

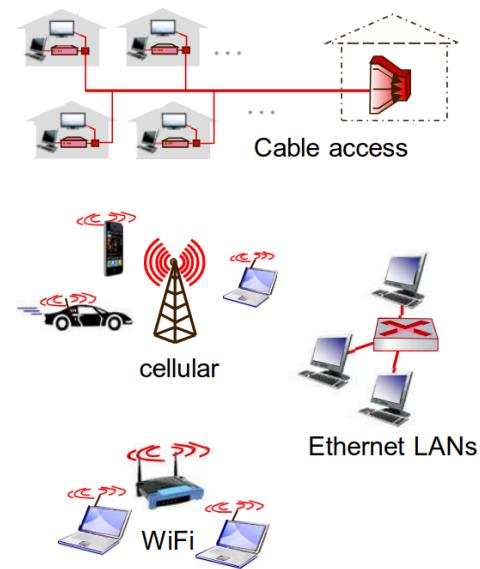


4. Livello di Collegamento

4.1 Servizi del livello di collegamento

Il **livello di collegamento dati** (data link layer) ha il compito di trasferire un pacchetto da un **nodo adiacente** a un altro attraverso un **collegamento fisico diretto**. Si tratta, quindi, di un servizio locale, che opera **hop-by-hop**, al contrario del livello di rete che lavora end-to-end.

Il ruolo fondamentale di questo livello è fornire un **canale di comunicazione affidabile** tra due dispositivi direttamente connessi. A seconda della complessità e delle esigenze del collegamento, questo livello può offrire una gamma più o meno ampia di servizi.



4.1.1 Framing

Il **framing** è il processo mediante il quale il livello di collegamento suddivide il flusso di bit proveniente dal livello fisico in **unità discrete chiamate frame**. Ogni frame incapsula un datagramma del livello di rete, arricchendolo con **informazioni di controllo** (header e talvolta trailer), necessarie per la trasmissione e l'elaborazione.

Un compito cruciale del framing è permettere al ricevente di **identificare correttamente l'inizio e la fine di ciascun frame**. Le tecniche utilizzate possono includere:

- **Byte-counting:** si specifica la lunghezza del frame nel campo di intestazione.
- **Flagging con caratteri speciali** (es. bit stuffing): si usano pattern distintivi per delimitare i frame.
- **Clock-based framing:** utilizzato nelle trasmissioni sincrone.

4.1.2 Rilevazione e correzione degli errori

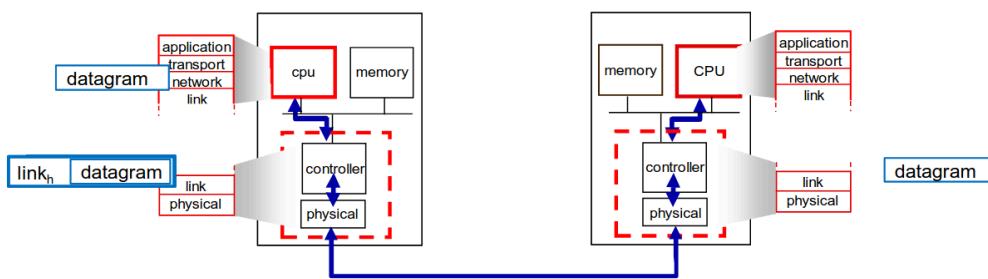
Durante la trasmissione, i bit possono essere alterati da **rumore, interferenze o degradazione del segnale**, specialmente nei canali wireless. Il livello di collegamento affronta questo problema implementando tecniche di:

- **Rilevazione degli errori**, tramite codici come:
 - **Parity bit**
 - **Checksum**

- **CRC (Cyclic Redundancy Check)**, il più diffuso per le sue prestazioni
- **Correzione degli errori**, se richiesta, mediante codici come:
 - **Hamming codes**
 - **Reed-Solomon**
 - **ARQ (Automatic Repeat reQuest)**: il livello chiede la ritrasmissione se rileva un errore (senza correggere direttamente)

4.1.3 Controllo di flusso

Il **controllo di flusso** assicura che un mittente non invii dati più velocemente di quanto il destinatario possa ricevere e processare. Questo è particolarmente importante quando il ricevente ha **buffer limitati** e può subire **sovraffollamento**.



Il meccanismo classico per il controllo di flusso a livello di collegamento è la **finestra scorrevole** (sliding window), che consente di inviare più frame prima di attendere l'acknowledgment, ma impone un limite alla quantità di dati "in volo".

Alcuni protocolli, come **PPP** o **HDLC**, implementano controllo di flusso esplicito, mentre Ethernet lo delega a protocolli superiori.

4.1.4 Controllo di accesso al mezzo (MAC)

Quando più dispositivi condividono lo stesso mezzo trasmissivo (es. nel caso di reti wireless), il livello di collegamento deve gestire **chi ha il diritto di trasmettere** in un dato momento.

I **protocolli MAC** (Medium Access Control) sono progettati per evitare collisioni e ottimizzare l'uso del mezzo. Esistono diverse categorie:

- **Random access**: i nodi trasmettono liberamente, ma gestiscono le collisioni (es. CSMA/CD nelle reti Ethernet legacy, CSMA/CA nelle reti wireless).
- **Accesso deterministico**: l'accesso è controllato da un token o da un sistema a turni (es. Token Ring, polling).
- **Accesso a priorità**: si dà precedenza a certi nodi in base a criteri predefiniti.

La progettazione di un protocollo MAC efficiente è cruciale nelle reti a elevata densità o in ambienti a bassa qualità di segnale.

4.2 Data Framing

Il **data framing** è il processo di organizzazione dei dati in **frame trasmissibili** che possono essere riconosciuti dal ricevente sul collegamento fisico. Oltre all'informazione utile, ogni frame include **bit di controllo** e **delimitatori**.

controllo per la sincronizzazione, la rilevazione degli errori e la delimitazione.

A differenza del framing logico, qui il termine si riferisce alla **codifica fisica dei dati binari** prima della trasmissione. Poiché un segnale elettrico o ottico non può trasportare direttamente una sequenza indefinita di bit uguali (es. sequenze lunghe di 0 o 1), vengono introdotti **schemi di codifica**, detti anche **encoding**, che trasformano i dati in sequenze più adatte al mezzo fisico.

Perché è un problema trasmettere bit "grezzi"?

Nel mondo reale, i dati digitali viaggiano sotto forma di **segnali fisici**: impulsi elettrici, segnali ottici, onde radio, ecc. Ma un lungo flusso di bit come **00000000000000** non genera **variazioni nel segnale**. Il ricevitore, senza cambiamenti di stato, **non ha modo di sapere quanti zeri sono passati**, né **dove finisce un bit e inizi il successivo**.

Il problema della sincronizzazione

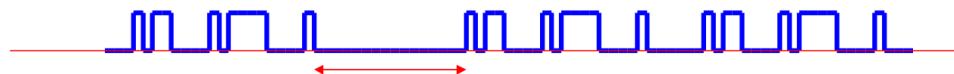
I dispositivi trasmettono e ricevono dati secondo un **clock interno**. Il ricevente deve sapere **quando leggere** i bit in arrivo. Se i due clock non sono perfettamente allineati (e **non lo sono mai** nel tempo), ci si può desincronizzare: leggere il bit nel punto sbagliato e ricevere dati corrotti.

Quando il segnale cambia frequentemente (es. da 0 a 1, da 1 a 0), il ricevente può **risincronizzare il proprio clock** su ogni variazione. Ma se i dati trasmessi sono troppo "piatti" (es. **00000000**), non ci sono variazioni e il clock si sfasatura. Questo genera **errori nella lettura dei dati**.

4.2.1 Codifiche

Le codifiche sono importanti perché **garantiscono transizioni regolari** nel segnale e quindi la possibilità di mantenere sincronizzati i clock.

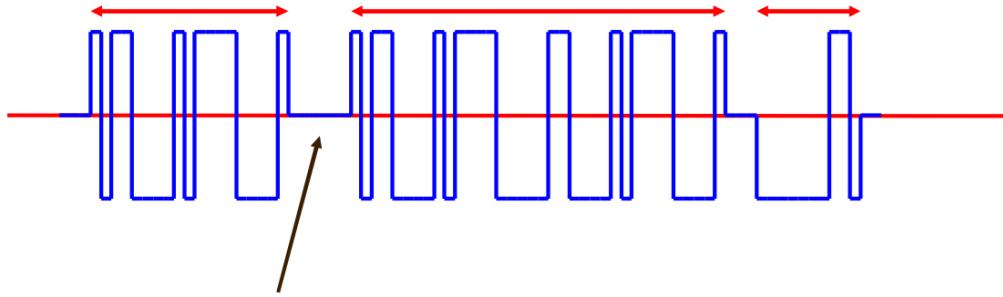
Codifica a 2 livelli



Se si utilizza la codifica a 2 livello non si ha un modo per delimitare inizio e fine di una frame. Infatti, la linea rossa indica un'assenza di segnale o l'inizio di una nuova frame?

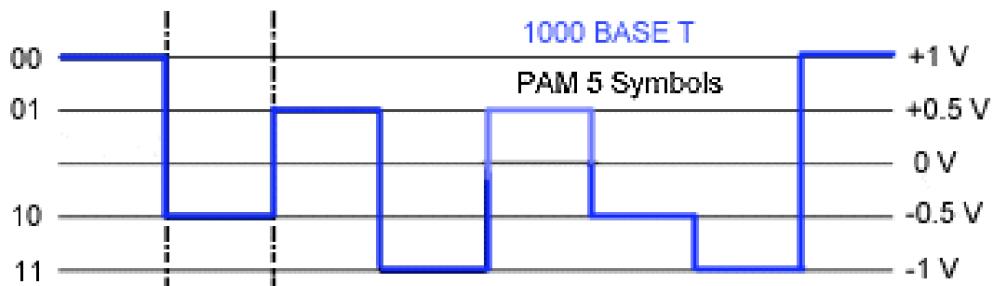
Codifica a 3 livelli

Si può pensare allora a una codifica a 3 livelli, dove il segnale intermedio sta per inizio di una nuova frame. In questo modo però si spreca bitrate, perché un livello è interamente buttato per le nuove frame.



4.2.2 Codifica 4B5B

La codifica **4B5B** trasforma ogni blocco di 4 bit in un blocco di 5 bit. Lo scopo è garantire che ci siano **abbastanza transizioni** nel segnale trasmesso per mantenere la **sincronizzazione tra mittente e destinatario**. Si ha quindi una codifica a 5 livelli e ad ognuno si associa una coppia di bit. Il livello intermedio va però buttato.



Perché aggiungere un bit?

Trasmettere sequenze lunghe di 0 (o di 1) può causare **perdita di sincronizzazione del clock**. L'aggiunta di un quinto bit consente di **controllare** la sequenza risultante: i 5-bit risultanti vengono scelti in modo da **non contenere più di tre 0 consecutivi**, evitando sequenze problematiche.

Funzionamento:

- Viene definita una **tabella di codifica** che associa ciascuna sequenza di 4 bit a una sequenza di 5 bit.
- Alcuni codici 5-bit vengono riservati per **codici speciali** (es. delimitatori, controllo).
- La decodifica è univoca: ogni 5-bit ricevuto può essere mappato al suo 4-bit corrispondente.

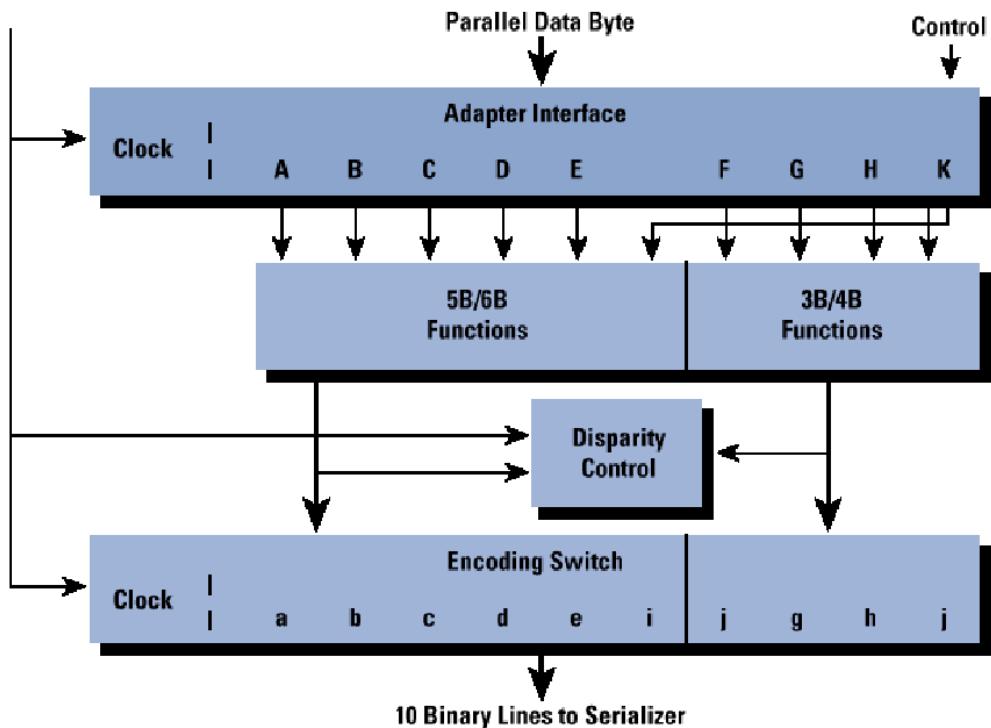
Nome	4B	5B	Descrizione
0	0000	11110	hex data 0
1	0001	01001	hex data 1
2	0010	10100	hex data 2
3	0011	10101	hex data 3
4	0100	01010	hex data 4
5	0101	01011	hex data 5
6	0110	01110	hex data 6
7	0111	01111	hex data 7
8	1000	10010	hex data 8
9	1001	10011	hex data 9
A	1010	10110	hex data A
B	1011	10111	hex data B
C	1100	11010	hex data C
D	1101	11011	hex data D
E	1110	11100	hex data E
F	1111	11101	hex data F
I	-NONE-	11111	Idle
J	-NONE-	11000	SSD #1
K	-NONE-	10001	SSD #2
T	-NONE-	01101	ESD #1
R	-NONE-	00111	ESD #2
H	-NONE-	00100	Halt

4.2.3 Codifica 8B10B

La codifica **8B10B** mappa ogni blocco di **8 bit** in **10 bit**. È stata progettata per:

- garantire **transizioni frequenti** (per la sincronizzazione),

- mantenere un bilanciamento tra 0 e 1 (per evitare **accumulo di carica** nel segnale),
- inserire **caratteri speciali** (es. delimitatori di frame).



Funzionamento:

La codifica avviene in due fasi:

1. **Separazione** degli 8 bit in due gruppi:
 - i primi 5 bit (detti **5b**)
 - i successivi 3 bit (**3b**)
2. Ognuno viene codificato separatamente:
 - **5b → 6b**
 - **3b → 4b**

Totale: $6 + 4 = 10$ bit

3. Il risultato tiene conto del **disparity** (bilanciamento tra 0 e 1). Se il flusso ha troppi 1, si può usare una variante **inversa** della codifica (inversione di alcuni bit), per mantenere il segnale bilanciato.

Esempio (semplificato):

- Input: **10101010** (8 bit)
- Output codificato: **1001110100** (10 bit)

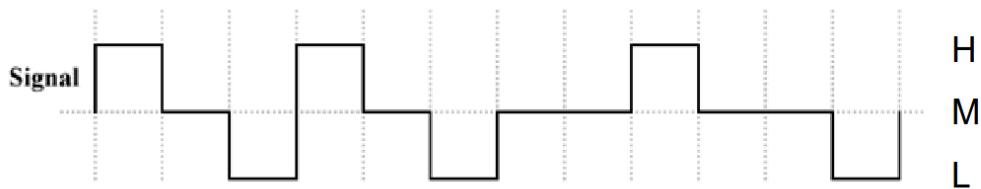
Il ricevitore utilizza una tabella nota per decodificare i blocchi ricevuti, riconoscendo anche eventuali errori se riceve combinazioni non valide.

4.2.4 Differenze tra 4B5B e 8B10B

Caratteristica	4B5B	8B10B
Bit in ingresso	4	8
Bit in uscita	5	10
Overhead	25%	25%
Obiettivo	Transizioni	Transizioni + bilanciamento
Utilizzo	FDDI, Token Ring	Gigabit Ethernet, Fibre Channel

4.2.5 Quantità di informazione

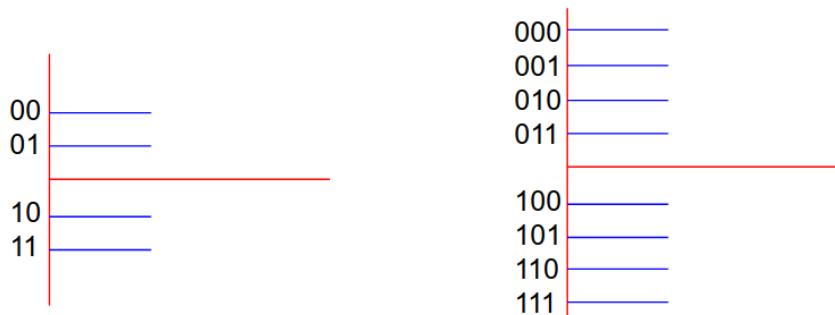
Con una codifica a 3 livello ogni step ha 3 possibili valori: **High**, **Medium** e **Low**.



Ipotizzando che la comunicazione si faccia a più "step", cioè il valore in un determinato istante di tempo, le combinazioni diventano tante. Con 2 step consecutivi, si hanno 9 possibili combinazioni: HH, HM, etc... Le combinazioni di bit utilizzabili quindi sono pari alla potenza di 2 subito minore del numero di combinazioni possibili.

Steps	Combinazioni	Bit	Combinazioni utilizzabili
1	3	1	$2^1 = 2$
2	9	3	$2^3 = 8$
3	27	4	$2^4 = 16$
4	81	6	$2^6 = 64$
5	243	7	$2^7 = 128$
6	729	9	$2^9 = 512$

Possiamo parlare di simboli come **l'unità elementare di informazione trasmessa sul canale fisico**. A differenza del **bit** (un concetto logico, 0 o 1), il simbolo è **cio che effettivamente viene inviato nel segnale**: può essere una variazione di tensione, una fase, una frequenza, un livello ottico ecc.



Se prendiamo un sistema con n **livelli fisici distinti**, allora ogni simbolo può rappresentare fino a:

$$\log_2 n \text{ bit}$$

Per esempio, un sistema con 4 livelli ha **2 bit per ogni simbolo**, invece uno a 8 ne avrà 3.

Quindi ogni simbolo trasporta una quantità di informazioni pari a

$$x = \log_2 n$$

Nel caso della codifica a 3 livelli, per esempio, ogni simbolo trasporta $x = \log_2 3 = 1,584\dots$

Steps	Combinations	Information (bits)	Usable bits	Usable combinations
1	3	1.584963	1	2
2	9	3.169925	3	8
3	27	4.754888	4	16
4	81	6.33985	6	64
5	243	7.924813	7	128
6	729	9.509775	9	512
7	2187	11.09474	11	2048
8	6561	12.6797	12	4096
9	19683	14.26466	14	16384
10	59049	15.84963	15	32768

Bitrate

Dalla quantità di informazione trasmessa possiamo estrapolare il **bitrate**: ovvero il numero di bit trasmessi in un determinato tempo in secondi.

Ipotizziamo di avere una velocità fissata di 12 sps (steps per secondo). Il limite teorico del bitrate è pari a $12 \cdot \log_2 3$.

Steps per symbol	Combinations per symbol	Usable combinations	Bits per symbol	bps
1	3	2	1	$12 \times 1 = 12 \text{ bps}$
2	9	8	3	$6 \times 3 = 18 \text{ bps}$
3	27	16	4	$4 \times 4 = 16 \text{ bps}$
4	81	64	6	$3 \times 6 = 18 \text{ bps}$
6	729	512	9	$2 \times 9 = 18 \text{ bps}$
12	531441	524288	19	$1 \times 19 = 19 \text{ bps}$

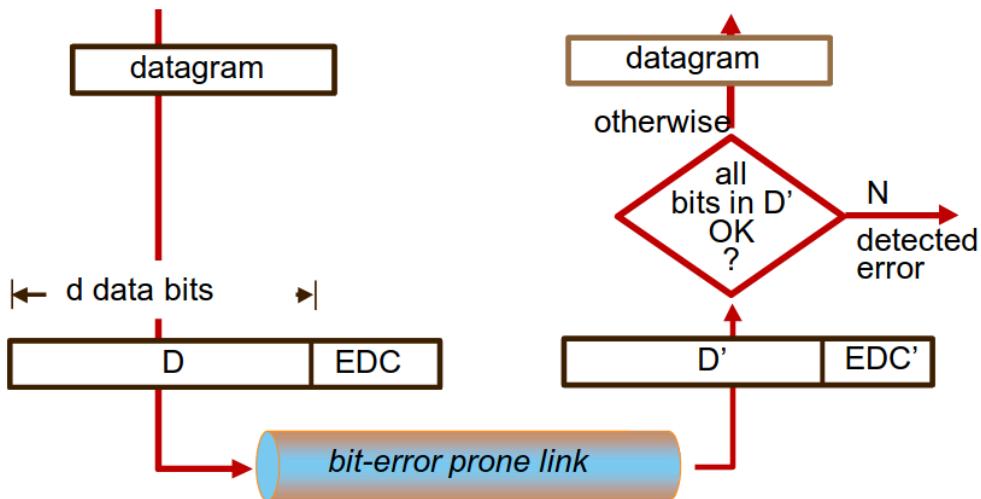
4.3 Rilevazione e correzione degli errori

La trasmissione su un canale fisico può introdurre errori nei bit. Il livello di collegamento affronta questo problema attraverso due approcci fondamentali: **rilevazione** e **correzione** degli errori.

4.3.1 Ridondanza e rilevazione

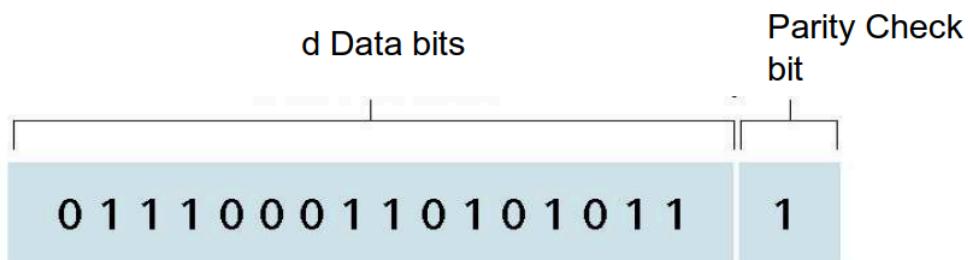
L'idea alla base della rilevazione degli errori è l'**introduzione di ridondanza**: si aggiungono bit extra (chiamati **EDC**, *Error Detection and Correction bits*) ai dati, in modo da poter identificare (e, in alcuni casi, correggere) eventuali modifiche non intenzionali ai bit originali durante la trasmissione.

Un esempio concreto è il codice fiscale: la lettera finale è calcolata in funzione dei caratteri precedenti, agendo come controllo di integrità.

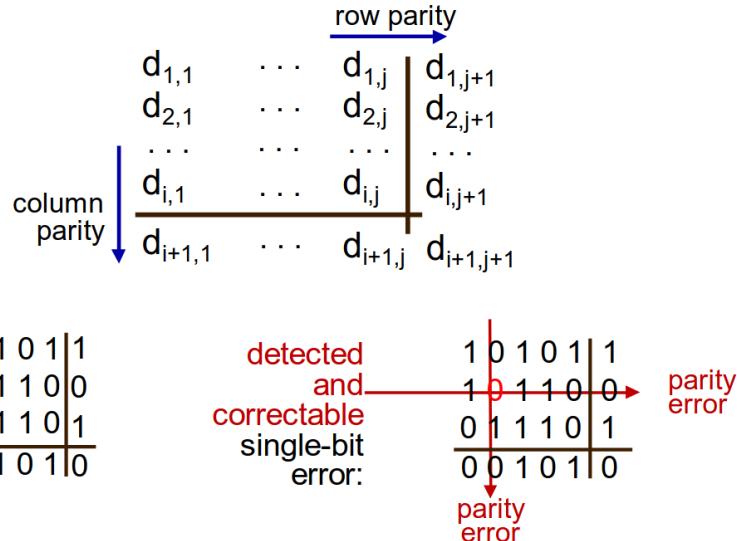


4.3.2 Parity check

Il **controllo di parità** è una delle tecniche più semplici. In esso si aggiunge un singolo bit che forza il numero totale di 1 nei dati a essere **pari (even parity)** o **dispari (odd parity)**. Alla ricezione, si verifica se il numero di bit "1" rispetta ancora la regola: se no, è stato introdotto un errore.



Per migliorare la capacità di localizzare e correggere l'errore si può usare la **parità bidimensionale** (2D parity), che aggiunge parità per righe e colonne di una matrice di bit. Questo consente di **individuare e correggere un singolo errore** di bit.

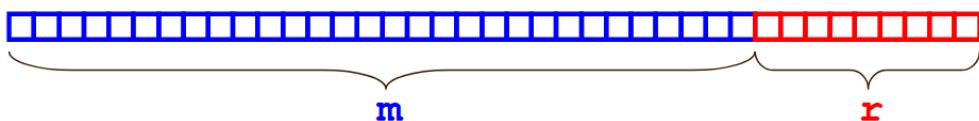


4.3.3 CRC: Controllo a ridondanza ciclica

Una tecnica di rilevazione dell'errore largamente utilizzata è basata sui **codici di controllo a ridondanza ciclica** (CRC, Cyclic Redundancy Check). I codici CRC sono anche detti **codici polinomiali** in quanto è possibile vedere la stringa di bit da trasmettere com un polinomio i cui coefficienti sono i bit della stringa, con le operazioni sulla stringa di bit interpretate come aritmetica polinomiale.

Sia $M(x)$ il polinomio del messaggio di bit m , si procede così:

1. Si sceglie un **polinomio generatore** $G(x)$ di grado r .
2. Si calcola il prodotto $x^r \cdot M(x)$
3. Si divide il prodotto per $G(x)$
4. Il resto della divisione $R(x)$ è la sequenza di controllo (CRC) e viene aggiunta in coda al messaggio



Il frame trasmesso diventa dunque:

$$T(x) = x^r M(x) + R(x)$$

E a destinazione, per controllare la presenza di errori esegue:

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

E se è verificata allora il messaggio dovrebbe essere corretto.

In generale, la formula della CRC è:

$$(x^r M(x) + R(x)) \bmod G(x) = 0$$

Esercizio

$D = 101110, G = 1001$

$$M(x) = x^6 + x^3 + x^2 + x$$

$$G(x) = x^3 + 1 \rightarrow r=3 \rightarrow \text{si aggiungono 3 bit a } D$$

$$x^r \cdot M(x) = x^3(x^6 + x^3 + x^2 + x) = x^8 + x^6 + x^5 + x^2$$

$$= 101110000$$

Poloceo $R(x) = 101110000 \mid 1001 \longrightarrow R(x) = 11$

$$\begin{array}{r} 1001 \\ 1010000 \\ 1001 \\ \hline 11000 \\ 1001 \\ \hline 1010 \\ 1001 \\ \hline 11 \end{array}$$

Il messaggio completo $T(x) \bmod G(x) = 0 ?$

$T(x)$ diventa 101110011 Si

$$\begin{array}{r} 101110011 \mid 1001 \\ 1001 \\ \hline 1010011 \\ 1001 \\ \hline 1011 \\ 1001 \\ \hline 1001 \\ 1001 \\ \hline 0 \end{array}$$

Limitazioni della CRC

CRC è molto efficace nel rilevare errori singoli o corti, ma alcune combinazioni di errori multi possono non essere rilevate se coincidono con un multiplo del generatore.

4.3.4 Distanza di Hamming

La **distanza di Hamming** è una misura utilizzata per valutare quanto due stringhe binarie differiscono tra di loro. Equivale, tra due stringhe di **uguale lunghezza** chiamate **codeword**, al numero di posizioni in cui bit differiscono.

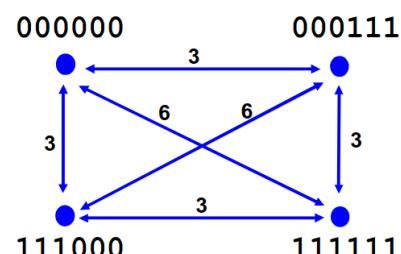
$$H(a, b) = \text{numero di bit diversa tra } a \text{ e } b$$

Esempio:

```
a: 1 0 1 1 1 0 1
b: 1 0 0 1 0 0 1
↑ ↑ ↑ → 3 differenze
```

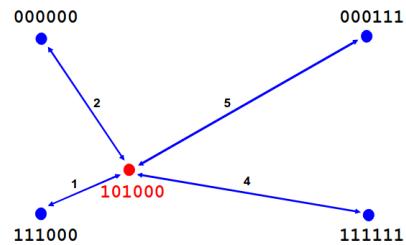
Possiamo definire un **vocabolario** come un insieme di **codeword** e si può calcolare la distanza tra di esso grazie alla distanza di **Hamming**. Un vocabolario, si dice **completo**, se contiene 2^n codewords, dove n è il numero di bit di ogni codeword.

Ipotizziamo un vocabolario con parole di lunghezza 6, formato dalle codewords $000000, 000111, 111000, 111111$. Chiamiamo la **distanza del vocabolario** il minimo tra tutte le distanze calcolate tra tutte le possibili coppie di codewords, in questo caso 3.



Correzioni degli errori

Ipotizzando sia stata trasmessa la codeword **101000**, dobbiamo calcolare la distanza tra il valore trasmesso e quelli validi. Se la probabilità di avere un solo errore è molto maggiore di quella di averne più di 1, si può tentare la correzione.



Se viene ricevuta una parola non valida, si cerca quella **valida più vicina**. Questo metodo si basa sull'assunzione che **gli errori singoli siano molto più probabili di errori multipli**.

Per poter correggere fino a e errori, le code words devono essere distanti almeno:

$$d_{min} \geq 2e + 1$$

Quindi, $d \geq 3$ corregge 1 errore, $d \geq 5$ corregge 2 errori, $d \geq 2$ può solo rilevare 1 errore ma non correggerlo.

Consideriamo tutte le combinazioni di 2 bit (00, 01, 10, 11). Abbiamo diversi scenari:

Scenario 1

Ipotizzando che tutte le combinazioni siano codewords valide, la distanza minima è solo 1. Quindi se riceviamo 01 possiamo dire se era corretto o meno, ma non sappiamo come correggerlo poiché dista da tutte le codewords possibili 1

Scenario 2

Ipotizzando che le codewords valide siano 00 e 11, la distanza tra di loro è 2. Quindi possiamo correggere 1 errore ma solo in alcuni casi.

Ricevuto	Distanza da 00	Distanza da 11	Correggo in
00	0	2	00
01	1	1	Ambiguo
10	1	1	Ambiguo
11	2	0	11

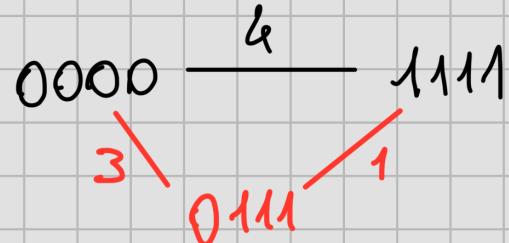
Notiamo dunque che questo controesempio dimostra che la distanza minima per poter correggere errori è di 3.

Esercizio

2

PSG: 0111

la correzione è 1111



Progettazione di un vocabolario

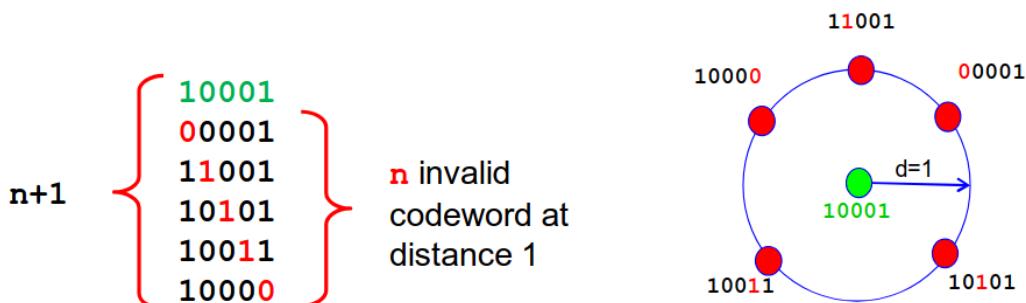
Voglio costruire un codice a

$$n = m + r \text{ bit}$$

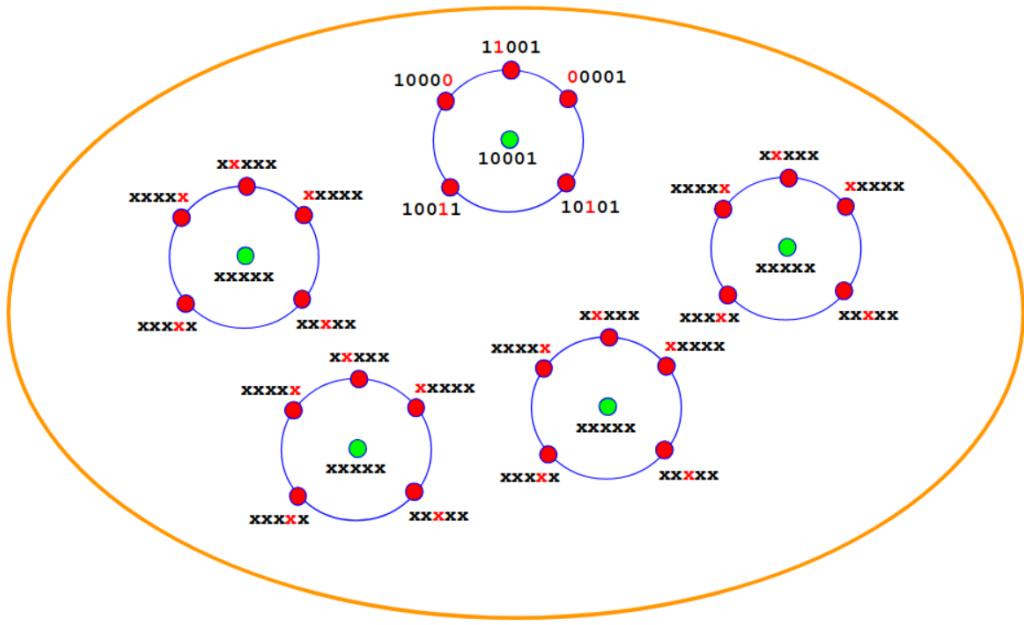
in grado di correggere 1 errore, con m bit del messaggio originale e r bit di ridondanza. Da questo sappiamo che ci sono 2^n codewords in totale, anche se non tutte valide (perché la distanza minima comunque deve essere 3).

Una codeword valida può generare **sé stessa o tutte le combinazioni con un solo bit cambiato** (in totale ce ne sono n). Quindi ogni codeword valida quindi può generare $1 + n$ codewords invalide.

Prendendo come esempio 10001:



Quindi, date 2^n codewords, solo 2^m sono valide. Sappiamo anche, che per correggere 1 bit di errore, la distanza minima deve essere 3.



Per costruire un vocabolario valido possiamo sfruttare l'ipotesi che la distanza minima è 3. Perché se, per assurdo, gli insiemi non fossero disgiunti, esisterebbero due codewords con distanza 2, ma questo appunto è assurdo.

Quindi possiamo dire:

$$\begin{aligned}
 (n+1)2^m &\leq 2^n && \text{sostituisco } n = m+r \\
 (m+r+1)2^m &\leq 2^{m+r} && \text{divido per } 2^m \\
 (m+r+1) &\leq 2^r \\
 m+1 &\leq 2^r - r
 \end{aligned}$$

(Questo è valido solo per distanza 3)

Esempio 1

$$\begin{aligned}
 m = 8 \Rightarrow r = 4 \\
 (8 + 4 + 1) \leq 2^4 \Rightarrow 13 < 16
 \end{aligned}$$

Esempio 2

$$\begin{aligned}
 m = 11 \Rightarrow r = 4 \\
 (11 + 4 + 1) \leq 2^4 \Rightarrow 16 = 16
 \end{aligned}$$

Vocabolario con $d = 5$

Vogliamo costruire un vocabolario con $d_{min} = 5$, ma sappiamo che $d_{min} \geq 2e + 1$ quindi possiamo dire:

$$e = \lfloor \frac{d_{min} - 1}{2} \rfloor = \lfloor \frac{5 - 1}{2} \rfloor = 2$$

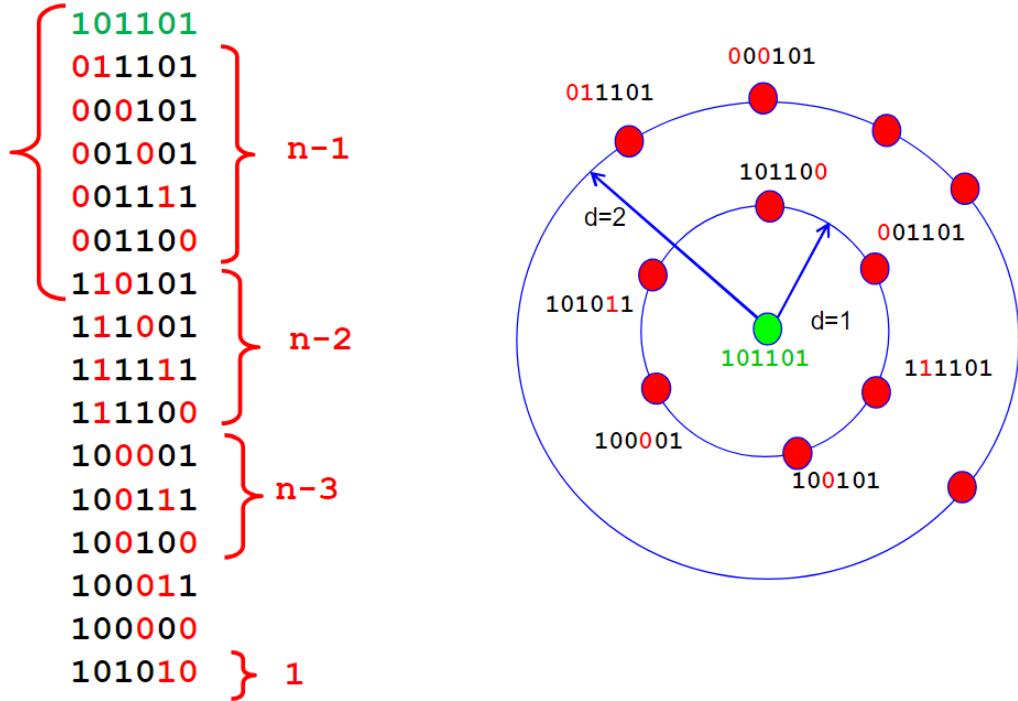
Quindi dobbiamo individuare una "sfera" attorno a ogni codeword di parole invalide, di raggio $d = 2$.

Quindi avendo una certa codeword v di lunghezza n , possiamo dire che:

- per $d = 0$ esiste solo la parola stessa come codeword invalida.
- per $d = 1$ esistono n codewords invalide, una per ogni bit cambiato di v .
- per $d = 2$ esistono codewords invalide in base alla posizione dei bit cambiati
 - se cambiamo il bit in posizione 1 mi rimangono le posizioni da 2 a n , quindi $n - 1$

- se cambiamo il bit in posizione 2 mi rimangono le posizioni da 3 a n , quindi $n - 2$
- se cambiamo il bit in posizione $k < n$, mi rimangono le posizioni da k a n , quindi $n - k$
- se cambiamo il bit in posizione $n - 1$, rimane solo il bit in posizione n , quindi 1 sola codeword.

Questo si può intendere come un coefficiente binomiale $\binom{n}{d}$.



Da questo possiamo costruire la sfera di codewords invalide V rispetto a v sommando le codewords a distanza 0, 1 e 2:

$$V = \sum_{d=0}^2 \binom{n}{d} = \underbrace{\binom{n}{0}}_{d=0} + \underbrace{\binom{n}{1}}_{d=1} + \underbrace{\binom{n}{2}}_{d=2} = 1 + n + \frac{n(n-1)}{2}.$$

Per dimostrare $d_{min} = 5$ nonostante la sfera di raggio 2, ragioniamo per assurdo: se due codewords fossero più vicini, esisterebbe almeno una stringa binaria che appartiene ad entrambe le loro sfere di raggio 2 e quindi non sarebbe più valido il vocabolario.

Quindi sia M il numero di **codewords valide**, allora esistono esattamente M sfere distinte (una per codeword). Se il codice ha $M = 2^m$ codewords valide, allora tutte le M sfere devono stare disgiunte nello spazio delle codewords di dimensione 2^n , quindi:

$$M \cdot V \leq 2^n$$

$$V = \sum_{d=0}^2 \binom{n}{d}$$

$$= 1 + n + \frac{n(n-1)}{2}$$

$M = 2^m$ codice lineare $[n, m]$ ha 2^m codeword

$$2^m \left(1 + n + \frac{n(n-1)}{2} \right) \leq 2^n$$

$n = m + r$ sostituisco $M = 2^m$, V

$$1 + (m+r) + \frac{(m+r)(m+r-1)}{2} \leq 2^r$$

$$2 \left[1 + m + r + \frac{(m+r)(m+r-1)}{2} \right] \leq 2^{r+1}$$

$$m^2 + (2r+1)m + r^2 + r + 2 \leq 2^{r+1}$$

introduco $r = n - m$

divido per 2^m

moltiplico per 2

sviluppo $(m+r)(m+r-1)$

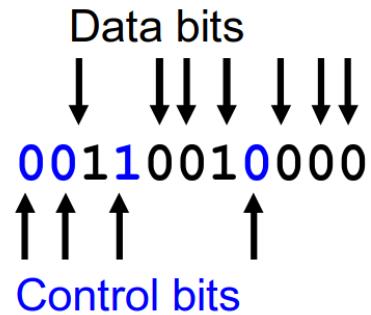
Quindi l'equazione per trovare r bit ridondanti, conoscendo m bit di messaggio, in un vocabolario con $d_{min} = 5$ è

$$m^2 + (2r+1)m + r^2 + r + 2 \leq 2^{r+1}$$

4.3.5 Codici di Hamming

I codici di Hamming sono un insieme di codici lineare ideati per correggere un singolo errore in trasmissione con ridondanza minima e distanza di Hamming 3.

Nelle **posizioni pari a potenze di 2** si pongono i **bit di controllo**, che vengono calcolati con la parità in base ai valori che si trovano nelle posizioni di indice pari alla somma di quelle posizioni stesse. Se dovessimo trovare il bit in prima posizione, si calcola in base ai valori dei bit in posizione 3, 5, 7, 9, 11, 13, 15.



Quindi, per calcolare il bit di parità in posizione 2^i si controllano tutte le posizioni il cui numero binario ha un 1 nell' i -esimo bit. Se abbiamo 0001, b_1 controlla tutte le posizioni in cui il numero binario ha un 1 nel primo bit (partendo da destra).

In definitiva la stringa, considerando p_i bit i -esimo di posizione con b_j bit j -esimo di controllo, si calcola con:

$$p_i = \bigoplus_{\substack{1 \leq j \leq n \\ (j \& 2^{i-1}) \neq 0}} b_j$$

Dove $\&$ indica l'AND bit a bit.

Esempio

Dati originali:

10010001100

Distribuzione dei bit (posizioni da sinistra a destra, 1-based):

Posizione:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Contenuto:	x	x	1	x	0	0	1	x	0	0	0	1	1	0	0
	b1	b2	d1	b4	d2	d3	d4	b8	d5	d6	d7	d8	d9	d10	d11

Posizione	Binario (1 + 2 + 4 + 8)
3	1 + 2
5	1 + 4
6	2 + 4
7	1 + 2 + 4
9	1 + 8
10	2 + 8
11	1 + 2 + 8
12	4 + 8
13	1 + 4 + 8
14	2 + 4 + 8
15	1 + 2 + 4 + 8

- **b1**: $3 \oplus 5 \oplus 7 \oplus 9 \oplus 11 \oplus 13 \oplus 15$
- **b2**: $3 \oplus 6 \oplus 7 \oplus 10 \oplus 11 \oplus 14 \oplus 15$
- **b4**: $5 \oplus 6 \oplus 7 \oplus 12 \oplus 13 \oplus 14 \oplus 15$
- **b8**: $9 \oplus 10 \oplus 11 \oplus 12 \oplus 13 \oplus 14 \oplus 15$

Ipotizzando il messaggio in figura, include 4 bit di parità (i posizioni 1, 2, 4 e 8).

Per ogni bit di parità si **ricalcola il valore XOR delle posizioni che dovrebbe controllare**, se uno o più bit di parità **risultano sbagliati** si ottiene un numero binario che rappresenta **la posizione del bit errato**.

101100100101100
 ↑↑↑↑↑↑
 1 2 4 8 10

Bit di parità	Valore
b_1	1
b_2	1
b_4	1
b_8	1

$$\begin{aligned} b_1 &= 1 \\ b_2 &= 1 \\ b_4 &= 1 \\ b_8 &= 1 \end{aligned}$$

$$2 + 8 = 10 !$$

Questo dà come risultato 1010, ovvero 10 in binario. Questo **significa che il bit in posizione 10 è errato** e basta invertirlo così che il codice **diventi corretto**.

4.3.6 CRC e Hamming

Nella realtà, i codici di Hamming e la CRC sono utilizzati insieme nei collegamenti molto rumorosi. L'hamming, data una sequenza in input, ritorna sempre una sequenza "corretta" ma non è detto che questo output sia davvero quello desiderato. Si introduce quindi la CRC come ridondanza, che ritorna se il valore è corretto o meno, in modo da controllare gli errori dell'Hamming.

4.4 Collegamenti e protocolli di accesso multiplo

4.4.1 Tipologie di collegamento

In una rete esistono 2 tipologie di collegamento:

1. **Punto a punto:** dove alle 2 estremità del collegamento si trovano ricevente e trasmettente.
2. **Broadcast:** dove possono esserci più nodi trasmittenti e riceventi connessi allo stesso canale broadcast condiviso. Il termine broadcast indica infatti che, quando un nodo trasmette una frame, il canale lo diffonde a tutti gli altri nodi che ne ricevono una copia.

Nel collegamento sorge un problema: chi comunica per primo e come? Servono dei modi per gestire la comunicazione.

4.4.2 Protocolli ad accesso multiplo

Queste tipologie di protocolli fissano le modalità con cui i nodi regolano le loro trasmissioni sul canale condiviso.

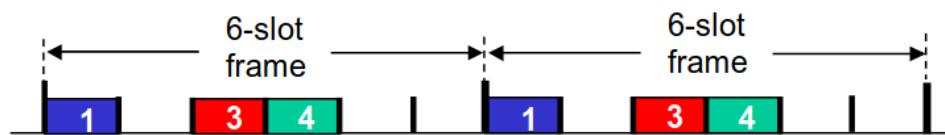


Esistono 3 tipologie di protocolli ad accesso multiplo:

1. **CDMA:** Code Division Multiple Access
2. **TDMA:** Time Division Multiple Access
3. **FDMA:** Frequency Division Multiple Access

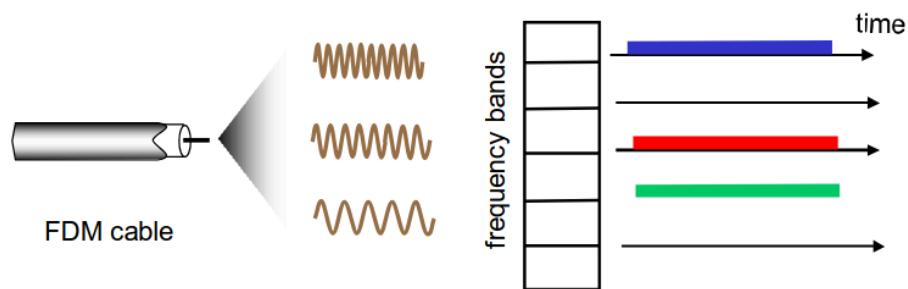
TDMA

TDM suddivide il tempo in intervalli, divide ciascuno di questi in N slot temporali, tanti quanti i nodi da gestire, e assegna a ogni nodo uno degli slot. Per esempio, ipotizziamo di avere 6 stazioni LAN: 1, 3 e 4 devono mandare pacchetti mentre gli altri sono in idle. Questo causa un rallentamento, perché degli slot di tempo non sono utilizzati. La massima capacità di uso del collegamento è di R/N con R bps e N numero di nodi.



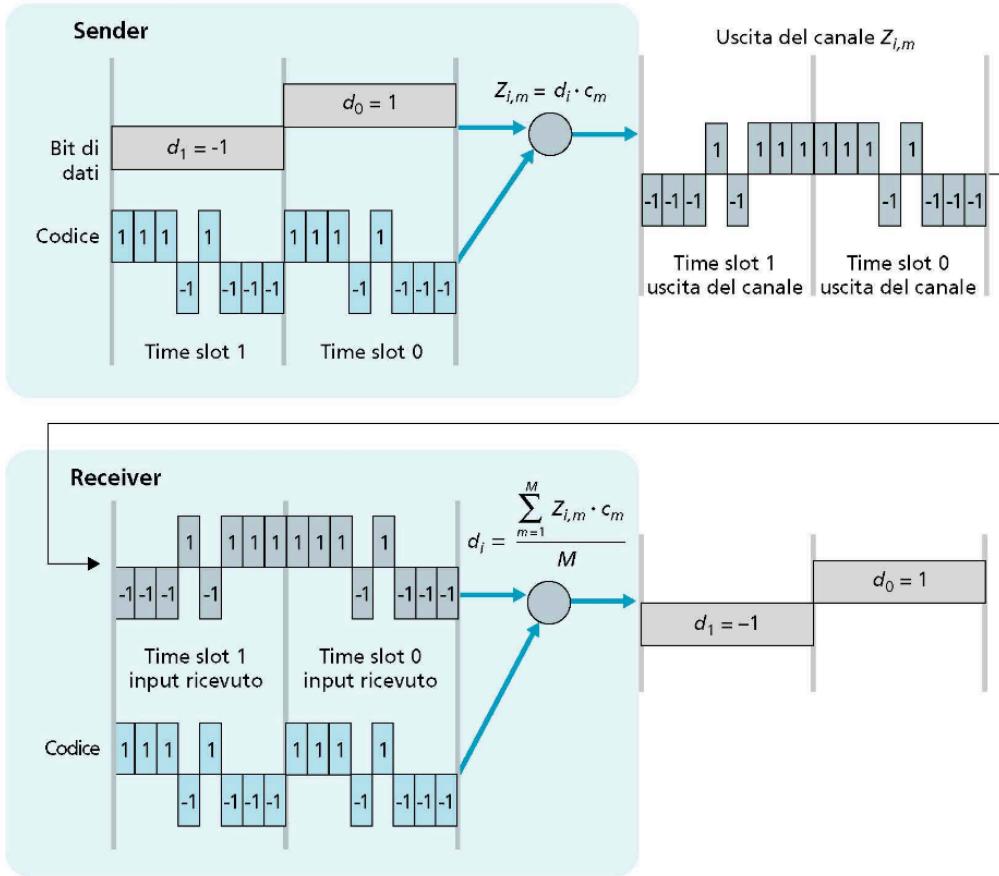
FDMA

FDM suddivide il canale in frequenze differenti (in termini di larghezza di banda) e assegna ciascuna frequenza a un nodo. I dispositivi riceventi distinguono quelli mittenti separando localmente i segnali in base alla frequenza. FDM evita le collisioni, ma anche in questo caso la larghezza di banda è limitata a R/N .



CDMA

CDMA assegna a ogni nodo un codice univoco per codificare i dati inviati. Le reti CDMA consentono a nodi differenti di trasmettere simultaneamente e ai rispettivi destinatari di ricevere correttamente i bit nonostante le interferenze derivanti dalle trasmissioni degli altri nodi.



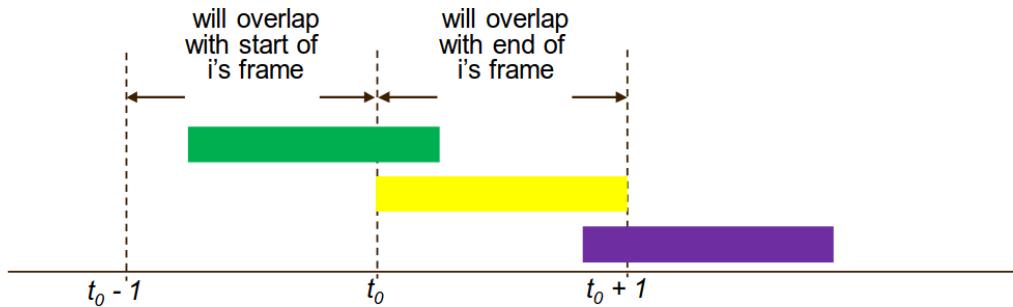
I codici devono formare una base linearmente indipendente, ovvero ogni elemento non può essere combinazioni lineari di altri. La larghezza di banda è di R/N anche in questo caso e nel caso più mittenti trasmettano in contemporanea il ricevente applica il filtro corrispondente al segnale che intende ricevere.

4.4.3 Protocolli ad accesso casuale

Se volessimo trasmettere sempre alla massima velocità consentita dal canale, cioè R bps, si utilizzano i **protocolli ad accesso casuale**. Quando si verifica una collisione, i nodi ritrasmettono più volte i loro frame finché non vengono ricevuti correttamente dalla destinazione. La ritrasmissione però, non è immediata, a il nodo attende per un periodo di tempo casuale, così che le differenti scelte di tempo diminuiscano la possibilità di avere collisioni.

Pure ALOHA

E' un protocollo semplice, senza sincronizzazione. Ogni nodo trasmette quando ha qualcosa da trasmettere, vi è infatti un'alta probabilità di collisione. Funziona quando le trasmissioni sono rare.



Slotted ALOHA

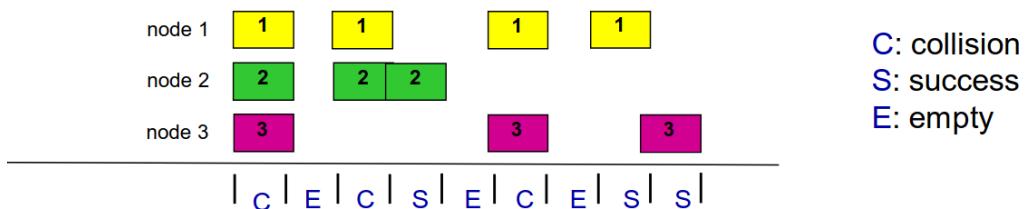
Quando un nodo ha una frame da spedire attende l'inizio dell slot successivo e poi trasmette l'intero frame. Se non si verifica una collisione il nodo può predisporre l'invio di un nuovo frame, altrimenti il nodo rileva la collisione prima del termine dello slot e ritrasmette con probabilità p il suo frame in uno degli slot successivi finché l'invio non ha successo.

Pro

- Ogni nodo può trasmettere alla massima velocità del canale.
- E' decentralizzato, solamente gli slot tra i nodi devono essere sincronizzati.
- E' semplice

Contro

- Collisioni con perdite di slot.
- Slot vuoti.
- I nodi non potrebbero accorgersi della collisioni in meno tempo rispetto alla trasmissione.
- Sincronizzazione di clock.

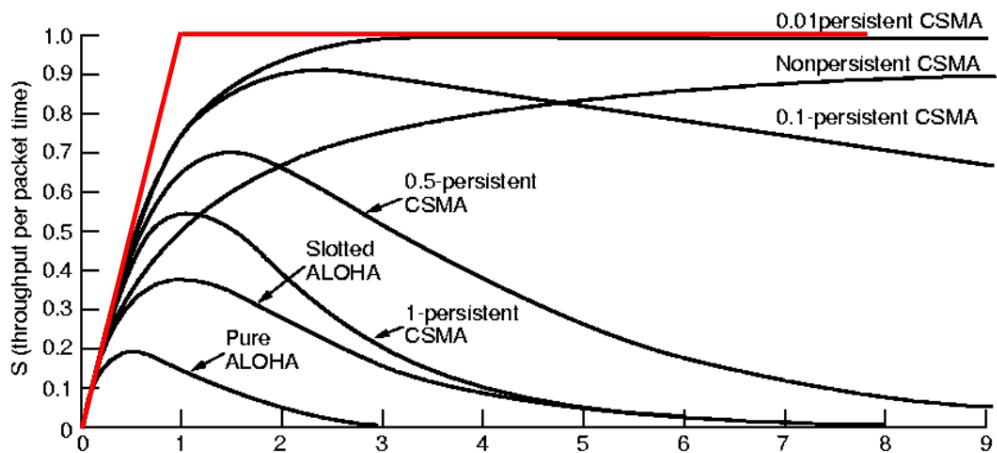


CSMA: accesso multiplo con rilevamento della portante

Il problema principale dei protocolli ALOHA è che i nodi decidono di trasmettere in modo indipendente dagli altri nodi collegati al canale broadcast. Il protocollo CSMA impone delle regole:

1. Ascoltare prima di comunicare.
2. Nessuna comunicazione contemporaneamente.

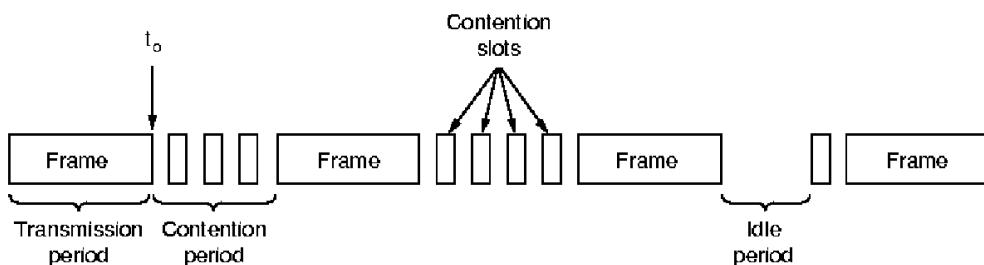
Quindi alla base di questi protocolli c'è il **rilevamento della portante** e il **rilevamento della collisione**. Un problema che nasce da questo protocollo è, maggiore è il ritardo di propagazione del segnale da un estremo all'altro del canale broadcast, maggiore è la probabilità che il nodo non si accorga che è già cominciata la trasmissione da parte di un altro nodo, generando collisione.



Esistono diverse versioni di CSMA:

- **CSMA-1-persistent**: una stazione trasmette una nuova frame quando non rileva la portante
- **CSMA-persistent**: una stazione trasmette, dopo l'assenza della portante, con una probabilità p .
- **CSMA-non-persistent**: la stazione aspetta un tempo random prima di controllare il canale.
- **CSMA/CD**: la stazione, dopo che ha rilevato una comunicazione, termina la sua. Ma qual è il **tempo di backoff ideale** da scegliere? Se l'intervallo è grande e il numero di nodi che collidono è piccolo allora i nodi dovranno attendere per tanto tempo, ma esiste anche il problema al contrario. Si utilizza l'algoritmo **binary exponential backoff**, quando il trasmettitore riscontra l' n -esima collisione durante la trasmissione di un frame, stabilisce un valore K casuale nell'insieme $\{0, 1, 2, \dots, 2^{n-1}\}$. Quindi più alto è il numero di collisioni più grande è l'intervallo da cui K viene estratto.

Ethernet utilizza questo algoritmo, quando riceve un pacchetto lo incapsula in una frame, dopodiché controlla il canale per verificare che sia vuoto così da poter iniziare la trasmissione.



4.4.4 Protocolli a rotazione

Polling

Uno dei nodi viene designato come **principale** e interella a turno gli altri. Il nodo principale invia un messaggio al nodo 1, comunicandogli che può trasmettere fino a un dato numero massimo di frame, così avanti per tutti i nodi. Il nodo principale determina che un nodo ha terminato di inviare i propri frame, osservando la mancanza di segnale sul canale.

Questo protocollo elimina le collisioni e gli slot vuoti che si riscontrano nei protocolli ad accesso casuale e ha quindi un'efficienza più elevata. Gli svantaggi però, sono diversi:

- Il primo è l'introduzione del ritardo di polling, è attivo un solo nodo ma trasmetterà a meno di R bps, poiché una parte della banda la prende il nodo principale.
- Il secondo è che se il nodo principale si guasta, l'intero canale diventa inattivo.

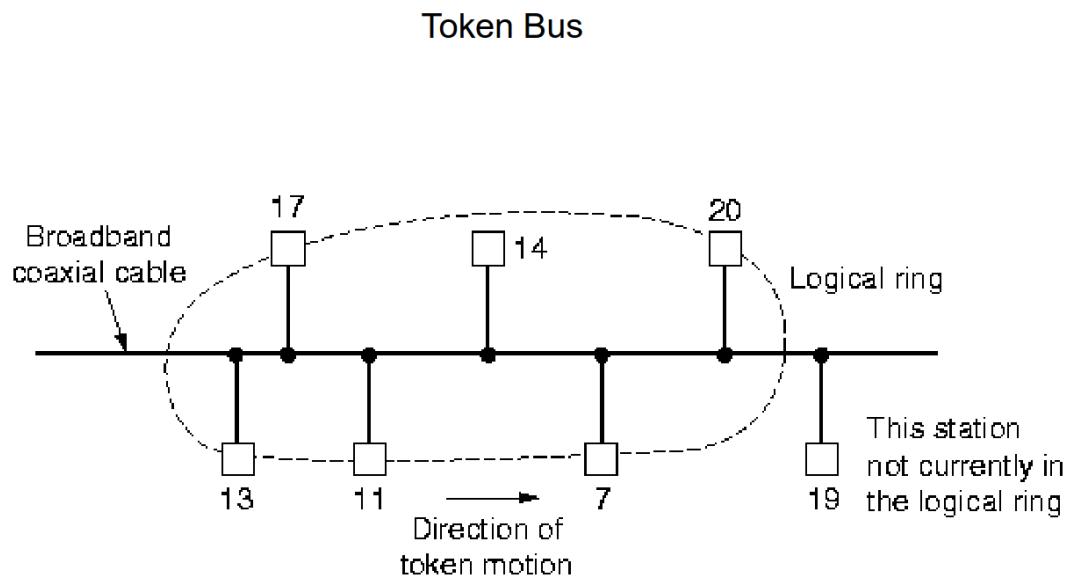
Bluetooth si basa su questo protocollo.

Token-Passing

In questo caso non esiste un nodo principale, ma un frame, un messaggio di controllo detto **token** che circola tra i nodi seguendo un ordine prefissato. Si forma così una chain, il nodo 1 spedisce al nodo 2 che spedisce al nodo k fino ad N (non per forza in ordine numerico crescente). Se il nodo che riceve il token non ha pacchetti da inviare, lo inoltra immediatamente al successivo, altrimenti procede a inviare.

Questo protocollo, per via della sua decentralizzazione, è altamente efficiente però non è privo di problemi:

- Il guasto di un nodo può mettere fuori servizio l'intero canale.
- Se un nodo non riesce a inoltrare un token, occorre invocare procedure di recupero



4.5 Cablaggio

4.5.1 Tipologie di mezzi fisiche con standard base-x

Ethernet compare in molte forme diverse e con varie denominazioni, in base alla velocità del segnale:

- **10BASE-T**: 10 Mbps
- **10BASE-2**: 100 Mbps
- **1000BASE-T**: 1 Gbps
- **10GBASE-T**: 40 Gbps

Il segnale si degrada con la distanza, quindi anche un cavo pessimo ma di lunghezza molto piccola può condurre bene un segnale. Negli standard, se si supera la lunghezza massima indicata, la velocità non è garantita, ma non è detto che il cavo non funzioni. È importante anche il numero di nodi per segmento, perché dopo un certo numero il segnale può arrivare corrotto.

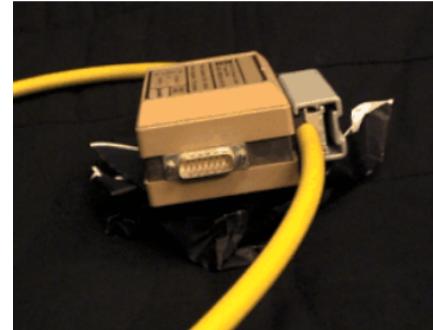
4.5.2 10BASE-T

Name	Cable	Max. seg.	Nodes/seg.	Advantages
10Base5	Thick coax	500 m	100	Original cable; now obsolete
10Base2	Thin coax	185 m	30	No hub needed
10Base-T	Twisted pair	100 m	1024	Cheapest system
10Base-F	Fiber optics	2000 m	1024	Best between buildings

Lo standard a 10 Mbit prevede 3 tipi di cablaggio.

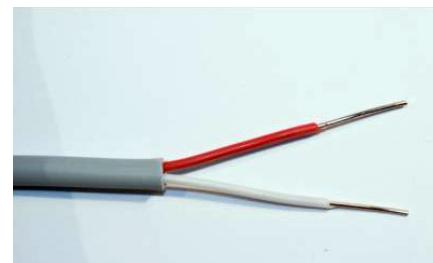
Cavo coassiale

Era immune alle interferenze elettromagnetiche. Una discontinuità può far rimbalzare una parte del segnale all'indietro, quindi il cavo non poteva essere tagliato di netto. Si usavano delle spine a vampiro che tagliavano la guaina esterna fissandosi tramite un perno all'anima del cavo. Alla fine del cavo dovevano esserci dei terminatori BNC per evitare il rimbalzo. Un segmento aveva dimensione massima 500 m, si poteva usare però un ripetitore per un massimo di 4 volte per far arrivare il segnale fino a 2.5 km.



Doppino di Rame

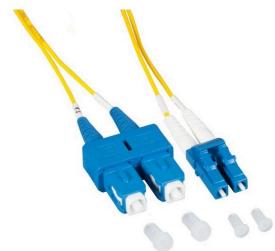
Funzionava per le connessioni a bassa velocità, sono intrecciati per non formare una spira e farlo diventare un'antenna che crea interferenze. Si usava insieme ad un concentratore (hub) a cui ogni dispositivo si collegava in modo punto a punto, in questo modo, se un collegamento smetteva di funzionare gli altri non erano coinvolti.



Fibra

E' immune a qualsiasi tipo di interferenza elettromagnetica, supportando collegamenti più lunghi senza riflessioni dei segnali.

Le componenti ottiche sono più difficili da gestire.



4.5.3 Fast Ethernet

Per passare allo standard a 100 Mbps si è decuplicata la velocità dividendo per 10 i tempi. Non si è utilizzato più il cavo coassiale perché troppo inaffidabile, si può utilizzare invece:

- Il 100Base-T4 usa 4 coppie di cui una in ciascuna direzione per inviare segnali e le altre 2 per modificare la direzione. Non è full-duplex. Usa la codifica 8B6T.
- Il 100Base-TX usa una coppia per ogni direzione permettendo di avere la stessa velocità in ogni direzione (full-duplex). Usa la codifica 4B5B.
- Il 100Base-FX usa 2 cavi di fibra, uno in ogni direzione. Non ha collisioni

4.5.4 Categorie di cavi

Category	Data Rate	Signal Frequency	Standard
Cat5	100 Mbps	100 MHz	TIA/EIA
Cat5e	100 Mbps / 1 Gbps	100 MHz	TIA/EIA-568-B
Cat6	1Gbps / 10 Gbps	250 MHz	TIA/EIA-568-B
Cat6a	1Gbps / 10 Gbps	500 MHz	ANSI/TIA/EIA-568-B.2-10

La differenza tra le categorie si basa su quanto è fitto l'intreccio. Salendo di velocità serve un cavo con prestazioni sempre migliori (e distanze sempre minori).

I cavi Cat5 e Cat5e non sono schermati e hanno come lunghezza massima di segmento 100 metri. Il Cat6 prevede una velocità di 10 Gbps con un massimo di 55 m, il Cat6a con un massimo di 100 m.

Nei cavi Cat6 vi era un'elica di plastica che permetteva di intrecciare le coppie tra loro e fungeva da isolante.

I cavi Cat7 hanno una schermatura STP o FTP e prevedono una velocità di 10 Gbps con una larghezza di banda di 600 MHz a una distanza massima di 100 metri.

I cavi Cat8 prevedono una velocità di 40 Gbps con una larghezza di banda di 2000 MHz a una distanza massima di 30 m.

Tipi di cavi a coppia intrecciata:

- UTP (Unshielded Twisted Pair)
- STP (Shielded Twisted Pair)
- FTP (Folded Twisted Pair)



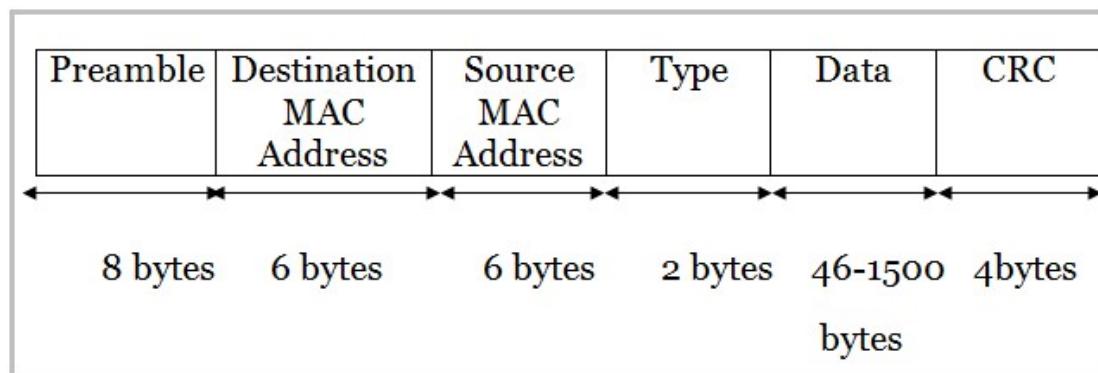
4.6 Ethernet

4.6.1 Struttura

Ethernet è una tecnologia di rete a livello collegamento dati che utilizza un formato di trama (frame) ben definito per la trasmissione di dati. La struttura base del frame Ethernet include i seguenti campi:

1. **Preamble (7 byte)**: una sequenza di bit (10101010) che serve per la sincronizzazione, segnalando ai dispositivi che sta per iniziare una trasmissione.
2. **Start Frame Delimiter - SFD (1 byte)**: un byte che segna l'inizio effettivo del frame dati; ha valore 10101011.
3. **Indirizzo MAC di destinazione (6 byte)**: identifica il destinatario del pacchetto.
4. **Indirizzo MAC di origine (6 byte)**: identifica il mittente.
5. **Tipo (2 byte)**: specifica il tipo di protocollo del livello superiore (es. IPv4 o IPv6).
6. **Dati e Padding (46-1500 byte)**: contiene i dati reali. Se i dati sono inferiori a 46 byte, viene aggiunto padding per raggiungere la lunghezza minima.
7. **CRC (4 byte)**: Cyclic Redundancy Check, per rilevare eventuali errori nella trasmissione.

Il frame Ethernet ha quindi una dimensione minima di 64 byte (compresi preamble e SFD) e una massima di 1518 byte.



Ethernet ha subito diverse evoluzioni, differenziandosi principalmente per la velocità di trasmissione e il tipo di mezzo fisico utilizzato:

- **Ethernet classica (10 Mbps)**: utilizza un mezzo condiviso e l'accesso al canale è regolato dal protocollo CSMA/CD.
- **Fast Ethernet (100 Mbps)**: miglioramento della velocità mantenendo la compatibilità con lo standard precedente.

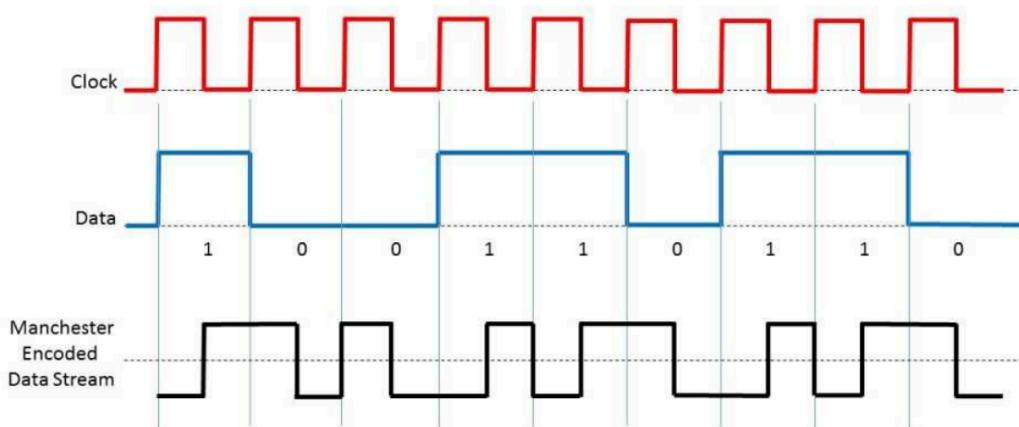
- **Gigabit Ethernet (1 Gbps)**: funziona principalmente in modalità full-duplex, con switching piuttosto che CSMA/CD.
- **10 Gigabit Ethernet e superiori**: progettate per backbone e data center, utilizzano collegamenti in fibra ottica.

4.6.2 Codifica di Manchester

La **codifica Manchester** è uno schema di codifica del segnale utilizzato nei primi standard Ethernet (fino a fast ethernet), progettato per migliorare la sincronizzazione tra trasmettitore e ricevitore. L'idea chiave è quella di combinare il segnale di **clock** con il segnale **dati**, consentendo al ricevitore di recuperare facilmente il timing senza una sorgente di clock separata.

Caratteristiche della codifica Manchester

- Ogni bit è rappresentato da **due intervalli temporali uguali**:
 - **Bit 0**: rappresentato da una **transizione da alto a basso** nella metà del bit.
 - **Bit 1**: rappresentato da una **transizione da basso ad alto**.
- Questo schema assicura che vi sia **almeno una transizione per ogni bit**, facilitando la sincronizzazione.
- Poiché ogni bit implica una transizione, la codifica Manchester raddoppia la frequenza del segnale rispetto al flusso dati (es. 10 Mbps → 20 MHz), rendendolo più **sensibile al rumore** e più **costoso in termini di larghezza di banda**.



Vantaggi

- Facilita la sincronizzazione temporale tra mittente e destinatario.
- Ha proprietà autodiagnostiche: un'assenza di transizioni segnala un errore.

Svantaggi

- Doppio consumo di larghezza di banda rispetto al flusso dati reale.
- Meno efficiente rispetto a schemi più recenti, come la codifica 4B/5B o 8B/10B usata in Fast/Gigabit Ethernet.

4.6.3 Gigabit Ethernet

Utilizzando cavi cat5e gli step per passare da fast ethernet a gigabit ethernet:

- Rimozione della codifica 4B5B, per passare da 100 a 125 Mpbs
- Uso di tutte le 4 coppie contemporaneamente, per passare da 125 a 500 Mpbs
- Trasmissione full-duplex, per passare a 500 Mpbs full duplex
- Uso di 5 livelli invece di 3 (PAM5), si può passare fino a 2 Gbps full duplex
- Introduzione di una tecnica FEC (Forward Error Correction) per recuperare 1 bit errato ogni 4, per passare da 2 a 1 Gbps.

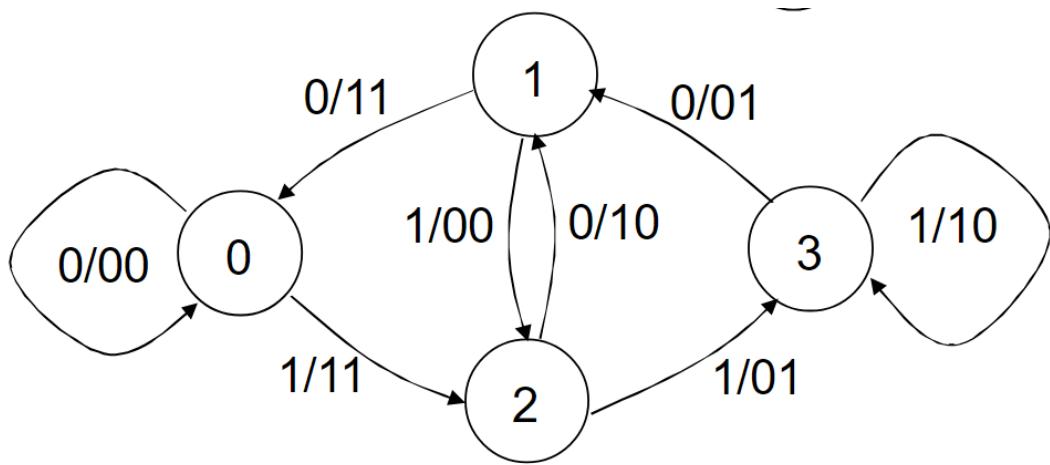
Si può utilizzare la fibra ottica monomodale, che viaggia lungo un unico asse ed è migliore di quella multimodale.

4.6.4 Codifica di Viterbi

La codifica di Viterbi è un termine più colloquiale che indica il **codice convoluzionale**, cioè un **algoritmo di codifica/decodifica** per la correzione degli errori.

In particolare si basa su una **macchina a stati finiti** dove ogni stato ha m bit in ingresso e k bit in uscita.

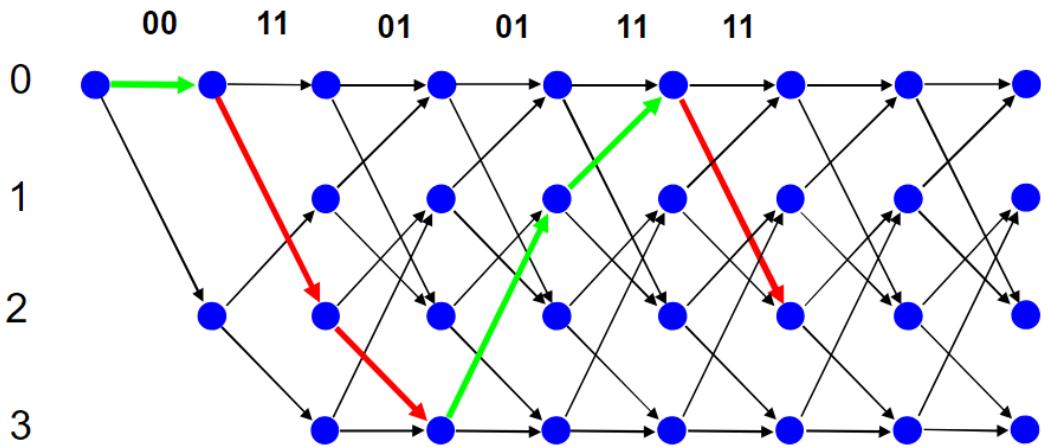
Ipotizziamo un esempio, con la macchina a stati finiti



Prendiamo come messaggio da codificare **011001**, grazie alla macchina risolviamo la codifica: in questo caso la macchina ha $m = 1$, $k = 2$ per ogni stato, con questa sintassi m/k (lo stato 0 ha $m = 0$, $k = 00$). Codificando il messaggio abbiamo **00 11 01 01 11 11**.

Decodifica senza errori

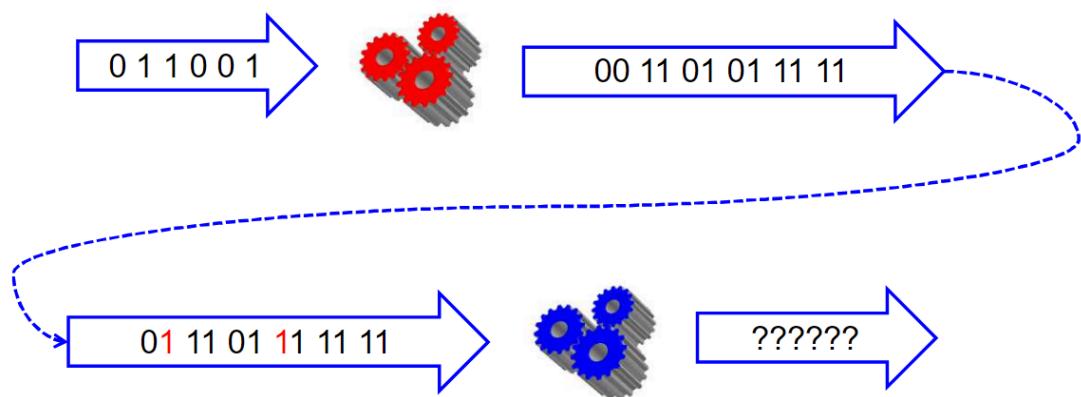
Per risalire al messaggio originale, avendo il messaggio codificato dobbiamo creare uno pseudo-grafico dove nell'asse verticale abbiamo gli stati, nell'asse orizzontale abbiamo le possibili uscite di ogni stato per bits decodificati.



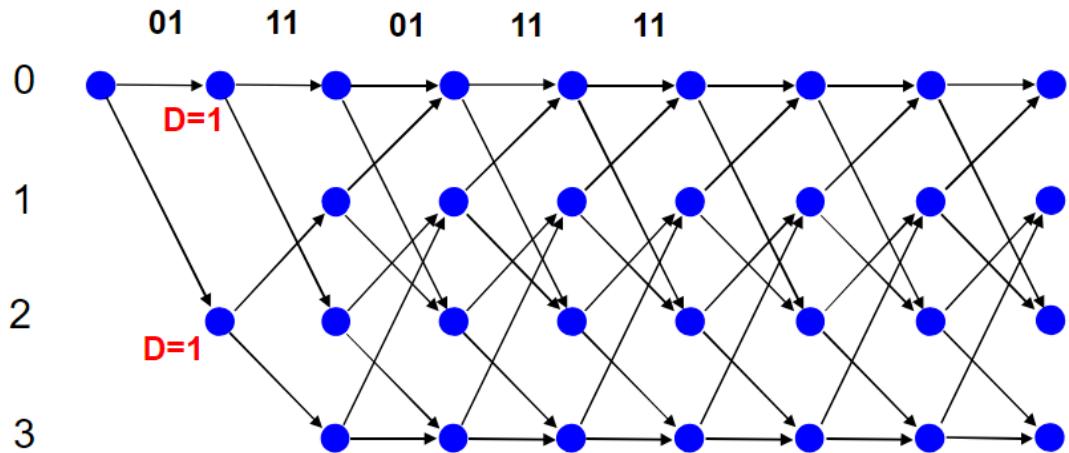
Quindi, ricomponendo il messaggio (da sinistra verso destra) abbiamo che il risultato è giusto, poiché ogni parola del messaggio corrisponde a una k di uno stato.

Decodifica con errori

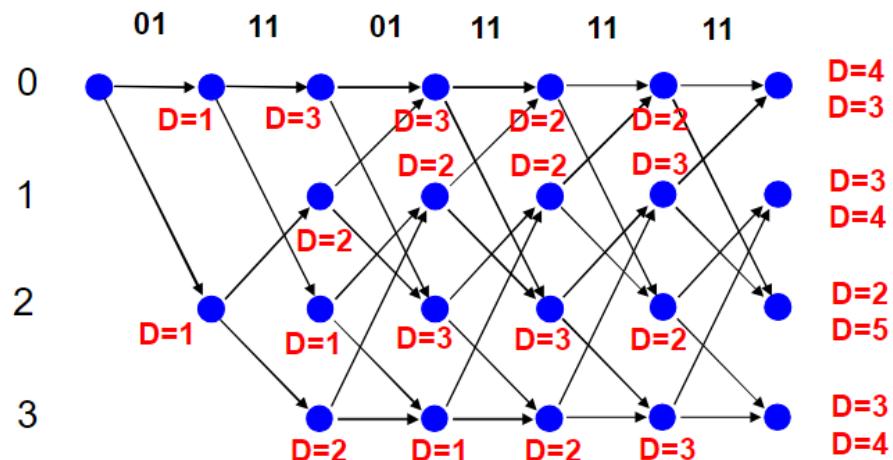
Se però durante la trasmissione uno dei bit viene alterato, la codifica di Viterbi oltre a notare l'errore può correggerlo.



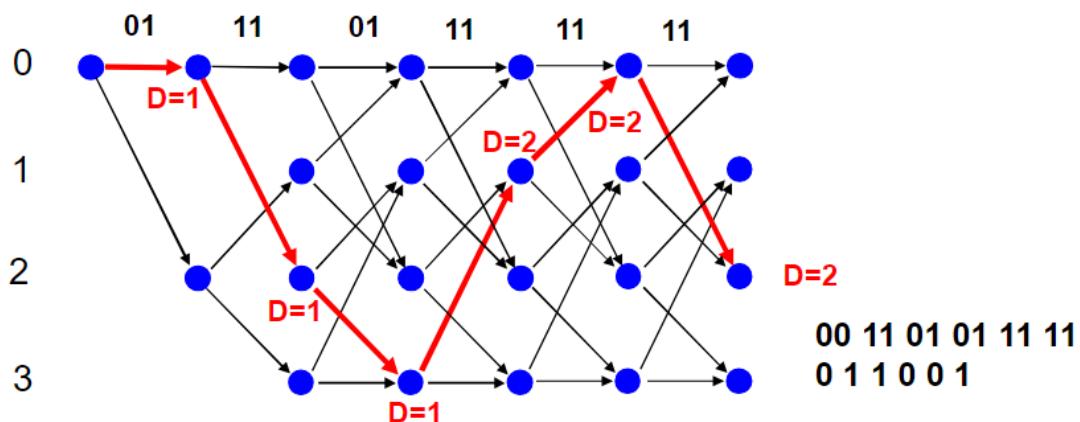
In particolare, con il messaggio `0 1 11 01 1 1111` con due errori, dobbiamo analizzare lo stesso pseudografico:



Come primo passaggio, notiamo che c'è un errore. In quanto dalla macchina a stati finiti, non esiste dallo stato 0 un valore $k = 01$, però sappiamo che la distanza di hamming tra 01 e i valori di uscita dello stato 0 dista 1. Quindi possiamo provare a seguire quei percorsi del grafo.



Dopo aver seguito l'intero percorso, possiamo **fare la somma di ogni singola distanza di Hamming e scegliere la minore**: in questo caso $d = 2$.



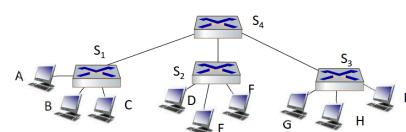
Una volta trovata la distanza di hamming, **abbiamo trovato il percorso** nella macchina a stati finiti e possiamo ricavare il messaggio originale seguendo l'uscita di ogni singolo stato e così decodificare il messaggio.

4.7 Hub, Bridge e Switch

4.7.1 Hub

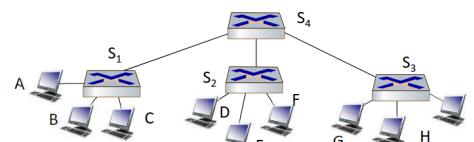
L'hub non trasforma i bit in dati, ma si limita a ritrasmettere il segnale fisico. Infatti, una rete di soli Hub può creare collisione elevata.

Lo switch riesce a gestire più traffico dell'hub a parità di velocità di connessione, che nel caso dell'hub viene condivisa tra tutte le connessioni.



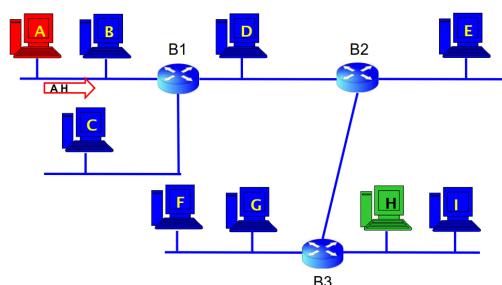
4.7.2 Bridge

Il Bridge permette la conversione tra protocolli LAN differenti. Ad esempio, può trasformare le frame Ethernet in frame Wifi (che hanno 4 indirizzi e un payload più lungo). Lavora a livello DLL e converte il segnale in bit.

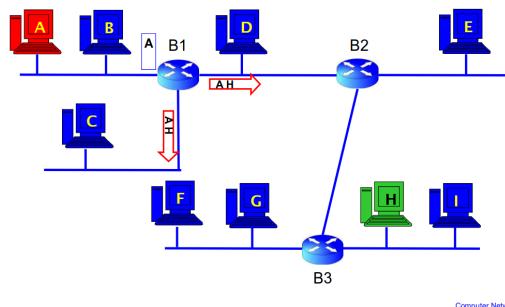


4.7.3 Bridge in una rete

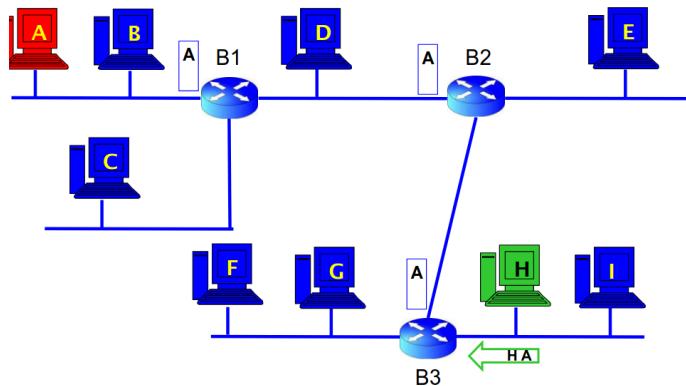
Ipotizziamo che A voglia mandare un messaggio ad H, invia quindi il suo messaggio al bridge B1. B1 però, non sa dove si trova la macchina di destinazione, quindi invia il messaggio a tutte le uscite.



Tutti i dispositivi ricevono il messaggio, ma solamente H non lo scarterà. In questo modo però, ogni dispositivo conosce la posizione di A nella rete.



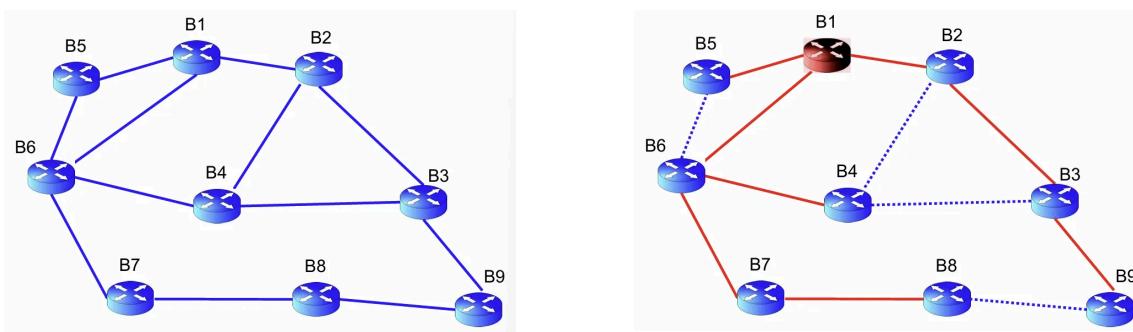
Al ritorno però, ogni hub conosce il percorso per A, quindi inoltra solo a quello.



4.7.4 STP

STP è un protocollo che permette di ottenere un albero a partire da un grafo, dividendo ogni nodo in livelli.

Questo può essere utile quando dobbiamo utilizzare flooding, poiché in questo modo non si crea il problema dei cicli all'interno dei grafi.



4.7.5 Switch

Il ruolo dello switch è ricevere i frame in ingresso e inoltrarli sui collegamenti in uscita. Lo switch è un componente della rete **trasparente**, ovvero ogni nodo indirizza una frame ad un altro nodo, che si occuperà di proseguire l'inoltro ad un altro ipotetico nodo.

La velocità con cui i frame arrivano ad una delle interfacce di uscita degli switch può temporaneamente eccedere la capacità del collegamento di quell'interfaccia, quindi le interfacce di

uscita sono dotate di buffer.

Gli switch coordinano le proprie trasmissioni e non inoltrano mai più di un frame sulla stessa interfaccia, inoltre quelli moderni sono full-duplex, quindi possono scambiare frame con un nodo in entrambe le direzioni senza interferire con quest'ultimo. Quindi, in una LAN Ethernet basata su switch non ci sono collisioni, motivo per cui non c'è bisogno di un protocollo MAC.

Inoltro e filtraggio

Il filtraggio è la funzionalità dello switch che **determina se un frame debba essere inoltrato o scartato**. Consiste nell'individuazione dell'interfaccia verso cui il frame deve essere diretto e l'invio a quell'interfaccia. Le operazioni di filtraggio e inoltro sono eseguite mediante una **tavella di commutazione** (switch table), composta da voci che contengono l'indirizzo MAC del nodo, l'interfaccia dello switch che conduce al nodo, e il momento in cui la voce è stata inserita nella tabella.

Indirizzo	Interfaccia	Tempo
62-FE-F7-11-89-A3	1	9:32
7C-BA-B2-B4-91-10	3	9:36
....

Supponiamo che un frame con indirizzo di destinazione **DD:DD:DD:DD:DD:DD** giunga allo switch sull'interfaccia x. Lo switch cerca nella sua tabella l'indirizzo, e vi sono 3 possibili casi:

- Non vi è una voce per l'indirizzo di destinazione → lo switch invia il frame su tutte le interfaccie eccetto quella di sorgente in broadcast.
- Vi è una voce che associa l'indirizzo di destinazione all'interfaccia x → non occorre inoltrare il frame poiché proviene da un segmento di rete che contiene la scheda di rete destinataria e lo switch esegue il filtraggio scartando il frame.
- Vi è una voce che associa l'indirizzo di destinazione ad un'interfaccia y diversa da x → il frame deve essere inoltrato al segmento LAN collegato all'interfaccia y, quindi lo switch pone il frame nel buffer dell'interfaccia y.

Autoapprendimento

Gli switch costruiscono **automaticamente le proprie tabelle senza l'intervento di un operatore** o un **protocollo di configurazione**. La capacità di autoapprendimento è ottenuta nel seguente modo:

1. La tabella è inizialmente vuota.
2. Di ogni frame che riceve, lo switch inserisce nella sua tabella l'indirizzo MAC del campo indirizzo sorgente del frame, l'interfaccia da cui arriva il frame e il momento di arrivo. La tabella sarà completa quando tutti i nodi della LAN avranno inviato un frame.
3. Dopo un periodo di tempo detto **aging time**, durante il quale lo switch non riceve frame da un dato indirizzo sorgente, lo cancella dalla tabella. Quindi se un calcolatore viene sostituito, l'indirizzo MAC vecchio viene eliminato automaticamente dalla tabella.

In conclusione gli switch sono dispositivi **plug-and-play**: non richiedono interventi dell'amministratore di rete o dell'utente, basta collegare i segmenti di LAN alle sue interfacce.

Proprietà della commutazione a livello di collegamento

- **Eliminazione delle collisioni:** In una LAN costituita da switch e senza hub **non vi è spreco di banda a causa delle collisioni in quanto gli switch non trasmettono più di un frame su ogni segmento di**

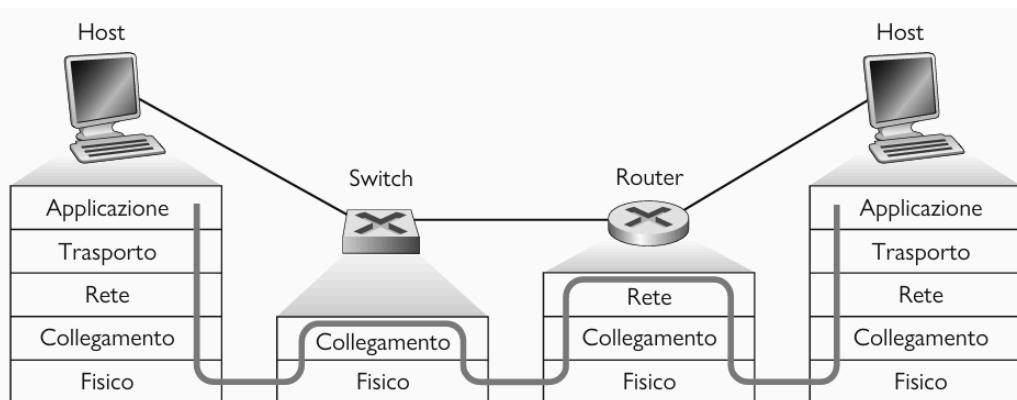
LAN in un certo istante. Gli switch forniscono quindi un miglioramento delle prestazioni su LAN con collegamenti broadcast. Il massimo throughput aggregato di uno switch è la somma dei tassi trasmissivi di tutte le sue interfacce.

- **Collegamenti eterogenei:** Dato che lo switch isola un collegamento da un altro, i collegamenti nella LAN possono funzionare a velocità diverse e usare mezzi trasmissivi diversi. Uno switch è quindi ideale per combinare dispositivi già installati con altri nuovi.
- **Gestione:** Gli switch facilitano la gestione di rete in vari modi. Se una scheda di rete ha un malfunzionamento e manda continuamente frame Ethernet (jabbering adapter), uno switch può individuare il problema e disconnettere la scheda di rete non funzionante. Analogamente, il taglio di un cavo disconnette solo il nodo che lo stava usando per collegarsi allo switch (quando veniva usato il cavo coassiale si doveva trovare il cavo guasto che aveva interrotto la rete). Gli switch raccolgono anche statistiche sull'uso di banda, tasso di collisioni e tipi di traffico che possono essere utili per rilevare e correggere dei problemi.

Intercettazioni in una LAN commutata: avvelenamento dello switch

Quando un nodo viene connesso a uno switch, riceve solo i frame che gli sono stati esplicitamente inviati, quindi in una LAN commutata è più difficile intercettare i frame. Tuttavia, poiché lo switch invia in broadcast i frame con indirizzo di destinazione non presente nella sua tabella, lo sniffer può intercettare questi frame. L'attacco detto avvelenamento dello switch consiste nell'invio di una grande quantità di pacchetti allo switch con indirizzi MAC fasulli diversi tra loro per riempire le tabelle degli switch con voci false, senza lasciare spazio agli indirizzi dei nodi legittimi. Ciò obbliga lo switch a inviare in broadcast la maggior parte dei frame, che può essere così intercettata dallo sniffer.

Switch e router a confronto



I **router** sono commutatori di pacchetto **di livello 3**, gli **switch** sono commutatori di pacchetto **di livello 2**. La topologia di una rete di switch **dove essere un albero** per evitare i cicli con i frame broadcast. Inoltre, reti di switch molto estese richiederebbero delle grandi tabelle ARP nei nodi e nei router generando un considerevole traffico ARP. Infine, gli switch non offrono alcuna protezione contro le tempeste di broadcast: se un host iniziasse a trasmettere un flusso ininterrotto di pacchetti broadcast gli switch li inoltrerebbero, causando il collasso della rete.

Nel caso dei router, **essendo gli indirizzi a livello di rete gerarchici**, i pacchetti non presentano **cicli attraverso i router anche nel caso in cui la rete presenti percorsi ridondanti**. Inoltre, IP usa un campo di intestazione del datagramma per limitare la percorrenza di cicli. Pertanto, i pacchetti non sono vincolati ad un albero di copertura e possono usare il percorso migliore tra sorgente e destinazione.

I router proteggono dalle tempeste di broadcast a livello 2 **ma non sono plug-and-play**, quindi il loro indirizzo IP e quello degli host a essi collegati va configurato. Presentano inoltre un temo di

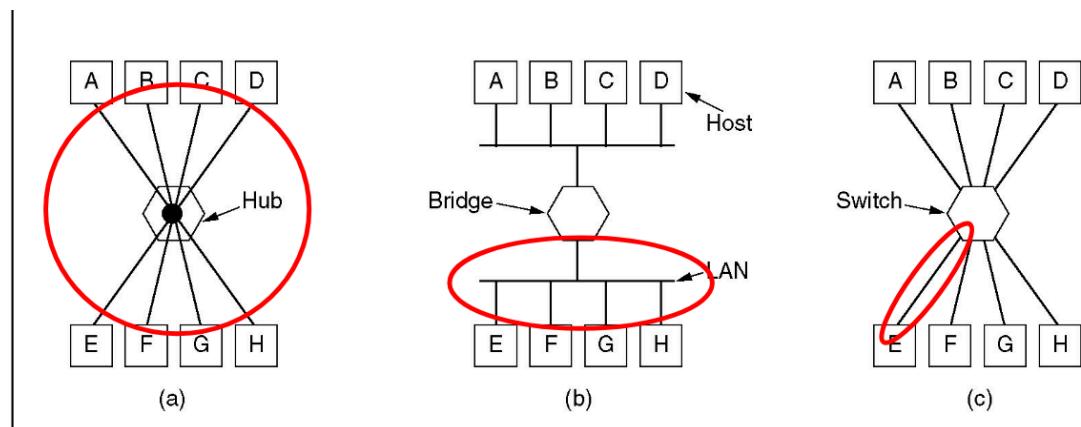
elaborazione per pacchetto più lungo di quello degli switch poichè devono eseguire l'elaborazione fino ai campi del livello 3.

Per le reti formate da alcune centinaia di host e da pochi segmenti sono sufficienti gli switch, le reti più grandi, invece, includono anche dei router.

Con gli switch non si può fare sniffing, ma si può configurare una porta in cui far passare tutto il traffico per motivi di sicurezza.

4.7.6 Hub vs Switch vs Bridge

Caratteristica	Hub	Bridge	Switch
Livello OSI	1 (Fisico)	2 (Data Link)	2 (Data Link)
Funzione base	Ripetitore multiporta elettrico	Filtraggio/inoltro tra 2 o più segmenti	Filtraggio/inoltro tra molti segmenti
Inoltro	Diffonde ogni segnale su tutte le porte	Inoltra solo tra i segmenti necessari	Inoltra solo alla porta di destinazione
Tabella MAC	✗ Non mantiene tabelle	✓ Mantiene tabella MAC (limitata)	✓ Mantiene tabella MAC (per ogni porta)
Collision domain	Unico (tutte le porte condividono)	Unico per segmento, isolati tra bridge	Uno per porta (ogni porta isola le collisioni)
Broadcast domain	Unico (tutte le porte condividono)	Unico (non segmenta i broadcast)	Unico (ma può essere segmentato con VLAN)
Full-duplex	✗ Solo half-duplex	⚠ Dipende dal modello (spesso half-duplex)	✓ Su tutte le porte
Larghezza di banda	Condivisa su tutte le porte	Condivisa per ogni coppia di segmenti	Dedicata per porta
Supporto VLAN	✗	✗	✓
Costo e complessità	Basso	Medio	Alto



4.8 VLAN

4.8.1 Caratteristiche

Le VLAN sono LAN separate logicamente, sebbene costruite nella stessa struttura fisica. I pacchetti broadcast in questo caso, sono confinati tra VLAN, ma per comunicare tra VLAN è necessario utilizzare

il routing del livello 3.

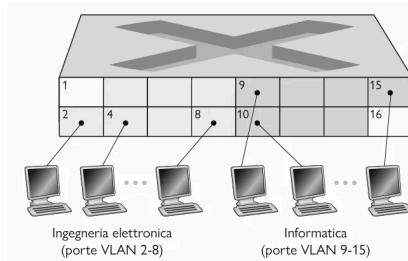
Una configurazione senza VLAN presenterebbe degli inconvenienti, quali:

- **Mancanza di isolamento del traffico**
- **Uso inefficiente degli switch**
- **Gestione degli utenti**

Queste difficoltà possono essere superate con una VLAN, ovvero una virtual local area network.

4.8.2 Switch nelle VLAN

Grazie alle VLAN possiamo definire più reti locali virtuali su una singola infrastruttura fisica di rete locale. Gli host all'interno di una VLAN comunicano tra loro come se fossero tutti connessi allo switch. In una VLAN basata su porte, le porte dello switch sono divise dal gestore di rete in gruppi, ognuno dei quali costituisce una VLAN le cui porte formano un dominio broadcast (il traffico broadcast proveniente da una porta raggiunge solo altre porte del gruppo).



Lo switch viene diviso logicamente (a livello software) in più parti, una per ogni VLAN. In questo modo i frame delle 2 VLAN sono isolati tra loro, basta un unico switch e se un utente si aggiunge ad un gruppo basta associare una nuova porta al gruppo.

Il gestore dichiara che una porta appartiene ad una LAN utilizzando un software per la gestione dello switch. Lo switch mantiene una tabella che associa porte e VLAN e l'hardware si limita a consegnare frame tra porte appartenenti alla stessa VLAN.

Come si trasmette il traffico tra gruppi diversi? Si potrebbe connettere una porta dello switch a un router esterno e configurarla in modo che appartenga ad entrambe le VLAN, come se i gruppi avessero switch separati connessi da un router. Tuttavia i vendori di switch semplificano la configurazione fornendo un dispositivo che contiene sia uno switch sia un router in modo che non sia necessario un router esterno.

4.8.3 Standard IEEE 802.1Q

Lo standard **IEEE 802.1Q** definisce il "VLAN tagging" per le reti Ethernet, cioè il modo in cui un singolo dominio di broadcast può essere virtualmente suddiviso in più LAN logiche (VLAN) senza modificare il cablaggio fisico.

Struttura del VLAN tag

Quando una porta è configurata in modalità *trunk* lo switch inserisce fra l'intestazione Ethernet e quella di livello 3 un campo di 4 byte:

