Frame : 1 1 0 1 0 1 1 0 1 1

Generator: 1 0 0 1 1

Message after 4 zero bits are appended: 1 1 0 1 0 1 1 0 1 1 0 0 0 0

Vediamo:

```
                1 1 0 0 0 0 1 0 1 0
        ___________________________
1 0 0 1 1 | 1 1 0 1 0 1 1 0 1 1 0 0 0 0
            1 0 0 1 1
            ---------
              1 0 0 1 1
              1 0 0 1 1
              ---------
                0 0 0 0 1
                0 0 0 0 0
                ---------
                  0 0 0 1 0
                  0 0 0 0 0
                  ---------
                    0 0 1 0 1
                    0 0 0 0 0
                    ---------
                      0 1 0 1 1
                      0 0 0 0 0
                      ---------
                        1 0 1 1 0
                        1 0 0 1 1
                        ---------
                          0 1 0 1 0
                          0 0 0 0 0
                          ---------
                            1 0 1 0 0
                            1 0 0 1 1
                            ---------
                              0 1 1 1 0
                              0 0 0 0 0
                              ---------
                                1 1 1 0   <- Remainder
```

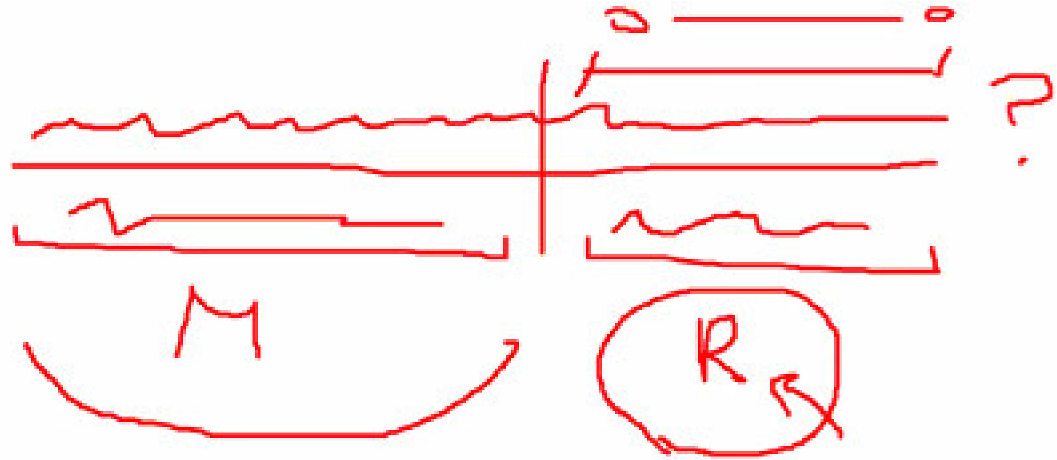
Transmitted frame: 1 1 0 1 0 1 1 0 1 1 1 1 1 0

Decoding?

- Il ***decoding*** è banale:
- Tagliamo via la parte che avevamo aggiunto, con uno shift a destra
- Con gli occhiali dei polinomi?
  → **si divide per x^r**

E l’error detection?

Vediamo…



- Abbiamo trasmesso M(x) seguito da R(x)
- “seguito da”…? Equivale a:
  x^r * M(x) + R(x)

Ma…

- **x^r * M(x) + R(x)** …
- … per le "magiche" proprietà di GF(2)[] è lo stesso che…
- **x^r * M(x) - R(x)**
- Quindi cos'abbiamo fatto? Abbiamo preso un numero (x^r * M(x)), diviso per G(x) trovando un certo resto, e sottratto il resto da quel numero
- →***abbiamo un numero divisibile per G(x)!***

Quindi…

- L’error detection è semplicemente prendere il numero trasmesso, e calcolare il resto della ***divisione*** per **G(x)**
- … che per quanto detto prima, dovrebbe essere **zero** (!)
- Quindi, se il resto è **0**, tutto ok, altrimenti c’è stato un ***errore***!

Potenza?

- Qual è la potenza di un tale metodo?
- Supponiamo ci sia un ***errore***, sono bits che sono stati invertiti
- In altre parole, per l'aritmetica di GF(2)[] è lo stesso che sommare un ***polinomio di errore***, diciamo E(x)

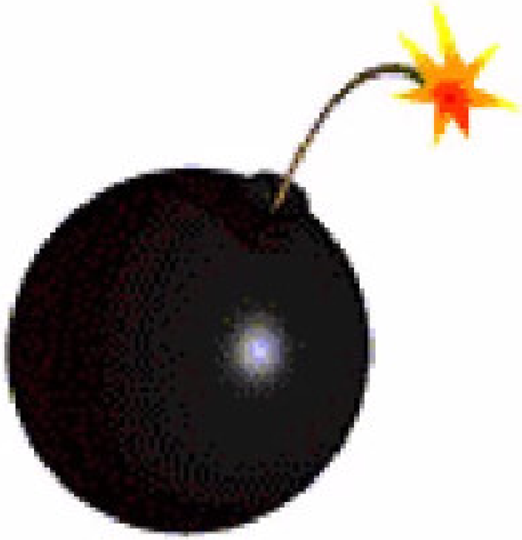
$$\underbrace{1\cdot x^2}+\underbrace{1\cdot x^1}+1\cdot x^0 \quad\Big|\quad 1 \;\; \overset{0}{1} \;\; 1$$

$$E(x)=1\cdot x^2+1\cdot x^1 \qquad 0$$

Allora…

- Abbiamo il polinomio trasmesso, T(x), più un ***polinomio di errore*** E(x), e calcoliamo il ***resto*** della divisione per G(x):
- (T(x)+E(x)) mod G(x) =
(sappiamo che T(x) mod G(x) = 0)
→ = E(x) mod G(x)
- Quindi, non riusciamo a trovare l'errore solo quando E(x) mod G(x) = 0
←→ **E(x) è divisibile per G(x)**

Singoli errori



- Abbiamo **E(x)=x^i**
- →basta che G(x) abbia due o più termini

x^{i+j}

$x^i \quad x^j$

x^i

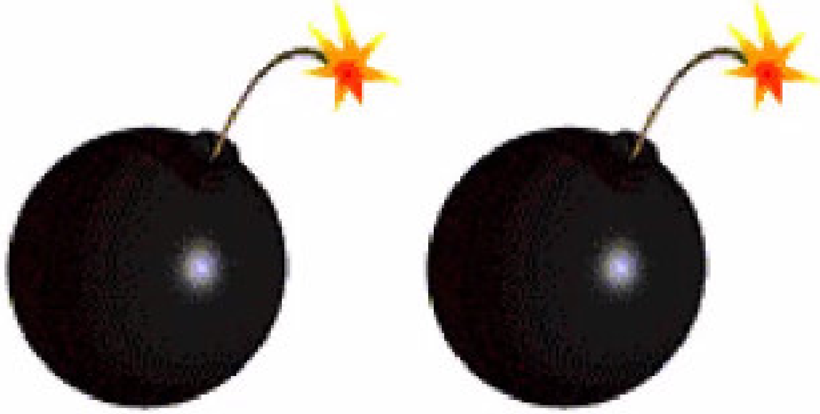
$\cancel{x}\; A(x) \cdot B(x)$

$x^i \quad x^{i-1} \quad x^{i-2} \; \ldots$

$(x+1) \cdot M$

$G(x)$

Doppi errori



- **$E(x)=x^i+x^j$**
- $= x^j * (x^{(i-j)}+1)$
- → basta che $G(x)$ non sia una potenza di x, e che non divida x^k+1 per ogni k fino al massimo valore di $i-j$

Ogni errore con un numero dispari di bits

$1 \quad (1+1)\cdot \ldots (1)$

$G(x) = (x+1)\cdot \ldots$

- Basta che x+1 sia un fattore di G(x) (deriva dagli zeri dei polinomi)

$E(x) \bmod G(x) = 0$

$E(x) = x^n + x^m + \ldots + x^k = 1$

DISPARI

$E(x) = G(x)\cdot \ldots$

$E(1) = G(1)\cdot \ldots(1) = 0$

$i+j$ x i

I burst errors

- **E(x)= x^i(x^j+…+1)** è il burst error di lunghezza j+1

In generale…

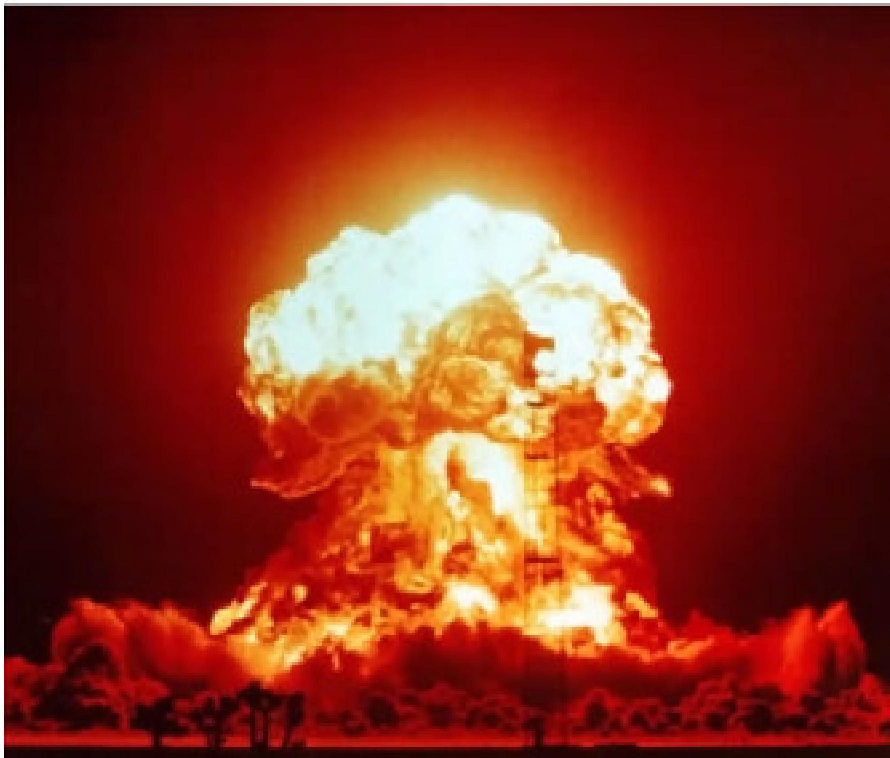
- "+1" buona scelta per un polinomio
- Vaste opzioni: possiamo ***combinare*** polinomi per avere il meglio che ognuno ci offre

Oppure

- Sceglierne (per quanto possibile) di ***irriducibili***

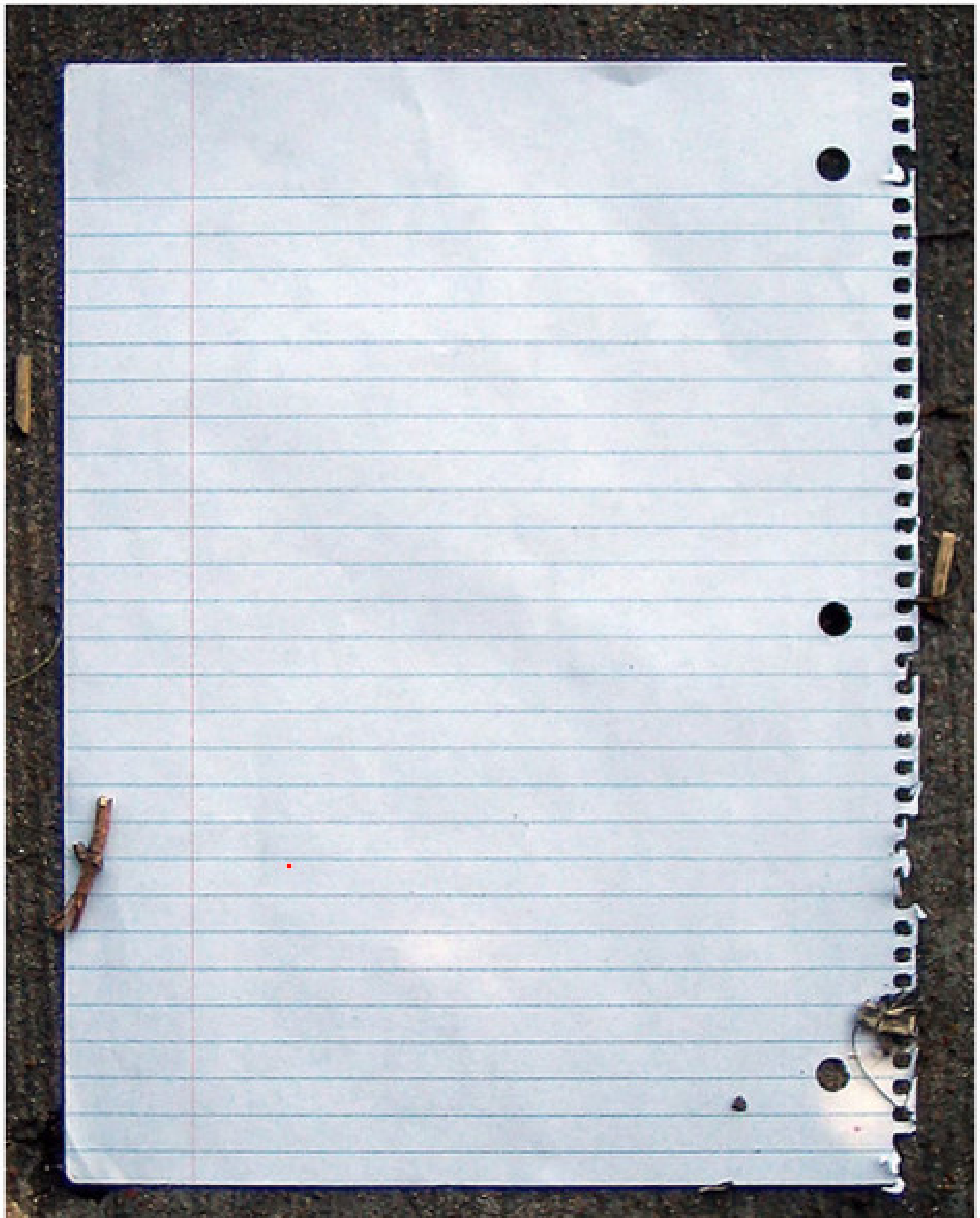
In generale

- All'aumentare del grado di G, aumenta la potenza: ogni burst error di lunghezza arbitraria più grande del grado di G
  → probabilità
  0.5^grado(G(x))



Cosa vuol dire in pratica?

- Protezione esponenziale col grado di G…
- Potenza dell’esponenziale (sempre sentito nominare, ma in pratica…?)



2011, record del mondo…

Pieghiamo…

- … 24 volte?

Pieghiamo…

- … 94 volte?

CRC-1

- X+1

CRC-5

- **X^5+x^2+1**
- E...

CRC-16

- X^16+x^15+x^2+1

USB!

CRC-16CCITT

- **X^16+x^12+x^2+1**
- Dove si usa…?

Bluetooth!

CRC-32

- **$X^{32}+x^{30}+x^{26}+x^{25}+x^{24}+x^{18}+x^{15}+x^{14}+x^{12}+x^{11}+x^{10}+x^{8}+x^{6}+x^{5}+x^{4}+x^{3}+x+1$**
- (Corregge tutti i burst fino a 32, e tutti i burst che alterano un numero dispari di bit)

Dove si usa? Ad esempio…

- Modem v.4
- Formato **.zip**
- FDDI (trasporto in Fibra ottica)

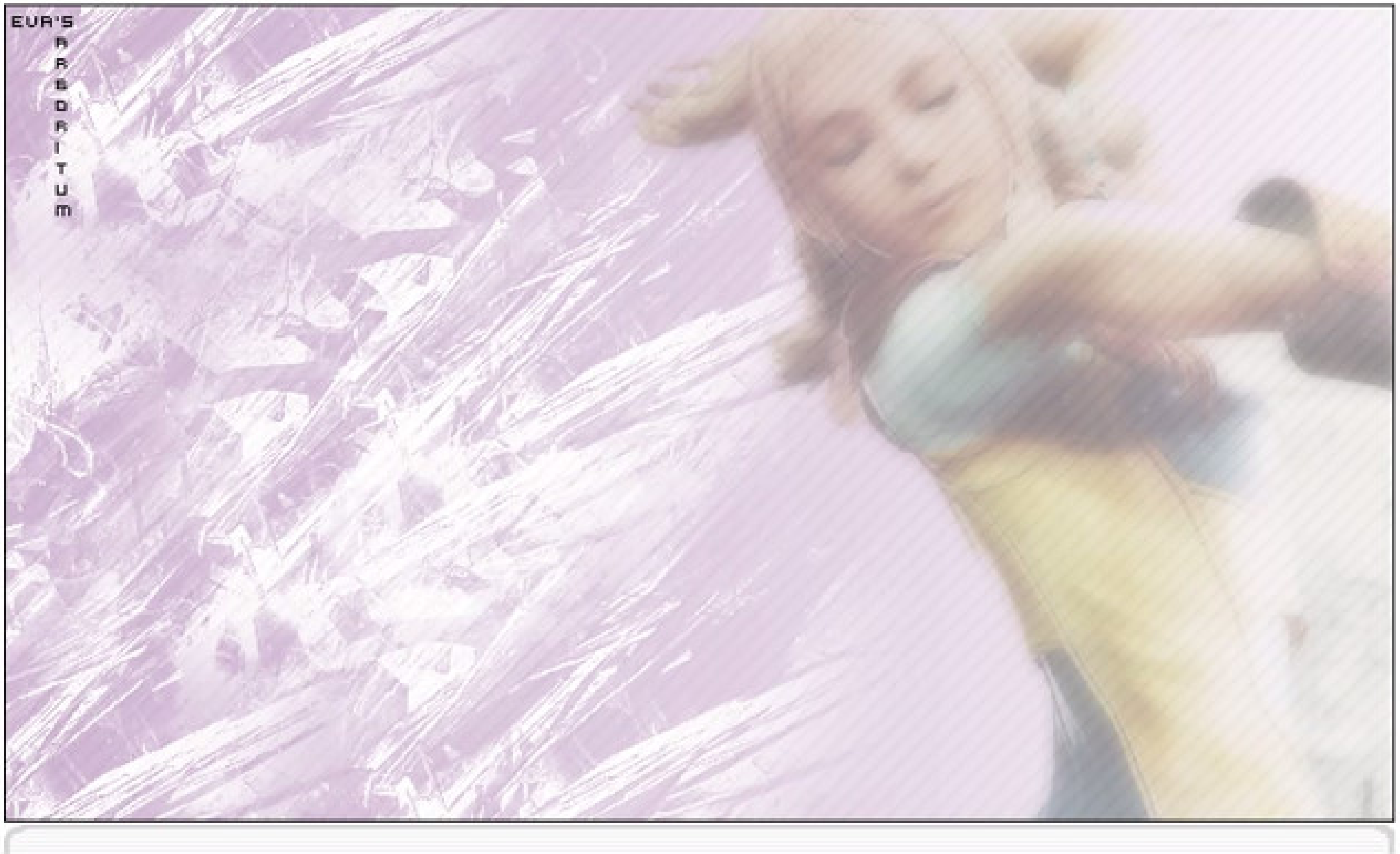
E ancora...

E ancora…

Ethernet (!)

E ancora…

PNG!



Inoltre...

- Come detto, questa tecnica “polinomiale” è alla base poi dei codici di ***error-correction*** più avanzati, come ***Reed-Solomon***
- Dove, informalmente, invece di usare GF(2) si va a “ordini superiori”, ad esempio GF(2^n)...

Esempio

- RS(255, 233) su GF(255) è uno dei principali standard NASA
- Potenza fino a bombe di ordine **16**
- Data rate: **91.4%**