Università degli Studi di Roma "Tor Vergata"
Laurea in Informatica

Sistemi Operativi e Reti
(modulo Reti)
a.a. 2024/2025

# Esercitazione: Livello di Collegamento

dr. Manuel Fiorelli

manuel.fiorelli@uniroma2.it

https://art.uniroma2.it/fiorelli

# Esercizio 1

Si vuole inviare la seguente sequenza di bit proteggendola attraverso l'aggiunta di un bit di parità (pari), posto alla fine della sequenza.

1101 1010 1001 0110

Quali bit saranno effettivamente trasmessi?

# Esercizio 1 (soluzione)

Nello schema di controllo di parità pari, si aggiunge ai dati un solo bit, detto bit di parità, in modo che il numero complessivo di bit a 1 risulti pari. (In alternativa, nello schema a parità dispari, il numero complessivo di bit a 1 deve risultare dispari).

Consideriamo la seguente sequenza di bit originali:

$$1101\ 1010\ 1001\ 0110$$

In totale ci sono 9 bit a 1, quindi per ottenere un numero pari di bit a 1, il bit di parità deve essere 1.

Pertanto, la sequenza da trasmettere sarà:

$$\underbrace{1101\ 1010\ 1001\ 0110}_{\text{D}}\ \underbrace{1}_{\text{EDC}}$$

# Esercizio 2

Dimostrare che, se in una sequenza di bit si verificano un numero pari di inversioni (da 0 a 1 o da 1 a 0), la parità della sequenza (cioè il fatto che contenga un numero pari o dispari di bit uguali a 1) rimane invariata.

# Esercizio 2 (soluzione 1)

(Ripassare concetti base di aritmetica modulare: https://it.wikipedia.org/wiki/Aritmetica_modulare)

Siano:

- $S$ il numero di bit a 1 nella sequenza originale
- $n = a + b$ un numero pari che esprime gli errori occorsi, dove
  - $a$ il numero di inversioni di 0 in 1
  - $b$ il numero di inversioni di 1 in 0

Il numero di bit a 1 nella sequenza ricevuta sarà dunque

$$S' = S + a - b$$

# Esercizio 2 (soluzione 1)

Sottraendo *S* da entrambi i lati dell'equazione, si ottiene

$$S' - S = a - b$$

da cui deriva

$$S' - S \equiv a - b \pmod 2$$

Siccome b $\equiv -b \pmod 2$ [perché $b - (-b) = 2b$ che è un multiplo del modulo], possiamo dedurre

$$S' - S \equiv a + b \pmod 2$$

ricordando che $n = a + b$, possiamo scrivere

$$S' - S \equiv n \pmod 2$$

ma *n* per ipotesi è un numero pari, cioè $n \equiv 0 \pmod 2$, pertanto

$$S' - S \equiv 0 \pmod 2$$

ovvero

$$S' \equiv S \pmod 2$$

In altre parole, *S* e *S'* producono lo stesso resto per la divisione per 2, quindi hanno la stessa parità.

# Esercizio 2 (soluzione 2)

Un altro modo per dimostrare il teorema è osservare che:

- è possibile suddividere un numero pari di inversioni di bit (bit flip, 0 $\leftrightarrow$1) in coppie
- ciascuna coppia di errori non altera la parità:
  - il primo errore cambia il numero di bit a 1 di $\pm 1$, cambiandone la parità (perché i pari e i dispari si alternano)
  - l'errore successivo, per le stesse ragioni, cambia nuovamente la parità, ripristinando quella originale

# Esercizio 3

Si supponga di aver ricevuto il seguente frame protetto da singolo bit di parità (pari):

1101 1010 1001 0110

Rispondere alle seguenti domande:

1. Il ricevente rileva un errore?
2. In caso affermativo, il ricevente può correggere l'errore?

# Esercizio 3 (soluzione)

Il ricevente calcola che nel frame ricevuto:

1101 1010 1001 0110

sono presenti 9 bit a 1, ovvero un numero dispari, mentre ci sarebbe dovuto essere un numero pari.

1. rileva quindi un errore nei bit all'interno del frame
2. non può correggerlo, poiché il bit di parità non fornisce informazioni sulla posizione dei bit invertiti

# Esercizio 3 (approfondimento)

In che senso il bit di parità non fornisce informazioni sulla posizione dei bit invertiti?

Nell'esempio sarebbe stato sufficiente invertire *uno qualsiasi* dei bit per ripristinare la parità pari. Le correzioni candidate hanno dunque tutte la stessa *distanza di Hamming* (= 1) rispetto alla sequenza ricevuta e, perciò non c'è criterio per preferirne una.

> Nella teoria dell'informazione, la **distanza di Hamming** tra due stringhe di ugual lunghezza è il numero di posizioni nelle quali i simboli corrispondenti sono diversi. In altri termini, la distanza di Hamming misura il numero di sostituzioni necessarie per convertire una stringa nell'altra, o, vista in altro modo, il numero minimo di errori che possono aver portato alla trasformazione di una stringa nell'altra.

https://it.wikipedia.org/wiki/Distanza_di_Hamming

# Esercizio 4

Si considerino i dati forniti di seguito in formato binario

1101 1010 1001 0110

Si calcolino i bit di EDC (error detection and correction) secondo lo schema di *controllo di parità (pari) bidimensionale*.

Quali sono i bit EDC da aggiungere (fornire la soluzione minima)?

# Esercizio 4 (soluzione)

Considerando i dati forniti di seguito in formato binario

1101 1010 1001 0110

per trovare il numero minimo di bit EDC dispongo i 16 bit in una matrice 4 x 4, quindi calcolo i bit di parità per ciascuna riga, colonna e per l'angolo in basso a destra (cioè per tutti i bit).

$$\begin{array}{cccc|c} 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 1 \end{array}$$

Bit EDC = 1000 1000 1

# Esercizio 5

Sono stati ricevuti i seguenti dati

10110010 01101100 11001110

I dati sono stati trasmessi in modo tale che:

- Il bit più significativo (il primo a sinistra) di ciascuno dei primi due byte è un bit di parità pari calcolato sui restanti 7 bit dello stesso byte.
- Il terzo byte contiene, in ciascuna posizione, la parità pari dei bit nella stessa posizione dei primi due byte

Rispondere alle seguenti domande:

1. Il ricevente rileva un errore?
2. In caso affermativo, può correggerlo?

# Esercizio 5 (soluzione)

I dati sono stati trasmessi con uno schema di parità pari bidimensionale.

Riarrangiamo i dati in forma di matrice e verifichiamo se tutti i bit di parità sono corretti:

$$
\begin{array}{c|ccccccc}
1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\
\hline
1 & 1 & 0 & 0 & 1 & 1 & 1 & 0
\end{array}
\longrightarrow
\begin{array}{c|ccccccc}
1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\
\hline
1 & 1 & 0 & 1 & 1 & 1 & 1 & 0
\end{array}
$$

1. Troviamo che i bit di parità della 3^ riga e della 4^ sono errati.
2. Invertendo il bit alla loro intersezione (che cade all'interno dei bit EDC) ripristiniamo la parità desiderata sulla riga e colonna incriminate, correggendo l'errore.

# Esercizio 5 (soluzione)

La sequenza di dati corretta è

`10110010 01101100 11011110`

Si noti che non c'è garanzia che questi siano effettivamente i dati inviati, così come non c'è garanzia che quando il controllo dell'EDC passa non ci sono errori.

# Esercizio 5 (approfondimento)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |

→

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |

Si noti che qualsiasi altra modifica di 1 solo bit non produce dati corretti:

- cambiando un bit nelle righe 1 o 2 (che hanno già la parità giusta), avremmo sbilanciato la loro parità
- cambiando un bit nelle colonne 1-3 o 5-8 (che hanno già la parità giusta), avremmo sbilanciato la loro parità

Quella proposta è l'unica modifica a singolo bit che ripristina la parità di tutte le righe e colonne e, dunque, può essere proposta come correzione.

# Esercizio 5 (approfondimento 2)

ricevuti

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |

correzione

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |

inviati

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |

I quattro bit colorati in rosso in alto a sinistra, presenti nei dati ricevuti, rappresentano degli errori che non sono stati rilevati (poiché si presentano in numero pari su ciascuna riga e colonna) e sono quindi rimasti anche dopo il processo di correzione.
Questo dimostra che non è possibile avere la certezza che la sequenza corretta corrisponda effettivamente a quella originariamente trasmessa.

# Esercizio 6

Si considerino i dati forniti di seguito in formato binario

1101 1010 1001 0110 1011 1100 0011 0101 1011 1100 0011 0101

Si calcolino i bit di EDC (error detection and correction) secondo la checksum di Internet.

# Esercizio 6 (soluzione)

Si considerino i dati forniti di seguito in formato binario

1101 1010 1001 0110 1011 1100 0011 0101 1011 1100 0011 0101

La si vede come una sequenza di numeri di 16 bit di cui si calcola la somma in complemento a 1 (quindi sommano al bit meno significativo un eventuale riporto).

```
  1101 1010 1001 0110 +
  1011 1100 0011 0101 =
-----------------------
1 1001 0110 1100 1011
                        end-around carry
  1001 0110 1100 1100 +
  1011 1100 0011 0101 =
-----------------------
1 0101 0011 0000 0001
                        end-around carry
  0101 0011 0000 0010
```

*complemento a 1* — 1010 1100 1111 1101 *Internet Checksum*

# Esercizio 6b

Si considerino i dati (identici a quelli dell'esercizio precedente) forniti di seguito in formato esadecimale

DA 96 BC 35 BC 35

Si calcolino i bit di EDC (error detection and correction) secondo la checksum di Internet.

# Esercizio 6b (soluzione)

Data la sequenza D di bit fornita di seguito in formato esadecimale

DA 96 BC 35 BC 35

La si vede come una sequenza di numeri di 16 bit di cui si calcola la somma in complemento a 1 (quindi sommano al bit meno significativo un eventuale riporto).

```
   DA 96 +
   BC 35 =
-----------
 1 96 CB +
   BC 35 =
-----------
 2 53 00
   53 02
```

*end-around carry (differito)*

11 +
5 =
----
16 = $(10)_{16}$ poiché 16 = 1 * 16 + 0

Dato un numero *x* di 4 bit (una cifra esadecimale), si ha che

$$x + \overline{x} = (1111)_2 = (15)_{10}$$

pertanto calcoliamo il complemento a 1 di $x$ come $(15)_{10} - x$

*complemento a 1* — AC FD *Internet Checksum* (in binario `1010 1100 1111 1101`)

| | |
|---|---|
| A | 10 |
| B | 11 |
| C | 12 |
| D | 13 |
| E | 14 |
| F | 15 |

# Esercizio 7

Data la sequenza D di bit fornita di seguito in formato binario

1101 1010 1001 0110

Si calcolino i bit di EDC (error detection and correction) secondo lo schema di CRC usando il generatore CRC-8-CCITT (1 0000 0111).

Quali sono i bit EDC da aggiungere?

Si noti che il polinomi standard sono chiamati in genere CRC-n-*blabla*, dove *n* è il numero di bit EDC, mentre il generatore è lungo n + 1. Poiché il bit più significativo del polinomio è 1, n corrisponde al grado del generatore quando interpretato come polinomio.

# Esercizio 7 (soluzione)

1. I bit di EDC sono calcolati come resto della seguente divisione

```
                   1101100010011101
          ---------------------------
100000111| 1101101010010110000000000
           100000111
            101100100
            100000111
              110001101
              100000111
               100010100
               100000111
                   100111100
                   100000111
                      111011000
                      100000111
                       110111110
                       100000111
                        101110010
                        100000111
                          111010100
                          100000111
                           11010011
```

Il divisore è il generatore

Il dividendo è formato dalla stringa dei dati cui sono stati appesi *r* zeri, dove *r* è la lunghezza del generatore - 1

Per motivi di spazio, sono state omesse le somme parziali quando la cifra è zero

EDC = 11010011

```
                    1101100010011101
            ------------------------
100000111 | 110110101001011000000000
            100000111
             101100100
             100000111
               11000110
               00000000
               110001101
               100000111
                100010100
                100000111
                    100111
                    000000
                    1001111
                    0000000
                    10011110
                    00000000
                    100111100
                    100000111
                       1110110
                       0000000
                       11101100
                       00000000
                       111011000
                       100000111
                        110111110
                        100000111
                         101110010
                         100000111
                           11101010
                           00000000
                           111010101
                           100000111
                            11010011
```

versione estesa con tutti i passaggi visibili

# Esercizio 7 (soluzione)

Come verifica è sufficiente calcolare:

(1101100010011101 x 100000111) $\oplus$ 11010011

```
            1101100010011101 x
                   100000111 =
------------------------------
            1101100010011101 ⊕
           11011000100111010 ⊕
          110110001001110100 ⊕
    110110001001110100000000 =
------------------------------
    110110101001011011010011 ⊕
                    11010011 =
------------------------------
    110110101001011000000000

    110110101001011000000000
```

# Esercizi 6 e 7: varianti

Vengono forniti i dati ricevuti comprensivi di bit EDC e si chiede di verificare se ci sono errori.

La checksum di Internet rileva un errore se la somma è diversa da tutti 1 (ovvero -0 in complemento a 1).

CRC rileva un errore se i bit ricevuti (comprensivi di EDC) non sono divisibili per il generatore (la divisione restituisce un resto diverso da zero).

# Esercizio 8

Dimostrare che lo XOR di n bit è 1 se e solo se il numero di bit a 1 è dispari. In altre parole che

$$x_1 \oplus \cdots \oplus x_n = 1 \Leftrightarrow \#\{i : x_i = 1\}\ e\grave{}\ dispari$$

# Esercizio 8 (soluzione)

Possiamo dimostrare questo teorema per induzione.

$$x_1 \oplus \cdots \oplus x_n = 1 \Leftrightarrow \#\{i: x_i = 1\}\ e`\ dispari$$

Caso base: $n = 1$

$x_1 = 1 \Leftrightarrow$ c'è un solo bit a 1, cioè un numero di dispari

Passo induttivo:

Supponendo che il teorema sia vero per ogni $n \geq 1$, dobbiamo dimostrare che è vero per $n + 1$.

$$x_1 \oplus \cdots \oplus x_n \oplus x_{n+1} = 1 \Leftrightarrow \#\{i: x_i = 1\}\ e`\ dispari$$

# Esercizio 8 (soluzione)

| a | b | a ⊕ b |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Possiamo riscrivere così:

$$\underbrace{x_1 \oplus \cdots \oplus x_n}_{y} \oplus x_{n+1} = y \oplus x_{n+1} = 1 \Leftrightarrow \#\{i: x_i = 1\}\ e\grave{}\ dispari$$

Dove $y_n$è lo XOR dei primi n bit. Per l'ipotesi induttiva sappiamo che

$$y = 1 \Leftrightarrow \#\{i: x_i = 1 \wedge i \leq n\}\ e\grave{}\ dispari$$

Caso 1:

$\boldsymbol{y = 0}$: $y \oplus x_{n+1} = 0 \oplus x_{n+1} = x_{n+1} = 1 \Leftrightarrow \#\{i: x_i = 1\}\ e\grave{}\ dispari$

è vero perché per ipotesi c'è un numero pari di bit a 1 tra i primi *n*, pertanto, complessivamente ce ne sarà un numero dispari se e solo se $x_{n+1} = 1$

$\boldsymbol{y = 1}$: $y \oplus x_{n+1} = 1 \oplus x_{n+1} = 1 \Leftrightarrow \#\{i: x_i = 1\}\ e\grave{}\ dispari$

è vero perché per ipotesi c'è un numero dispari di bit a 1 tra i primi *n*, pertanto, complessivamente ce ne sarà un numero dispari se e solo se $x_{n+1} = 0$

# Esercizio 9

Un frame di $N$ bit viene trasmesso attraverso un collegamento soggetto a errori su bit indipendenti con probabilità $p$.

Qual è la probabilità che:

1. Il frame sia ricevuto senza errori
2. Il frame sia ricevuto con un errore
3. Il frame sia ricevuto con errori multipli (2 o più)

Applicare queste probabilità alla discussione dell'efficacia del bit di parità.

# Esercizio 9 (soluzione)

Il numero *S* di bit invertiti è una variabile aleatoria binomiale con parametri *N* e *p* (https://it.wikipedia.org/wiki/Distribuzione_binomiale)

$$S \sim B(N, p)$$

La probabilità che non si verifichi alcun errore è

$$P_{no\ errore} = P(S = 0) = \binom{N}{0} p^0 (1-p)^N = (1-p)^N$$

Se $p \to 0$ , allora $P_{no\ errore} \to 1$: è assai probabile che il frame arrivi senza errori se $p \ll 1$ (molto più piccolo di).

# Esercizio 8 (soluzione)

La probabilità che si verifichi un errore singolo

$$P_{1\ errore} = P(S=1) = \binom{N}{1} p^1 (1-p)^{N-1} = Np(1-p)^{N-1}$$

Ricordando che se $|x| < 1$, vale il seguente sviluppo di Taylor nell'intorno di x = 0:

$$(1+x)^N = 1 + Nx + \frac{N(N-1)}{2}x^2 + \cdots + \frac{N(N-1)(N-2)(N-k+1)}{k!}x^k + o\left(x^k\right)$$

Siccome *p* è prossimo a zero, possiamo utilizzare uno sviluppo del primo ordine per scrivere:

$$P_{1\ errore} = Np(1-(N-1)p + o(p)) = Np - N(N-1)p^2 + o(p^2) = Np + o(p)$$

Ma anche

$$P_{no\ errore} = 1 - Np + o(p)$$

# Esercizio 8 (soluzione)

La probabilità che si verifichino errori multipli è

$$P_{errore\ multiplo} = \sum_{m=2}^{N} \binom{N}{m} p^m \, (1-p)^{N-m}$$

Osserviamo però che quelle discusse sono le uniche tre possibilità, mutualmente incompatibili:

$$\Omega = \{no\ errore\} \cup \{errore\ singolo\} \cup \{errore\ multiplo\}$$

Quindi,

$$P_{errore\ multiplo} = 1 - P_{no\ errore} - P_{1\ errore} = 1 - (1-p)^N - Np(1-p)^{N-1}$$

# Esercizio 8 (soluzione)

Osservando che p è prossimo a zero, possiamo approssimmare come prima

$$P_{errore\ multiplo} = 1 - (1-p)^N - Np(1-p)^{N-1}$$

$$= 1 - \left(1 - Np + \frac{N(N-1)}{2}p^2 + o(p^2)\right) - Np(1 - (N-1)p + o(p))$$

$$= 1 - 1 + Np - \frac{N(N-1)}{2}p^2 + o(p^2) - Np + N(N-1)p^2 + o(p^2)$$

$$= \frac{N(N-1)}{2}p^2 + o(p^2)$$

# Esercizio 8 (soluzione)

Riassumendo, per *p* molto vicino a zero

$$P_{no\ errore} \sim 1 - Np$$
$$P_{1\ errore} \sim Np$$
$$P_{errore\ multiplo} \sim \frac{N(N-1)}{2} p^2$$

Da ciò possiamo calcolare un *upper bound* per la probabilità di errore non rilevato:

$$P(errore\ non\ rilevato) = P(errore\ pari) < P(errore\ multiplo) \sim \frac{N(N-1)}{2} p^2$$

Sotto l'assunzione di errori indipendenti e identicamente distribuiti (*i.i.d.),* la probabilità di fallimento di un bit di parità è molto bassa (grazie all'infinitesimo di grado 2) con i valori di *p* molto bassi tipici dei collegamenti cablati moderni.

Considerando, per esempio, $N = 12000$ (cioè 1500 byte) e $p = 10^{-12}$, la probabilità di un errore non rilevato è inferiore a

$$7.2 \cdot 10^{7} \cdot 10^{-24} = 7.2 * 10^{-17}$$

# Esercizio 8 (soluzione)

Un altro modo per rispondere è calcolare la probabilità di una combinazione di *m* errori (sempre tenendo conto che *p* è prossimo a zero):

$$P_{m\ erorri} = P(S = m) = \binom{N}{m} p^m\ (1-p)^{N-m}$$

$$= \binom{N}{m} p^m \big(1 - (N-m)p + o(p)\big)$$

$$= \binom{N}{m} p^m \big(1 - (N-m)p + o(p)\big) = \binom{N}{m} p^m - \binom{N}{m} (N-m) p^{m+1} + o(p^{m+1})$$

$$= \binom{N}{m} p^m + o(p^m)$$

# Esercizio 8 (soluzione)

Avendo calcolato che $P_{m\ errori} = P(S = m) = \binom{N}{m} p^m + o(p^m)$

Si può calcolare

$$P_{errori\ pari} = P(S = 2) + P(S = 4) + \cdots + S\left(S = \left\lfloor \frac{N}{2} \right\rfloor \times 2\right)$$

$$= \left(\binom{N}{2} p^2 + o(p^2)\right) + \left(\binom{N}{4} p^4 + o(p^4)\right) + \cdots$$

$$= \binom{N}{2} p^2 + o(p^2) = \frac{N(N-1)}{2} p^2 + o(p^2)$$

Se consideriamo $N\ = 12000$ (per 1500 byte) e $p = 10^{-12}$, viene fuori una probabilità pari a

$$7.2 * 10^7 \cdot 10^{-24} = 7.2 * 10^{-17}$$

# Esercizi interattivi forniti dal libro

- https://gaia.cs.umass.edu/kurose_ross/interactive/2d_parity.php
- https://gaia.cs.umass.edu/kurose_ross/interactive/CRC.php
- https://gaia.cs.umass.edu/kurose_ross/interactive/internet_checksum.php