame più il checksum, divide il tutto per G(x). Se il risultato è zero è tutto OK, altrimenti c'è stato un errore.

Il calcolo del checksum si effettua come segue:

1. Appendere r bit a destra del frame, che quindi ha m+r bit, e corrisponde ad $x^r M(x)$;
2. Dividere $x^r M(x)$ per G(x);
3. Sottrarre ad $x^r M(x)$ il resto della divisione effettuata al passo precedente. Ciò che si ottiene è il frame più il checksum da trasmettere, che è ovviamente divisibile per G(x). Si noti che di fatto questa è un'operazione di XOR fatta sugli r bit meno significativi, e quindi non modifica il frame.

Questo metodo è molto potente, infatti un codice polinomiale con r bit:

- rileva tutti gli errori singoli e doppi;
- rileva tutti gli errori di x bit, x dispari;
- rileva tutti i burst di errori di lunghezza <= r.

Tra i polinomi sono diventati standard internazionali:

- ***CRC-12***: $x^{12} + x^{11} + x^3 + x^2 + x^1 + 1$;
- ***CRC-16***: $x^{16} + x^{15} + x^2 + 1$;
- ***CRC-CCITT***: $x^{16} + x^{12} + x^5 + 1$;

Un checksum a 16 bit corregge:

- errori singoli e doppi;
- errori di numero dispari di bit;
- errori burst di lunghezza <= 16;
- 99.997% di burst lunghi 17;
- 99.998% di burst lunghi 18.

Questi risultati valgono sotto l'ipotesi che gli m bit del messaggio siano distribuiti casualmente, il che però non è vero nella realtà, per cui i burst di 17 e 18 possono sfuggire più spesso di quanto si creda.