



# **IL LIVELLO DI LINK**

**Prof. Salvatore Marano**

**Prof. S. Marano**

**Università della Calabria**

**A.A. 2012-2013**



# Il livello Data Link

- **Funzioni elementari di un protocollo del livello di linea**
  - Delimitazione e identificazione delle trame
  - Rivelazione degli errori trasmissivi
  - Recupero del corretto trasferimento delle trame in caso di errore
  - Controllo di flusso
  - Gestione della DL-connessione (instaurazione, abbattimento e reinizializzazione)



# Il livello Data Link

- Il livello di linea si occupa del trasferimento dei dati su un link seriale
  - modo di trasmissione asincrono o sincrono
  - protocollo orientato al carattere o al bit
  - servizio connectionless o connection-oriented
    - connectionless: trame errate scartate e ritrasmissioni a carico dei livelli superiori (LAN, ISDN dove BER è basso)



# *Protocolli di linea*

## **classificazione**

- orientati al carattere
- orientati al bit

## **colloquio**

- half-duplex
- full-duplex

## **relazioni tra le stazioni**

- master-slave
- peer-to-peer

## **canali**

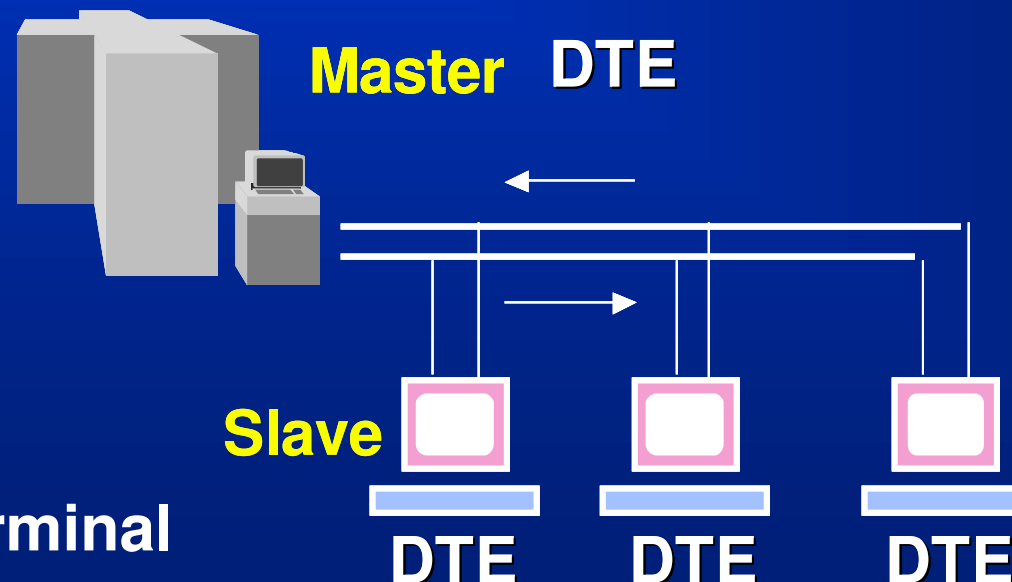
- punto-punto
- punto-multipunto
- necessità dell'indirizzo nei canali punto-multipunto



# Configurazioni di linea

## Canale Multi-Punto

**Più nodi collegati ad un unico canale: un nodo master e numerosi slave**



**DTE: Data Terminal  
Equipment**



# Configurazioni di linea

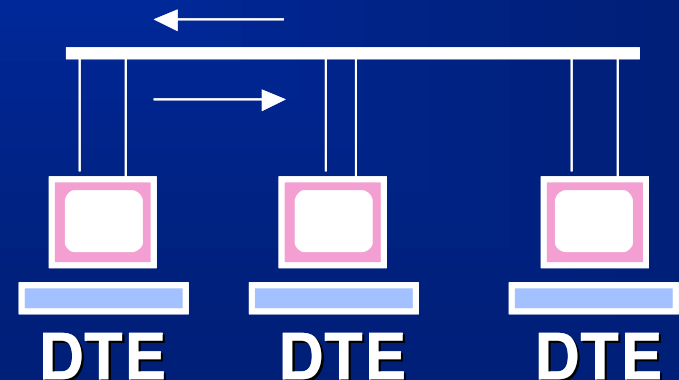
## Canale Broadcast

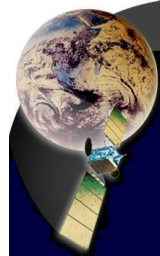
**Un unico canale di comunicazione,  
condiviso da tutti i nodi**

**L'informazione inviata da un nodo è ricevuta  
da tutti gli altri**

**I dati trasmessi contengono l'indirizzo del  
nodo destinazione**

**Tipicamente usati nelle LAN**

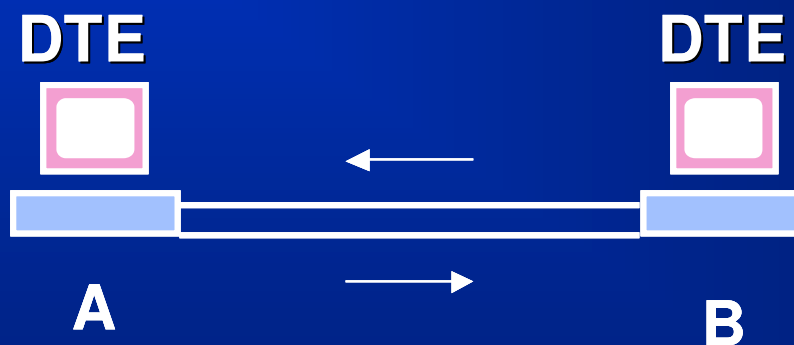




# Configurazioni di linea

## Canale Punto-Punto

**Due soli nodi collegati agli estremi del canale che viene utilizzato in modo paritetico**



**DTE =Data Terminal  
Equipment**



# Configurazioni di linea

## Canale Punto-Punto

I nodi sono collegati attraverso una rete (il DLC opera end-to-end tra i DTE)



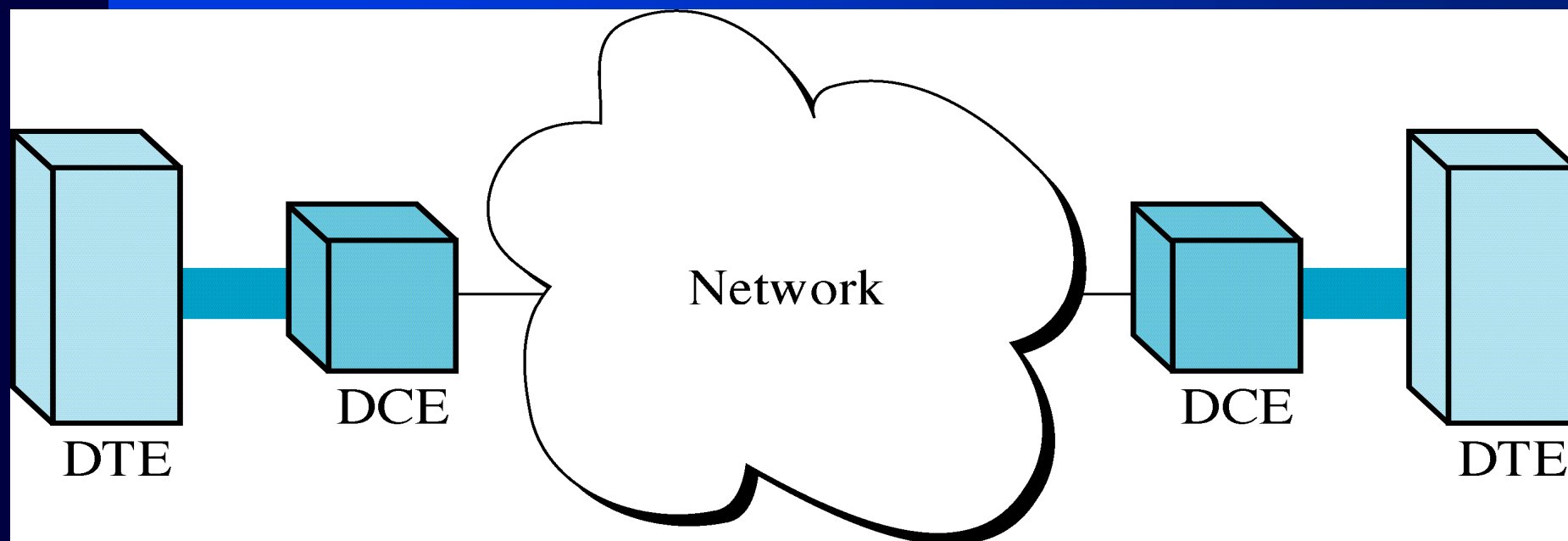
DTE =Data Terminal  
Equipment

DCE =Data Circuit-  
terminating Equipment





# DTE e DCE





## Esempio di utilizzo di MODEM





## **MODEM: MOdulatore e DEModulatore**

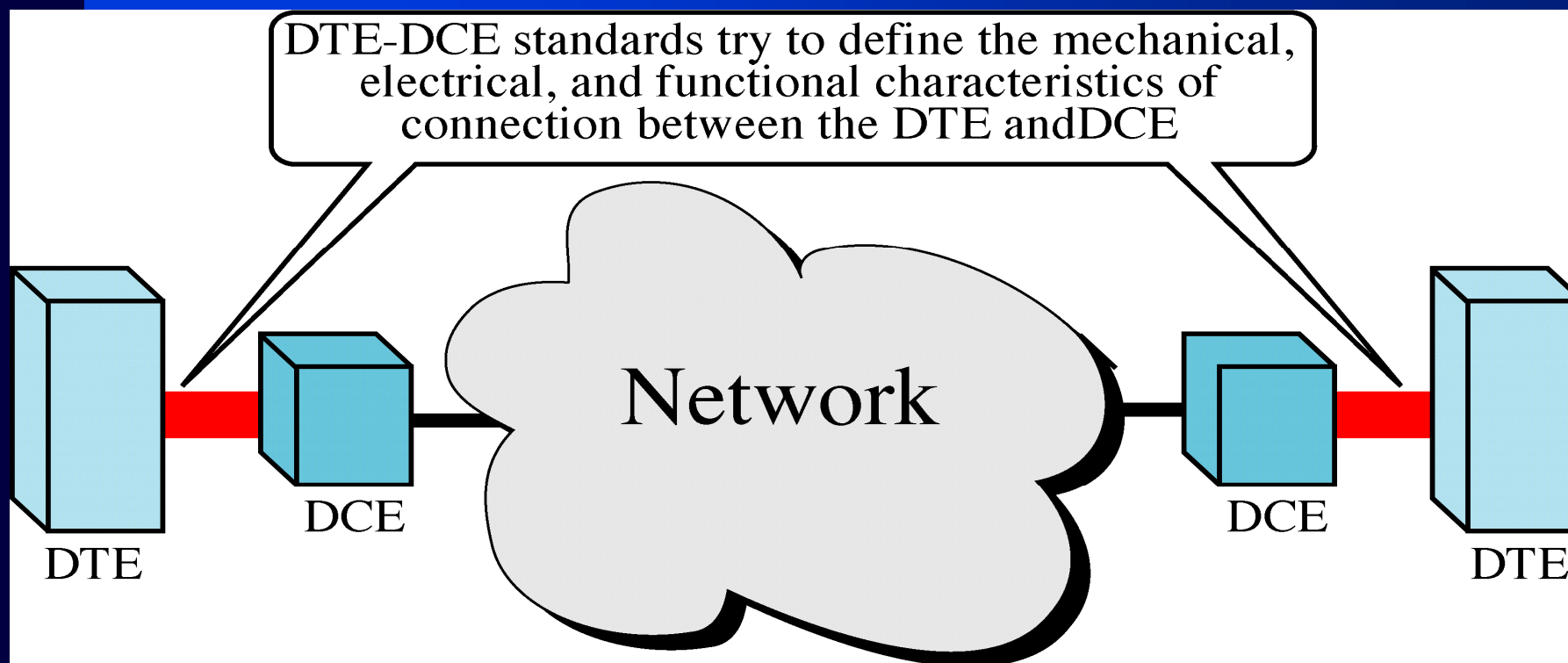
**Si utilizzano per effettuare trasmissioni seriali su rete pubblica**

**Trasformano il segnale da digitale ad analogico e viceversa**

**Rendono il segnale idoneo alla trasmissione su rete pubblica**



# DTE-DCE interface





## Standard di Interfaccia

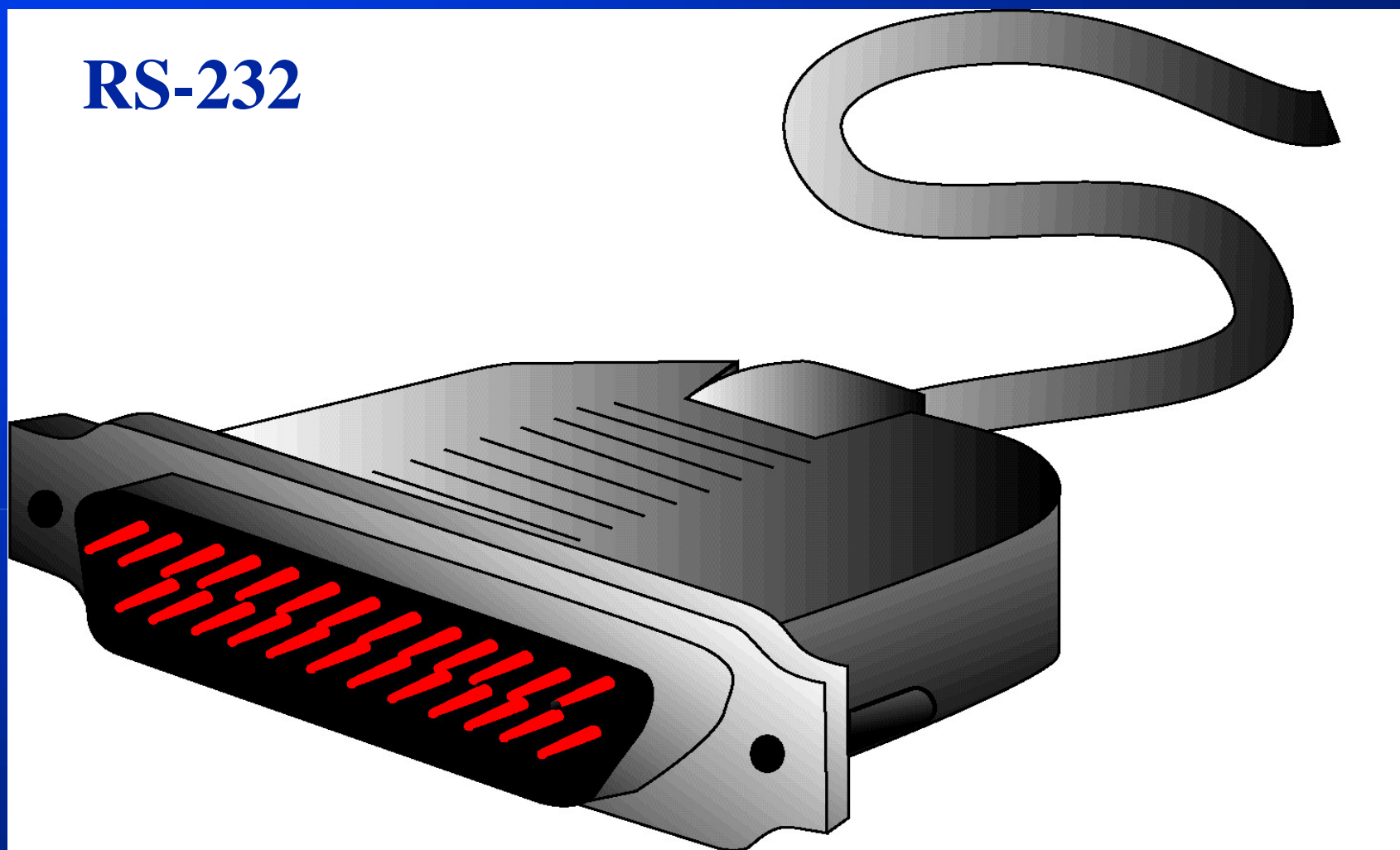
Specificano le interconnessioni tra DTE e DCE

**I principali sono:**

- + RS-232 (o equivalenti CCITT V.24 e V.28)**
- + V.35**
- + G.703 e G.704**



## RS-232





## RS-232

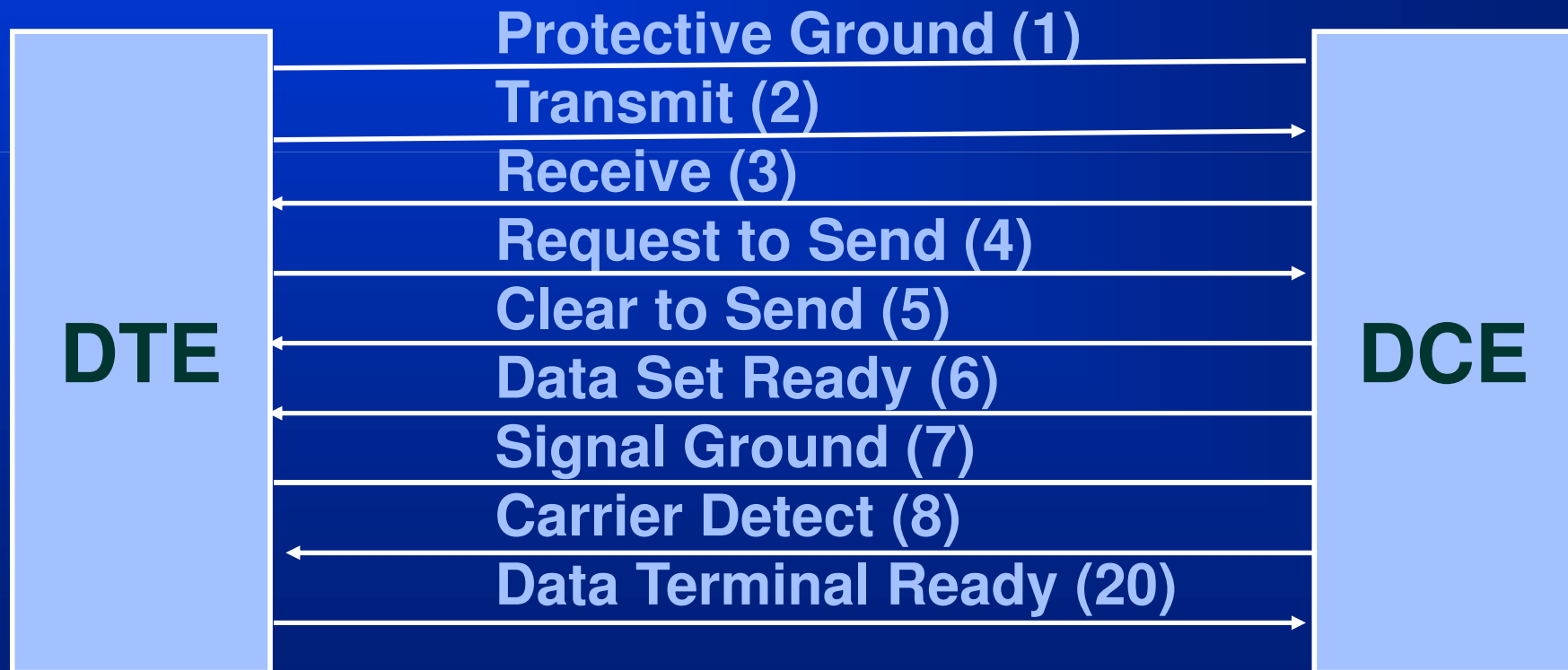
**Standard per la trasmissione seriale a bassa velocità (sino a 19200 b/s)**

**Utilizza un connettore a 9 o 25 pin (vaschetta)**

**Prevede 8 segnali + 1 schermatura**



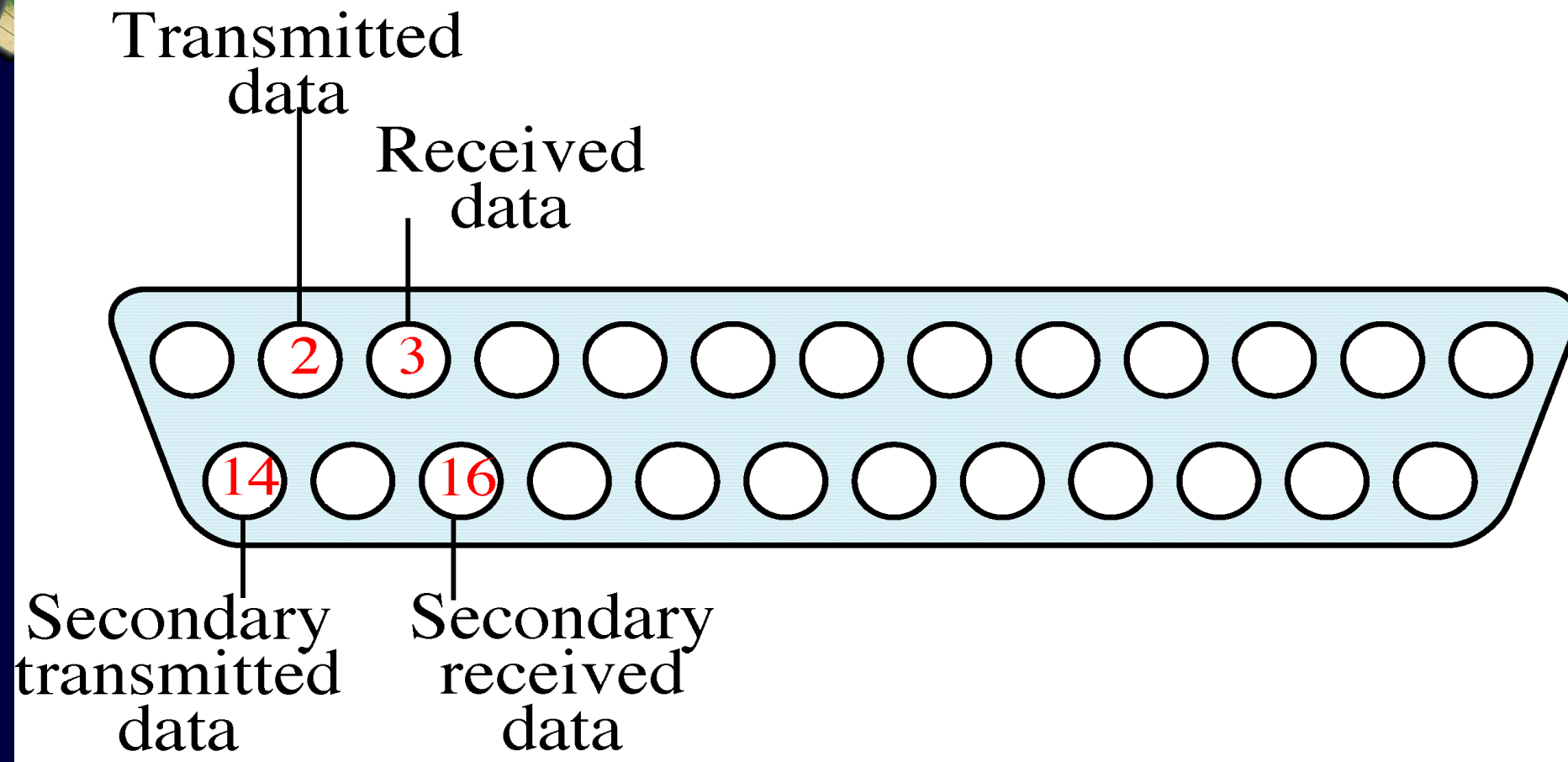
## RS-232 (9 fili)

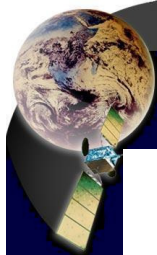




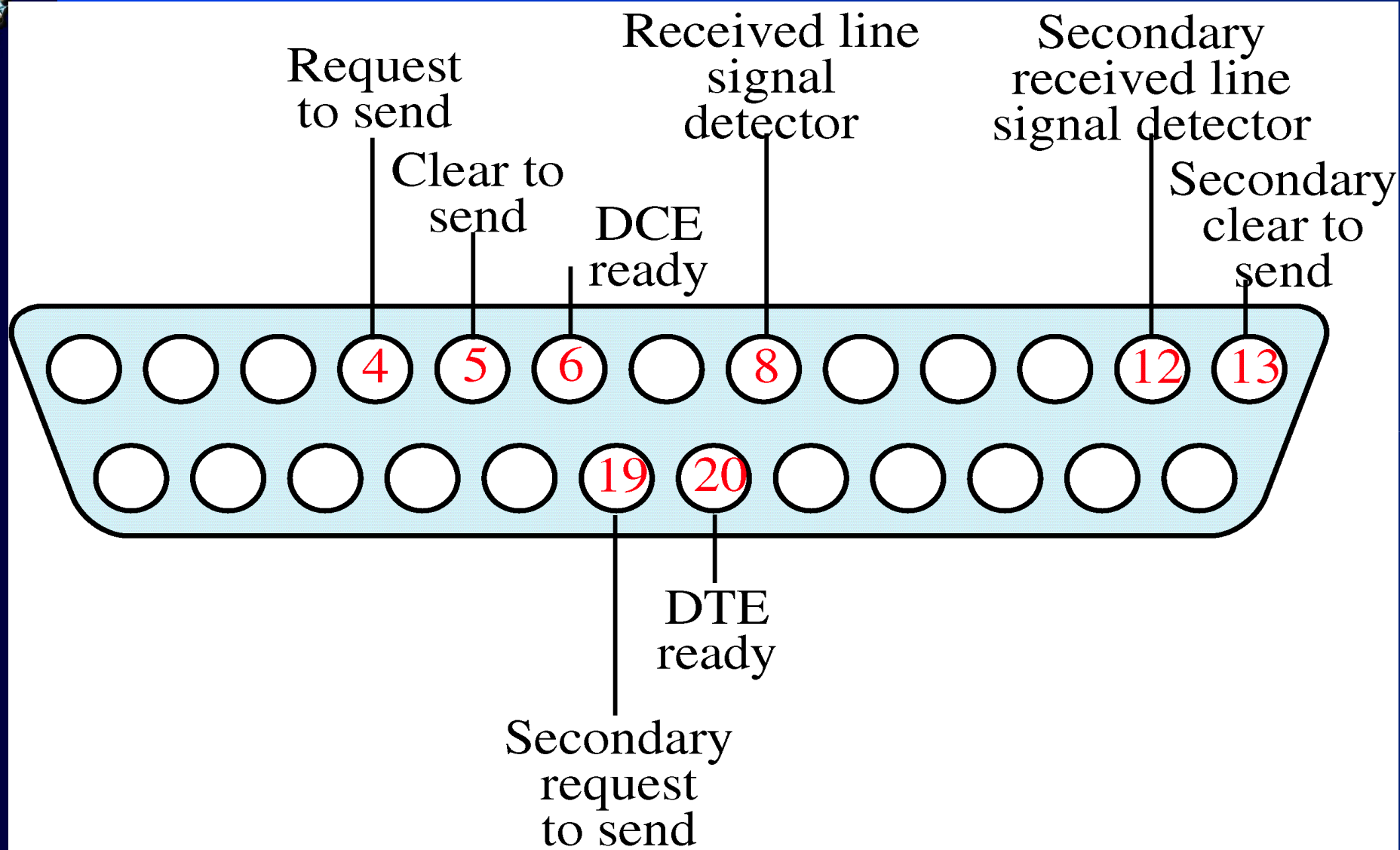


# Data Pins



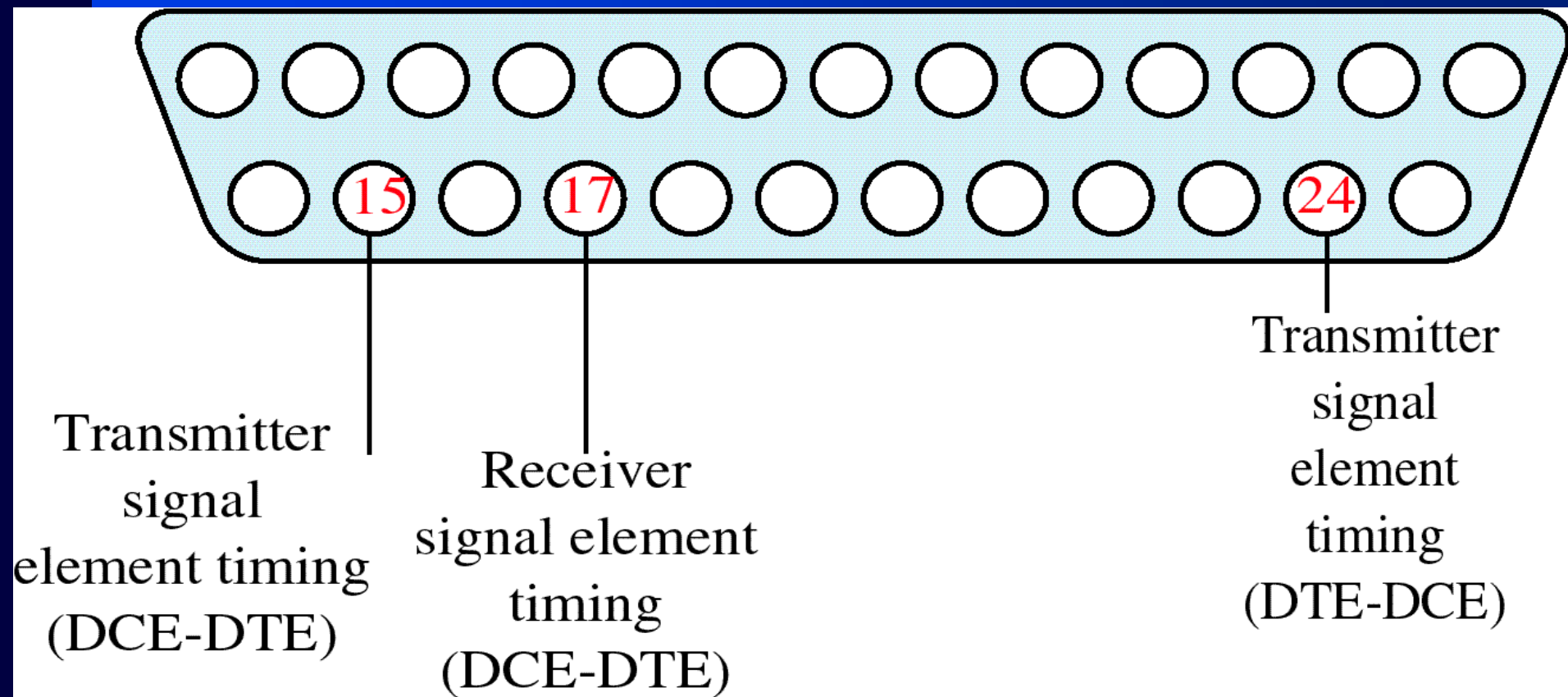


# Control Pins



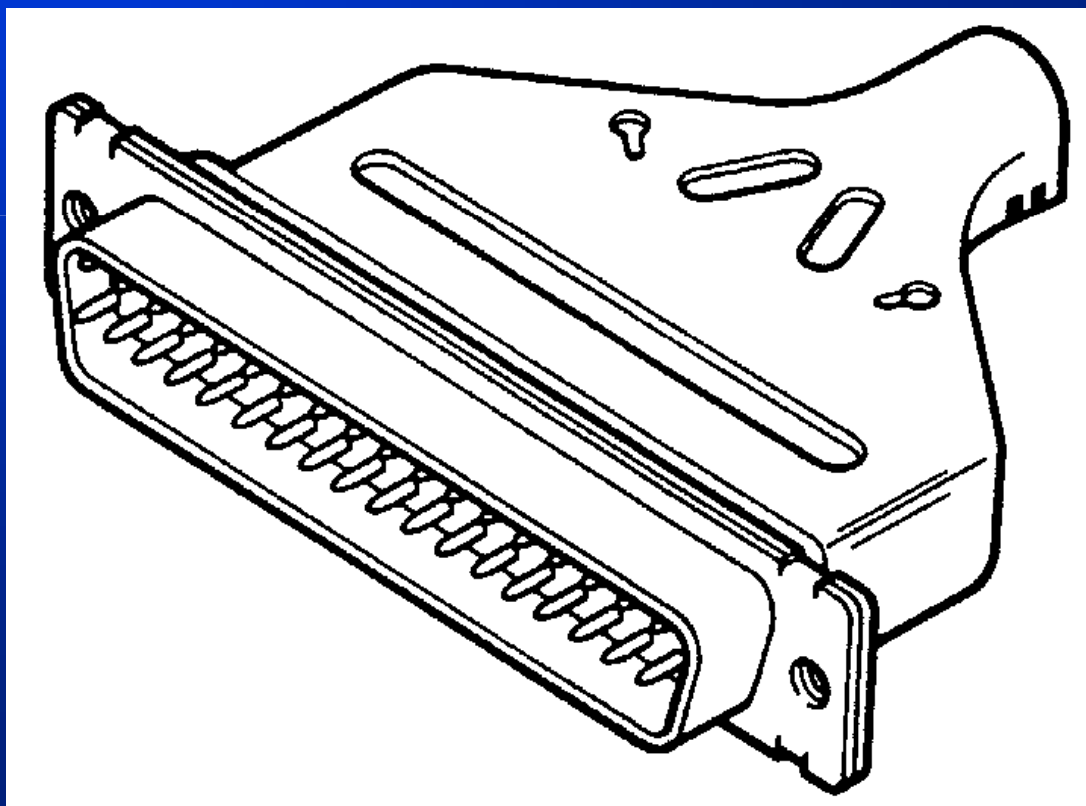


# Timing Pins





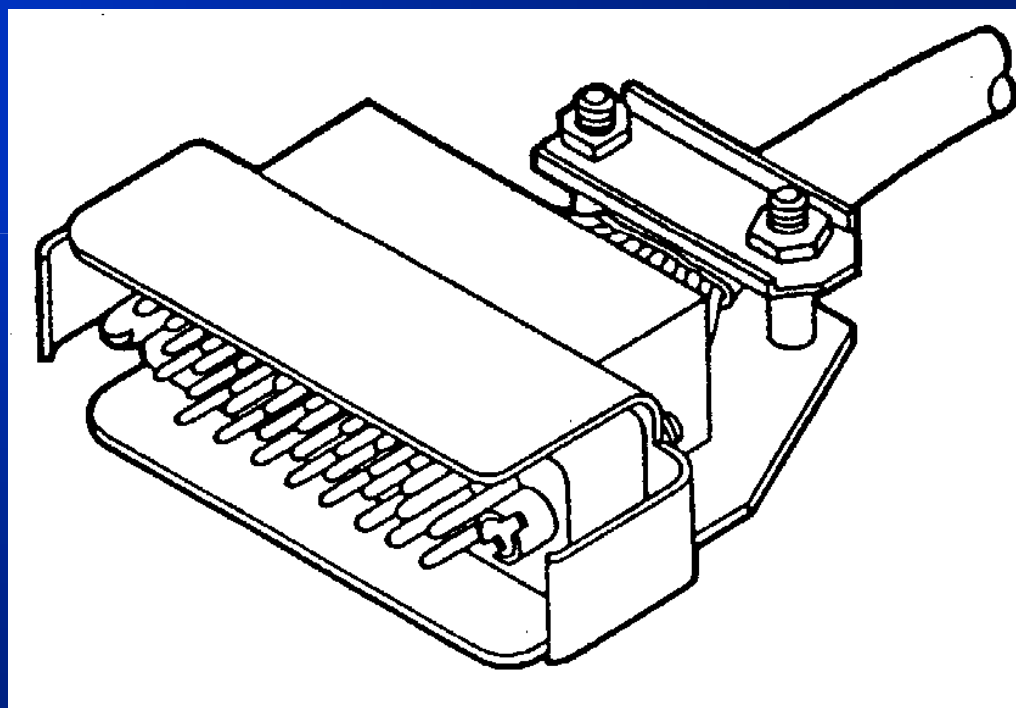
## RS-232 Connettore 25 pin (vaschetta)





## V.35

Standard simile  
a RS-232, ma  
per velocità  
superiori a  
19200 b/s

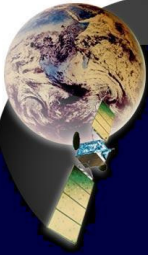




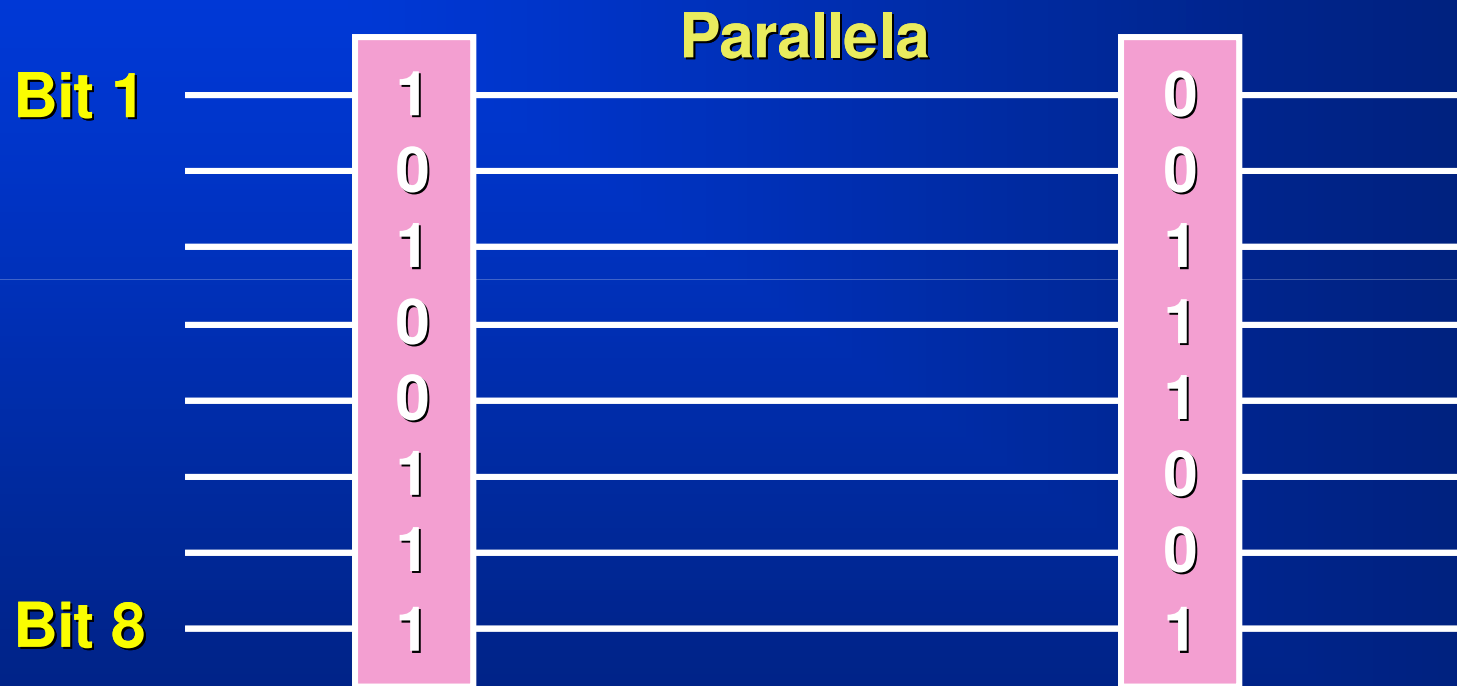
## Tipi di trasmissione

### PARALLELA

**L'informazione viene trasferita in parallelo (tipicamente 8 bit = 1 byte alla volta) su un bus di comunicazione contenente segnali di dati e segnali di temporizzazione (clock).**

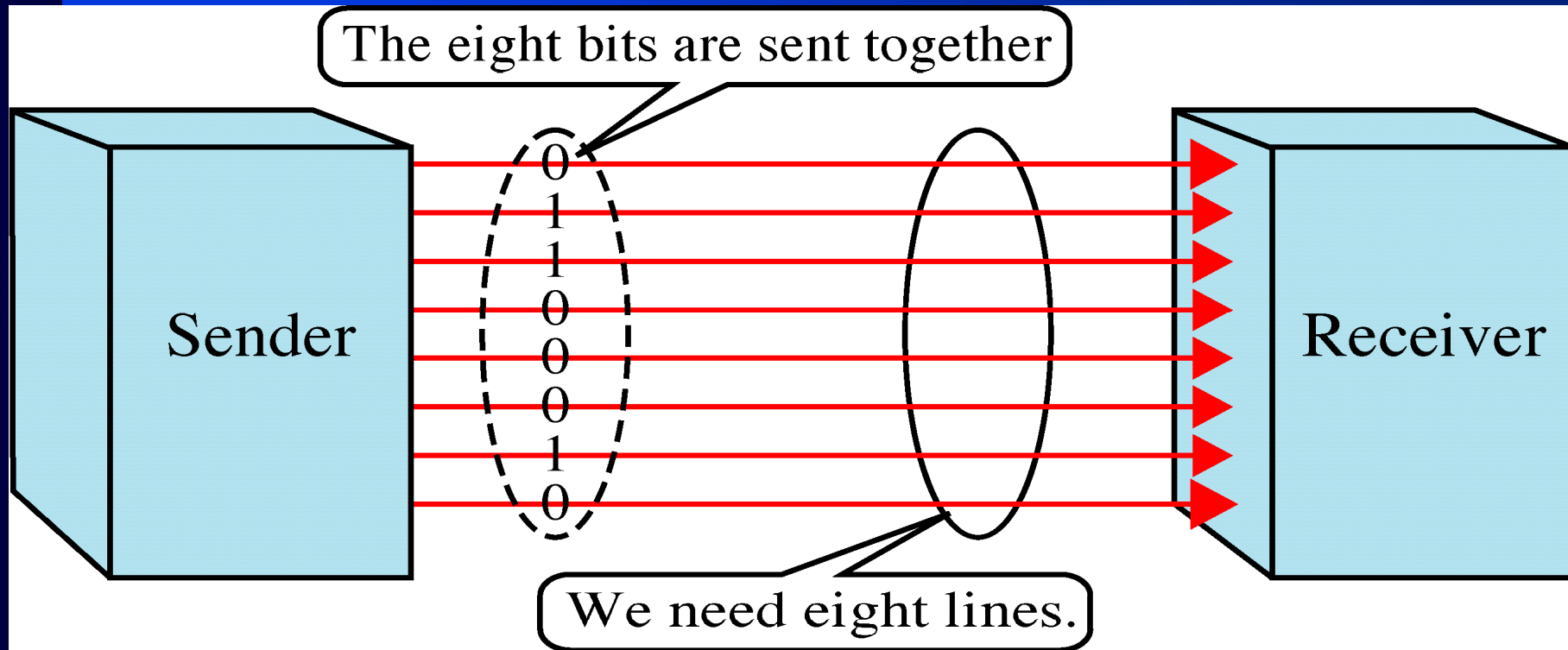


# *Trasmissione Parallela*





# Parallel Transmission



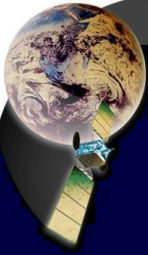




## Tipi di trasmissione

### SERIALE

**L'informazione viene prima serializzata e quindi trasmessa un bit alla volta. Esistono meccanismi di sincronizzazione che evitano l'uso di segnali aggiuntivi di temporizzazione.**

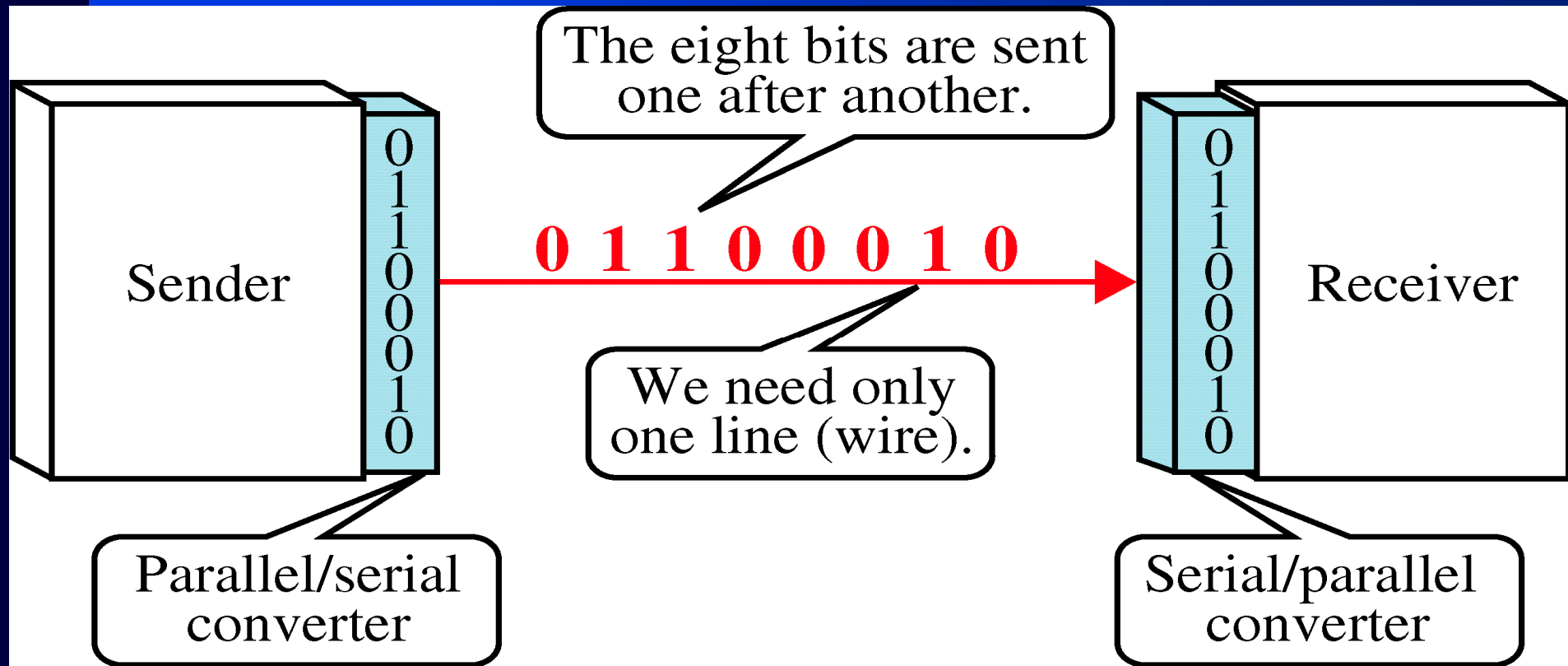


# *Trasmissione Seriale*





# Serial Transmission

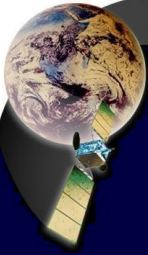




## Tipi di trasmissione seriale

### ASINCRONA

**Ogni byte di informazione viene trasmesso separatamente dagli altri. Il clock di ricezione è solo nominalmente uguale a quello di trasmissione.**

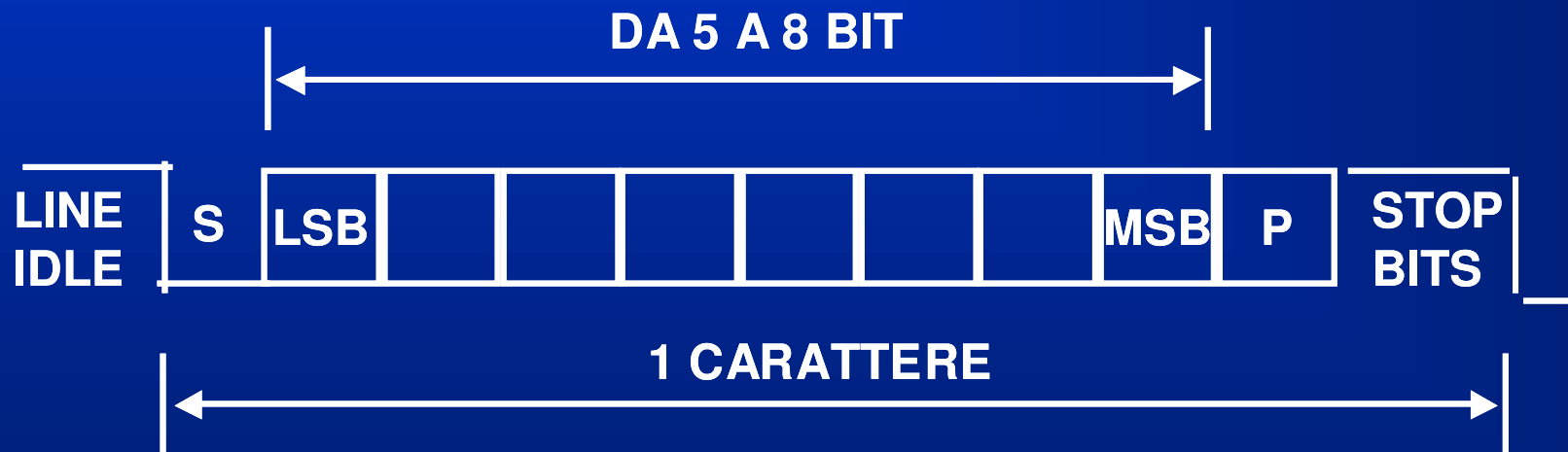


# *Trasmissione Asincrona*

**S: Start Bit**

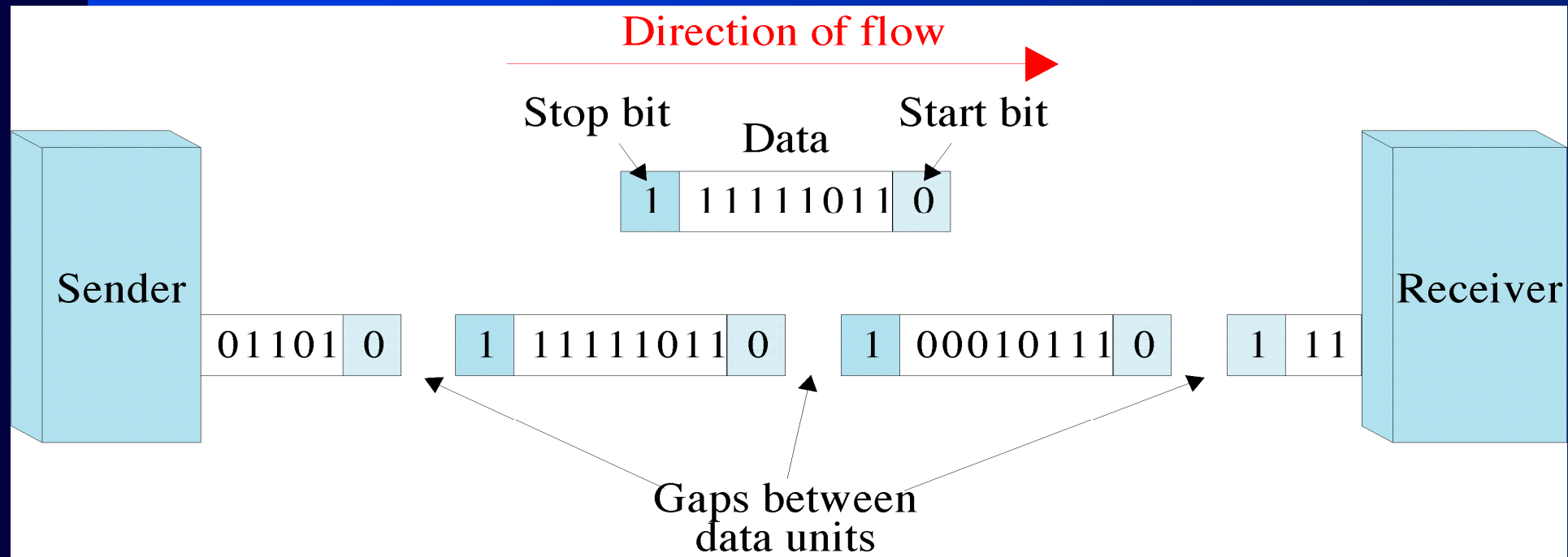
**P: Parity Bit**

**Stop Bits 1, 1.5, 2**





# Asynchronous Transmission





## Tipi di trasmissione seriale

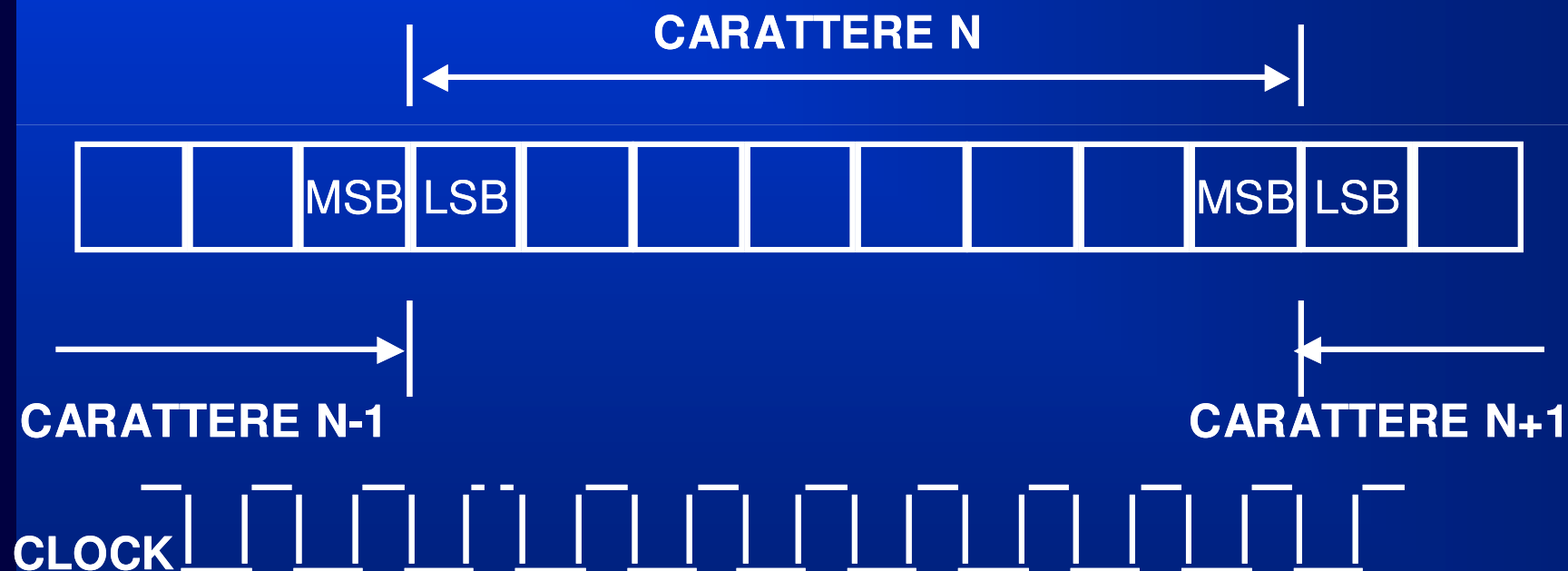
### SINCRONA

**Le informazioni da trasmettere sono strutturate in trame. Il trasmettitore e il ricevitore sincronizzano i loro clock prima della trasmissione e li mantengono sincronizzati per tutta la durata della trama.**

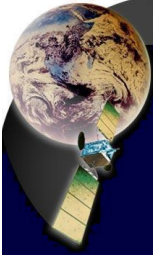


# *Trasmissione Sincrona*

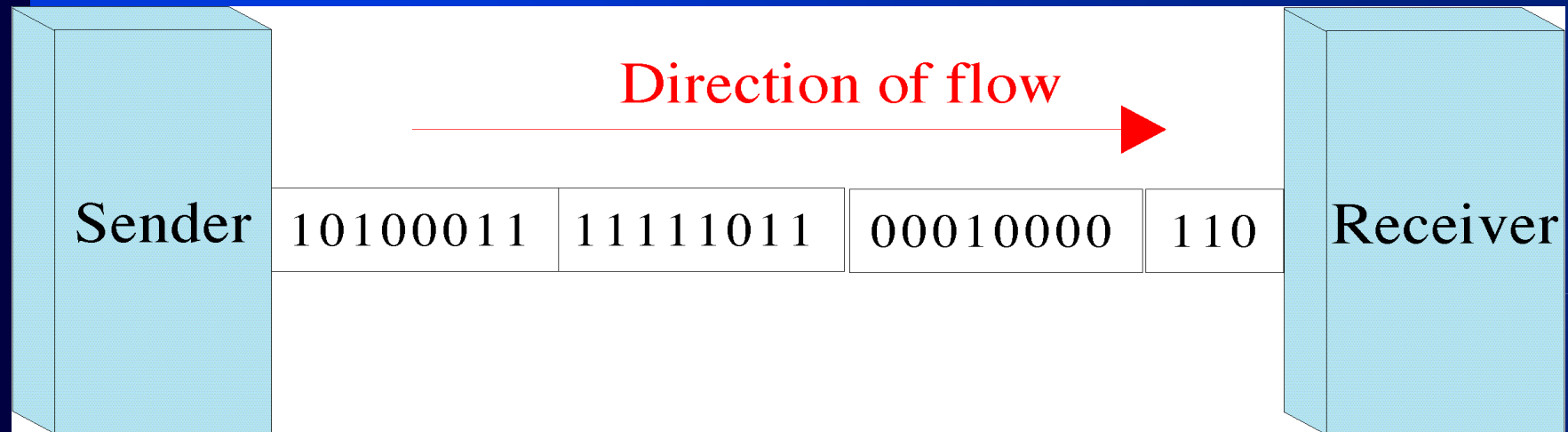
**L'overhead di sincronizzazione è ridotto**







# Synchronous Transmission





# *Tipi di protocolli di linea*

I protocolli sincroni possono essere:

## **orientati al carattere (ormai in disuso)**

- informazione numerica interpretata a gruppi di 8 bit sulla base dei caratteri di un alfabeto (es. ASCII)

## **orientati al bit**

- informazione senza interpretazione diretta
- trasferimento trasparente dei bit



## *Tipi di protocolli di linea*

**Il tipo di protocollo usato è funzione della separazione tra i DTE e del bit rate della linea**

- su linee a basso bit rate, come quelle usate coi modem, si preferisce un protocollo orientato al carattere (es. Kermit, X-Modem simplex, BSC half-duplex) che usa lo stop&wait
- su linee lunghe e ad alta velocità si preferiscono i protocolli orientati al bit (es. HDLC full-duplex) che usano protocolli di controllo a finestra



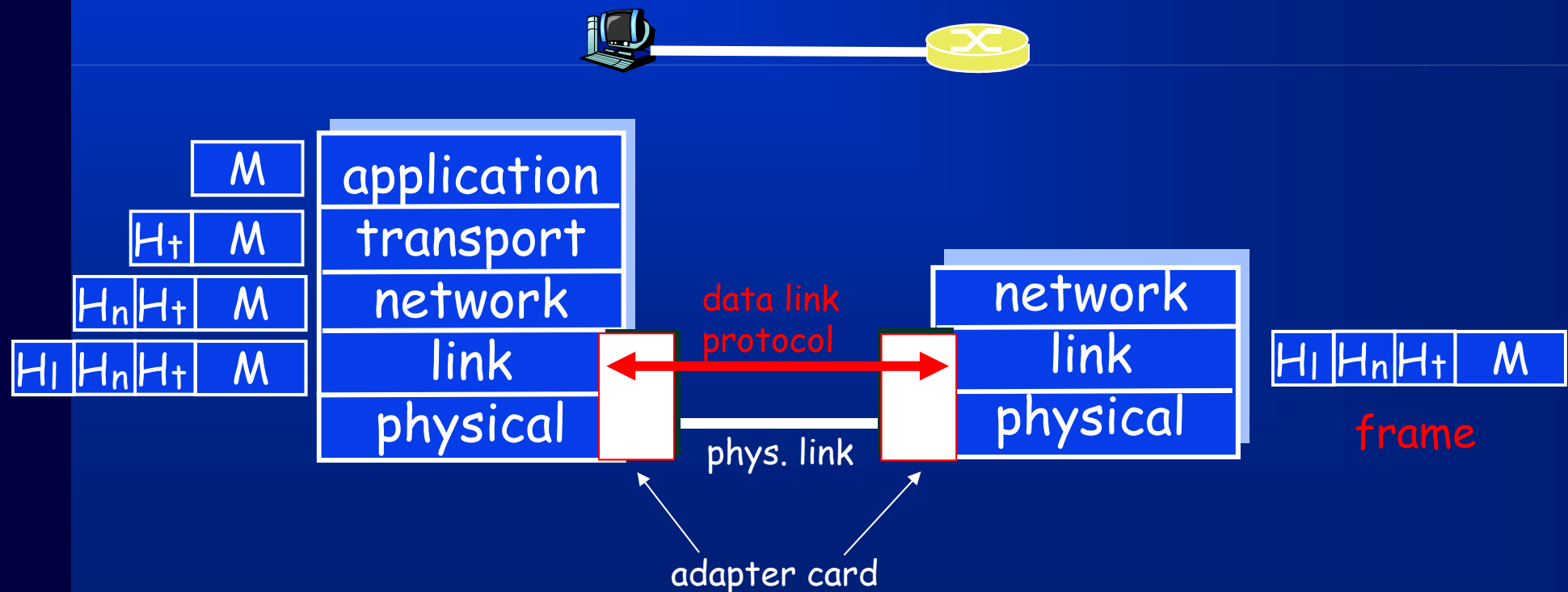
## *Link Layer: Implementazione*

- Il protocollo del livello di link è in gran parte implementato in un “adattatore”
  - es. scheda PCMCIA, scheda Ethernet
  - tipicamente contiene: RAM, chip DSP, interfaccia per il bus dell’host e un’interfaccia del link
  - gli adattatori sono anche conosciuti col nome di schede di interfaccia di rete (Network Interface Card, NIC)

# Link Layer: Implementazione

Per saperne di più sugli adattatori:

<http://www.3com.com/products/nics.html>





## *Funzioni del livello di Link*

- Delimitazione delle trame
- Controllo e recupero da errori
- Controllo di flusso
- Gestione della connessione



## *PDU del livello Link: Trame*

- Nella trasmissione dati i bit trasmessi dal livello fisico sono organizzati in gruppi logici: *trame*
- Le trame sono separate con metodi opportuni e identificate tramite un *header* dal livello di linea
  - trame fisiche (es. TDM) e trame dati
- Perché dividere il flusso in trame:
  - controllo d'errore (checksum)
  - indirizzamento
  - numerazione trame, ecc.



## ***Delimitazione delle trame***

- **la delimitazione delle trame consente al ricevitore di riconoscere l'inizio e la fine senza ambiguità**
- **I delimitatori di trama possono essere:**
  - **sequenze di bit (flag) o caratteri speciali inseriti a inizio e fine trama**
  - **violazione del codice di linea usato a livello fisico (es. manchester)**
  - **temporizzazioni ricevute su canali di servizio**
  - **conteggio dei caratteri o dei bit**





# *Protocolli orientati al carattere*

## **Binary Synchronous Communications (BSC)**

- vecchio protocollo a caratteri di IBM
- uso di caratteri speciali per delimitare trame (SYN) e all'interno header (SOH) e dati (STX, ETX)





# *Protocolli orientati al carattere*

**I problemi nascono quando si vuole trasmettere informazione binaria e non caratteri**

- **trasmettere carattere 1 e 0 invece dei bit**
  - 1 bit => 8 bit: dispendioso!!!
- **organizzare i bit in gruppi di 8 e trasmettere il carattere corrispondente**
  - **problema della trasparenza: i bit del flusso possono casualmente codificare i caratteri speciali usati per la delimitazione**



# *Protocolli orientati al carattere*

- uso del carattere DLE (Data Link Escape) prima di ogni carattere di controllo: ha significato solo la sequenza DLE-carattere di controllo
- i bit del flusso possono codificare casualmente anche il DLE: tecnica del character stuffing

Informazione  
in modalità  
trasparente

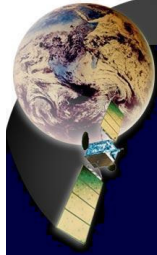
A	b	9	DLE	v	7	X	h
---	---	---	-----	---	---	---	---

character stuffing

A	b	9	DLE	DLE	v	7	X	h
---	---	---	-----	-----	---	---	---	---

trama

DLE	STX	A	b	9	DLE	DLE	v	7	X	h	DLE	ETX
-----	-----	---	---	---	-----	-----	---	---	---	---	-----	-----



# *Protocolli orientati al bit*

**Uso di flag: sequenze di 8 bit**

**Esempio: HDLC**

- sequenza di flag all'inizio e alla fine di una trama

*0 1 1 1 1 1 1 0*

- come impedire una casuale presenza della sequenza di flag nei dati



# Protocolli orientati al bit

informazione  
da trasmettere

111100011111110001001010111111

inserimento bit di stuffing

111100011111011000100101011111011

trama

01111110 111100011111011000100101011111011

flag

01111110

flag

ricezione

~~01111110~~ 1111000111110~~1~~10001001010111110~~1~~

↑  
riconoscimento  
flag d'inizio

↑  
eliminazione di un bit  
dopo 5 uno consecutivi

~~01111110~~

↑  
riconoscimento  
flag di fine



## *Funzioni del livello di Link*

- Delimitazione delle trame
- Controllo e recupero da errori
- Controllo di flusso
- Gestione della connessione



# Controllo d'errore

- In ricezione è possibile che venga riconosciuta una sequenza di bit diversa da quella trasmessa (bit errati)

*10011010100100100101000101000*

*10001010100110100101000111000*

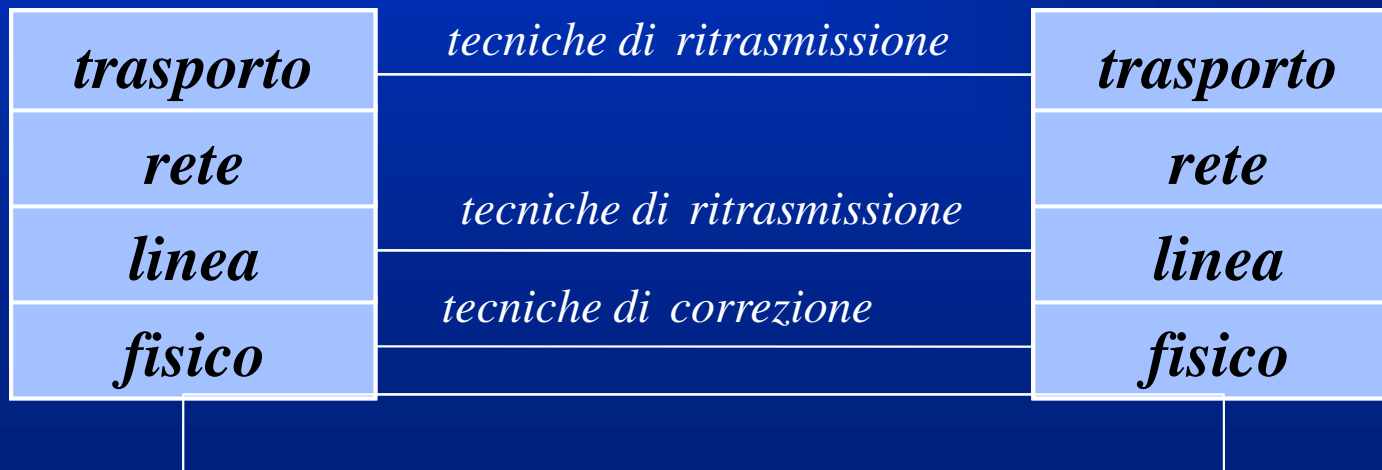
## cause:

- rumore termico (mezzi trasmissivi, apparati di ricezione e trasmissione)
- interferenza da altre trasmissioni sullo stesso mezzo
- disturbi elettromagnetici
- perdite di sincronismo
- ecc.



# Controllo d'errore

- I livelli della pila possono introdurre dei rimedi agli errori collaborando ad offrire un servizio con basso tasso di errore

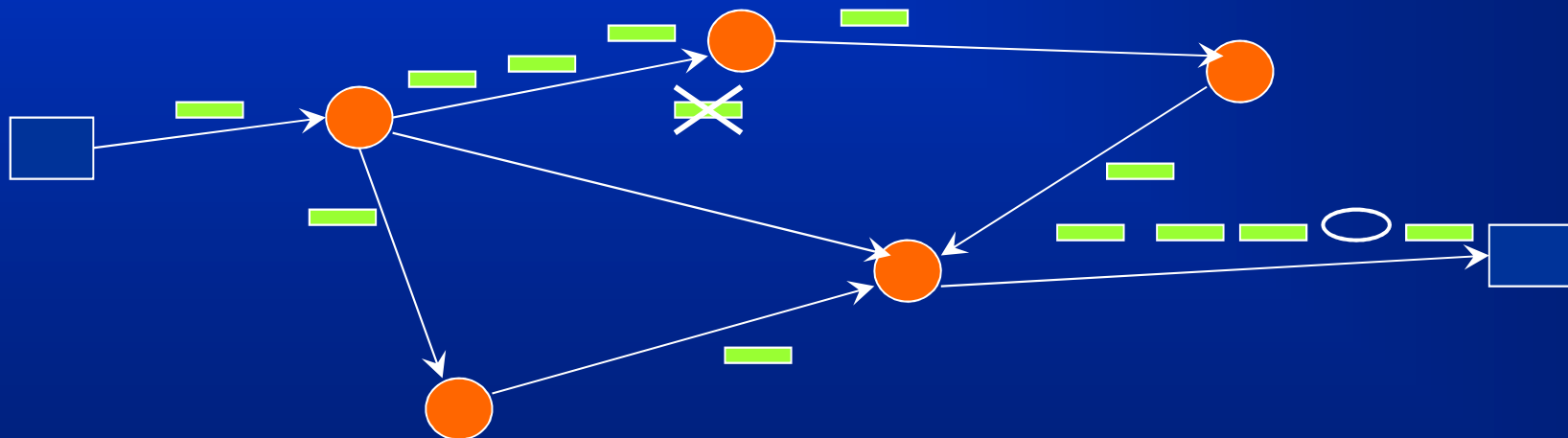






## *Perdita di pacchetti*

- In una rete è possibile che una unità informativa vada persa per overflow dei buffer di rete
- le tecniche di controllo d'errore a livello di trasporto vengono usate di solito per il recupero dei pacchetti persi per overflow





# *Controllo di errore*

I meccanismi per il controllo d'errore possono essere distinti in due classi:

- meccanismi di correzione automatica degli errori
- procedure di ritrasmissione



## **Tecniche per la protezione dagli errori di trasmissione**

- **FEC (forward error correction)**
- **ARQ (automatic retransmission request)**



# *Tecniche di controllo*

## **Correzione (FEC - forward error correction)**

- **codifica di canale**
- **aggiunta di ridondanza in trasmissione**
- **uso della ridondanza in ricezione per rimediare ai bit errati (se in numero contenuto)**
- **teoria dei codici**
- **esempio: codice a ripetizione (N volte lo stesso bit); correzione di  $N/2 - 1$  errori**



# *Tecniche di controllo*

## **Ritrasmissione (ARQ - Automatic Repeat reQuest)**

- aggiunta di ridondanza in trasmissione per ogni unità informativa (FCS - Frame Check Sequence)
- uso della ridondanza in ricezione per rivelare la presenza di errori e non per correggerli
- uso di messaggi di servizio per la richiesta di ritrasmissioni o la conferma di corretta ricezione
  - Richiedono collegamenti half o full duplex



# *Tecniche di controllo*

## **Ritrasmissione (ARQ - Automatic Repeat reQuest)**

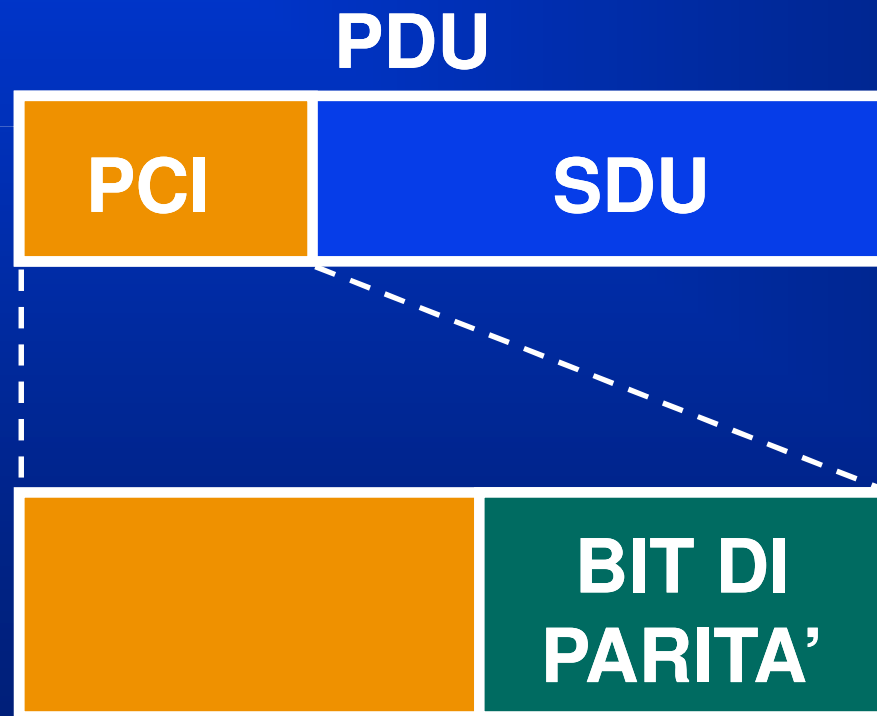
- La ridondanza richiesta per rivelare gli errori è molto più contenuta di quella richiesta per la correzione
  - un'aggiunta di 16-32 bit ai pacchetti è di solito sufficiente
- Il più semplice codice a rivelazione di errore è costituito dal bit di parità: alla fine del pacchetto viene aggiunto un bit pari a 1 se il numero di 1 nel pacchetto è dispari e pari a 0 se il numero di 1 è pari

*0011001100111001101001001110*

*0011010110001111001010001101*



**Si introducono bit di parità  
tra le informazioni di controllo  
all'interno delle PDU**





## Esempi di protezione dagli errori

0 1 1 0 1 0 1 0	0
0 1 0 0 1 0 1 0	1

bit di parità (riconosce  
errori in numero dispari)

0 1 1 0 1 0 1 0
0 1 1 0 1 0 1 0
0 1 1 0 1 0 1 0

codice a ripetizione  
(decisione a maggioranza:  
permette di correggere errori)





## Esempi di protezione dagli errori

0	1	1	0	1	0	1	0	0
0	1	0	0	1	0	1	0	0
0	0	0	1	0	1	0	1	1
1	1	0	0	0	0	1	0	1
1	1	1	0	1	0	1	0	1
0	0	0	0	1	0	1	0	0
0	1	1	1	1	0	1	0	1
0	1	0	0	0	0	1	0	0
0	0	1	0	0	1	1	1	0

parità di riga e colonna  
(consente la correzione  
di errori singoli)



# *Tecniche di controllo*

## **Correzione e rivelazione di errore**

- Date due stringhe binarie di uguale lunghezza,  $X$  e  $Y$ , si definisce **Distanza di Hamming (HD)** tra  $X$  e  $Y$  il numero di bit di cui differiscono
  - Numero di 1 nell'OR esclusivo tra  $X$  e  $Y$
- Dati simboli di  $m$  bit, aggiungendo  $r=n-m$  bit di ridondanza si ottiene un codice con parole (codeword) di  $n$  bit
  - Non tutte le  $2^n$  codeword sono legali
- La minima HD tra due codeword è la HD del codice



# Codifica a blocco per controllo di errore

**n bit**



**m bit utente**

**n-m bit parità**

**$2^m$  possibili combinazioni**

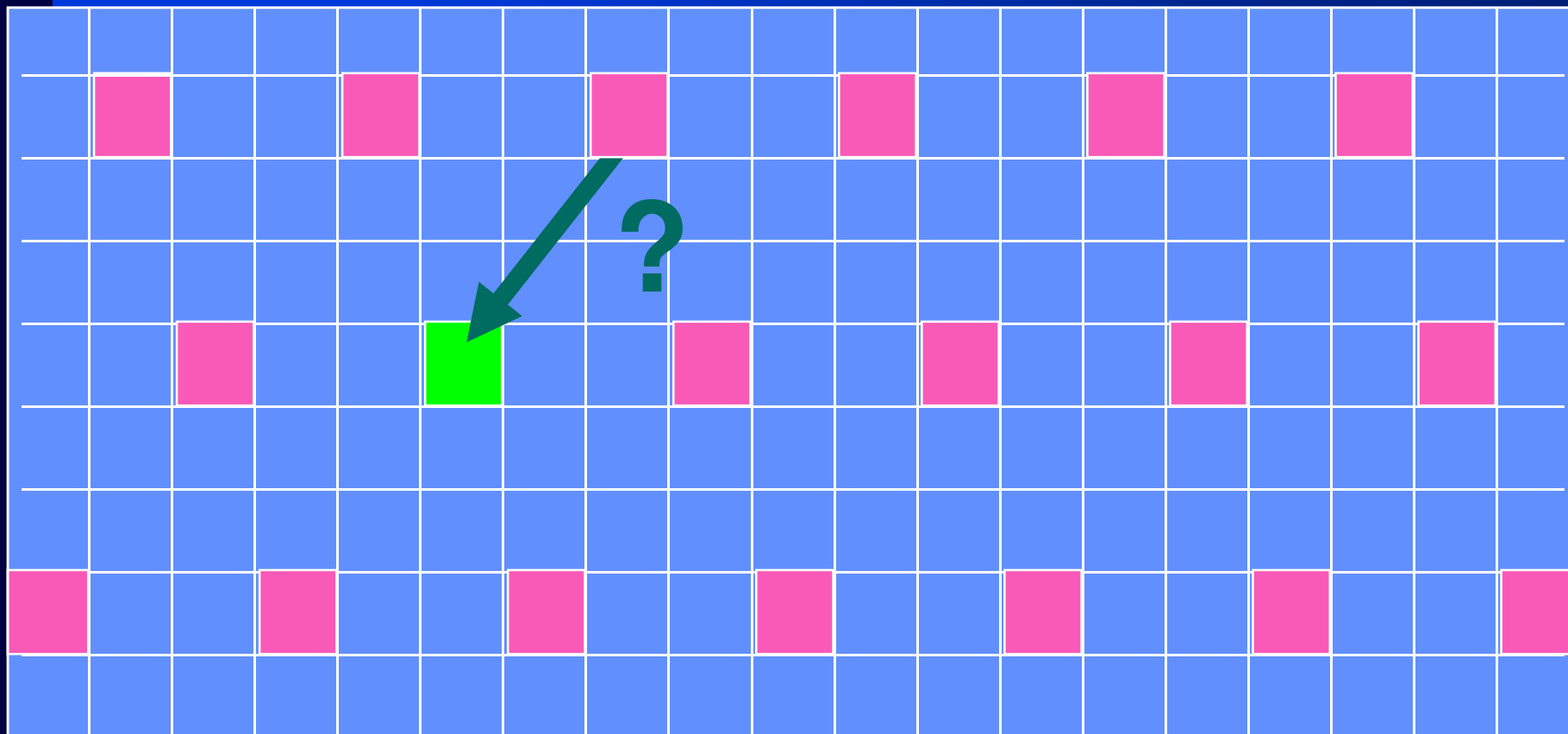






  $2^n$

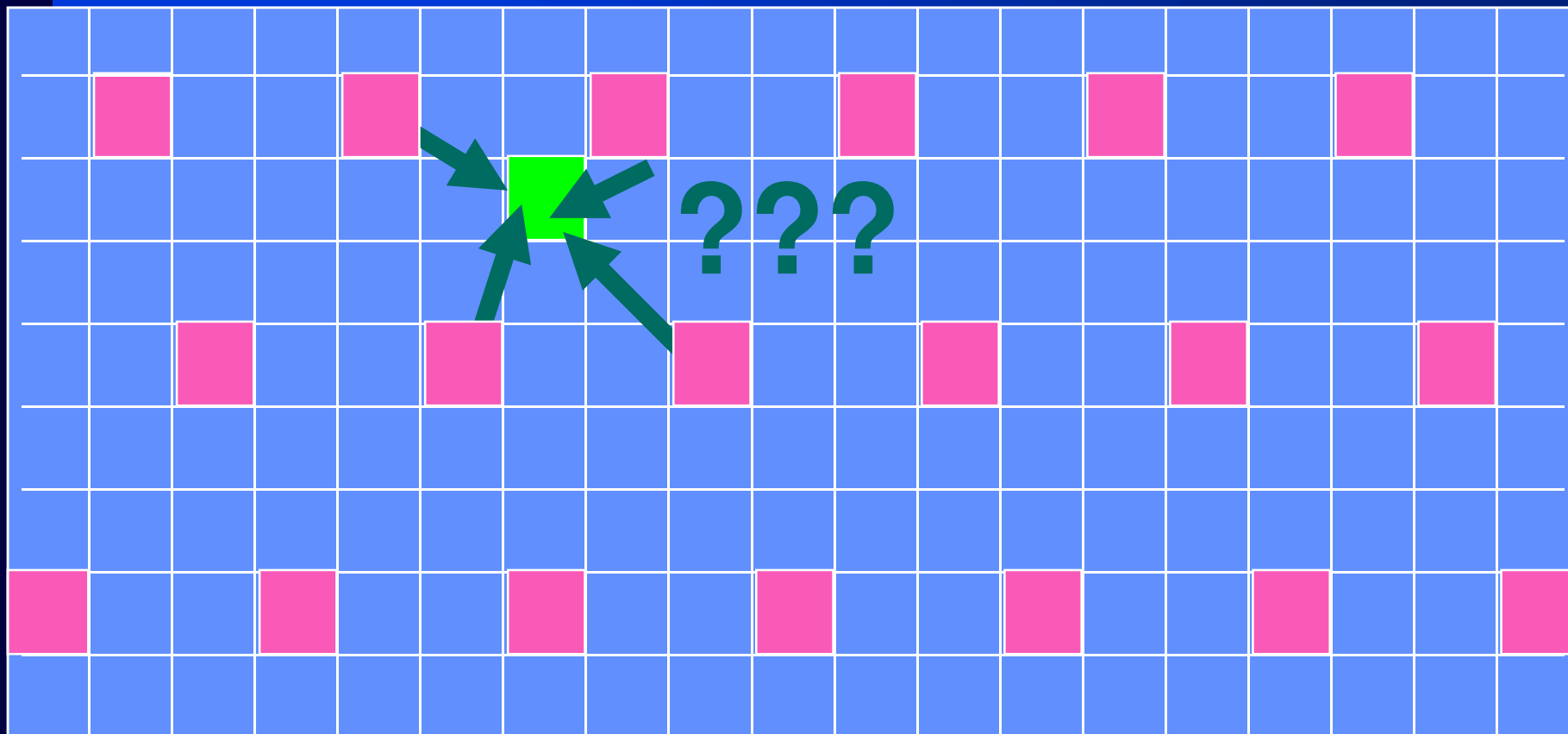
  $2^m$

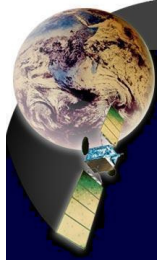




  $2^n$

  $2^m$





# *Tecniche di controllo*

## Correzione e rivelazione di errore

- Le proprietà di correzione e rivelazione di un codice dipendono dalla HD del codice
- Per **rivelare d errori** deve essere  **$HD=d+1$** , così non è possibile che d errori singoli cambino una codeword in un'altra codeword valida
- Per **correggere d errori** il codice deve avere  **$HD=2d+1$**  (può rivelarne fino a 2d), così anche con d errori la codeword alterata è più vicina all'originale che ad ogni altra





# *Tecniche di controllo*

## **Controllo di parità a blocchi**

- Per correggere errori a burst con i bit di parità, si può organizzare una sequenza di  $k$  codeword di  $n$  bit in una matrice  $k \times n$ , una codeword per riga
- Un bit di parità è calcolato per ogni colonna e aggiunto come ultima riga; si trasmette quindi una riga alla volta
- Un errore a burst di lunghezza  $n$  influenzerà al più 1 bit per codeword, così si riesce a correggere l'intero blocco



# Checksum

**Obiettivo:** rivelare errori nei segmenti trasmessi (nota: è usato *solo* al livello di trasporto)

## **Sender:**

- tratta il contenuto del segmento dati come una sequenza di interi a 16-bit
- checksum: somma (somma complemento a 1) del contenuto del segmento
- il sender inserisce il valore della checksum nel campo checksum dell'header

## **Receiver:**

calcola la checksum del segmento ricevuto

controlla se la checksum calcolata è uguale a quella nel campo checksum :

- NO - error detected
- YES - no error detected.  
*Ma potrebbero essercene comunque?*



# *Checksum: Cyclic Redundancy Check*

## **Codici polinomiali (CRC=Cyclic Redundancy Code)**

- I bit di una stringa  $M$  di  $m$  bit da proteggere sono visti come i coefficienti (0 e 1) di un polinomio  $M(x)$ ; l' $i$ -esimo bit è il coefficiente di  $x^{i-1}$
- Ad una stringa di  $k$  bit corrisponde un polinomio di grado  $k-1$
- Per generare il CRC, le DL-entità trasmittente e ricevente si accordano su un polinomio comune detto polinomio generatore  $G(x)$  di grado  $r$ 
  - i bit LSB e MSB devono essere 1



# *Checksum: Cyclic Redundancy Check*

## Codici polinomiali (CRC=Cyclic Redundancy Code)

- L'idea è di appendere una checksum alla fine della sequenza  $M(x)$  in modo che il polinomio complessivo sia divisibile per  $G(x)$
- Il ricevitore prova a dividere la trama ricevuta per  $G(x)$ , se il resto è zero la trama è corretta



## *Checksum: Cyclic Redundancy Check*

- Si appendono  $r$  zeri in coda alla stringa da proteggere, così si ottiene una stringa di  $m+r$  bit che corrisponde al polinomio  $x^rM(x)$
- Si divide  $x^rM(x)$  per  $G(x)$ , usando la divisione modulo 2, e si calcolano il quoziente  $Q(x)$  e il resto  $R(x)$  (polinomio di grado  $r-1$ )
- Si sottrae il resto (formato al più da  $r$  bit) da  $x^rM(x)$  usando la sottrazione modulo 2, il risultato è la trama da trasmettere

$$T(x) = x^rM(x) + R(x) = G(x)Q(x)$$

- Il polinomio risultante  $T(x)$  è divisibile per  $G(x)$



# *Checksum: Cyclic Redundancy Check*

- In ricezione si ottiene  $Y(x) = T(x) + E(x)$ , dove  $E(x)$  rappresenta l'eventuale sequenza di errori
- Si calcola il resto della divisione di  $Y(x)/G(x)$ ; se il resto è zero la stringa ricevuta è corretta (cioè  $E(x)=0$ )
- **Nota: il CRC fallisce se  $E(x)$  è divisibile per  $G(x)$** 
  - Per errori singoli  $E(x)=x^j$ , che non è divisibile per  $G(x)$  se  $G(x)$  contiene almeno due termini
  - Per un qualunque numero dispari di errori,  $E(x)$  non è divisibile per  $G(x)$  se  $G(x)$  contiene il fattore  $(x+1)$
  - Un qualsiasi burst di errori è rivelato, in quanto  $E(x)=x^j(x^{k-1}+\dots+1)$ , con  $k=r$  e  $G(0)=1$
  - Per ogni  $G(x)$  esiste un valore minimo di  $n$  tale che  $x^n+1$  è divisibile per  $G(x)$ , e  $n=2^r-1$



## ***Checksum: Cyclic Redundancy Check***

- Il CRC rivela tutti gli errori a burst con lunghezza minore di  $r+1$  bit
  - tecnica usata da ATM, HDLC, ecc.

## Esempio CRC

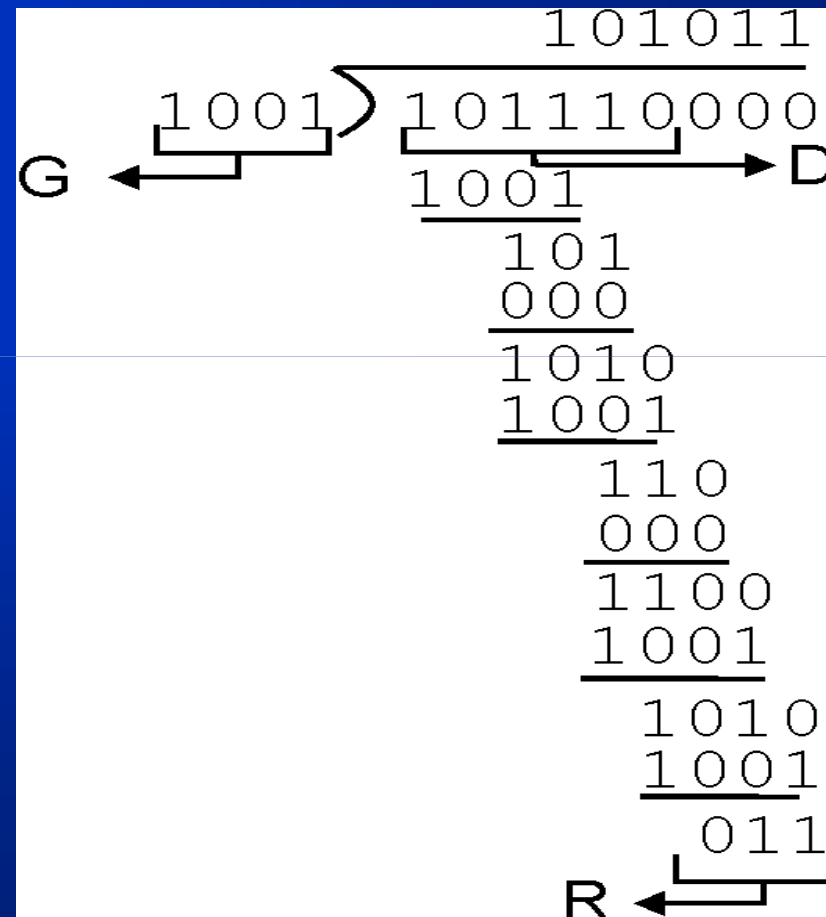
D=messaggio  
originale

G=polinomio  
generatore

R=resto

r=3 bit di ridondanza

$$R(x) = \text{resto} \left[ \frac{x^r D(x)}{G(x)} \right]$$







## **ARQ (automatic retransmission request)**

**controllo congiunto di**

- errore**
- flusso**
- sequenza**

**su una connessione**



# *Controllo di flusso*

## **Obiettivo:**

- regolare la velocità di invio delle unità informative da una sorgente ad una destinazione in modo che tale velocità non sia superiore a quella con la quale le unità informative vengono smaltite a destinazione

## **Livelli**

- livello di linea (2)
- livello di trasporto (4)



## *Schema di esempio*



- **Buffer di ricezione limitato a K posizioni**
- **ritmo di assorbimento dell'utente arbitrario**
- **obiettivo: evitare che pacchetti vadano persi perché all'arrivo trovano il buffer pieno**



## Tre tecniche ARQ

- Stop and wait (Alternating bit)
- Go back N
- Selective repeat