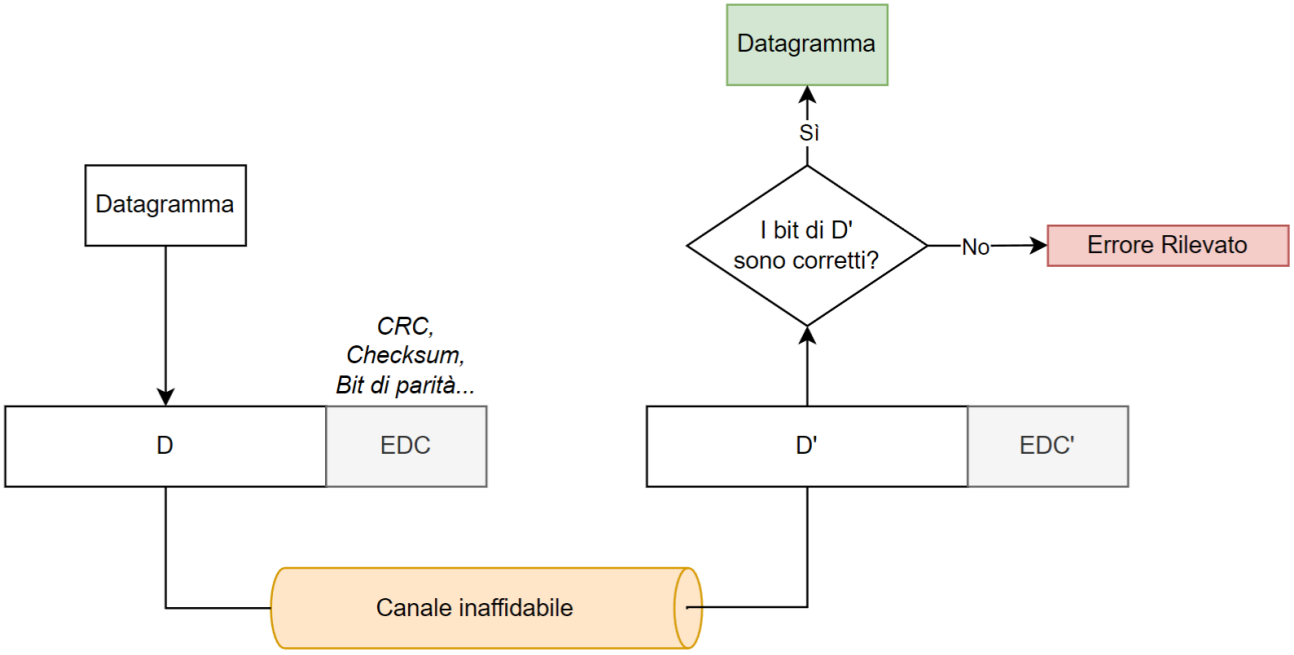
12_5_maggio

ERROR DETECTION

Ridondanza: elemento di base per l'error detection
Immaginiamo di voler trasmettere dei dati anagrafici all'interno di un generico pacchetto. Aggiungere allo stesso pacchetto un codice fiscale, mi fornisce delle informazioni si ridondanti, ma che mi permettono di individuare possibili errori di trasmissione successivamente alla comunicazione.

Error Detection

Nella realtà i pacchetti trasmetti contengono una sequenza di bit D (i dati) e un EDC (error detection and correction bits). L'EDC può essere una qualsiasi tecnica di controllo: CRC, Checksum, bit di parità ecc...
Una volta che il datagramma con EDC passa per il canale inaffidabile al pacchetto viene applicata la funzione inversa a quella di partenza per risalire ai dati originali e capire se ci sono stati degli errori



ERROR CORRECTION

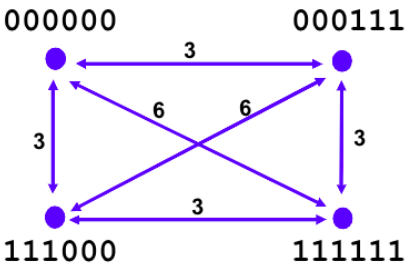
Distanza Di Hamming
Misura la distanza tra 2 code word:
1000110
1100110
distanza = 1

Vocabolario
Definiamo vocabolario un insieme di codeword. Un vocabolario si dirà completo se, con

n bit, avrà 2^n codeword

Consider the following vocabulary, with only 4 codewords:

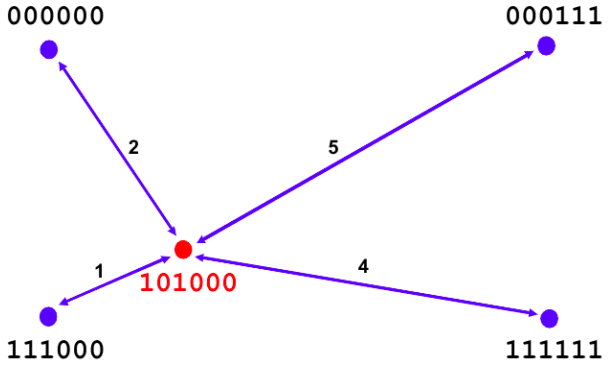
000000
000111
111000
111111



The distance in this vocabulary is 3.

In un vocabolario la distanza è misurata come il minimo tra tutte le distanze
Per comunicare si possono usare solo le code word appartenenti al vocabolario.

Se arriva questa code word: 101000 che non appartiene al vocabolario posso risalire in maniera probabilistica a capire qual'era la code word originale in base alla distanza dal vocabolario



P(e=1) >> P(e=2) >> P(e=3) …

Basi teoriche per poter fare rilevazione e correzione degli errori

Se abbiamo e errori e vogliamo correggerli abbiamo bisogno di una distanza $d = 2e + 1$, invece per rilevarli abbiamo bisogno di una distanza $d = e + 1$

Reminder - Formula coefficiente binomiale

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Formula generale per la correzione degli errori:

Dato un vocabolario con m bit e r control bit.

$$m + r = n$$

Avremo quindi 2^n combinazioni, con 2^m valide.
Data una codeword valida, si ottengono n codeword non valide modificando un singolo bit.

Formula generale

$$\sum_{i=0}^{e} \binom{m+i}{r} \leq 2^r$$

Per correggere un errore, avremo:

m	r	n
1	2	3
2	3	5
3	3	6
4	3	7
5	4	9
6	4	10
7	4	11
8	4	12
9	4	13
10	4	14
11	4	15
12	5	17

[...]

Per esempio, volessimo trasferire un messaggio di 5 bit, e volessimo verificare un solo errore e, possiamo calcolare:

$$\sum_{i=0}^{1} \binom{m+i}{r} \leq 2^r$$

$$\binom{5}{r} + \binom{6}{r} \leq 2^r$$

- Con $r = 0$, $\binom{5}{0} + \binom{6}{0} \leq 2^0 \rightarrow 1 + 1 \leq 1 \rightarrow 2 \leq 1$ - falso!
- Con $r = 1$, $\binom{5}{1} + \binom{6}{1} = \frac{5!}{4!} + \frac{6!}{5!} = \frac{120}{24} + \frac{720}{120} = 5 + 6 = 11 \leq 2^1$ - falso!
- Con $r = 2$, $\ldots$ - falso
- Con $r = 3$, $\ldots$ - falso
- Con $r = 4$, $\ldots$ - falso
- Con $r = 5$, $\binom{5}{5} + \binom{6}{5} = 1 + \frac{6!}{5!} = 1 + \frac{720}{120} = 1 + 6 = 7 \leq 2^5$ - vero!

Codice di Hamming

Questo algoritmo serve ad inserire dei bit di controllo prima di inviare il messaggio in modo che il destinatario possa risalire al messaggio originale in caso di errori dovuti al canale inaffidabile.

E' composto da 2 fasi: una di creazione del messaggio e una di verifica e correzione (per il destinatario)

- **Parte 1**
  L'obbiettivo qui è prendere i bit contenenti i dati e infarcirli con altri bit di controllo in modo che diventino resistenti agli errori
  Indicheremo con b_r i bit di controllo, questi bit vanno posizionati solo negli indici che sono potenze di 2.
  Prendiamo la stringa di esempio: `xx1x001x0`
  Mettiamo in fila le posizioni da 1 a 9 e vediamo cosa c'è dentro:
  - Pos 1: b_1 (bit di controllo, indicato con `x`)
  - Pos 2: b_2 (bit di controllo, indicato con `x`)
  - Pos 3: $b_3 = 1$ (dato)
  - Pos 4: b_4 (bit di controllo, indicato con `x`)
  - Pos 5: $b_5 = 0$ (dato)
  - Pos 6: $b_6 = 0$ (dato)
  - Pos 7: $b_7 = 1$ (dato)
  - Pos 8: b_8 (bit di controllo, indicato con `x`)
  - Pos 9: $b_9 = 0$ (dato)

Come troviamo il valore delle x?
La regola dice che ogni bit di controllo "sorveglia" uno specifico numero di bit:

> **REGOLA**
>
> Un bit di controllo in posizione X sorveglia tutte e sole le posizioni che, scritte in binario, hanno un **1** nello stesso posto in cui ce l'ha la posizione X.
> Facciamo gli esempi concreti (usiamo 4 cifre binarie per comodità):
>
> - **Il bit b_1 (posizione 1):** In binario il numero 1 si scrive `0001`. L'uno si trova nell'**ultima cifra a destra**. Quindi b_1 sorveglia tutte le posizioni che in binario finiscono con 1. Quali sono i numeri che finiscono con 1 in binario? Esattamente i **numeri dispari**: 1 (`0001`), 3 (`0011`), 5 (`0101`), 7 (`0111`), 9 (`1001`), ecc.

L'operazione usata è lo *XOR* (se i bit sono uguali il risultato è 0, se sono diversi è 1

- $b_1 = b_3 \otimes b_5 \otimes b_7 \otimes b_9 = 1 \otimes 0 \otimes 1 \otimes 0 = 0$
- $b_2 = b_3 \otimes b_6 \otimes b_7 = 1 \otimes 0 \otimes 1 = 0$
- $b_4 = b_5 \otimes b_6 \otimes b_7 = 0 \otimes 0 \otimes 1 = 1$
- $b_8 = b_9 = 0$

Sostituendo i valori calcolati ($b_1 = 0, b_2 = 0, b_4 = 1, b_8 = 0$) al posto delle `x` nella stringa di partenza, otteniamo il messaggio pronto per essere spedito: `001100100`

- **Parte 2**

```
001101100
^^ ^ ! ^

^ = bit da controllare
| = errore di comunicazione
```

Adesso il destinatario (che non sa ancora se c'è un errore) riesegue la stessa operazione di XOR di prima sugli stessi bit, i risultati trovati vanno confrontati con i bit di controllo (nelle pos. potenze di 2) e se sono uguali non ci sono errori, se sono diversi c'è un errore:

- $b_1 = b_3 \otimes b_5 \otimes b_7 \otimes b_9 = 1 \otimes 0 \otimes 1 \otimes 0 = 0$
- $b_2 = b_3 \otimes b_6 \otimes b_7 = 1 \otimes 1 \otimes 1 = 1$ Errore! Nella stringa ricevuta, $b_2 = 0$.
- $b_4 = b_5 \otimes b_6 \otimes b_7 = 0 \otimes 1 \otimes 1 = 0$ Errore! Nella stringa ricevuta, $b_3 = 0$.
- $b_8 = b_9 = 0$

Per trovare il bit errato basta sommare gli indici dei bit di controllo che hanno fallito il test: in questo caso $b_2 + b_4 = b_6$ → l'errore è in posizione 6.

Correzione

Per correggere semplicemente eseguiamo un bit flip

Gestire gli accessi multipli

In una rete, possiamo individuare due tipi di collegamento:

• *Collegamenti punto a punto.* Sono collegamenti diretti tra trasmittente e ricevente.

• *Collegamenti broadcast.* Sono collegamenti in cui più nodi trasmittenti e riceventi sfruttano lo stesso canale broadcast.

Collisioni

Quando due nodi cercano di trasmettere sullo stesso canale nello stesso istante, avviene una collisione. Le collisioni causano perdite di frame, e, in eccessiva frequenza, un generale spreco di banda. I protocolli di accesso multiplo gestiscono i canali in maniera consona per non rientrare in situazioni del genere. Esistono tre tipi principali di protocolli per l'accesso multiplo:

• **Protocolli a suddivisione del canale**. Basati sulla suddivisione del canale in porzioni accessibili esclusivamente ad un nodo.

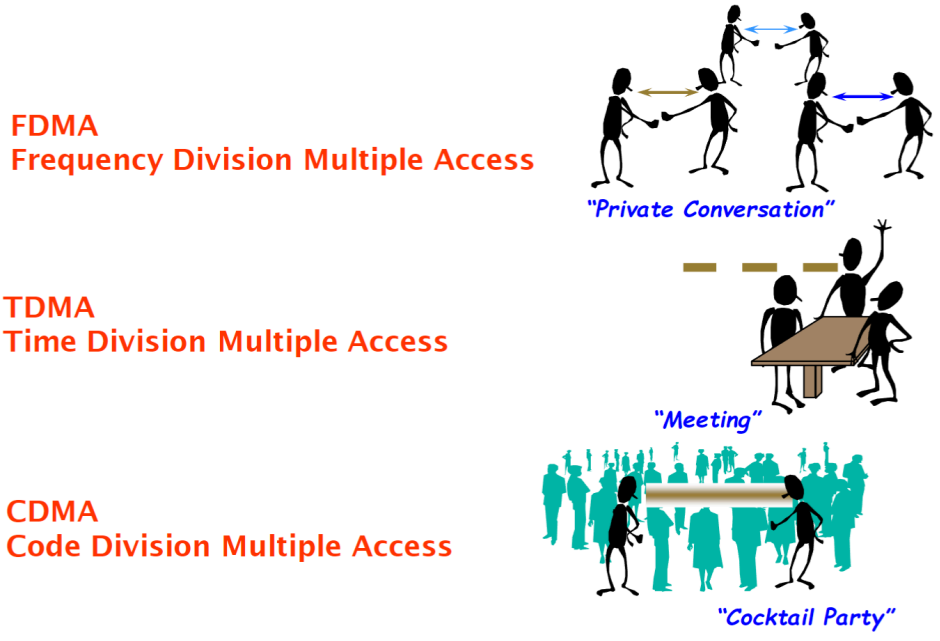
• **Protocolli ad accesso casuale**. In cui l'accesso al canale avviene in maniera casuale. Il metodo in questione ammette collisioni, che verranno gestite in maniera opportuna.

• **Protocolli a rotazione** - senza collisioni. Coordinando opportunamente l'accesso al canale, è possibile evitare le collisioni.

Iniziamo dai primi - *Protocolli a suddivisione del canale*
Abbiamo 3 protocolli diversi:
FDMA - Frequency Division Multiple Access. Consiste nella suddivisione dello spettro del canale in varie bande di frequenza. A ogni nodo è associata una banda fissa. Questa suddivisione limita la banda, sempre ad R/N , ma non blocca la trasmissione dei nodi.

TDMA - Time Division Multiple Access.
Supponiamo che un canale broadcast supporti N nodi, e che abbia velocità R bps. TDMA, suddividerà il tempo in intervalli di tempo, e ogni intervallo di tempo in N slot temporali. Ogni slot sarà dedicato ad un nodo: come conseguenza, ogni nodo avrà un tasso di trasmissione di R/N bps. Risolve effettivamente il problema delle collisioni, ma quando si è in uno slot temporale dedicato a un nodo che non deve trasmettere nel canale, avverrà un'attesa non necessaria. E' un protocollo equo, che previene le collisioni, ma limita la banda di ogni nodo a R/N , e porta i nodi in attesa anche quando gli altri non devono trasmettere nulla.

CDMA - Code Division Multiple Access. Consiste nell'associazione di un codice univoco ad ogni nodo. I dati inviati sono codificati secondo quel codice. Con codici opportuni, diventa possibile trasmettere simultaneamente dati a più nodi, e per i nodi, di interpretare in maniera corretta i bit dei dati codificati. Ogni nodo deve conoscere i codici altrui per interpretare correttamente i dati.



Protocolli ad accesso casuale
Ogni nodo trasmette alla massima velocità consentita (R bps), e ogni volta che avviene una collisione, i nodi ritrasmetteranno dopo un periodo di tempo casuale (random delay). Questo ritardo casuale è indipendente tra i vari nodi, e consente, con buona probabilità, di non subire ulteriori collisioni.

Slotted ALOHA

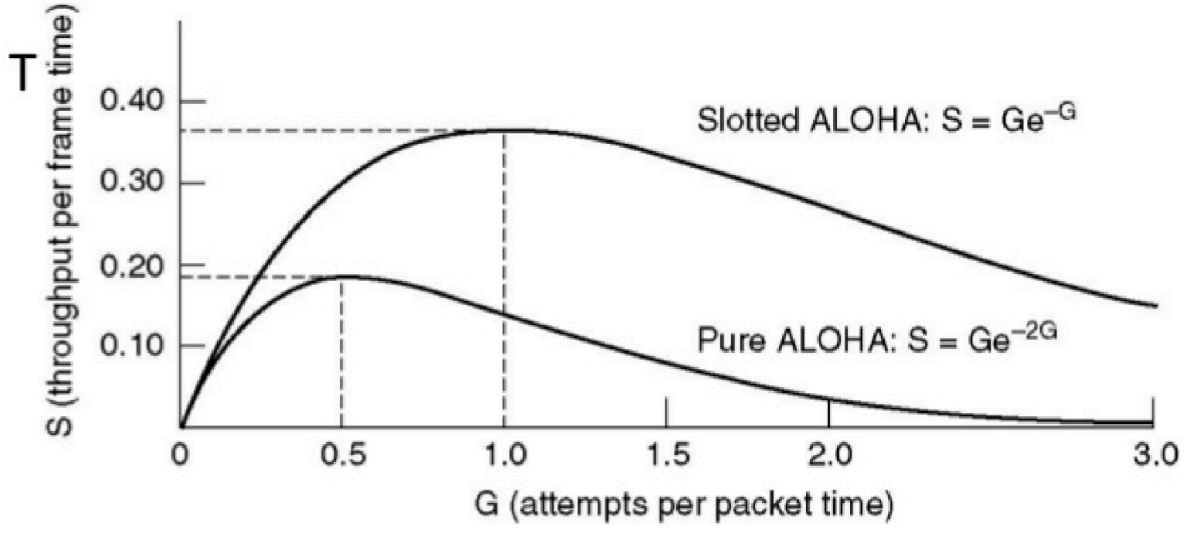
• Tutti i frame contengono L bit.
• Il tempo è suddiviso in slot L/R secondi.
• La trasmissione del frame da parte del nodo, inizia all'inizio degli slot. • I nodi sono sincronizzati in modo che tutti sappiano quando iniziano gli slot.
• Se in uno slot due o più frame collidono, tutti i nodi della rete rilevano l'evento prima della fine dello slot. Indichiamo $k \in [0, 1-K], K \in N$. I nodi slotted ALOHA seguono il seguente comportamento:

1. Quando un nodo ha un nuovo frame da spedire, attende l'inizio dello slot successivo, per poi trasmettere il frame.
2. Se non si verifica una collisione, la trasmissione è conclusa.
3. Se si verifica una collisione, il nodo la rileva prima del termine dello slot e ritrasmette il frame dopo k frame, fino a quando l'operazione non ha successo. k è un valore casuale minore di K. K cresce ad ogni collisione. `

E' un protocollo semplice, che consente ad ogni nodo di lavorare al massimo delle performance possibili, senza limitare i bps a priori. Inoltre, è un protocollo altamente scalabile e abbastanza decentralizzato: l'unica condizione, è che gli slot siano sincronizzati per tutti i nodi. Tuttavia, ammette collisioni e idle slots, uno spreco generale di slot e un aumento della latenza.

Unslotted ALOHA o Pure ALOHA

E' una versione totalmente asincrona dello slotted ALOHA. Il nodo mittente aspetterà un tempo pari al Round Trip Time, in attesa di un ACK. Se l'ACK non arriva, il nodo intuirà che è avvenuta una collisione. In tal caso, aspetterà un quanto di tempo di durata casuale (ma inclusa in un range determinato). Le performance migliori, sono ottenute dalla versione con gli slot temporali.



E un grande prezzo da pagare quello per la completa asincronia: l'algoritmo slotted ha un throughput complessivo due volte più grande di quello unslotted. L'intuizione è banale: nella versione slotted di ALOHA, due frame possono essere mandati solo

all'inizio dello slot temporale. Nella versione unslotted, i frame possono collidere anche a metà trasmissione.

CSMA - Carrier Sense Multiple Access

I protocolli ALOHA, portano i nodi a decidere di trasmettere indipendentemente dall'attività degli altri nodi. *I protocolli CSMA introducono un meccanismo di rilevamento della portante del canale, per "ascoltare" se altri nodi stanno già trasmettendo. In questo modo, i nodi trasmetteranno esclusivamente quando non rilevano alcuna trasmissione già presente nel canale.* Al contrario di ciò che si potrebbe pensare, questo protocollo non sconfigge definitivamente la problematica delle collisioni. La velocità di trasmissione (nonostante spesso sia altissima) è un fattore non trascurabile.

1. Il nodo A trasmette all'istante t_0.
2. B misura la portata del canale all'istante $t_1 > t_0$.
3. All'istante t_1 i bit trasmessi da A potrebbero non essersi propagati fino a B.
4. B trasmette. Collisione!

CSMA-CD - Carrier Sense Multiple Acces con Collision Detection

Un miglioramento è ottenuto se si introduce un meccanismo di collision detection sopra quello di rileva- mento del carico: la scheda che avrà trasmesso un determinato frame continuerà a misurare il carico del canale. Se rileva rumore o energia di altre trasmissioni nel canale, il nodo interromperà istantaneamente la trasmissione del frame, manderà un segnale di jamming con backoff a tutti gli host, informando tutti i nodi della collisione, e attenderà un quantitativo di tempo casuale per poi ritrasmettere. Questo tempo è chiamato tempo di backoff, non è fissato (intuitivamente renderebbe inutilizzabile il protocollo), e dipende dall'algoritmo scelto dal protocollo.

Università di Catania

CSMA vs CSMA /CD