Data Link: consente comunicazioni affidabili ed efficienti tra due macchine *adiacenti* (connesse da un cavo coassiale, doppino, linea telefonica,..).

I protocolli del livello Data Link devono tener conto del fatto che:

***ci sono *errori e disturbi occasionali*;**

***il canale ha un *data rate finito*;**

***c'è un *ritardo nella propagazione*.**

Quando attraverso un canale (ad es. una linea fisica) arrivano dei bit alla scheda di rete di un Router, l'***HW*** della scheda di rete, preposto alle operazioni del ***livello 1***, li rileva e li passa alla sezione ***HW/SW*** (anch'essa presente sulla scheda di rete) preposta alle operazioni del ***livello 2*** (***data link***).

L'***HW/SW*** di ***livello 2*** effettuati i controlli di:

framing;
errori di trasmissione;
numero di frame;

se tutto è OK, genera un interrupt alla cpu del router.

L’interrupt attiva e passa il frame arrivato ad un ***processo*** di sistema (SW di livello tre).

Il ***processo*** ricostruisce il pacchetto ed in base all'indirizzo del destinatario, contenuto nel pacchetto, determina su quale linea rimetterlo in uscita.

Restituisce il pacchetto all’ HW/SW di ***livello due*** che lo imbusta in un nuovo frame, e lo consegna al sottostante ***livello fisico***, che provvederà ad inoltrarlo sulla linea prescelta.

Il ***livello 1*** accetta un flusso di bit grezzi e cerca di farli arrivare a destinazione.

Purtroppo

*il flusso non è esente da errori;

*possono arrivare più o meno bit di quanti sono stati inviati.

E' compito del ***livello 2*** rilevare, e se possibile correggere, tali errori.

Operazioni del livello due

in trasmissione:

*suddivide il flusso di bit, che arriva dal ***livello tre***, in una serie di frame;

*calcola una funzione (***checksum***) per ciascun frame;

*inserisce il checksum nel frame;

*consegna il frame al livello uno, il quale lo spedirà come sequenza di bit.

in ricezione:

°riceve una sequenza di bit dal livello uno;

°ricostruisce da essa un frame dopo l'altro;

°per ciascun frame ricalcola il checksum;

°se il checksum ricalcolato è diverso da quello contenuto nel frame, il frame viene scartato.

Per delimitare i frame è rischioso usare lo spazio temporale che li separa, perché nelle reti (in genere) è difficile garantire una perfetta temporizzazione.

Metodi impiegati per indicare l'inizio e la fine dei frame:

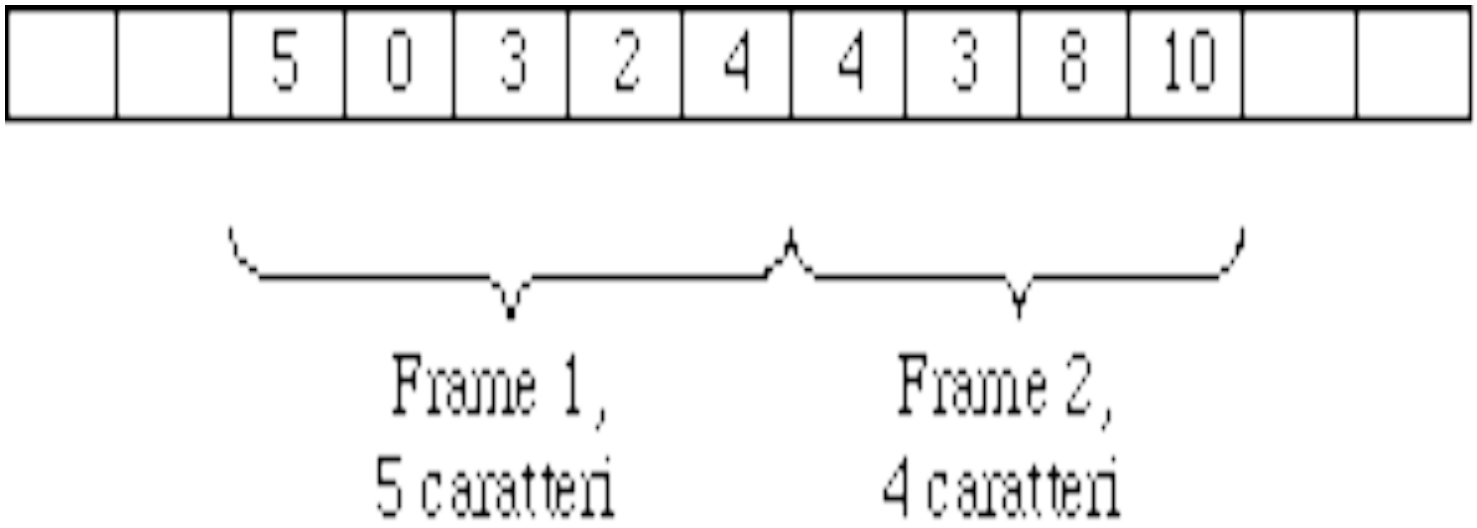
* ***conteggio dei caratteri***;

* ***caratteri di inizio e fine*** (***character stuffing***);

* ***bit pattern di inizio e fine*** (***bit stuffing***);

* ***violazioni della codifica*** del livello fisico.

Un campo dell'header indica quanti sono i caratteri del frame



Eventuali errori o perdita di tale campo rendono impossibile l'individuazione dell'inizio del prossimo frame e dei successivi (metodo poco usato per la scarsa affidabilità).

Character stuffing: Ogni frame inizia e finisce con una particolare sequenza di caratteri ASCII (***DLE***: Data Link Escape):

*****DLE STX*** (Start of TeXt): inizio frame;

* ***DLE ETX*** (End of TeXt): fine frame.

Se si perde traccia dei confini di un frame la si riacquista all'arrivo della prossima coppia di ***DLE STX*** - ***DLE ETX***.

Character stuffing: poiché il byte corrispondente alla codifica dei *DLE* potrebbe coincidere con byte del frame da trasmettere

*il Data Link aggiunge, per ciascuno di tali byte, un altro *DLE* (quindi solo i singoli *DLE* segnano i confini dei frame).

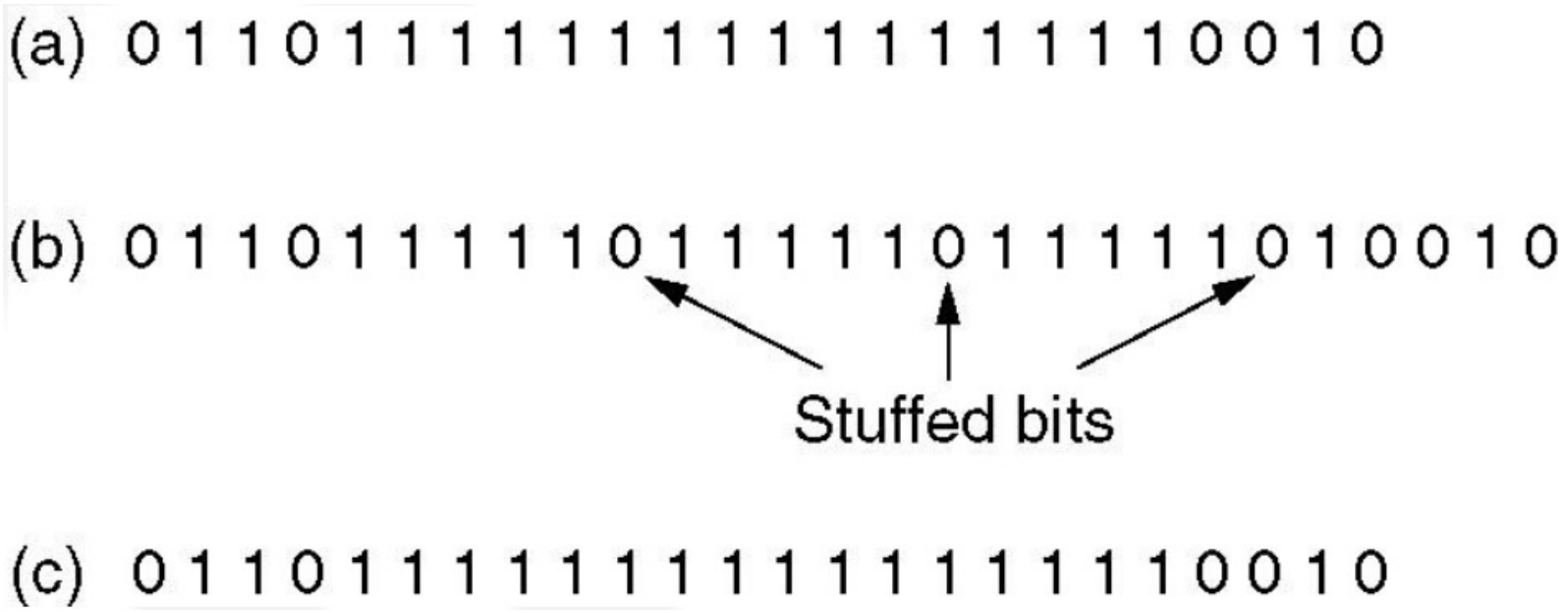
*A destinazione il Data Link rimuove tutti i DLE prima di consegnarli al livello tre.

Bit stuffing: ogni frame inizia e finisce con una specifica sequenza di bit (***flag byte***), ad es.: 01111110

Poiché il ***flag byte*** può far parte dei dati che devono essere trasmessi, il data link provvede ad inserire specifici bit che, coerentemente saranno rimossi all'arrivo.

Ad esempio nel caso in esame, ogni volta che il Data Link incontra 5 bit 1 consecutivi inserisce uno zero aggiuntivo. In tal modo il flag byte può apparire solo all'inizio ed alla fine dei frame.

Coerentemente, quando a destinazione compaiono 5 bit uguali a uno, il Data Link rimuove lo zero che li segue.



a) Dati originali
b) Dati trasmessi sul canale (stuffed)
c) Dati ricevuti dopo il de-stuffing

Per motivi fisici in molte reti (sopratutto LAN) i bit si codificano con una certa ridondanza.

Manchester encoding (IEEE 802.3):

bit dati 1: coppia di bit fisici ***high/low*** ;
bit dati 0: coppia di bit fisici ***low/high*** .

Le coppie ***low/low*** ed ***high/high*** non sono utilizzate, quindi possono essere impiegate per delimitare i frame.

La ***Manchester encoding*** (bandwidth doppia) prevede una transizione per ogni bit dato.

Molti fattori possono provocare errori, sopratutto sul local loop e nelle trasmissioni wireless (mentre sono piuttosto rari nei mezzi più moderni quali le fibre ottiche).

Gli errori sono dovuti in generale a:

*rumore di fondo;
*disturbi improvvisi (ad es. fulmini) ;
*interferenze (ad es. motori elettrici).

Due approcci al trattamento degli errori:

Rilevazione dell'errore: includere informazione aggiuntiva, in modo da accorgersi che c'è stato un errore;

Correzione dell'errore: includere abbastanza informazione aggiuntiva in modo da poter ricostruire, in caso di errori, il messaggio originario.

Un frame (a parte i delimitatori) consiste di

n = (m + r) bit

m bit di messaggio vero e proprio;

r bit ridondanti (***reduntant bit*** o ***check bit***).

Codeword (***parola codice***): sequenza di **m** + **r** bit.

Attraverso una XOR bit a bit tra due codeword è possibile determinare la loro ***distanza di Hamming*** (ossia il numero di bit in cui esse differiscono).

Se due codeword hanno una distanza di Hamming uguale a ***d***, ci vogliono esattamente ***d*** errori su singoli bit per trasformare l'una nell'altra.

Codice (***code***): insieme prefissato di codeword.

La ***distanza di Hamming di un codice*** *è il minimo delle distanze di Hamming fra tutte le possibili coppie di codeword del codice.*

Per rilevare d errori serve un codice con distanza di Hamming (d+1).

Qualunque combinazione di ***d*** errori non riesce a trasformare una codeword valida in un altra codeword valida.

Per correggere d errori, serve un codice con distanza di Hamming (2d +1).

Una codeword contenente fino a ***d*** errori è più vicino a quello originario che a qualunque altro codeword valido.

A seconda degli scopi, ed in funzione di ***m***, si progetta un algoritmo per il calcolo degli ***r*** *check bit,* in modo che le parole codice risultanti di ***n*** = ***m*** + ***r*** bit costituiscano un codice con la desiderata distanza di Hamming.

Codice con bit di parità pari (even parity):

<-- m -->	r
1011	0101	1
1000	0111	0

Si contano il numero di bit 1 nella codeword!
Dispari: 1
Pari: 0

Il ***codice di parità*** (***parity code***) ha distanza di Hamming uguale a due (per ottenere una codeword valida devono cambiare due bit, infatti un singolo errore produce un numero dispari di 1 e quindi una parola codice non valida).

Si considerino le seguenti codeword a distanza 5:

00000 00000
00000 11111
11111 00000
11111 11111

Se si invia la codeword ***00000 11111***
ma arriva: **00000 00111** (2 errori)

la codeword più vicina è: ***00000 11111***

Se invece arriva: **00000 00011** (tre errori)

la codeword più vicina è: **00000 00000**

Per correggere un singolo errore su ***m*** bit, si devono impiegare almeno ***r*** check bit, con ***r*** tale che:

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

Se le codeword legali sono 2^m, poiché per ogni codeword di $n=m+r$ bit abbiamo $n+1$ codeword a distanza 1, il numero di bit n deve essere tale che

$$2^n = 2^{m+r} > 2^m * (n+1) \implies 2^r > (n+1) = (m+r+1)$$

Per correggere un singolo errore su m bit, si devono impiegare almeno r check bit, con r tale che:
$2^r >= (m + r + 1)$

Per correggere un solo errore occorrono quindi circa $\lg_2 n$ redundant ceck bit.

Se le informazioni che intendiamo trasmettere sono caratteri ASCII, quindi m = 7, poiché il numero ***r*** di redundant bit deve essere tale che

$2^r >= (m + r + 1) >= (7+r+1)$

segue $r = 4 \quad \Longrightarrow \quad n = m+r = 7+4 = 11$

Occorre un algoritmo che, in base alla posizione ed al valore dei quattro bit di controllo, consenta la correzione del bit errato.

Es. si dispongono gli ***r*** redundant bit nelle posizioni 2^i con $0 \leq i < r$, assegnandogli come valore …

Consideriamo un carattere ASCII ed ipotizziamo che nella trasmissione si verifichi un errore. Cioè inviamo (A): 100 0001, ed arriva (Q) : 101 0001

11= 8+**2**+1; 10=8+**2**; 9=8+1; **8*=8**;
7=4+**2**+1; 6=4+**2**; 5=4+1; 4*=4;
3=**2**+1; 2*=**2**; 1*=1;

il bit di parità **2*** è calcolato sui bit di posto: 11, 10, 7, 6, 3, 2

1*	2*	3	4*	5	6	7	8*	9	10	11
0	0	1	0	0	0	0	1	0	0	1

+ -> Addizione senza riporto == XOR

Inviato (A) è arrivato (Q)

1 *	2 *	3	4 *	5	6	7	8 *	9	10	11
0	0	1	0	0	**0**	0	1	0	0	1
0	0	1	**0**	0	***1***	0	1	0	0	1

11= 8+**2**+1; 10=8+**2**; 9=8+1; 8=8;

7=**4**+**2**+1; 6=**4**+**2**; 5=**4**+1; 4= **4**;

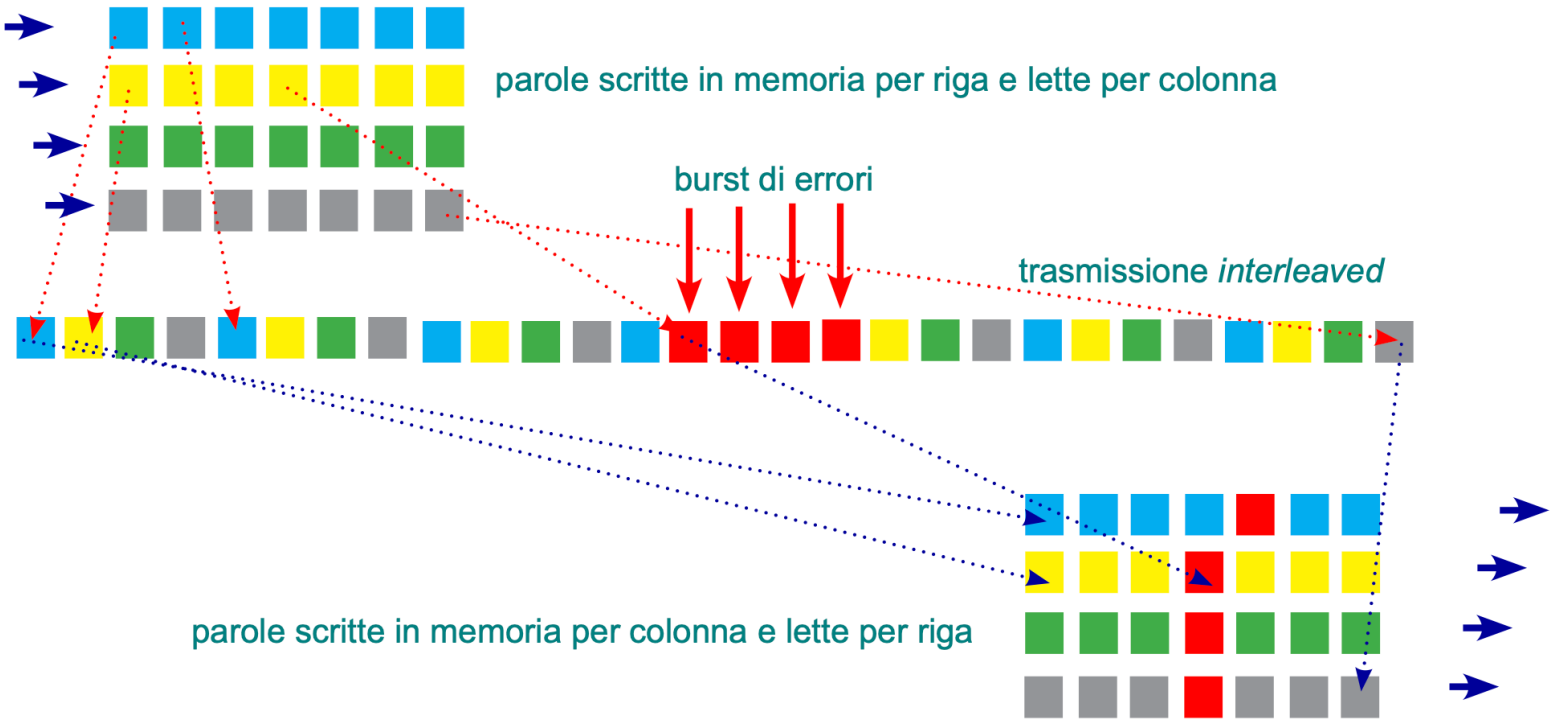
3=**2**+1; 2=**2**; 1=1;

Risultano errati i bit di parità di posto **2*** e **4***, quindi il bit errato è quello di posto 6=2+4

Tecnica per correggere ***burst di errori*** (gruppi contigui di errori) di lunghezza massima prefissata k:

- si accumulano k codeword, riga per riga;
- i codeword si trasmettono per colonne;
- quando arrivano si riassemblano per righe.

Poiché un burst di k errori comporta un singolo errore in ciascuna delle K codeword; correggendo ciascuna codeword ricostruiamo l'intero burst.



Elettronica e Telecomunicazioni N° 1, Aprile 2008 (www.crit.rai.it)

Tecnica per rilevare ***burst di errori*** di lunghezza massima k:

- si accumulano k codeword di n bit, secondo una matrice di K righe e n colonne;
- si aggiunge il parity bit per ciascuna riga formando una colonna aggiuntiva;
- i dati vengono trasmessi per colonne;
- ricevuto l'intero blocco, si controllano tutti i bit di parità e nel caso di errori si richiede la ***ritrasmissione del blocco***.

I codici correttori di errore, ad eccezione delle trasmissioni simplex dove non è possibile inviare al mittente una richiesta di ritrasmissione, sono usati molto raramente.

E' più efficiente limitarsi a rilevare gli errori, e ritrasmettere saltuariamente i dati errati, piuttosto che impiegare un codice (più dispendioso in termini di ridondanza) per la correzione degli errori.

Nel caso di trasmissione di 1 Mbit, con frame di 1000 bit e tasso d'errore di 10^{-6}, solo un blocco su 1000 deve essere ritrasmesso.

Overhead per la trasmissione di un Mbit:

*parity bit: 1.000 bit di parità (uno per blocco) più 1.001 bit per il blocco errato che va ritrasmesso (overhead di 2001 bit);

* codici correttori: per ogni blocco di 1000 bit occorrono 10 redundant ceck bit (ossia un overhead di 10*1000 bit).

Codici correttori e parity bit riga-colonna sono raramente usati.

Nella pratica viene quasi sempre usato il ***CRC*** (***Cyclic Redundancy Code*** o ***polynomial code***): un numero di m bit corrisponde ad un polinomio di grado *m-1*

Ad esempio, la stringa di bit **1101** corrisponde al polinomio $\mathbf{x}^3 + \mathbf{x}^2 + \mathbf{x}^0$.

L'aritmetica polinomiale utilizzata per il calcolo del ***CRC*** è sviluppata modulo 2, in particolare:

*addizione senza riporto e sottrazione senza prestito (equivalenti all'or esclusivo);

*divisione calcolata attraverso la sottrazione modulo 2.

Mittente e destinatario adottano lo stesso polinomio generatore G(x) (il bit più significativo e quello meno significativo devono essere entrambi 1).

Se ***r*** è il grado di **G(x)**, ed ***m*** è il numero di bit del frame **M(x)**, ***m*** deve essere maggiore di ***r*** { **M(x)** più lungo di **G(x)** }.

Si appende in coda al frame un checksum tale da originare un ***polinomio*** (***frame + cecksum***) di grado (***m*** + ***r***) che sia divisibile per **G(x)**.

Quando il ricevitore riceve il polinomio, lo divide 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) Si appendono ***r*** bit **0** a destra del frame ottenendo $\mathbf{M(x)x^r}$ di (***m***+***r***) bit;

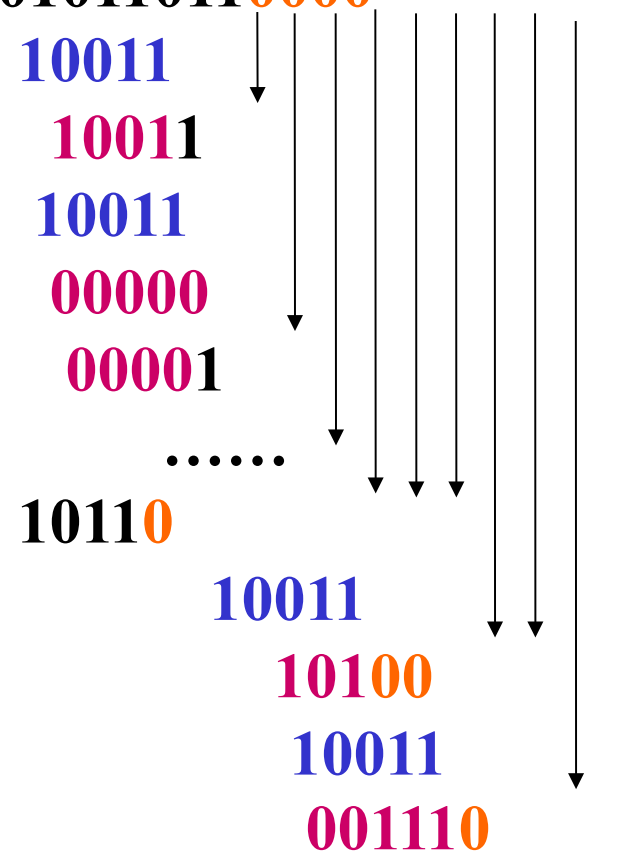
2) Si divide $\mathbf{M(x)x^r}$ per **G(x)**;

3) Si ottiene il frame da trasmettere sottraendo ad $\mathbf{M(x)x^r}$ il resto della divisione.

La sottrazione (XOR) fatta sugli ***r*** bit meno significativi, non modifica il frame che, vista la costruzione, risulta divisibile per **G(x)**.

Frame **1101011011** Generatore G(x)=**10011** (grado 4)

Frame + quattro 0: **11010110110000**



1110 (Resto)

11010110110000

1110

11010110111110 ***Frame trasmesso***

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***

Polinomi divenuti 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 rileva:

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.

Risultati validi nell'ipotesi che gli ***m*** bit del messaggio siano distribuiti casualmente (ipotesi poco probabile, quindi i burst di 17 e 18 sfuggono più spesso di quanto si creda.)