Riassunto di

# Reti e sicurezza

Fonti: 5° edizione del libro “Reti di calcolatori” di A. S. Tanenbaum e materiale spiegato a lezione nell’A.A. 2018/2019

# Sommario

**Il livello data link**



**Il sottolivello MAC**

# Introduzione

## Hardware di rete

Capitolo 1.2 della 5° edizione del libro “Reti di calcolatori” di A. S. Tanenbaum

Ci sono due tipi principali di tecnologie di trasmissione:

- **Broadcast**
- **Punto a punto (Point to point)**

La tecnologia **broadcast** permette la connessione di tutte le macchine di una rete attraverso un singolo canale condiviso:

- un pacchetto inviato da una qualsiasi macchina viene ricevuto da tutte le altre che, se vedono che il pacchetto è indirizzato a loro lo processano, altrimenti lo ignorano.
- Modalità operative presenti in questa tecnologia:
  - **broadcast**: pacchetto inviato da una macchina e processato da tutte le altre;
  - **multicast**: pacchetto inviato da una macchina a tutti e processato da un sottoinsieme di macchine.
- Esempio: reti wireless.

La tecnologia **punto a punto** permette la connessione di coppie di computer che comunicano mediante **pacchetti** (brevi messaggi). Le reti punto a punto con solo un trasmettitore e un ricevitore si dicono **unicast**.

Le reti possono essere classificate in base alla loro dimensione fisica (o estensione geografica):

1. Personal Area Network (**PAN**)
2. Local Area Network (**LAN**)
3. Metropolitan Area Network (**MAN**)
4. Wide Area Network (**WAN**)
5. **Internetwork**

**PAN**: rete che permette ai dispositivi di comunicare nello spazio fisico alla portata di una persona. Alcuni esempi di PAN sono:

- le reti wireless presenti fra i PC e le loro periferiche (ovviamente quelle senza fili);
- le reti con la tecnologia **Bluetooth**, che adottano il paradigma **master-slave** (nell’esempio precedente: il PC è *master*, la periferica è *slave*).

**LAN**: rete privata che opera all’interno, o nelle vicinanze, di un singolo edificio. È implementata sia in versione cablata (wired), con lo standard **IEEE 802.3** (**Ethernet**), che in versione senza fili (wireless), con lo standard **IEEE 802.11** (**Wi-Fi**), rinominando la rete in **WLAN**.

**MAN**: rete che copre un’intera città. Alcuni esempi di MAN sono:

- la rete TV via cavo cittadina, implementata con una **stazione di testa** (**cable headset**) che si occupa di distribuire il segnale a tutte le varie abitazioni;
- le reti **WiMAX**, definite dallo standard **IEEE 802.16**.

**WAN**: rete che copre un’area geograficamente estesa (solitamente una nazione).

**Internetwork**: è un insieme di reti interconnesse conosciute anche come **internet** (con la i minuscola), generico, da non confondere con la rete **Internet**, globale, che è una internet specifica:

- Internet usa le reti degli **Internet Service Provider** (**ISP**) per connettere reti aziendali, domestiche e altre.

# Software di rete

Capitolo 1.3 della 5° edizione del libro "Reti di calcolatori" di A. S. Tanenbaum

La maggior parte delle reti è organizzata come una **pila di livelli** (**layer**) o **strati**, costruiti uno sopra l'altro.

Ogni livello fornisce dei servizi al livello superiore, senza fornire dettagli implementativi (come per l'information hiding in OOP).

**Protocollo**: regole adottate da entrambi i computer a un determinato livello della pila. È un accordo tra entrambe le parti che comunicano su come deve procedere la comunicazione.

**Peer**: entità che formano i livelli di pari grado sui diversi computer. Possono essere processi software oppure dispositivi hardware. I peer comunicano attraverso il protocollo del loro livello.

I peer non comunicano **direttamente**: infatti ogni livello che deve trasmettere, per scambiare informazioni con lo stesso livello del ricevente, deve prima far passare dati e informazioni di controllo prodotte ai livelli sottostanti, fino a sotto il 1° livello dove c'è il **supporto fisico** attraverso cui avviene la comunicazione vera e propria.

**Interfaccia**: definisce le operazioni elementari e i servizi che ci sono fra coppie di livelli successivi.

**Architettura di rete**: insieme di livelli e protocolli.

**Pila di protocolli** (**protocol stack**): elenco dei protocolli usati da uno specifico sistema.

Esempio di comunicazione tra livelli, in un'architettura a 5 livelli (il 1° è il supporto fisico):

- Messaggio *M* prodotto da un processo applicativo al livello 5, passato poi al livello 4 per la trasmissione.
- Il livello 4 antepone un **header** (**intestazione**) ad *M* e passa quindi $\mathbf{header_4}$+*M* al livello 3. L'header include informazioni di controllo per consentire al suo peer di identificare *M.*
  Il livello 4 non ha limiti di dimensione per *M*, però il livello 3 sì, quindi a questo livello è necessario spezzare *M* in pacchetti che, dovendo essere numerati per essere ricomposti correttamente dal destinatario, richiedono l'aggiunta di un ulteriore header, ottenendo $\mathbf{header_3}$+$\mathbf{header_4}$+*M*.
- Il livello 3 decide la linea di uscita da usare e passa i pacchetti al livello 2, che aggiunge un header e anche un **trailer** (**informazione aggiuntiva in coda**) ottenendo $\mathbf{header_2}$+$\mathbf{header_3}$+$\mathbf{header_4}$+*M*+$\mathbf{trailer_2}$ che poi dà al livello 1 per la comunicazione fisica.
- Nel destinatario viene ricomposto il messaggio analizzando ad ogni livello *n* i propri $\mathbf{header_n}$ e $\mathbf{trailer_n}$ (se presenti), che vengono rimossi gradualmente da *M*. Il livello 5 del destinatario otterrà il messaggio *M* così come spedito dal mittente.

Nella comunicazione in rete potrebbero presentarsi degli errori in trasmissione o ricezione. Per questo motivo vengono aggiunti ad ogni livello header e/o trailer contenenti codici per l'**individuazione degli errori** in modo tale da poter richiedere la ritrasmissione dei messaggi, se necessario. Codici più complessi permettono non solo l'individuazione di errori, ma anche la loro **correzione**.

Un'altra decisione che deve essere presa nella comunicazione è come spedire i messaggi, ovvero quali "strade" prendere: potrebbero essercene alcune di non funzionanti, altre di troppo lunghe.

Queste decisioni devono essere prese nel miglior modo possibile e automatico tramite un processo chiamato **routing** (instradamento).

**Progetti scalabili**: progetti di rete che continuano a funzionare a dovere anche all'aumentare della dimensione della rete stessa.

# Modelli di riferimento

Capitolo 1.4 della 5° edizione del libro "Reti di calcolatori" di A. S. Tanenbaum

## ISO OSI

Sviluppato dall'**International Standards Organization** (**ISO**). **OSI** sta per **Open Systems Interconnection**, perché si occupa della connessione di sistemi "aperti" alla comunicazione con altri. Il nome completo del modello è **ISO OSI**. Per brevità verrà definito **modello OSI**.

Il modello si compone di sette livelli, in cui ognuno possiede queste caratteristiche:

- ha un'astrazione diversa dagli altri;
- ha una funzione da svolgere ben definita, con uno sguardo rivolto alla definizione di standard per protocolli internazionali;
- il flusso di informazioni in uscita attraverso le interfacce è ridotto;
- è pensato in modo da non contenere troppe funzioni distinte ma al contempo di non contenerne troppo poche, appesantendo l'architettura globale.

I livelli sono i seguenti:

- livello 1, **fisico**: si occupa della trasmissione di bit grezzi sul canale di comunicazione. Viene controllato che i bit arrivino a destinazione così come inviati (ovvero che se è stato spedito 1 venga interpretato 1 e non 0). Le specifiche riguardano le interfacce meccaniche o elettriche e temporizzazioni, oltre al mezzo di trasmissione sotto al livello fisico;
- livello 2, **data link**: si occupa di far diventare una trasmissione grezza in una linea che appare priva di errori (non rilevati), mascherando gli errori reali al livello successivo. Questo viene compiuto forzando il trasmittente a suddividere il messaggio in più **frame** (sequenze di byte) trasmessi sequenzialmente. Se il servizio è affidabile, il trasmittente riceve una conferma di ricezione attraverso un **frame di acknowledgement**. Inoltre, evita che il trasmittente riempia il buffer troppo velocemente, mediante una regolazione del traffico. Nelle reti broadcast c'è un sottolivello in più a questo punto, il **medium access control**, che si occupa di controllare l'accesso al canale condiviso;
- livello 3, **rete**: si occupa di controllare il funzionamento della sottorete. Controlla le modalità in cui i pacchetti vengono inoltrati dalla sorgente alla destinazione, mediante tabelle statiche "cablate" dentro la rete, oppure aggiornandoli automaticamente per aggirare componenti guaste. Un punto chiave di questo livello è la gestione della **qualità del servizio offerto**, ovvero il controllo dei ritardi, tempi di transito, jitter, ecc. Inoltre, si occupa di controllare gli indirizzi di destinazione dei pacchetti, in modo da cambiarli se non corrispondenti a quelli della sottorete in cui opera;
- livello 4, **trasporto**: si occupa di prendere i dati dal livello superiore, dividerli in unità più piccole, se necessario, e di passarle al livello di rete, assicurandosi che tutti queste arrivano a destinazione: infatti, a questo livello ci si occupa di controllare tutto il tragitto da sorgente a destinazione mediante intestazioni e messaggi di controllo nei pacchetti, a differenza dei livelli precedenti che invece il loro obiettivo è occuparsi di comunicare con i propri vicini, non fra computer sorgente e computer destinatario, che possono essere molto distanti fra loro;

- livello 5, **sessione**: permette l'instaurazione di una **sessione** fra computer diversi, che offrono vari servizi come **controllo del dialogo** (chi deve trasmettere e quando), **gestione dei token** (per evitare che vengano eseguite operazioni critiche contemporaneamente) e **sincronizzazione** (supervisionare la comunicazione per consentirne la ripresa in caso di crash);
- livello 6, **presentazione**: si occupa della sintassi e semantica dell'informazione trasmessa;
- livello 7, **applicazione**: comprende una varietà di protocolli utilizzati dagli utenti, ad esempio: **http**, **ftp**.

## TCP/IP

Architettura di rete nota come **modello di riferimento TCP/IP**, dovuta ai nomi dei suoi due protocolli principali, **TCP** e **IP**.

I livelli sono i seguenti:

- **link**: è connection-less e in grado di operare attraverso varie reti. Descrive cosa devono fare i diversi collegamenti (per esempio: linee seriali ed Ethernet) per esaudire le proprie necessità senza connessione. Fa quindi da interfaccia fra l'host e il mezzo trasmissivo;
- **internet**: si occupa di permettere agli host di inviare pacchetti in qualsiasi rete (che tra l'altro potrebbero arrivare in ordine diverso da quello di trasmissione e sarebbe compito dei livelli superiori riordinarli). Il nome del livello è inteso in senso generico (non ha niente a che vedere con Internet). Il protocollo ufficiale a questo livello è Internet Protocol (**IP**), assieme a uno di supporto chiamato Internet Control Message Protocol (**ICMP**);
- **trasporto**: si occupa di gestire la comunicazione fra peer host, sorgente e destinazione, come nel livello 4 del modello OSI. Ci sono due protocolli principali a questo livello: Transmission Control Protocol (**TCP**) e User Datagram Protocol (**UDP**). Il primo è *affidabile*, orientato alla connessione e permette che il flusso di byte arrivi integro a destinazione (spezzettando il messaggio in invio in parti più piccole poi ricomposte a destinazione), oltre che a controllare il flusso. Il secondo è *inaffidabile*, senza connessione e controllo di flusso, per le applicazioni che preferiscono gestirsi queste operazioni in maniere autonoma;
- **applicazione**: contiene tutti i protocolli presenti al livello 7 del modello OSI, che si gestiscono sessione e presentazione (rispettivamente livello 5 e livello 6 del modello OSI) in maniera autonoma, senza la necessità di due livelli che si occupino di questo.

## Ibrido

È una via di mezzo fra le parti migliori di entrambi i modelli. È definito **modello ibrido**, formato da 5 livelli.

I livelli sono i seguenti:

- **fisico**: specifica come trasmettere i bit sui differenti tipi di mezzo di trasporto, sottoforma di segnali elettrici;
- **link**: coinvolto quando bisogna spedire messaggi di lunghezza finita direttamente tra computer connessi con un livello prefissato di affidabilità. Alcuni protocolli a questo livello sono **Ethernet** e **Wi-Fi**;
- **rete**: collega i dispositivi nelle reti e nelle internetwork in modo da spedire messaggi fra computer distanti. Cerca il percorso migliore per svolgere questo compito. Il protocollo principale usato a questo livello resta **IP**;
- **trasporto**: si occupa di dare garanzia di consegna al livello di rete, aumentandone l'affidabilità fornendo astrazioni sulle modalità di consegna dei byte. Il protocollo principale usato è **TCP**;

- **applicazione**: è lo stesso del livello **applicazione** del modello TCP/IP.

## Confronto fra modelli OSI e TCP/IP (e il motivo per cui si usa quello ibrido)

- Entrambi sono basati sul concetto di pila di protocolli indipendenti e le funzioni dei livelli sono abbastanza simili.
- Entrambi, sopra il livello di trasporto forniscono le astrazioni per gli utenti e sono orientati alle applicazioni.
- Il modello OSI presenta tre concetti fondamentali ben esplicitati: **servizi**, **interfacce**, **protocolli**. I servizi sono ciò che viene offerto da un livello al loro superiore; le interfacce sono le modalità di accesso ai processi sovrastanti, ovvero specificano parametri e risultati, ma non come sono realizzati; i protocolli (paritetici, ovvero fra pari) usati all'interno di ogni livello, che possono cambiare a piacimento ma devono rispettare gli input e gli output attesi del livello.
- Il modello TCP/IP non specifica i concetti fondamentali appena descritti, ed è molto vago, poiché sono stati realizzati prima i protocolli e poi il modello per poterlo standardizzare, a differenza del modello OSI.
- Il modello OSI ha 7 livelli, forse troppi. Il modello TCP/IP ne ha 4 con il primo, **link**, che non fa una corretta distinzione fra **fisico** e **data link**, che sono due astrazioni completamente diverse e non dovrebbero stare assieme.
- Il modello OSI non ha avuto successo perché è stato realizzato con scarso tempismo (quando l'altro modello era già ampiamente utilizzato, almeno in ambito accademico, che spingeva per l'adozione per tutti i costruttori), con tecnologia e implementazione scadente (lo standard c'è, i protocolli da utilizzare invece non sono stati definiti) e incapacità politica di farlo adottare come standard.

# Il livello fisico

## Basi teoriche della comunicazione dati

Capitolo 2.1 della 5° edizione del libro “Reti di calcolatori” di A. S. Tanenbaum

Le informazioni possono essere trasmesse via cavo variando alcune proprietà fisiche, per esempio tensione e corrente. Rappresentando il valore di questa tensione o corrente attraverso una funzione a un valore nel tempo, *f(t)*, è possibile modellare il comportamento del segnale e analizzarlo matematicamente.

Alle basi della comunicazione dati c’è l’**analisi di Fourier** che afferma che qualsiasi funzione periodica sufficientemente regolare può essere scomposta sommando un numero (potenzialmente infinito) di funzioni seno e coseno:

$$g(t) = \frac{1}{2}c + \sum_{n=1}^{\infty} a_n \sin(2\pi n f t) + \sum_{n=1}^{\infty} b_n \cos(2\pi n f t)$$

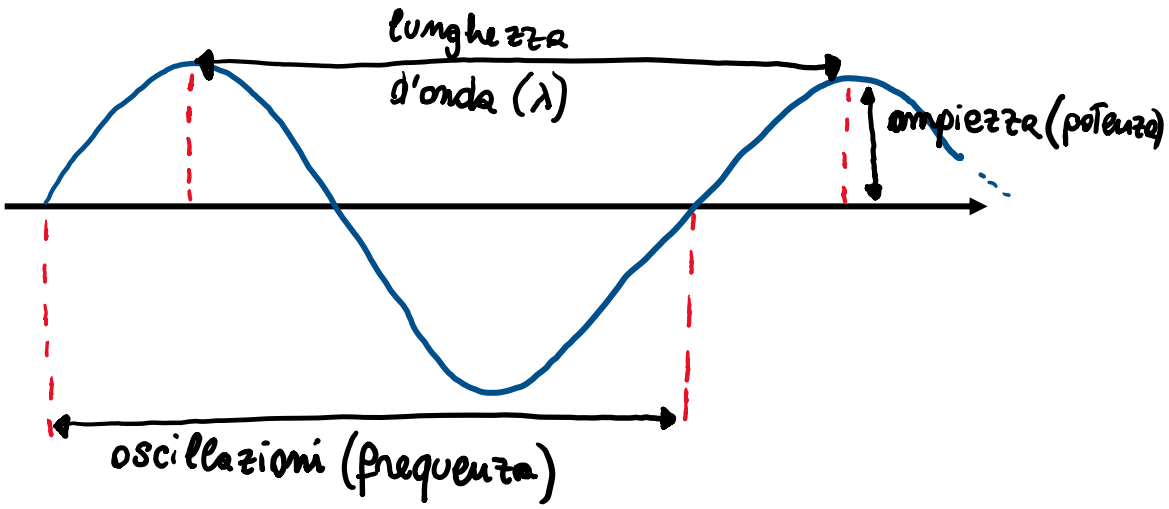
in cui:

- $f = 1/T$ rappresenta la frequenza fondamentale;
- $a_n$ e $b_n$ sono rispettivamente le ampiezze seno e coseno delle **armoniche** (termini) $n$-esime;
- $c$ rappresenta una costante.

Questa scomposizione è chiamata **serie di Fourier**.

La serie di Fourier è utile per ricostruire la funzione di partenza, conoscendo il periodo T e le ampiezze delle armoniche. Infatti, un segnale di durata finita (caratteristica comune a tutti i segnali) può essere gestito immaginandolo come se fosse ripetuto infinite volte.

Un segnale non è altro che una sinusoide:



Tutto questo è importante perché i canali reali di comunicazione, a frequenze diverse, influiscono in modo non omogeneo sui segnali. Inoltre, il segnale inviato nel mezzo trasmissivo perde intensità durante il suo tragitto.

L'obiettivo della comunicazione è quello di riuscire a inviare un messaggio abbastanza fedele che possa poi essere ricondotto a quello originale in maniera abbastanza precisa da poter riconoscere le sequenze di bit inviate.

**Banda** (**bandwidth**): è l'intervallo di frequenze (misurate in Hz) trasmettibili su un mezzo trasmissivo, solitamente da 0 a una certa frequenza $f_c$, in cui il segnale non viene attenuato fortemente (limite non molto preciso, spesso viene indicata la **banda passante*** da 0 a dove la potenza è attenuata del 50%). È detta anche *larghezza di banda* e l'informazione trasportata dipende solamente da questa e non dalle frequenze di inizio e fine di un certo canale. Da notare che gli ingegneri e gli informatici usano questo termine differentemente: i primi intendono una quantità misurabile in Hz (come la definizione appena data); i secondi intendono il massimo tasso di trasporto dati di un canale, misurabile in bps (bit per second).

***Banda passante**: è una proprietà fisica del mezzo di trasmissione e dipende dalla sua costruzione, dal suo spessore e dalla sua lunghezza. Solitamente è limitata da un filtro che lascia passare solo le frequenze che servono allo scopo della comunicazione.

Anche in un canale perfetto (senza interferenze) c'è un limite massimo di trasmissione, che dipende dalla banda. Trasmesso un segnale su un mezzo trasmissivo in cui:

- $B$ è l'ampiezza di banda massima (in Hz);
- $bit\ rate$ è il numero di bit trasmessi al secondo
- $V$ è il numero di simboli presi da un alfabeto di simboli (nel caso di segnali binari, l'alfabeto è 0, 1);

si può ottenere il $bit\ rate$ (informalmente, le informazioni trasmesse al secondo) mediante questa formula (**teorema di Nyquist**):

$$bit\ rate = 2B\ \cdot \log_2 V$$

Visto che ogni canale ha sempre del rumore termico (provocato dal movimento delle molecole), il massimo $bit\ rate$ non è calcolabile con la precedente formula ma con una più specifica, in cui:

- $B$ è l'ampiezza di banda massima (in Hz);
- $bit\ rate$ è il numero di bit trasmessi al secondo;
- $S/N$ è il **SNR** (**Signal-to-Noise ratio**, rapporto segnale-rumore), indicato in forma $10 \log_{10} S/N$, che è misurata i **decibel** (**dB**);

che è la seguente (**teorema di Shannon**):

$$bit\ rate = B\ \cdot \log_2(1 + S/N).$$

# Mezzi di trasmissione vincolati

Capitolo 2.2 della 5° edizione del libro "Reti di calcolatori" di A. S. Tanenbaum

Lo scopo di un mezzo fisico è trasmettere bit grezzi da un dispositivo a un altro. Ne esistono diversi, ognuno caratterizzato dalla propria banda passante, ritardo, costo e facilità di installazione e manutenzione.

Si classificano in:

- mezzi vincolati, ovvero cavi in rame, fibre ottiche, ecc.
- mezzi non vincolati, come onde radio, laser, ecc.

I tipi di collegamenti possono essere:

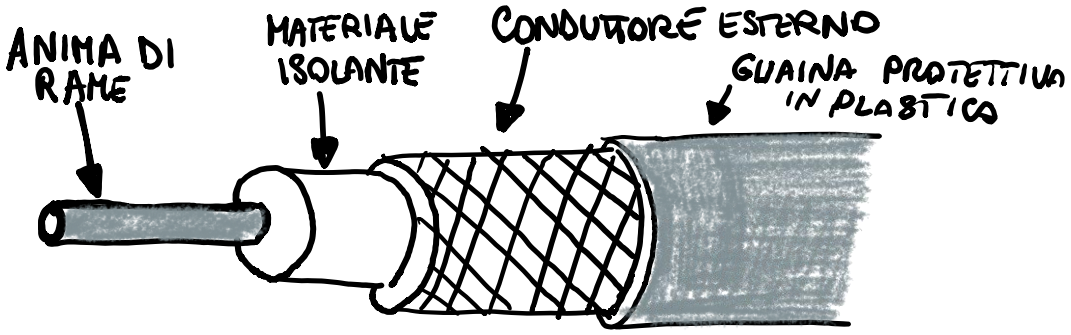
- **full-duplex**: mezzo utilizzabile in entrambe le direzioni contemporaneamente;
- **half-duplex**: mezzo utilizzabile in entrambe le direzioni singolarmente (prima un verso poi l'altro);
- **simplex**: mezzo utilizzabile in un solo verso.

## Doppino (twisted pair)

È un cavo composto da due conduttori di rame isolati, avvolti a forma di elica. L'intreccio è dovuto al fatto che due cavi paralleli formano due campi magnetici che si interferiscono a vicenda (**crosstalk**), mentre se intrecciati i campi magnetici vengono annullati nei punti di intersezione. Questo fa pensare, correttamente, che aumentando l'intreccio dei fili si riduce l'interferenza, ovvero più ci sono punti di intersezione fra i due cavi in rame più il cavo è performante. Il doppino viene usato sia per la trasmissione di segnali analogici che digitali. La sua ampiezza di banda dipende dal diametro del cavo e dalla distanza percorsa. Ha un basso costo e prestazione discrete. I doppini attualmente in uso sono **UTP** (**Unshielded Twisted Pair**, **doppino non schermato**) di categoria **Cat. 5** (cavi del doppino isolati singolarmente, quattro doppini raggruppati in un unico cavo da una guaina di plastica protettiva), **Cat. 6** e **Cat. 7**.

## Cavo coassiale

È un cavo più schermato del doppino e quindi può estendersi per distanze più lunghe e con velocità maggiori. È formato come segue:




La sua costruzione garantisce un'ottima immunità al rumore e un'ampiezza di banda elevata.

## Linee elettriche

Sono utilizzate per trasmettere dati ad un tasso molto basso. Viene sfruttata la linea elettrica domestica.

## Fibre ottiche

Sono mezzi utilizzati per la trasmissione a lunga distanza e con un'ampiezza di banda molto elevata, soprattutto nelle dorsali di rete ma anche per l'accesso a Internet ad alta velocità come **FTTH** (**Fiber To The Home**).

Il nucleo, o **core**, è formato da tre componenti: la sorgente luminosa, il mezzo trasmissivo e il rilevatore. L'impulso di luce corrisponde a 1, la sua assenza a 0. Il mezzo trasmissivo è una fibra di vetro sottilissima che riesce a trasportare il segnale luminoso che deve essere captato dal rilevatore. Questo sistema funziona grazie ad un principio della fisica che permette la non dispersione della luce: infatti, se il raggio luminoso colpisce la fibra di vetro ad un certo angolo di inclinazione pari o superiore a un certo angolo critico (se inferiore si disperderebbe) il segnale può essere trasmesso per chilometri senza alcuna perdita. Il nucleo è circondato da un rivestimento in vetro (cladding) che costringe la luce a rimanere nel nucleo, dato il suo indice di rifrazione inferiore. A sua volta il cladding è rivestito da una fodera di plastica.

Essendo queste tre parti molto sottili, di fatto vengono raggruppate più fibre in fasci protetti da una guaina esterna.

La fibra può essere **multimodale** (con un nucleo di diametro 50 µm) se all'interno della stessa possono essere contenuti più raggi a diverse angolazioni; **monomodale** (con un nucleo di diametro di circa 8-10 µm) se è permesso alla luce di viaggiare solo in linea retta, utile soprattutto nelle lunghe distanze.

**Dispersione cromatica**: fenomeno che dipende dalla lunghezza d'onda trasmessa all'interno della fibra ottica e riguarda gli impulsi luminosi trasmessi. Durante la propagazione questi si espandono e per evitare che si sovrappongano sarebbe necessario ridurre la velocità del segnale (cosa che non si vuole fare, date le buone proprietà della fibra ottica). Per fortuna si sono scoperte delle particolari onde, chiamate **solitoniche** (derivanti dalla scoperta dei **solitoni** da parte di J. S. Russell), che riescono a resistere per migliaia di km senza attenuazione sensibile.

Le fibre hanno tre modi per essere collegate fra loro (a differenza dei cavi in rame che basta sostanzialmente annodarli fra loro):

- tramite **connettori**, che fanno perdere circa il 10-20% della luce in ingresso, ma sono semplici da usare e comodi soprattutto per la riconfigurazione di sistemi;
- tramite un'**attaccatura meccanica**, svolta con degli strumenti detti allineatori meccanici che, allineate le fibre da unire, le pinzano facendo però perdere sempre circa il 10% del segnale;
- tramite **fusione**, fornendo una connessione solida fra i cavi con un'attenuazione del segnale di circa l'1%.

Il segnale viene trasmesso tramite un impulso luminoso generato da tre possibili sorgenti luminose:

- **LED** (**Light Emitting Diode**);
- **laser**;
- **a incandescenza**.

### Confronto tra fibre ottiche e cavi in rame

La fibra ottica, rispetto ad un cavo in rame, offre una maggiore ampiezza di banda, un segnale migliore, un bassissimo livello di interferenza (se non nullo) ed è difficile da intercettare, però è più costosa ed è molto meno flessibile, oltre che difficile da unire e derivare.

## Trasmissioni wireless

Capitolo 2.3 della 5° edizione del libro "Reti di calcolatori" di A. S. Tanenbaum

Le comunicazioni non via cavo possono essere implementate in vari modi, utilizzando lo **spettro elettromagnetico**, le **trasmissioni radio**, le **microonde** oppure le **onde infrarossi**.

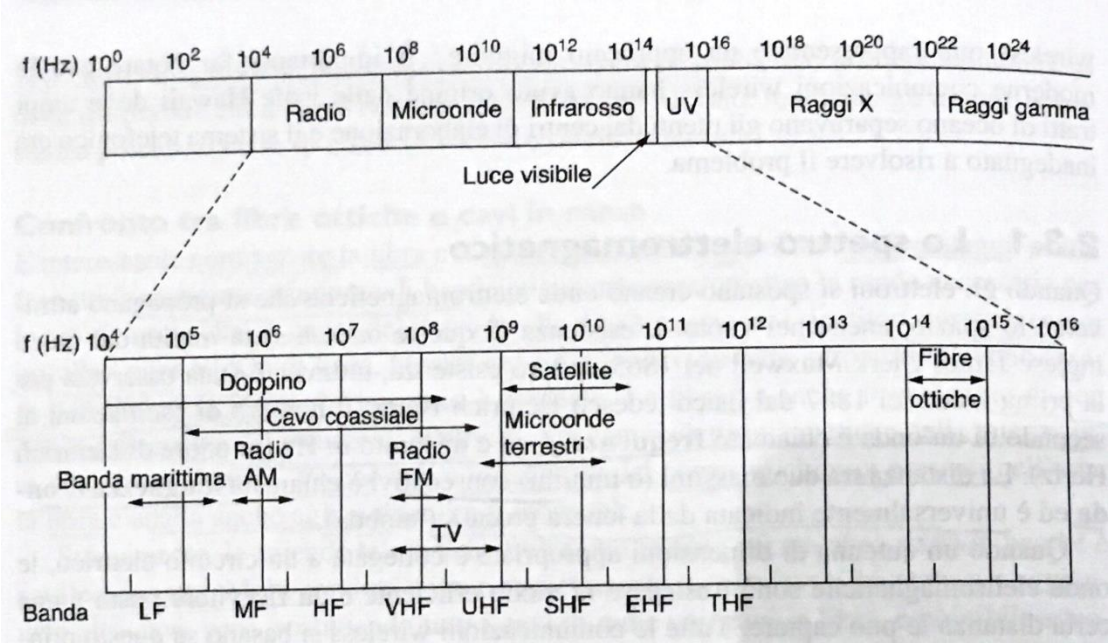
## Spettro elettromagnetico

Le comunicazioni attraverso lo **spettro elettromagnetico** sfruttano una relazione fondamentale della fisica che è la seguente:

$$\lambda f = c$$

dove $\lambda$ è la **lunghezza d'onda**, $f$ è la frequenza (in Hz) di un'onda elettromagnetica qualsiasi e $c$ è una costante corrispondente a **3 x $10^8$ m/s** (la velocità della luce). Questa relazione è verificata nel vuoto.

Lo spettro elettromagnetico è stato suddiviso e normato per il suo utilizzo dall'**International Telecommunication Union** (**ITU**).




*Le sigle che identificano la banda sono: LF (Low Freq.), MF (Medium Freq.), HF (High Freq.), VHF (Very High Freq.), UHF (Ultra High Freq.), SHF (Super High Freq.), EHF (Extremely High Freq.), THF (Tremendously High Freq.)*

Lo spettro di frequenze è regolato da accordi nazionali e internazionali, per poter assegnare le varie bande. Ci sono tre possibili modalità per assegnare le frequenze:

- "**beauty contest**", a chi dimostra di voler utilizzarle per un maggior pubblico interesse;
- "**lotteria**", in maniera casuale;
- "**vendita all'asta**", a chi offre di più.

Ci sono tre bande che vengono raggruppate sotto il nome di **banda ISM** (**Industrial, Scientific, Medical**) che sono libere e utilizzabili da chiunque per gli scopi industriali, scientifici e medici. Un esempio di tecnologia che lavora in queste frequenze è **Bluetooth**, ma anche telefoni senza filo, forni a microonde, periferiche per computer wireless e telecomandi.

La maggior parte delle comunicazioni utilizza una banda di frequenze ristretta ($\frac{\Delta f}{f} \ll 1$) per ottenere una miglior ricezione (molti Watt/Hz), ma in alcuni casi si usa una banda larga con due varianti:

- **spettro distribuito a frequenza variabile**, in cui il trasmettitore salta da una frequenza a un'altra un centinaio di volte al secondo, utilizzato soprattutto in ambito militare;
- **spettro distribuito a sequenza diretta** in cui si usa una sequenza codificata per distribuire il segnale su una banda di frequenza molto più ampia ed è molto efficiente nel permettere a più segnali di condividere le stesse bande di frequenze. A ogni segnale viene assegnato un codice diverso con un approccio detto **CDMA** (**Code Division Multiple Access**) usato nelle reti 3G e GPS.

### Trasmissioni radio

Le onde in radio frequenza sono semplici da generare e possono viaggiare per lunghe distanze, attraversando facilmente gli edifici. Sono largamente utilizzate per le comunicazioni perché sono omnidirezionali e ciò permette che trasmettitore e ricevitore non siano fisicamente allineati.

Le proprietà delle onde radio dipendono dalla loro frequenza: più basse sono, più attraversano bene gli ostacoli ma la potenza diminuisce facilmente; più alte sono più tendono a viaggiare in linea retta e a rimbalzare contro gli ostacoli (oltre che, ad esempio, essere assorbite facilmente dalla pioggia). A tutte le frequenze le onde sono comunque soggette a interferenze.

Le onde radio VLF, LF e MF (radio AM) seguono il terreno, ovvero la curvatura terrestre.

Le onde radio HF e VHF (radio FM) tendono a essere assorbite dal pianeta, ma quelle che raggiungono la ionosfera vengono riflesse e tornano nel pianeta.

### Trasmissioni a microonde

Sopra i 100 MHz le onde viaggiano quasi in linea retta e possono quindi focalizzarsi e viaggiare più a lungo. Il trasmettitore e il ricevente devono però essere meglio allineati.

### Trasmissioni a infrarossi

Sono trasmissioni che sfruttano onde direzionali, facili da costruire e soprattutto economiche (usate ad esempio per i telecomandi) ma che hanno il grave difetto di non attraversare ostacoli solidi, che a sua volta può essere un grande vantaggio, in quanto comporta maggiore sicurezza (rispetto ai sistemi basati su onde radio).

## Comunicazioni satellitari

Capitolo 2.4 della 5° edizione del libro "Reti di calcolatori" di A. S. Tanenbaum

Nella sua forma più semplice, un satellite di comunicazione è un grande ripetitore di microonde in cielo che contiene diversi **transponder** (ricetrasmettitori satellitari) che ricevono il segnale in ingresso e lo ritrasmettono su un'altra frequenza per evitare interferenze col segnale in arrivo (modalità operativa **bent pipe**).

**Fasce di Van Allen**: strati di particelle molto cariche intrappolate dal campo magnetico terrestre che, se attraversate da un satellite, lo distruggerebbero immediatamente. Sono due e vanno quindi a definire quali sono le zone in cui i satelliti possono essere collocati senza pericolo.

### Satelliti geostazionari – GEO (Geostationary Earth Orbit)

Sono satelliti collocati su orbite molto alte (circa 35000km di altitudine), posti sull'Equatore. Il fatto che questi satelliti possano stare solo sopra l'Equatore (per coprire al meglio la superficie terrestre) comporta che questi debbano stare ad un minimo di 2° (su 360°, ovviamente) di distanza l'uno dall'altro per evitare interferenze, portando ad un massimo di 180 satelliti di questo tipo in orbita. L'allocazione dei satelliti è gestita dall'ITU.

Applicazioni importanti per questo tipo di satelliti sono ad esempio il meteo e le TV satellitari, ma anche satelliti spia.

### Satelliti su orbite medie – MEO (Medium Earth Orbit)

Sono satelliti collocati fra le due fasce di Van Allen. Si muovono lungo la longitudine e impiegano circa 6 ore per compiere un giro completo del pianeta e quindi devono essere seguiti per poterci comunicare, a differenza dei satelliti GEO che sono fermi. Essendo più bassi, coprono un'area più piccola e sono necessari più satelliti di questo tipo per coprire l'intera superficie terrestre ma in compenso servono anche trasmettitori meno potenti diversamente da quelli per le comunicazioni con i satelliti geostazionari.

Applicazioni importanti per questo tipo di satelliti sono ad esempio il **GPS** (**Global Positioning System**) di invenzione americana, il **GLONASS** (**GLObal NAvigation Satellite System**, in inglese, **Global'naja Navigacionnaja Sputnikovaja Sistema**, in russo) di invenzione russa, **A-GPS** (**Assisted GPS**) e il barometro.

Nota storica: il primo satellite della storia fu spedito in orbita media e fu lo **Sputnik**, di invenzione russa, che non era altro che una sfera di 55 cm che emetteva segnali acustici.

### Satelliti su orbite basse – LEO (Low Earth Orbit)

Sono satelliti collocati sopra l'atmosfera e si spostano molto rapidamente, comportando l'utilizzo di numerosi satelliti per una copertura terrestre completa. Le stazioni terrestri per le comunicazioni con questi satelliti non richiedono molta potenza.

Applicazioni importanti per questo tipo di satelliti sono ad esempio **Iridium**, compagnia di telecomunicazioni satellitari (è dotata di 66 satelliti, anche se originariamente dovevano essere 77, ovvero il numero atomico dell'elemento iridio che ha portato al nome; ha un data rate molto basso e ogni satellite ha 48 celle), **Globalstar**, anche lei come Iridium (è dotata invece di 52 satelliti perché copre meno zone con satelliti meno costosi di quelli della concorrente principale poiché hanno una durata inferiore).

**Space debris**: i satelliti che non funzionano più o hanno finito il loro compito restano in orbita, aumentando la spazzatura sopra l'atmosfera.

**Sindrome di Kessler**: scenario nel quale il volume di detriti spaziali che si trovano in orbita bassa intorno alla Terra diventa così elevato che gli oggetti in orbita vengono spesso in collisione. La conseguenza diretta del realizzarsi di tale scenario consiste nel fatto che il crescente numero di rifiuti in orbita renderebbe impossibile per molte generazioni l'esplorazione spaziale e anche l'uso dei satelliti artificiali.

### Satelliti o fibra ottica?

Fino a venti anni fa i satelliti sembravano il futuro delle telecomunicazioni visto che la rete telefonica non stava in alcun modo progredendo dal punto di vista delle tecnologie impiegate (l'intera rete in USA era in mano al monopolista AT&T). Nel 1984 la situazione cambia, infatti negli Stati Uniti e in Europa si inizia a cambiare radicalmente il modo in cui erano concepite le telecomunicazioni, dando di fatto il via ad un mercato di concorrenza di cui ne potevano beneficiare tutti. Ci fu un rapido progresso della rete, un passaggio ai cavi coassiali e alla fibra ottica, oltre che l'introduzione delle tecnologie wireless.

## Modulazione digitale e multiplexing

Capitolo 2.5 della 5° edizione del libro "Reti di calcolatori" di A. S. Tanenbaum

Per inviare informazioni digitali si devono utilizzare segnali analogici rappresentanti bit.

**Modulazione digitale**: processo di conversione tra bit e segnali analogici.

**Trasmissione in banda base**: i segnali vengono trasmessi con frequenze che vanno da zero a un massimo che dipende dal tasso di trasmissione (esempio: da 0 a *B* Hz).

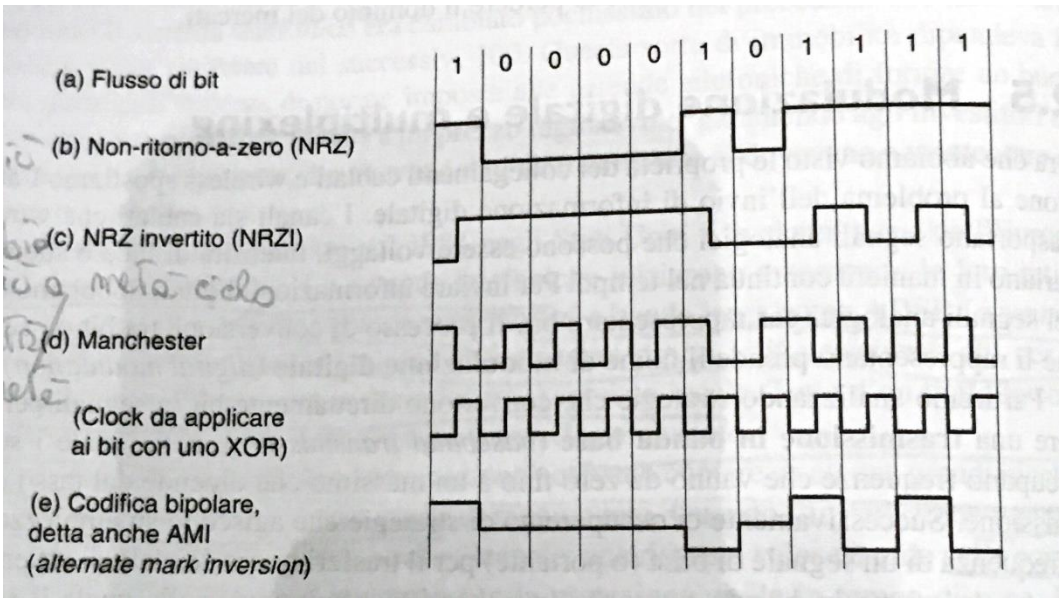
**Trasmissione in banda passante**: il segnale viene trasmesso su una banda di frequenze attorno a quella portante, con strategie che agiscono sull'ampiezza, fase e frequenza (esempio: da *S* a *S+B* Hz).

## Trasmissione in banda base

**Non-return-to-zero (NRZ)**: è la forma più naturale di trasmissione di un segnale. Consiste nel far corrispondere a una tensione positiva il bit 1, negativa al bit 0; per la fibra ottica invece la presenza di luce il bit 1, l'assenza il bit 0. Semplice da implementare, ma è difficile che venga utilizzato così come è descritto perché lungo il segnale potrebbero esserci distorsioni e rumore.

**Codifica Manchester**: consiste nel far corrispondere al bit 0 una transizione verso l'alto, al bit 1 verso il basso. A ogni bit trasmesso avviene una transizione. Se ci sono molte transizioni nel segnale è più facile che trasmettitore e ricevitore restino sincronizzati.

**Non-return-to-zero-inverted (NRZI)**: consiste nel far corrispondere ad un bit 1 una situazione di transizione, al bit 0 invece una situazione di stallo.




## Trasmissione in banda passante

**Amplitude Shift Keying (ASK)**: modulazione in cui i due livelli di ampiezza del segnale sono usati per rappresentare 0 e 1. Con più di due livelli si possono rappresentare più simboli.

**Frequency Shift Keying (FSK)**: modulazione in cui due frequenze sono usate per rappresentare 0 e 1. Con più di due frequenze si possono rappresentare più frequenze.

**Phase Shift Keying (PSK)**: modulazione in cui l'onda viene traslata di 0° o 180° all'inizio della trasmissione di ogni simbolo. In questo caso con due simboli ci sono due fasi, e la modulazione si chiama **Binary Phase Shift Keying (BPSK)**, con binary che sta per i due simboli 0 e 1.

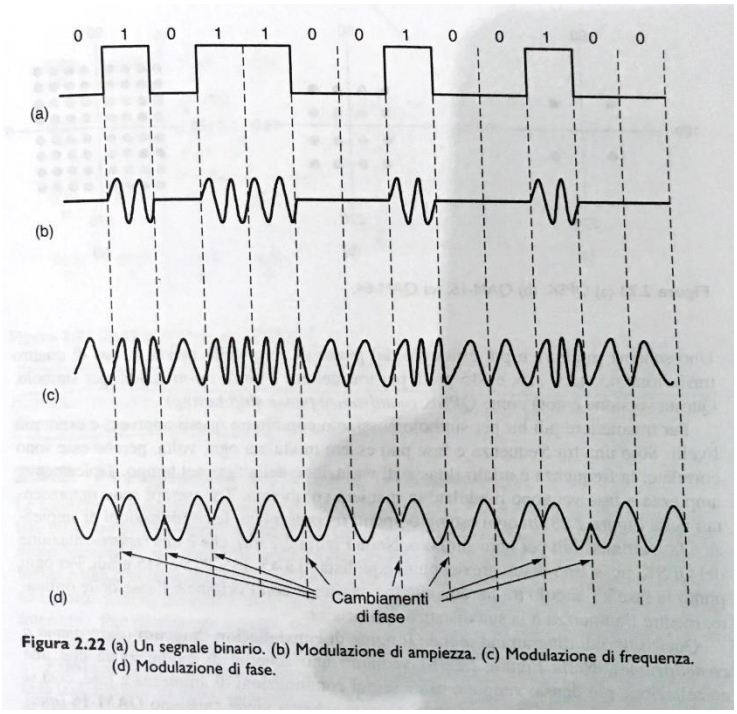



**Figura 2.22** (a) Un segnale binario. (b) Modulazione di ampiezza. (c) Modulazione di frequenza. (d) Modulazione di fase.

La **BPSK** come implementazione della **PSK** non è l'unica: se con due traslazioni (a 0° e a 180°) possiamo ottenere due simboli, con *n* traslazioni si possono ottenere *n* simboli.

**Quadrature Phase Shifting Keying (QPSK)**: modulazione in cui l'onda viene traslata di 45°, 135°, 225° o 315° all'inizio della trasmissione di ogni simbolo.

Aumentare il numero di simboli (e quindi di sfasamenti) comporta però che in caso di rumore nel segnale aumenta il rischio di interpretarlo erroneamente. Per risolvere questo, si adotta una modulazione oltre che in fase, anche in ampiezza, usando la massima frequenza disponibile.

Usando dei **grafici a costellazione** si possono introdurre altre due modulazioni chiamate **Quadrature Amplitude Modulation (QAM) *x***, dove *x* sta per il numero di simboli trasmettibili.

Alcuni esempi:

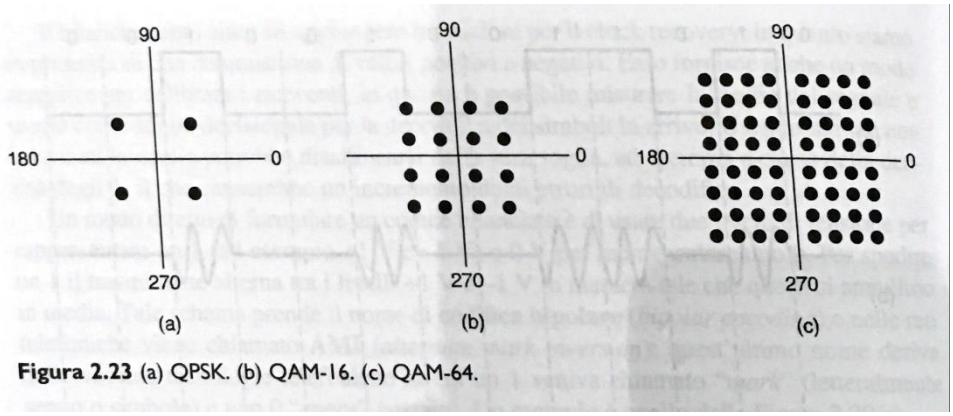



**Figura 2.23** (a) QPSK. (b) QAM-16. (c) QAM-64.

## Multiplexing

Il multiplexing è una tecnica che permette a molti segnali di condividere uno stesso canale trasmissivo. È utilizzato nelle trasmissioni in banda passante.

**Frequency Division Multiplexing (FDM)**: permette la divisione del canale in più spettri di frequenze in cui ogni utente ha un proprio uso esclusivo. Le frequenze di ogni utente vengono semplicemente traslate.

**Wavelength Division Multiplexing (WDM)**: applica lo stesso principio dell'FDM ma a frequenze molto più elevate (è utilizzato dalla fibra ottica, quindi non sono segnali elettrici ma ottici). La divisione delle frequenze, fatta da una componente attiva nel FDM, è fatta qui da una griglia di diffrazione (prisma), che è passiva e quindi molto più affidabile.

**Timelength Division Multiplexing (TDM)**: il canale trasmissivo viene diviso per periodi temporali e non per frequenze, con politica round-robin.

# La rete telefonica pubblica commutata

Capitolo 2.6 della 5° edizione del libro "Reti di calcolatori" di A. S. Tanenbaum

La **PSTN** (Public Switched Telephone Network) progettata molti anni fa era stata creata con lo scopo di trasmettere la voce umana in un modo comprensibile, non è quindi adatta per la comunicazione fra computer. Il suo funzionamento è strettamente correlato a le reti di calcolatori di grande scala.

Il suo funzionamento è il seguente:

- da ogni telefono partono due cavi di rame collegati all'**end office** o **centrale locale** più vicina (tra 1 e 10 km). Questo collegamento si definisce **local loop** o **ultimo miglio**;
- se due telefoni che devono comunicare sono collegati alla stessa centrale locale si crea una connessione elettrica fra i due, altrimenti il segnale viene inoltrato a un **tool office** o **centrale interurbana**, a cui sono collegate più centrali locali tramite le **tool connecting trunk** o **linee di connessione interurbana**;
- se due telefoni di due diverse centrali locali devono comunicare e hanno in comune la centrale interurbana possono farlo tramite quest'ultima, altrimenti i segnali devono attraversare le rispettive centrali locali e interurbane, collegate fra loro da linee a banda larga dette **intertoll trunk** o **interoffice trunk**.

Il segnale era in principio analogico, poi con l'avvento delle fibre ottiche la comunicazione è passata completamente al digitale (preferibile perché più semplice da decodificare e comunque abbastanza accurato) con l'eccezione dell'ultimo miglio che è ancora spesso realizzato in rame.

## Collegamenti locali: modem, DSL e fibre

**Modem**: dispositivo che converte un flusso di bit in un segnale analogico che li rappresenta. Modem sta per "mo*dulator* dem*odulator*". Ne esistono di vari tipi: telefonici, DSL, cable, wireless, ecc. I modem telefonici funzionano ad un massimo di 56 kbps, un limite non casuale, raggiunto dopo anni di avanzamenti tecnologici nel campo degli standard per i modem. Si è partiti dallo standard **V.21** (1964) con 300 bps trasmessi in FSK fino al **V.32** (1984) con 9600 bps trasmessi in QAM-32. Dal **V.32bis** (1991) si è poi raggiunto il **V.34bis** (1996) raggiungendo un massimo di 33,6 kbps, corrispondente al limite fisico del canale, con filtri applicati. Dato che nelle centrali telefoniche il segnale viene convertito in digitale, può essere sicuramente rimosso almeno un filtro, aumentando il SNR del Teorema di Shannon ottenendo un bit rate raddoppiato, di circa 70 kbps, quindi utilizzabili 64 kbps. Per il rispetto degli standard internazionali, americani, si usano però 56 kbps.

***x* Digital Subscriber Line (*x*DSL)**: linee di tipo DSL con più banda di quelli telefoniche, definite **broadband** (a banda larga, più per fini commerciali che tecnologici). La *x* in *x*DSL sta per un valore possibile, essendocene vari. Il più conosciuto è **ADSL** (**A** sta per **asymmetric**). Il mezzo trasmissivo

usato è lo stesso del sistema telefonico, ma quando viene attivato un abbonamento il provider si occupa di rimuovere il cavo dell'utente dal filtro applicatovi per filtrare le frequenze in modo da tenere solo quelle della voce (circa dai 300 Hz ai 3100 Hz), riuscendo a portare l'ampiezza di banda a 1.1 MHz. Avendo tutta questa banda a disposizione, ci si può quindi permettere di trasmettere anche dati (l'obiettivo delle DSL), attraverso il multiplexing del mezzo trasmissivo. L'ADSL usa FDM di tipo Discrete Multitone (**DMT**), in cui ogni canale si autogestisce con standard V.34 dei modem telefonici per il controllo costante degli errori, dividendo il mezzo trasmissivo in 256 canali da circa 4 KHz, usandone 1 (il canale 0) per la voce, 5 vuoti per creare una **banda di guardia**, 32 per l'upstream, i restanti per il downstream. Il canale 0 serve quindi a mantenere la compatibilità con il vecchio sistema telefonico, i canali dal 1 al 5 vengono tenuti liberi per limitare la possibilità di interferenze fra la voce e i dati, trasmessi nei restanti canali. ADSL ha utilizzato numerosi standard, che si sono adattati agli avanzamenti tecnologici:

- **ADSL** (noto come **G.dtm**), 1999, downstream: 8 Mbps – upstream: 1 Mbps;
- **ADSL2**, 2002, downstream: 12 Mbps – upstream: 1 Mbps;
- **ADSL2+**, 2009, downstream: 24 Mbps – upstream: 1,5 Mbps.

In particolare, ADSL2+, può raggiungere il suo downstream solo grazie al raddoppiamento della banda utilizzabile nel doppino, fino a 2,2 MHz, rimuovendo tutti i filtri, funzionante però solo in condizioni ottimali (difficili da raggiungere).

**Fiber To The Home (FTTH)**: risolve il problema dell'ultimo miglio, realizzato con cavi UTP Cat. 3 (un vero e proprio collo di bottiglia), sostituendoli con fibra ottica che arriva fino all'abitazione o all'ufficio.

# Il sistema telefonico mobile

Capitolo 2.7 della 5° edizione del libro "Reti di calcolatori" di A. S. Tanenbaum

Sistema usato per le comunicazioni a grande distanza sia voce che mobile. I **telefoni mobili** sono detti **cellulari**. Ci sono state diverse generazioni di telefonia mobile:

- ***"0G"***, una versione primordiale, di invenzione americana, fallita;
- **1G**, prima generazione, voce analogica;
- **2G**, seconda generazione, voce digitale;
- **3G**, terza generazione, voce e dati digitali.

Attualmente è in diffusione il **4G**, quarta generazione, e in fase di sviluppo il **5G**, quinta generazione.

## *"0G"*, la generazione primordiale

Questa prima generazione di telefonia mobile è molto distante dall'implementazione odierna. Risale alla fine degli anni '50 ed era una naturale evoluzione delle trasmissioni radio in **PTT** (Push-to-talk, premi per parlare), un sistema di tipo half-duplex (o trasmissione o ricezione).

**Improved Mobile Telephone System** (**IMTS**): sistema ideato negli anni '60 che va a sostituire il vecchio PTT, mettendo a disposizione due canali diversi per la trasmissione e la ricezione, in modo da non rendere più necessario la *pressione del pulsante* per parlare. Introdusse anche il concetto di privacy nelle telefonate, in quanto per la trasmissione veniva usato un canale diverso da quello per le chiamate in arrivo. Supportava solamente 23 canali bidirezionali, in un range di frequenze da 150 a 450 MHz.

## 1G, la prima generazione, analogica

**Advanced Mobile Telephone System** (**AMPS**): sistema in uso dal 1982, chiamato **TACS** in Italia. Questo sistema divide il territorio in celle grandi 10-20 Km, tutte adiacenti fra loro, che per evitare interferenze usano insiemi di frequenze diverse. Celle vicine possono avere le stesse frequenze, celle adiacenti non possono (per evitare interferenze). Le celle più sono piccole più aumentano la possibilità di supportare più utenti contemporaneamente. Le celle sono organizzate in gruppi di 7 generalmente, ma in aree in cui la densità di utenti è maggiore, queste celle vengono ridotte in potenza e aumentate di numero, come nella figura che segue:

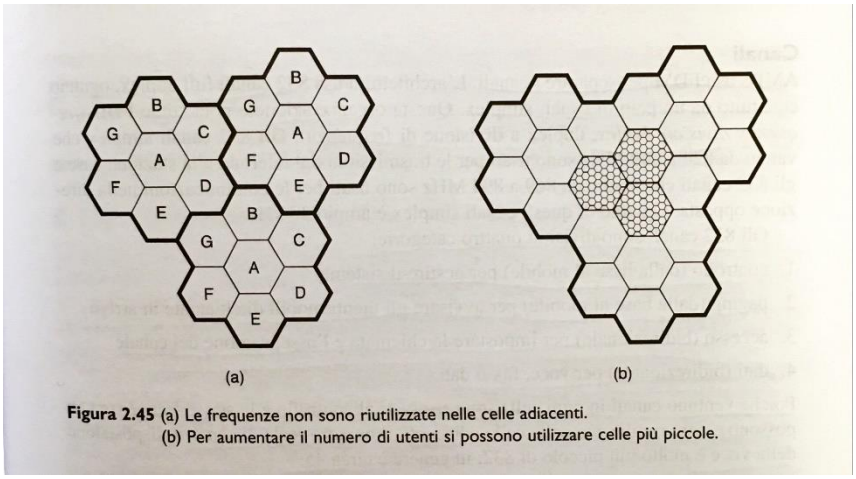



**Figura 2.45** (a) Le frequenze non sono riutilizzate nelle celle adiacenti. (b) Per aumentare il numero di utenti si possono utilizzare celle più piccole.

Ogni cella, piccola o grande, è dotata di una stazione base che comunica e gestisce tutti i cellulari presenti all'interno della cella. La stazione base è composta da un computer e un trasmettitore/ricevitore con un'antenna. In sistemi più piccoli tutte le celle sono connesse ad un unico **MSC** (Mobile Switching Center) che svolgono lo stesso compito delle centrali locali nel sistema telefonico cablato.

I cellulari ad ogni istante sono logicamente collegati ad un'unica cella e sotto il controllo della stazione base che la gestisce.

**Handoff**: un telefono si accorge di stare abbandonando (fisicamente) una cella quando il suo segnale diventa molto debole. La stazione base che lo gestisce si occupa in questa situazione di verificare quali altre celle adiacenti a quella in cui si trovava il cellulare sono disposti ad accoglierlo, fino a cederlo completamente quando la cella è stata trovata. Lo scambio di cella richiede circa 300 ms (percebili quando si sta effettuando una telefonata). L'handoff può essere di due tipi:

- *hard* handoff: cella lasciata quando il segnale è troppo debole con conseguente riaggancio ad un'altra successivamente;
- *soft* handoff: cella lasciata quando si è già agganciati alla successiva (non disponibile in questa generazione, solamente dal 3G in poi, ma deve essere supportato dall'hardware del cellulare).

Il multiplexing usato per l'1G è di tipo FDM, con 832 canali full-duplex (realizzati con coppie di canali simplex, quindi di fatto sono 832 per la trasmissione, in un range di frequenze da 824 MHz a 849 MHz, e 832 per la ricezione, da 869 MHz a 894 MHz). Questi canali sono divisi in quattro categorie:

- controllo (dalla base al mobile) per la gestione del sistema;
- paging (dalla base al mobile) per avvisare gli utenti di chiamate in arrivo;
- accesso (bidirezionale) per impostare la chiamata e assegnare i canali;

- dati (bidirezionale) per voce, fax, dati.

Dato che il sistema è molto complesso da gestire, di 832 canali solo 45 sono effettivamente utilizzabili per le telefonate.

Ogni cellulare ha un numero seriale di 32 bit e un numero di 10 cifre (34 bit) che ogni 15 minuti invia in broadcast per cercare una cella a cui potersi attaccare.

## 2G, la seconda generazione, digitale

Il passaggio da digitale ad analogico permette molti vantaggi dal punto di vista di efficienza e dalla capacità di utenti in contemporanea, comportando però una perdita di qualità nel segnale trasmesso e ricevuto (è meno naturale).

**Digital-AMPS** (**D-AMPS**): negli Stati Uniti si continua a usare, almeno inizialmente, AMPS nella sua versione digitale che permette la piena retrocompatibilità di AMPS e il riutilizzo delle frequenze, oltre che altre di nuove, gestite con modulazione TDM. Il flusso dati è compresso. In questo sistema sono i cellulari a gestire l'handoff, con un sistema chiamato **Mobile Assisted Handoff** (**MAHO**), durante i tempi di attesa di trasmissione o ricezione (dato che il canale è in multiplexing a divisione di tempo).

**Global System for Mobile communication** (**GSM**): standard di origine europea per unificare le comunicazioni mobili del continente. Qui ebbe subito una grande diffusione e rapidamente anche nel resto del mondo (in Australia, ad esempio). Il funzionamento di GSM è simile a quello di D-AMPS, con piccole differenze ma abbastanza rilevanti. Si usa anche qui una combinazione di FDM e TDM, i canali (simplex) impiegati però sono molto più ampi (200 KHz rispetto ai 30 KHz per D-AMPS) e questo permette di supportare più del doppio degli utenti (8 rispetto a 3) e un data rate più alto (avendo 270,833 Kbps totali, vengono assegnati 33,854 Kbps ad utente). Sono implementati 124 coppie di canali simplex, ciascuno con 8 slot temporali, con un data rate di 33,854 Kbps (senza overhead per il controllo del flusso e correzione degli errori circa 13 Kbps, comunque superiore a D-AMPS).

La gestione del territorio è fatta con celle di grandezza differente in base all'impiego:

- *macro* (circa 35 Km di raggio);
- *micro* (più piccole, di altezza massima un edificio);
- *pico* (ancora più piccole, per permettere la gestione di più utenti in aree molto dense, come le metropoli);
- *umbrella* (per coprire i buchi di copertura delle altre celle).

Mentre in AMPS e D-AMPS per identificare un utente si utilizzavano un numero seriale e un numero di telefono legato al dispositivo, in GSM si iniziano ad utilizzare le schede **SIM** (**Subscriber Identity Module**), vere schede di memoria di taglie molto basse (nell'ordine dei KB), che al loro interno contenevano (e contengono tutt'ora, dato che vengono ancora utilizzate) due codici:

- **IMSI** (**Internation Mobile Subscriber Identity**), un identificativo per la SIM;
- **Ki**, una chiave di autenticazione.

GSM inizio ad implementare l'autenticazione crittografica a chiave condivisa delle utenze, dal funzionamento qui di seguito spiegato:

1. il dispositivo invia il proprio IMSI all'operatore;
2. l'operatore genera un numero casuale e lo invia al cellulare;
3. il cellulare firma con la Ki il numero e lo spedisce indietro all'operatore;

4. l'operatore firma il numero casuale con la Ki (che dovrebbe essere uguale all'utente, se non ci sono problemi di autenticazione) e lo confronta con il numero firmato ricevuto dal cellulare: se sono uguali, l'utente è autenticato.

## 3G, la terza generazione, digitale

La terza generazione, 3G, decide di staccarsi dal passato e di utilizzare standard completamente diversi da GSM e D-AMPS (non usa né FDM né TDM).

**General Packet Radio Service** (**GPRS**): standard classificato come 2.5G, è un overlay del 2G che permette la gestione del traffico a pacchetti e non a messaggi interi (permette di non utilizzare il GSM per i dati, in quanto riserverebbe un intero canale indipendentemente se utilizzato o meno). Supporta nativamente i protocolli IP e PPP. Alloca dinamicamente i canali Internet e i canali voce (a seconda delle richieste di traffico).

Per supportare l'utilizzo di GPRS, sono stati introdotte varie classi di dispositivi:

- classe **C**: il dispositivo può utilizzare gli standard GPRS o GSM, impostabili manualmente;
- classe **B**: il dispositivo può utilizzare gli standard GPRS o GSM, automaticamente impostati in base all'utilizzo;
- (pseudo) classe **A**: il dispositivo può utilizzare gli standard GPRS o GSM assieme, se supportato dall'operatore tramite il Dual Transfer Mode (DTM);
- classe **A**: il dispositivo piò utilizzare gli standard GPRS o GSM assieme.

**Enhanced Data rates for GSM Evolution** (**EDGE**): conosciuto anche come EGPRS, classificato come *2.75G*. Utilizza sia la modulazione di fase che di frequenza.

**Code Division Multiple Access** (**CDMA**): questo standard permette agli utenti di comunicare contemporaneamente. Ogni utente spedisce il proprio segnale (una sinusoide) quando ne ha bisogno. Il funzionamento è spiegato brevemente:

- dovendo spedire più segnali contemporaneamente, per semplicità, si considera che il picco positivo della sinusoide valga **1** e che il picco negativo valga **-1**. Questo è diverso dalla solita interpretazione in cui il picco negativo vale **0**. Questo ha un impatto molto importante nell'implementazione di CDMA e permette di comprendere se un utente ha inviato un segnale o se non lo ha inviato;
- a ogni segnale (quindi utente) facciamo corrispondere un vettore. I vettori di ogni segnale, spediti contemporaneamente, si possono interpretare come una matrice. Si sta lavorando quindi in uno spazio multidimensionale;
- per fare encoding/decoding dei segnali si utilizzano tecniche di composizione (somma) e proiezione (prodotto scalare) di vettori;
- si creano assi, utilizzando 1 e -1, che siano mutuamente ortogonali (ogni coppia di righe diverse rappresenta vettori ortogonali), utilizzando matrici di Hadamard, in particolare con la costruzione di Sylvester per matrici di ordine *2n*. Una matrice di ordine *n* soddisfa la seguente relazione:

$$H^T H = nI_n$$

$$H_1 = [1]$$

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$H_{2^k} = \begin{bmatrix} H_{2^{k-1}} & H_{2^{k-1}} \\ H_{2^{k-1}} & -H_{2^{k-1}} \end{bmatrix}, 2 \leq k \in \mathbb{N}$$

- viene assunto che tutti gli utenti abbiano segnali con la stessa ampiezza, quindi i cellulari devono aumentare o diminuire la potenza del loro segnale in base alla distanza dal trasmettitore.

CDMA permette quindi, oltre al traffico dati, di far funzionare in maniera più efficiente il traffico voce, infatti, mentre TDM e FDM indipendentemente dalla presenza di segnale o meno riservano il canale per l'utente (sprecandolo), questo standard non ne ha bisogno.

**Wideband CDMA** (**W-CDMA**): conosciuto in Europa con il nome di **Universal Mobile Telecommunication System** (**UMTS**). Utilizza canali da 5 MHz, portando ad un data rate di 384 Kbps.

**CDMA2000**: standard in uso negli Stati Uniti. Utilizza canali più piccoli di W-CDMA, da 1,25 MHz, permettendo di raggiungere un data rate di 144 Kbps.

In ordine temporale sono poi stati sviluppati ulteriori standard, qui di seguito elencati.

**High Speed Downlink Packet Access** (HSDPA): conosciuto come *3.5G*, sfrutta una combinazione di CDMA e QAM.

**High Speed Uplink Packet Access** (HSUPA): conosciuto come *3.75G*, un'evoluzione dell'HSDPA delle origini. Purtroppo, quasi subito soppiantato perché nel frattempo HSDPA era migliorato e aveva superato di gran lunga questo standard.

**Evolved HSDPA** (HSDPA): evoluzione di HSDPA che soppianta HSUPA.

## Oltre il 3G

**High Speed OFDM Packet Access** (HSOPA): conosciuto anche come **Long Term Evolution** (**LTE**) o 4G, sfrutta bande differenti in base all'utilizzo, da 1,25 MHz a 20 MHz. Permette un downlink di 1,2 Gbps e un uplink di 600 Mbps.

# Il livello data link

## Progettazione del livello data link

Capitolo 3.1 della 5° edizione del libro “Reti di calcolatori” di A. S. Tanenbaum

Il livello data link usa i servizi offerti dal livello fisico per inviare e ricevere bit sui canali di comunicazione.

Alcune sue funzioni sono:

- fornire un’**interfaccia** per il livello di rete;
- gestire gli errori di trasmissione;
- regolare il flusso dati in modo che i dispositivi lenti non vengano inondati di messaggi da parte di dispositivi veloci.

Prende i pacchetti provenienti dal livello di rete e li incapsula in **frame** contenenti un’intestazione (**frame header**), il pacchetto ricevuto (**payload field**) e una sequenza di chiusura (**frame trailer**).

I servizi offerti al livello di rete possono essere:

1. senza conferma (**unacknowledged**) senza connessione (**connection-less**): la macchina sorgente invia frame indipendenti alla macchina destinazione, senza che questa debba dare conferma dell’avvenuta ricezione (utilizzato quando la frequenza degli errori è molto bassa ma anche per il traffico real-time dove il ritardo è un problema peggiore della trasmissione di dati sbagliati);
2. con conferma (**acknowledged**) senza connessione (**connection-less**): ciascun frame è inviato individualmente e per ognuno viene inviata una conferma di ricezione entro uno specifico intervallo di tempo (se non viene inviato, il frame viene rispedito);
3. con conferma (**acknowledged**) orientato alla connessione (**connection-oriented**): viene instaurata una connessione fra sorgente e destinazione, ciascun frame è quindi inviato individualmente e per ognuno viene inviata una conferma di ricezione.

Il servizio svolto a questo livello dipende molto da quello fornitogli dal livello fisico: se il mezzo trasmissivo è poco affidabile o soggetto a rumore, il messaggio potrebbe non essere corretto. È compito di questo livello accorgersene e agire in modo da risolvere gli errori.

Ogni frame spedito contiene nel suo header un campo checksum calcolato a partire dal frame, successivamente ricalcolato a destinazione: se quello calcolato a destinazione è diverso da quello ricevuto il livello data link riesce ad accorgersi che ci sono stati degli errori nella ricezione e deve quindi agire.

Un’operazione preliminare da svolgere è riconoscere l’inizio e la fine di ciascun frame. Si potrebbe usare un timer che permette di contare e analizzare i frame in base alla loro lunghezza, ma data l’alta velocità dei canali trasmissivi questo diventa troppo complesso, oltre che troppo a rischio di errori dovuti a una errata sincronizzazione. Per questo motivo, vengono impiegati altri metodi:

1. **conteggio dei byte**: nell’header è indicata la dimensione in byte del frame che segue (con i problemi che ne conseguono, infatti se questo valore della dimensione viene alterato nella trasmissione, il frame viene letto erroneamente);
2. **flag byte con byte stuffing**: i frame sono delimitati da un **flag byte** che permette sia la sincronizzazione del ricevitore che il riconoscimento dei frame. Se casualmente il flag byte è

presente all'interno del frame come byte dato, esso viene preceduto da un byte di **escape** (permette di far interpretare il byte come parte del payload e non come flag byte). Se anche il byte di escape è presente all'interno dei dati come byte dato, lo si fa precedere da un altro byte di escape (e così via);

3. **flag bit con bit stuffing**: metodo sviluppato per il protocollo **HDLC**, delimita ogni frame con byte 01111110. Ogni qual volta che all'interno dei dati appare una sequenza di cinque bit a 1 consecutivi, viene inserito un bit a 0 (svolge il lavoro del byte di escape). Il ricevente quando interpreterà il payload saprà che i bit 0 dopo i cinque bit 1 andranno rimossi per interpretare correttamente il messaggio;
4. **violazioni della codifica del livello fisico**: si delimitano i frame con segnali che non possono essere presenti nei dati perché è stata introdotta una ridondanza. Questo permette di non dover aggiungere dati ulteriori.

# Rilevazione e correzione degli errori

Capitolo 3.2 della 5° edizione del libro "Reti di calcolatori" di A. S. Tanenbaum

**Codici a rilevazione d'errore**: informazioni ridondanti aggiunte ai dati in numero sufficiente da permettere al destinatario di dedurre se c'è stato un errore. Vengono impiegati in canali altamente affidabili o con bassa probabilità di errore (per esempio le fibre ottiche, dove rilevare un errore e farsi rispedire i dati è più efficiente che correggerli).

**Codici a correzione d'errore**: informazioni ridondanti aggiunte ai dati in numero sufficiente da permettere al destinatario di dedurre il messaggio originariamente trasmesso. Vengono impiegati in canali con alta probabilità di errore (per esempio le reti wireless, altamente soggette a rumori esterni).

Il processo di creazione dei messaggi da inviare con i codici di protezione (siano essi di correzione o di rilevazione) si chiama **encoding**. Il processo inverso invece, ovvero quando avviene il controllo degli errori e l'ottenimento dei dati inviati dal trasmittente si chiama **decoding**.

**Distanza di Hamming**: indicata con *d*, corrisponde al numero di bit diversi fra due sequenze di bit. È ottenuta mediante la somma dei bit a 1 dello XOR bit a bit fra le due sequenze. Se due sequenze di bit sono distanti *d* l'una dall'altra saranno necessari *d* errori sui singoli bit per ottenere la medesima sequenza.

## Tipologie di errori

Gli errori possibili sono diversi e in base alla loro tipologia vengono trattati diversamente.

**Errori isolati**: errori singoli (sul singolo bit) causati tipicamente da del rumore termico.

**Errori a burst**: catena consecutiva di errori, basata su processi fisici come gravi dissolvenze su di un canale wireless o un'interferenza elettrica. Per risolverli si procede tramite **interleaving** (metodo della matrice invertita), che cambia l'ordine in cui i bit dei frame vengono trasmessi facendo in modo che se avvengono degli errori questi siano spalmati su più frame e quindi più facilmente correggibili.

**Canale a cancellazione**: avviene quando un errore ha cambiato i dati e li ha fatti risultare completamente diversi dal loro valore atteso. In questo caso il dato viene dichiarato perso. Questa tipologia di errore è "migliore" perché fa subito notare che c'è stato un errore, a differenza delle altre due che nonostante alterino i bit, comunque i loro valori rientrano fra quelli attendibili e quindi tramite i codici di protezione devono venire rilevati.

## Rilevazione degli errori

$$\#massimo\ di\ errori\ trattabili = \min(d)\ \ fra\ messaggi\ corretti - 1$$

**Bit di parità**: è il più semplice metodo di rilevazione dell'errore. Ogni certo numero di bit ne aggiungo uno che indica se la somma dei precedenti bit è pari (0) o dispari (1). Ha **potenza 1**, ovvero riesce a rilevare al massimo errori su di un singolo bit.

**Repetition code $R_n$**: in una sequenza di bit ripeto *n* volte ciascun bit. Con questa tipologia di codifica ottengo $d = n$.

## Correzione di errori

La correzione di errori si potrebbe fare con i Repetition code. Possono intervenire su *k* errori e il messaggio ha distanza *k* da un messaggio possibile:

$$k = \begin{cases} \frac{n}{2} - 1, n\ pari \\ \frac{n-1}{2}, n\ dispari \end{cases}$$

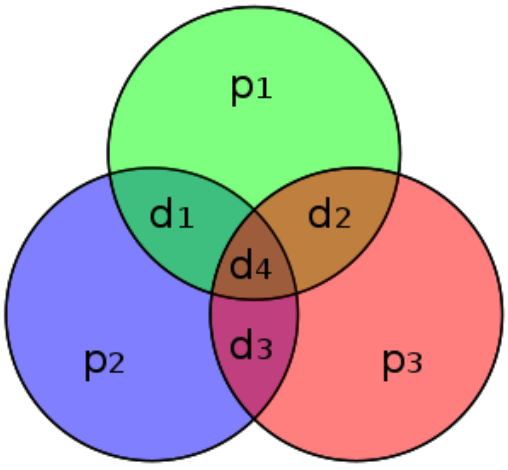
Questi codici però non vengono utilizzati in pratica per un semplice motivo: il data rate diventerebbe $\frac{1}{n}$, mentre vengono utilizzati bit di protezione direttamente all'interno dei dati trasmessi. Alcuni codici utilizzati in pratica sono il **codice di Luhn** (impiegato ad esempio nelle carte di credito) e i **codici di Hamming** (definiti codici lineari perché per essere calcolati vengono impiegate solo operazioni lineari, ovvero somme e prodotti).

Un frame formato da *n* bit contiene sempre *m* bit dati e *r* bit di controllo degli errori, con $n = m + r$.

I codici che vengono spiegati di seguito sono:

- **a blocco**: gli *r* bit di controllo sono calcolati unicamente in funzione degli *m* bit associati;
- **sistematici**: gli *m* bit di dati sono trasmessi assieme a quelli di controllo;
- **lineari**: gli *r* bit di controllo sono una funzione lineare degli *m* bit (come ad esempio XOR o somma in modulo 2.

I codici si indicano con **nome(n, m)**. Una sequenza di *n* bit viene chiamata **codeword**. Il **code rate** è la frazione di parole che trasportano dati pari a $\frac{m}{n}$.




Hamming(7, 4): $p_1$ è il bit di parità per $d_1$, $d_2$ e $d_4$; $p_2$ è il bit di parità per $d_1$, $d_3$ e $d_4$; $p_3$ è il bit di parità per $d_2$, $d_3$ e $d_4$.

**Codice di Hamming - Hamming(x, y)**: codice che codifica *y* bit usandone *x*. Un esempio è il codice Hamming(7, 4), qui a sinistra rappresentato graficamente.

**Codice di Reed-Solomon – RS(x, y)**: è nato come evoluzione del codice di Hamming ed è basato sull'aritmetica polinomiale (con resti polinomiali). Come per Hamming(x, y), vengono codificate *y* parole dati usandone *x*. La potenza di correzione massima è $\frac{x-y}{2}$. L'unità di base su cui fare correzione non è il singolo bit, ma blocchi (genericamente byte). In caso di errori dovuti a cancellazione (per esempio, un graffio su un CD), possono essere corretti fino a $x - y$ errori (il doppio). Può gestire contemporaneamente errori isolati e errori dovuti

a cancellazione per un massimo di $2 \cdot errori + cancellazioni < x - y$. Vengono impiegati in CD, DVD, Blu-Ray, DSL, WiMax, TV digitale (in questo caso usato RS(204, 188)).

**Codice sparso LDPC (Low-Density Parity Check)**: è un tipo di codice che si avvicina al limite di Shannon (teoricamente potrebbe far mantenere un data rate abbastanza alto facendo anche correzione degli errori). È purtroppo un problema di tipo NP-Completo, quindi i tempi di calcolo sono molto lunghi.

**Codici CRC (Cyclic Redundancy Check)**: è un tipo di codice simile a Reed-Solomon (sfrutta l'aritmetica polinomiale) ma è molto più semplice da implementare e anche per questo motivo non permette la correzione di errori ma solo la rilevazione (crescente, in base alla potenza impiegata).

L'aritmetica polinomiale utilizzata è quella in base 2 (corrispondente alle operazioni in GF(2)[]). In questa aritmetica si fa corrispondere una stringa di *m* bit in un polinomio di grado *m*-1 in cui ogni bit è coefficiente del termine corrispondente al proprio grado. L'addizione e la sottrazione corrispondono allo XOR.

La sorgente e la destinazione si accordano su un polinomio ***G(x)*** con cui dividere il messaggio da trasmettere ***M(x)*** per trovare il resto ***R(x)***. *G(x)* deve avere il termine di grado *m*-1 e 0 con coefficiente 1 (primo e ultimo bit di *G(x)* devono essere 1).

**Encoding**: Se *M(x)* < *G(x)* (in generale vero), si moltiplica M(x) per $x^r$, dove *r* = *deg(G(x))* (che equivale a fare lo shift a sinistra del messaggio *M(x)* di *r* posizioni, ovvero aggiungere *r* bit a 0 a destra di *M(x)*). Si procede come segue:

1. si divide ($x^r \cdot M(x)$) per *G(x)*, ottenendo un resto *R(x)*;
2. si sottrae *R(x)* a ($x^r \cdot M(x)$), ottenendo il polinomio *T(x)* che è certamente divisibile per *G(x)* (ovvero *T(x)* mod *G(x)* = 0).

Viene trasmesso *T(x)* nel canale di trasmissione.

**Decoding**: Ottenuto il polinomio *T(x)*, si procede come segue:

1. si effettua la divisione *T(x)* per $x^r$ e quello che si ottiene è uno shift della sequenza di bit di *r* posizioni verso destra;
2. si divide per il polinomio *G(x)* accordato il polinomio *T(x)* e si ottiene un certo resto *R(x)*. Se questo è 0, la trasmissione è avvenuta correttamente e si ha a disposizione il messaggio *M(x)*, altrimenti c'è stato un errore (o più).

Questo semplice ma potente sistema funziona, e nella pratica:

- in caso di bit invertiti nel messaggio ricevuto, questa situazione è equivalente ad avere *T(x)*+*E(x)*, dove *E(x)* è detto il polinomio di errore. Questa divisione avrà un certo resto *R(x)* diverso da 0, perché sappiamo che *T(x)* mod *G(x)* = 0. Questo resto sarà *T(x)*+*E(x)* mod *G(x)* = *E(x)*. È quindi vero che:
$$errore\ non\ trovato \leftrightarrow E(x)\ mod\ G(x) = 0$$
- in caso di errori singoli, *G(x)* viene scelto con due o più fattori, in modo da non riuscire a dividere mai $E(x) = x^i$, ∀i;
- in caso di errori doppi, *G(x)* viene scelto non multiplo di *x*, in modo da non riuscire a dividere $x^k+1$, *k* = *i* – *j*, *i* > *j*, dato che l'errore è $E(x) = x^i + x^j$, ∀i, ∀j;
- in caso di errori di numero dispari, *G(x)* = *x* + 1 è sufficiente per determinare gli *E(x)*;
- in caso di errori burst, *G(x)* viene scelto di grado *deg(G(x))* > *j*, con il termine di grado 0 con coefficiente 1, dato che gli errori sono di tipo $E(x) = x^i(x^j + \ldots + 1)$ e i burst di lunghezza *j* + 1.

In generale, la probabilità di errore è $1/2^{deg(G(x))}$.

Alcuni tipi di G(x):

- CRC-1, $G(x) = x + 1$ (corrisponde a parity bit 1);
- CRC-5, $G(x) = x^5 + x^2 + 1$;
- CRC-16, $G(x) = x^{16} + x^{15} + x^2 + 1$ (impiegato nel USB);
- CRC-16 CCITT, $G(x) = x^{16} + x^{12} + x^2 + 1$ (impiegato nel Bluetooth);
- CRC-32, $G(x) = x^{32} + \dots$ (corregge burst fino a 32 errori, tutti gli errori dispari, ed è impiegato nei modem, nei file .zip, nella fibra ottica).

La differenza principale con Reed-Solomon è che CRC usa GF(2), RS usa GF($2^n$).

# Protocolli data link elementari

Capitolo 3.3 della 5° edizione del libro "Reti di calcolatori" di A. S. Tanenbaum

**Protocollo**: regola tramite la quale si gestiscono i flussi dati.

Il controllo del flusso (**flow control**) è necessario perché se vengono inviati dal mittente più dati ti quanti possa gestirne il ricevente, i dati non gestiti vanno persi.

## Protocollo stop-and-wait (per reti dinamiche)

Tipo di protocollo in cui il mittente e il ricevente si scambiano informazioni, indipendentemente da chi sia il mittente e il ricevente.

Viene inviato un frame e si aspetta (**stop-and-wait**) che l'arrivo di un frame di conferma di avvenuta ricezione (**acknowledgement**) prima di trasmettere il successivo. Con questo tipo di implementazione, bastano dei canali half-duplex per poter trasmettere. È ovviamente un tipo di protocollo intrinsecamente lento, in quanto il mittente è spesso in attesa.

Se il frame di acknowledgement (ACK) non arriva entro un certo periodo di tempo, scatta un timeout. Quando il timeout scatta, il frame viene inviato nuovamente: avviene una duplicazione (che non va bene) e quindi i frame devono essere dotati di controllo dell'errore e di controllo di flusso (che deve tener conto se è un pacchetto successivo al suo precedente o no (tanto vengono inviati in ordine, quindi questa informazione basta e avanza).

Famiglie di protocolli di questo tipo sono **PAR** (**Positive Acknowledgement with Retransmission**) e **ARQ** (**Automatic Repeat reQuest**).

Nei canali half-duplex ci sono solo un mittente e un solo destinatario (ovvero c'è un verso di trasmissione). In generale però si è in presenza di canali full-duplex, in cui c'è trasmissione in entrambi i versi. Un grosso problema dello stop-and-wait è l'overhead: se per ogni pacchetto inviato, ne devo inviare uno di conferma di ricezione, occupo due volte il canale anziché una. Una tecnica per ovviare a questo problema è il **piggybacking**, ovvero l'invio della conferma "a cavallo" del primo frame che viene inviato dopo aver ricevuto un frame, spedendo un frame con dei dati effettivi assieme all'ACK, e non un frame completamente dedicato all'ACK. Questa tecnica non è però utilizzabile in ogni situazione perché non si può far aspettare il mittente in eterno (prima o poi scade il timeout) e quindi è utilizzabile solamente in canali in cui c'è una comunicazione abbastanza equilibrata.

Si sta comunque sottoutilizzando il canale, perché il prodotto di **bandwidth** per il **roundtrip-delay** è troppo grande. Se un canale ha capacità ***C*** bps, il frame di dimensione ***S*** bit e il tempo di roundtrip

***R*** s, l'utilizzo della linea con protocolli con ACK è: $S/(S + C \cdot R)$. Se $S < C \cdot R$, l'efficienza del canale è < 50%.

Quello che si vorrebbe fare è invece spedire più frame contemporaneamente, e poi controllare man mano di quali ho ricevuto l'ACK e di quali no, ed eventualmente rispedire i pacchetti di cui non ho ricevuto ACK. Questa tecnica è chiamata **pipelining** ed è sfruttata nel prossimo tipo di protocollo.

# Protocolli a finestra scorrevole

Capitolo 3.4 della 5° edizione del libro "Reti di calcolatori" di A. S. Tanenbaum

Nei protocolli che sfruttano la tecnica del pipelining vengono inviati contemporaneamente più frame, tipicamente potenze di 2 (per motivi implementativi). Il numero di frame che può essere inviato contemporaneamente corrisponde alla taglia della **finestra di invio**, presente nel mittente; il numero di frame che può essere ricevuto contemporaneamente corrisponde invece alla taglia della **finestra di ricezione**, presente nel destinatario. Finestra di invio e ricezione non devono essere necessariamente uguali. Il numero di sequenza nella finestra rappresenta i frame che sono già stati trasmessi correttamente o che sono in transito, ma di cui non si è ricevuto l'ACK. Quando arriva un nuovo pacchetto dal livello di rete da trasmettere, si incrementa il limite superiore della finestra; quando si riceve un ACK dal destinatario si incrementa il limite inferiore della finestra (risultato ottenuto: la finestra si sposta man mano che i frame vengono trasmessi). La finestra deve poter tenere in un buffer i frame inviati fino a che questi non ricevono il loro ACK corrispondente, per poter eventualmente rinviarli all'occorrenza. Una finestra di taglia *n* deve poter tenere in un buffer *n* frame inviati.

## Protocollo Go-Back-*N*

È un tipo di protocollo con finestra scorrevole in cui la taglia della finestra di invio è *N* e la taglia della finestra di ricezione è 1. Funziona bene nel caso il canale abbia pochi errori e in cui il prodotto $C \cdot R$ è alto. Bisogna tenere conto del fatto che:

- ci sono *N* timer che scattano se non ricevono l'ACK in tempo;
- c'è un buffer grande *N* contenente i frame di cui non ho ricevuto l'ACK;
- se scatta un timeout, tutti gli *N* frame vengono rispediti.

## Protocollo Selective Repeat

È una versione di protocollo con finestra scorrevole, che da parte di chi trasmette non ha differenze con Go-Back-*N*, mentre chi riceve ha anch'esso un buffer di dimensione *N*. Il principio di funzionamento è il medesimo: vengono spediti gli *N* frame in contemporanea, vengono attivati i timeout da parte del mittente e il ricevente è pronto ad accoglierli assieme. La possibilità che avvengano errori non è sparita: se qualche frame non arriva a destinazione o arriva fuori flusso o il campo checksum che viene calcolato è errato, il ricevitore può spedire al mittente un messaggio NAK (Not AcKnowledgement, l'opposto dell'ACK), in cui rende nota al mittente l'anomalia riscontrata. Anche i NAK, come gli ACK, sono attivabili tramite lo scattare di un timeout. Sono un'ottimizzazione, perché permettono di ottimizzare il flusso dei dati.

I protocolli a finestra scorrevole funzionano bene se l'apertura della finestra è al più uguale alla metà della dimensione del buffer, altrimenti si rischia la perdita di sincronizzazione.

## Protocollo HDLC (High-level Data Link Control)

È un protocollo ideato e utilizzato in pratica, a differenza dei precedenti che erano solo semplici esempi. È stato impiegato in tecnologie come fax, vecchi ATM e in alcuni modem. Di questo protocollo esistono delle varianti, chiamate LAP – LAPB (entrambe Link Access Procedure).

I frame, come si era già visto, sono delimitati tramite bit stuffing. Di tutti i campi del frame, sono interessanti in particolare il campo **Data** del frame, che contiene il **payload** (il carico, ovvero i dati); il campo **Checksum**, che contiene il codice di controllo del frame (calcolato con CRC); il campo **Address**, che contiene l'indirizzo del destinatario del pacchetto; infine, il campo **Control**, che serve a riconoscere se il pacchetto è di tipo "Information", "Supervision" o "Unnumbered".

Il flow control è gestito tramite sliding window con grandezza massima 3 bit (ovvero 8 frame paralleli, ampiezza massima della finestra 4).

Un frame di tipo "Information", nei primi 8 bit, contiene i seguenti campi:

| **0** | **Seq** | **P/F** | **Next** |
|---|---|---|---|

Descrizione dei campi:

- **0** (1 bit): indica che si tratta di un frame "Information";
- **Seq** (3 bit): indica il numero di sequenza del frame per gestire il flusso;
- **P/F** (1 bit): indica la richiesta verso il ricevente (Poll: inizia la trasmissione, Final: chiudi la trasmissione);
- **Next** (3 bit): indica se il frame è ACK (in piggyback).

Un frame di tipo "Supervision", nei primi 8 bit, contiene i seguenti campi:

| **0** | **1** | **Type** | **P/F** | **Next** |
|---|---|---|---|---|

Descrizione dei campi:

- **0** (1 bit), **1** (1 bit): indicano che si tratta di un frame "Supervision";
- **Type** (2 bit): indica il tipo di controllo che viene effettuato;
- **P/F** (1 bit): indica la richiesta verso il ricevente (Poll: inizia la trasmissione, Final: chiudi la trasmissione);
- **Next** (3 bit): indica se il frame è ACK (in piggyback).

Di **Type** sono presenti 4 tipi (dati i 2 bit, ovviamente):

- **Type 0**: RECEIVE READY è un ACK usato quando il flusso è sbilanciato e non si riesce a mandare un ACK in piggyback (viene impostato un timer);
- **Type 1**: REJECT è un NAK generalizzato che chiede la ritrasmissione di tutto a partire dal frame indicato nel campo **Next**;
- **Type 2**: RECEIVE NOT READY è un segnale inviato in casi di congestione che indica di non spedire più nulla fino a che non viene inviato il prossimo ACK;
- **Type 3**: SELECTIVE REJECT è un NAK che viene usato per richiedere la ritrasmissione di un singolo frame indicato nel campo **Next**.

Un frame di tipo "Unnumbered", nei primi 8 bit, contiene i seguenti campi:

| **1** | **1** | **Type** | **P/F** | **Modifier** |
|---|---|---|---|---|

Descrizione dei campi:

- **1** (1 bit), **1** (1 bit): indicano che si tratta di un frame “Unnumbered”;
- **Type** (2 bit): indica la richiesta da mettere in atto;
- **P/F** (1 bit): indica la richiesta verso il ricevente (Poll: inizia la trasmissione, Final: chiudi la trasmissione);
- **Modifier** (3 bit).

Di comandi in **Type** sono presenti molti tipi, vediamo i principali:

- **DISC** (DISConnect): richiesta la disconnessione di una macchina in maniera definitiva;
- **SNRM** (Set Normal Response Mode): richiesta la connessione di una nuova macchina, in modalità asimmetrica (il nuovo entrato è meno importante, retaggio storico dai vecchi tipi di rete con mainframe e terminali stupidi);
- **SABM** (Set Asynchronous Balanced Mode): richiesta la connessione di una nuova macchina, in modalità simmetrica (il nuovo entrato è importante come gli altri, opposto al comando SNRM);
- **FRMR** (FRaMe Reject): segnalazione che è arrivato un frame con un numero di sequenza non atteso.

Anche per i frame di tipo “Unnumbered” possono esistere errori, quindi anche per questi c’è la necessità di conferme di ricezione tramite comandi dedicati, detti **UA** (**Unnumbered Acknowledgement**).

## Protocollo PPP (Point-to-Point Protocol)

PPP è un altro protocollo dello strato data link utilizzato in pratica, per collegare dispositivi in Internet.

PPP è definito meta-protocollo, perché i veri protocolli in realtà sono le due parti in cui è suddiviso:

- **LCP** (**Link Control Protocol**), che controlla il flusso per attivare le connessioni, i test, le negoziazioni e la chiusura delle connessioni;
- **NCP** (**Network Control Protocol**), che gestisce il flusso di dati con il livello di rete sovrastante.

È stato ideato per essere il più simile possibile a HDLC. PPP delimita i suoi frame con byte stuffing, non con bit stuffing come HDLC.

Per retrocompatibilità sono stati mantenuti i campi Address, ma non viene usato, e quindi ha il valore fisso 11111111, Control, anch’esso non utilizzato anche se sarebbe necessario e quindi ha il valore fisso 00000011. Dunque, PPP non numera i pacchetti e non usa gli ACK per creare una comunicazione affidabile.

Nel campo Protocol si specifica a che protocollo si riferisce il frame inviato (LCP o NCP, ecco perché è definito meta-protocollo). I frame del protocollo LCP iniziano col bit 1, gli altri con il bit 0.

Nel campo Payload sono contenuti i dati: la dimensione di questo campo può variare, in base al protocollo utilizzato.

Nel campo Checksum è contenuto il codice di rilevazione degli errori, calcolato con CRC.

I tipi di frame **LCP** sono i seguenti:

- Configurazione (4):

1. **Configure-request**, dal mittente al ricevente, propone azioni per la configurazione della linea.
2. **Configure-ack**, dal ricevente al mittente, ACK per frame di tipo **Configure-request** (ecco che la mancanza di ACK in PPP viene ripristinata in LCP).
3. **Configure-nak**, dal ricevente al mittente, NAK per frame di tipo **Configure-request** in cui si richiedono configurazioni diverse.
4. **Configure-reject**, dal ricevente al mittente, per frame di tipo **Configure-request** in cui si rifiuta completamente la possibile configurazione.

Con alcune opzioni è possibile impostare il campo per l'error detection (2 o 4 byte), la lunghezza del campo Protocol (1 o 2 byte), la lunghezza del campo Payload (1500 byte di default), la rimozione dei campi Address e Control (che tanto hanno valori fissi), risparmiando 2 byte a frame.

- Terminazione (2):
  1. **Terminate-request**, dal mittente al ricevente, chiusura della comunicazione.
  2. **Terminate-ack**, dal ricevente al mittente, ACK per frame di tipo **Terminate-request**.
- Rifiuto (2):
  1. **Code-reject**, dal ricevente al mittente, comunicazione di mal comprensione del messaggio.
  2. **Protocol-reject**, dal ricevente al mittente, comunicazione di impossibilità di comprendere il protocollo utilizzato o di non supportarlo.
- Echo (2):
  1. **Echo-request**, dal mittente al ricevente, richiesta di rinvio del precedente frame.
  2. **Echo-reply**, dal ricevente al mittente, rinvio del pacchetto richiesto.
- Test (1):
  1. **Discard-requset**, dal mittente al ricevente, comunicazione di test della linea e richiesta di non rispondere.

## PPP e le ADSL

All'interno di una rete è possibile decidere la dimensione di un frame, impostando il valore **MTU** (**Maximum Transmission Unit**). Esiste comunque una dimensione prefissata dell'MTU ed è decisa nello standard di PPP.

Si potrebbe tenere un valore alto per l'MTU e si avrebbero quindi frame più grandi: si otterrebbe come risultato meno overhead e quindi più banda disponibile (andrebbe bene in canali affidabili).

Si potrebbe tenere un valore basso per l'MTU e si avrebbero quindi frame più piccoli: si otterrebbe come risultato più overhead e quindi meno banda disponibile (andrebbe bene in canali poco affidabili).

Queste due affermazioni però non sono sempre vere, poiché le reti non sono isolate, devono interfacciarsi con altre e quindi la modifica dell'MTU potrebbe portare benefici solo nella propria rete, ma al di fuori di questa le cose si complicano parecchio.

Per collegare le varie reti, PPP viene istanziato in due varianti:

- **PPPoA** (**PPP over ATM**);
- **PPPoE** (**PPP over Ethernet**).

Una versione (molto semplificata) dei passi che avvengono quando un computer si connette a Internet:

1. PC → Router;
2. Router → Modem;
3. Modem → DSLAM (DSL Access Multiplexer);
4. DSLAM → Provider;
5. Provider → Internet.

Per ogni passo, c'è un cambio di protocollo. Ad esempio, quando si passa al provider non si è arrivati in fondo: possono esserci più reti cablate assieme, ma di tipo diverso. Solitamente, il primo tratto è ATM, il secondo è Ethernet. ATM (Asynchronous Transfer Mode) è un'evoluzione di HDLC, usa TDM e gestisce il flusso con una sliding window (con apertura 16), rilevazione degli errori con CRC-8 e indirizzamento di due tipi gerarchici:

- cammini (path) (nei modem, identificati con il campo **Virtual Path Identifier**, **VPI**);
- canali (channels) (nei modem, identificati con il campo **Virtual Channel Identifier**, **VCI**).

ATM è connection-oriented, ecco il perché dei Virtual Channel.

Una connessione ADSL inizia nel seguente modo:

- il computer/modem invia un frame PPPoE (**Active Discovery Initiation**), col suo indirizzo fisico (MAC);
- i servizi ADSL disponibili rispondono con un **PADO** (**PPPoE Active Discovery Offer**);
- il computer/modem risponde con un **PADR** (**PPPoE Active Discovery Request**) in cui segnala il servizio ADSL che ha scelto;
- il servizio fa l'ACK usando un frame **PADS** (**PPPoE Active Discovery Session-confirmation**);
- la connessione è terminata da un frame **PADT** (**PPPoE Active Discovery Termination**).

# Il sottolivello MAC

## Protocolli ad accesso multiplo

Capitolo 4.2 della 5° edizione del libro “Reti di calcolatori” di A. S. Tanenbaum

Questi protocolli sono diversi da quelli Point-to-Point in cui uno parla e uno ascolta, come PPP, perché ora più persone vogliono utilizzare un canale condiviso. Si parla in questo caso di **contention**.

Bisogna fare delle assunzioni prima di iniziare a parlarne.

**Station model**: sono le entità che trasmettono. Dopo che hanno iniziato la trasmissione di un frame, non fanno altro fino a che non è trasmesso.

**Single channel**: c’è un unico canale, condiviso, per tutti.

**Collision**: se due frame si sovrappongono (vengono inviati allo stesso istante), avviene una collisione e diventano inutilizzabili (non si può utilizzare una tecnica simile a CDMA in questo caso).

**Tempo continuo**: non c’è un orologio centrale, quindi non c’è sincronizzazione.

**Tempo discreto** (**slotted**): ci sono più intervalli di tempo, quindi c’è sincronizzazione.

**Senso del canale** (**carrier sense**): le stazioni possono sapere se il canale è in uso.

**Nessun senso del canale** (**no carrier sense**): le stazioni non possono sapere se il canale è in uso.

### ALOHA

Protocollo inventato alle isole Hawaii negli anni ’70 dall’idea di un ricercatore dell’Università delle Hawaii, Norman Abramson. L’idea è nata perché doveva essere realizzato un modo per poter trasmettere su un’unica frequenza più stazioni radio. L’idea era molto semplice, ma molto efficace: lo sfruttamento del caso: ogni volta che una stazione deve trasmettere, lancia un dado che segna il tempo da aspettare prima di provare a trasmettere. Quando è passato il tempo, la stazione prova a trasmette: se ci riesce, continua fino a che non finisce; se non riesce (avviene una collisione), rilancia il dado e ripete il procedimento.

Di ALOHA esistono due implementazioni: **puro** e **slotted**.

### ALOHA Puro

L’idea alla base dell’ALOHA puro è quella sopra descritta. Ogni stazione trasmette quando ha necessità, se riesce bene, se non riesce deve innanzitutto venire a conoscenza di questo fatto, poi deve aspettare un tempo casuale prima di ritrasmettere. Il tempo deve essere casuale altrimenti vengono generati cicli di attesa infiniti. Questo tipo di sistema si dice **a contesa**.

La capacità di trasporto dei sistemi ALOHA è massima quando tutti i frame hanno la stessa dimensione.

Se due frame collidono, i checksum calcolati risultano sballati e quindi vengono scartati.

Questo caos riesce sorprendentemente ad avere dei “buoni” risultati. L’efficienza di un canale gestito con ALOHA puro è infatti circa il 18%, calcolato utilizzando la distribuzione di Poisson:

$$\Pr[k] = \frac{G^k e^{-G}}{k!}$$

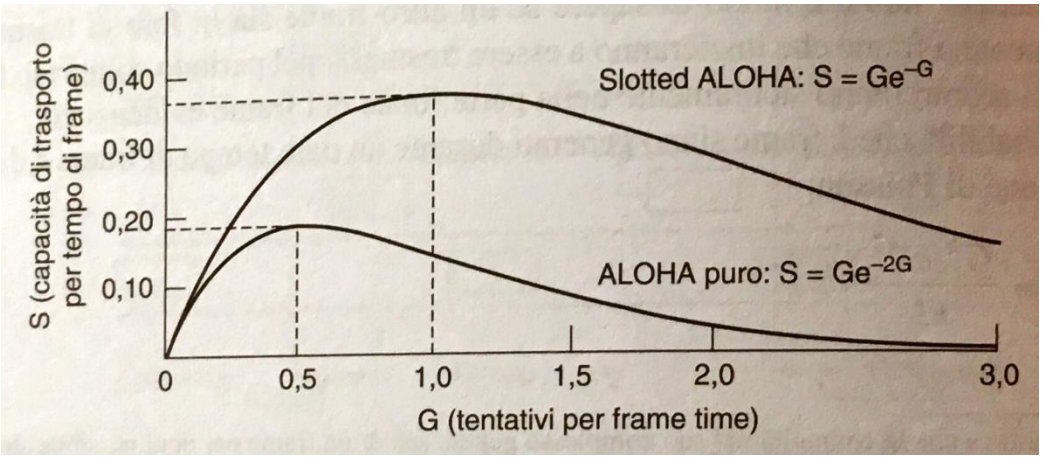
$\Pr[k]$ è la probabilità che vengano generati $k$ frame durante un dato tempo di frame (lunghezza del frame diviso capacità del collegamento), $G$ è la media di frame per tempo di frame.

Il 18% è poco, è vero, ma è pur vero che non dipende in nessun modo dal numero di stazioni (e questo è un grande vantaggio).

## Slotted ALOHA

Dopo l'invenzione di ALOHA Puro, viene pubblicato un metodo per duplicare la capacità del metodo originale, semplicemente dividendo il tempo in intervalli, detti **slot**. La sincronizzazione avviene attraverso una stazione che a ogni inizio intervallo emette un segnale che tutte le altre stazioni ricevono, come un metronomo.

L'efficienza di un canale gestito con Slotted ALOHA riesce a raggiungere il 36%, che è già un buon risultato (ci stiamo pur sempre basando sul caso).




*La capacità di trasporto in funzione del traffico per un sistema ALOHA.*

ALOHA Puro e Slotted ALOHA riescono a raggiungere rispettivamente circa il 18% e il 36% dell'efficienza solo basandosi sul caso. Non è abbastanza ovviamente, ma per fortuna è possibile migliorare la situazione. È stato introdotto il concetto di **carrier sense**: né il primo, né il secondo per il momento ne sono dotati, ecco perché vengono ideati sistemi che invece lo adottano.

## 1-persistent CSMA

**1-persistent CSMA** (**Carrier Sense Multiple Access**) è il più semplice dei protocolli che hanno il senso del canale. Il funzionamento è semplice:

- ogni qualvolta una stazione debba trasmettere, controlla il canale per vedere se è occupato;
- se il canale è libero, trasmette normalmente;
- se il canale è occupato, si mette in attesa fino alla liberazione del canale. Appena si libera il canale, trasmette (l'**1** di **1-persistent** sta per probabilità 1/1 che la stazione trasmetta appena il canale si libera);
- se avviene una collisione, attende un tempo casuale e poi si rimette in attesa.

Il fatto che la stazione trasmetta sicuramente ogni qualvolta il canale si liberi è il maggior problema di questo protocollo. Infatti, se più stazioni vogliono trasmettere e stanno attendendo che il canale si

liberi, stando in ascolto, appena questo si libera si genera una collisione. Diminuendo la probabilità che la stazione trasmetta ogni volta che il canale si libera migliorerebbe le cose.

In ogni caso, questo protocollo ha prestazioni migliori degli ALOHA (si raggiunge più del 50% di utilizzo del canale), in quanto almeno attendono la fine della trasmissione della stazione in corso prima di provare loro a trasmettere.

## Non-persistent CSMA

**Non-persistent CSMA** è un'altra versione del protocollo CSMA. Il funzionamento è simile a quello del precedente, ma con alcune differenze importanti:

- ogni qualvolta una stazione debba trasmettere, controlla il canale per vedere se è occupato;
- se il canale è libero, trasmette normalmente;
- se il canale è occupato, genera un tempo casuale dopo il quale riprova a controllare il canale.

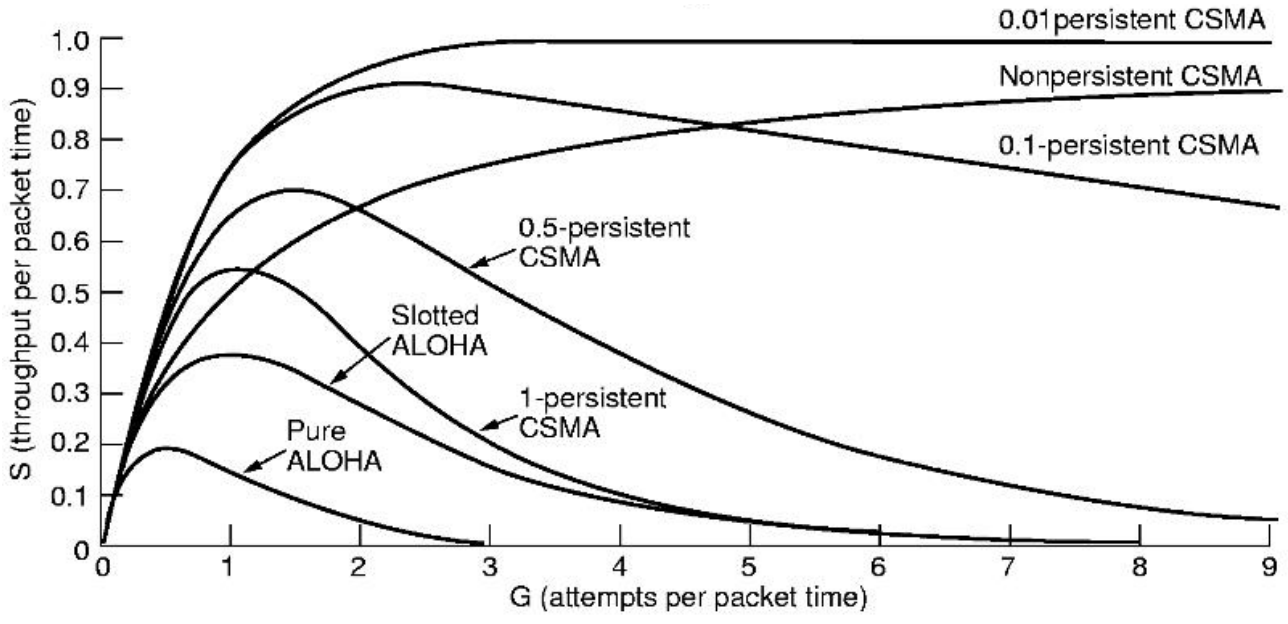
In questo caso, si allungano certamente i tempi di attesa per la trasmissione, ma si è quasi certi di non generare collisioni. L'efficienza del canale migliora parecchio, si è a circa il 90%.

## p-persistent CSMA

L'ultima versione di questo tipo di protocolli è **p-persistent CSMA**. Innanzitutto, questo protocollo si basa sulla divisione temporale del canale, a differenza dei precedenti. Il funzionamento è un po' diverso dai precedenti, ma si basa sugli stessi principi:

- ogni qualvolta una stazione debba trasmettere, controlla il canale per vedere se è occupato;
- se il canale è libero, trasmette con probabilità ***p*** (quindi non è detto che trasmetta). Se non riesce a trasmettere, riprova nello slot successivo
- se il canale è occupato, genera un tempo casuale dopo il quale riprova a controllare il canale.

Con questo tipo di protocollo, con particolari *p*, si può raggiungere un'efficienza del 100%.




*Confronto tra gli utilizzi del canale in funzione del carico per protocolli ad accesso casuale.*

## CSMA/CD

Il protocollo **CSMA/CD** (**CSMA Collision Detection**) cerca di risolvere una grave lacuna dei protocolli precedenti. Tutti e tre infatti, in caso si verifichi una collisione, non sono in grado di saperlo

subito e quindi continuano a trasmettere fino in fondo, scoprendo solo dopo la trasmissione (consumando banda inutilmente) del problema. Questo tipo di protocollo vuole invece aggiungere il controllo del canale da parte delle stazioni anche mentre trasmettono in modo che, se notano che i messaggi inviati sono diversi da quelli che leggono nel canale, vuol dire che si sono verificate delle collisioni e quindi possono interrompere immediatamente la trasmissione, ritrasmettendo più tardi, che tanto ormai i messaggi trasmessi si sono danneggiati.

A questo punto, la rete può assumere tre diversi stati:

- **trasmissione**, in cui una stazione ha il controllo del canale e trasmette;
- **contesa**, in cui si decide il prossimo che deve trasmettere. Il tempo da assegnare sarà due volte il tempo di **round-trip** (andata e ritorno fra le due stazioni più distanti nella rete), in modo che tutti quelli che stanno trasmettendo si accorgano della collisione;
- **idle**, in cui tutte le stazioni sono in attesa di trasmettere.

## CSMA/CA

I protocolli precedenti hanno il problema delle collisioni, sebbene si cerchi di risolvere la situazione con CSMA/CD. Il protocollo **CSMA/CA** (**CSMA Collision Avoidance**) intende eliminare il problema delle collisioni, facendo in modo che nel periodo di **contesa** venga scelto chi ha il controllo esclusivo del canale fino a che non finisce di trasmettere.

Esempio di implementazione del protocollo in cui nella rete ci sono *N* stazioni, con indirizzi da 0 a *N-1*:

- **Basic bitmap**: il periodo di contesa dura esattamente *N* slot (uno per stazione). Per partecipare a un periodo di contesa e prenotare la trasmissione (**protocollo a prenotazione**), una stazione deve essere pronta in anticipo, altrimenti deve aspettare il successivo. In ogni slot *j*, la *j*-esima stazione trasmette un bit a 1 se vuole trasmettere, un bit a 0 se non vuole trasmettere. Trascorsi gli *N* slot, in ordine numerico, le stazioni che avevano segnalato la necessità trasmettono senza essere disturbate. Il periodo di contesa successivo comincia quando l'ultima stazione che si era prenotata per trasmettere ha trasmesso. A pieno carico (tutte le *N* stazioni hanno frame da trasmettere), l'efficienza è $\frac{d}{d+1}$, con la taglia del frame indicata con *d* (ovvero spedisco un solo bit in più). Il problema di questo sistema è che il periodo di contesa aumenta linearmente con l'aumentare di *N*.

## Protocolli a contesa limitata

Fino ad ora si è capito che:

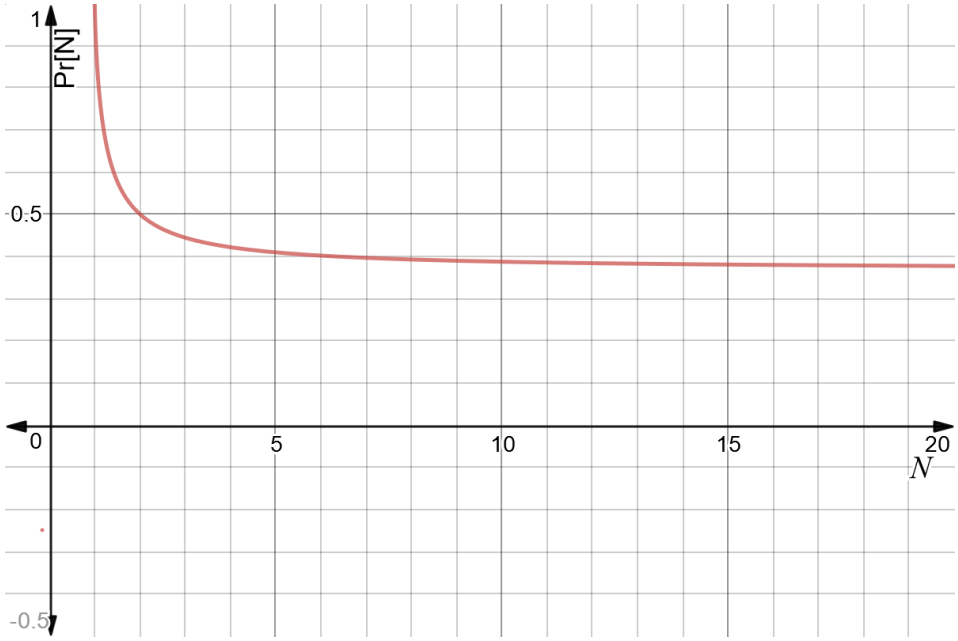
- i protocolli come CSMA/CD e precedenti funzionano bene con carico basso, male invece con carico alto (perché aumentano le collisioni);
- i protocolli che evitano le collisioni funzionano bene invece con carico alto, male invece con carico basso (perché aumenta l'overhead per la contesa).

Sarebbe utile poter prendere il meglio da entrambi.

Innanzitutto, è bene analizzare cosa succederebbe se ai protocolli CSMA aggiungessimo l'informazione sul numero di stazioni (che adesso non hanno):

- il tempo è suddiviso in slot;
- ogni stazione ha probabilità *p* di trasmettere;
- ci sono *N* stazioni nella rete.

La probabilità $p$ che una stazione trasmetta, e che le altre non lo facciano (probabilità 1-$p$) è $N \cdot p \cdot (1-p)^{N-1}$. Il caso migliore è quando $p = \frac{1}{N}$. Quindi, la migliore probabilità è $\Pr[N] = (\frac{N-1}{N})^{N-1}$, rappresentata nel grafico sottostante. Si può notare come la probabilità di trasmettere è buona solo se $N \leq 5$, poi degrada.




Il fatto che *N* debba essere minore di 5 affinché le prestazioni siano buone fa comprendere il motivo per cui si è cercato di realizzare protocolli a contesa limitata, ovvero che limitino la competizione in moda da evitare le collisioni. L'obiettivo è realizzare un sistema per cui:

- con carico alto, poche stazioni si contendono il canale;
- con carico basso, molte stazioni si contendono il canale.

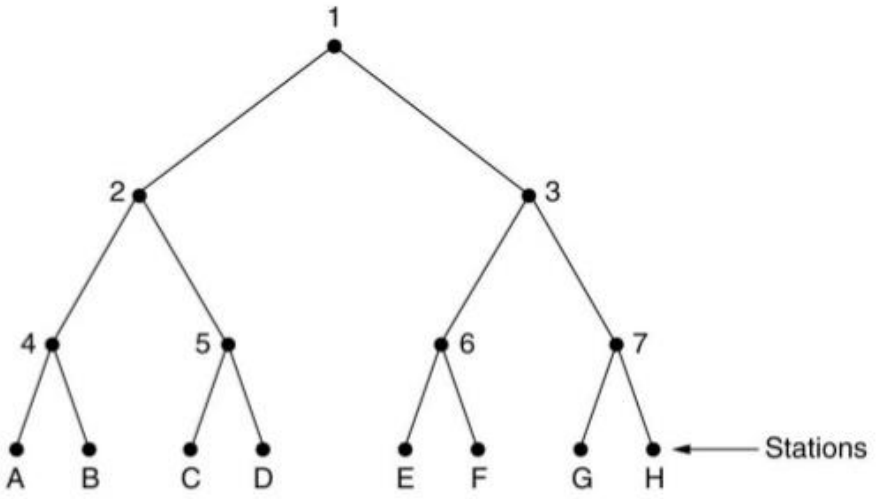
## ATWP (Adaptive Tree Walk Protocol)

Il modo più semplice per realizzare l'obiettivo è utilizzare lo stesso algoritmo impiegato dall'esercito americano durante la II Guerra Mondiale per sapere quanti soldati erano affetti da sifilide. Veniva prelevato un campione di sangue da *N* soldati, una parte di ciascun campione veniva messa in un'unica provetta e veniva verificata la presenza di anticorpi: se non ne erano presenti, gli *N* soldati erano sani; se invece ne trovavano, iniziava un procedimento ricorsivo. Veniva effettuato lo stesso test ma su due provette diverse: la prima contenente il campione di sangue dei soldati da 1 a *N*/2, la seconda contenente i campioni rimanenti. Questo procedimento veniva fatto fino a che non venivano individuati i singoli campioni con gli anticorpi e quindi i soldati affetti.

Nel caso del protocollo nelle reti:

- i soldati sono le stazioni;
- i soldati affetti da sifilide sono le stazioni che vogliono trasmettere;
- il ritrovamento di anticorpi nei campioni di sangue è il rilevamento di collisioni.

La situazione è rappresentabile con un albero binario, in cui le foglie sono le stazioni della rete.




Dopo una trasmissione senza collisioni, viene considerato lo slot che segue **Slot 0**, e viene impiegato il seguente algoritmo:

- **Slot 0**: tutte le stazioni attive della rete, dette stazioni di **Tree(1)**, provano a trasmettere. Se non ci sono collisioni, bene. Se ci sono collisioni l'algoritmo segue.
- **Slot 1**: tutte le stazioni attive di **Tree(2)** provano a trasmettere. Se non ci sono collisioni, l'algoritmo segue da Slot 2 (a). Se ci sono collisioni, l'algoritmo segue da Slot 2 (b).
- **Slot 2 (a)**: tutte le stazioni attive di **Tree(3)** provano a trasmettere. Se non ci sono collisioni, l'algoritmo segue per ricorsività.
- **Slot 2 (b)**: tutte le stazioni attive di **Tree(4)** provano a trasmettere. Se non ci sono collisioni, l'algoritmo segue per ricorsività.

L'algoritmo termina quando le stazioni che dovevano trasmettere lo hanno fatto e quindi si può ricominciare da **Tree(1)**.

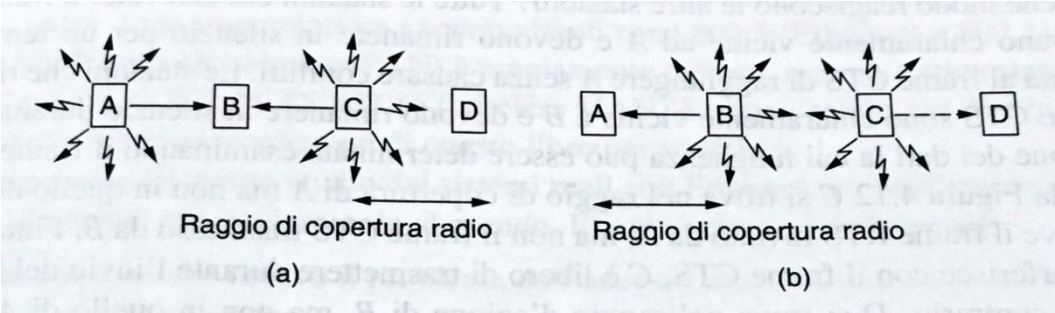
Partire a visionare le stazioni dall'alto dell'albero per capire chi deve trasmettere non ha sempre senso. Se il carico è molto alto, ovvero se molte stazioni devono trasmettere, il controllo al nodo 1 può evitare, partendo direttamente dal nodo 2 (e poi 3).

Se le stazioni attive (che vogliono trasmettere) sono *A* (si può conoscere *A* analizzando l'andamento della rete), e i nodi vengono distribuiti in livelli in base alla loro distanza *P* dalla base (il nodo 1 è al livello 0). Ogni nodo al livello *P* ha sotto di esso $\frac{1}{2^P}$ delle stazioni totali. Se le *A* stazioni sono distribuite uniformemente, ogni nodo ha sotto di esso $\frac{A}{2^P}$ stazioni attive. Quindi il livello corretto da cui partire è quando le stazioni attive sotto il nodo sono uniche, ovvero $\frac{A}{2^P} = 1$ e quindi $P = \log_2 A$.

## Protocolli per LAN Wireless

Nel caso di reti LAN wireless, le assunzioni fatte fino ad ora non bastano. Mentre in una rete cablata tutte le stazioni ricevono i frame inviati, nelle reti wireless le stazioni cambiano spesso (si possono collegare e scollegare molto spesso) e l'ambiente in cui sono inserite può generare interferenze o limitare il segnale. Per questo motivo, sono impiegati più canali nelle reti wireless, dove alcuni stazioni interagiscono e altre no. Il problema passa da un controllo globale a un controllo locale.

Nelle spiegazioni dei seguenti problemi, si assuma che A, B, C, D siano stazioni in canali diversi. Il fatto che ci sia o meno carrier-sense non cambia la situazione seguente.




**Hidden station problem** (figura a): è il problema della stazione nascosta. Il caso è il seguente:

- A trasmette a B.
- C vuole trasmettere a B e lo fa, perché nota che nessun altro sta trasmettendo nel suo canale (non può saper nulla di A).
- Dentro al canale di B arrivano due frame per cui avviene una collisione e nessuno dei due è riuscito a trasmettere.

**Exposed station problem** (figura b): è il problema duale della stazione nascosta. Il caso è il seguente:

- B trasmette ad A.
- C vuole trasmettere a D, ma nota che vicino a lei stanno trasmettendo. Per evitare collisioni, non trasmette (ma avrebbe potuto, perché inviando a D non avrebbe creato una collisione).

Entrambi i problemi sprecano banda, il primo generando collisioni, il secondo invece non sfruttando il canale quando potrebbe.

Una soluzione a questi problemi è introdotta nei sistemi **MACA** (**Multiple Access Collision Avoidance**) e la sua versione per reti wireless **MACAW** (**MACA Wireless**).

Con MACA e MACAW, le stazioni rendono il proprio spazio locale a conoscenza. Quando intendono trasmettere inviano al destinatario un frame **RTS** (**Request To Send**), in cui è presente la dimensione del frame che devono inviare successivamente. Se il destinatario accetta, il mittente riceverà un ACK chiamato **CTS** (**Clear To Send**).

Il sistema non risolve le collisioni al 100%: ci possono essere casi in cui, sempre per problemi di visibilità fra stazioni, alcune stazioni non sentono che una stazione ha ricevuto un frame RTS e quindi trasmettono ugualmente, generando una collisione. Nonostante ciò, il sistema funziona comunque bene.

# Ethernet

Capitolo 4.3 della 5° edizione del libro “Reti di calcolatori” di A. S. Tanenbaum

Il protocollo che si sta per descrivere è detto **IEEE 802.3** (**Ethernet**) ed è stato sviluppato da uno studente laureatosi al MIT, Bob Metcalfe, che ideò sia la tecnologia che i cavi, ispirandosi alla rete ALOHA che era in sviluppo negli stessi anni. Il nome Ethernet deriva dall’*etere luminifero*, dove fino al XIX secolo si pensava passassero le onde elettromagnetiche (ancora non si sapeva che queste onde potrebbero propagarsi anche nel vuoto).

I cavi utilizzati in Ethernet impiegano la nomenclatura *X*Base*Y*, in cui *X* è la banda in Mbps, Base indica “Base band” ovvero trasmissione in singola frequenza, *Y* è il tipo di cavo.

Alcuni esempi di cavo:

- **10Base5**: cavo coassiale (spesso) base band con banda ampia 10 Mbps. Ogni 2,5 m c'è una tacca per inserire un altro cavo. Il 5 indica che fino a 500 m non servono interventi di ripetizione. Questo tipo di cavo supporta al massimo 100 utenti. L'interconnessione fra cavi avviene tramite dei cosiddetti **vampire taps** (morsetti che innestano i cavi bucando il cavo a cui innestare). Il transceiver, ovvero la componente hardware che si occupa di carrier e collision detection è inserita in ogni giunzione.
- **10Base2**: cavo coassiale (fine) base band con banda ampia 10 Mbps. Le giunzioni sono a T, il transceiver è rimosso dal cavo e inserito nei PC che supportano Ethernet (con apposita scheda di supporto). È più economico e più affidabile del precedente cavo, ma 2 sta per 200 m e quindi devono aumentare gli interventi di ripetizione. Inoltre, ogni segmento tiene al massimo 30 utenti.
- **10Base-T**: cavo twisted pair (doppino) con banda ampia 10 Mbps. La T sta appunto per twisted pair, è poco costoso ed è ancora più affidabile. È possibile utilizzare strumenti come gli hub per connettere più computer assieme senza giunzioni nei cavi. La lunghezza massima è nuovamente scesa (100 m) ma è salito vertiginosamente il numero massimo di utenti: 1024.
- **10Base-F**: cavo in fibra ottica, molto fine, con banda ampia 10 Mbps. Permette una lunghezza massima di 1000 m.

**Confronto fra i tipi di cavo**:

| ***Cavo*** | **Lunghezza** | **Densità** | **Densità/Km** |
|---|---|---|---|
| *10Base5* | 500 | 100 | 200 |
| *10Base2* | 200 | 30 | 150 |
| *10Base-T* | 100 | 1024 | 10240 |
| *10Base-F* | 1000 | - | - |

Per Ethernet, Metcalfe poteva scegliere vari tipi di codifiche per lo 0 e per l'1, per esempio 0 Volt per lo 0, *x* Volt per 1 oppure *-x* Volt per lo 0 e *+x* Volt per l'1 (come CDMA) ma si sarebbero potuti presentare problemi di sincronizzazione con lunghe sequenze di 0 o 1; invece scelse il **Manchester encoding** che è facile da implementare, non richiede hardware specifico, ma dimezza la banda. È comunque stato scelto per l'alta affidabilità che gli ha poi permesso di incrementare la banda senza far aumentare i costi.

## Protocollo del sottolivello MAC di Ethernet classica

I frame dello standard Ethernet sono storicamente due:

- il primo, adottato prima dell'IEEE 802.3 (Ethernet DIX);
- il secondo, adottato dall'IEEE 802.3 (Ethernet 802.3).

Fra i due cambia sostanzialmente un solo campo, che verrà opportunamente segnalato.

I campi di un frame Ethernet sono:

- **Preamble** (8 Byte): preambolo necessario per sincronizzare il trasmettitore e il ricevitore. Sono tutti byte 01010101. Nella versione Ethernet 802.3, gli ultimi due bit dell'ottavo byte sono impostati a 11 (l'ultimo byte è 01010111) per segnalare l'inizio vero e proprio del frame;
- **Destination** (6 Byte): indirizzo del destinatario del messaggio. Il primo bit, se è a 0 segnala una comunicazione ordinaria, se è a 1 segnala una comunicazione multicast (a un gruppo). Se tutti i bit sono a 1 segnalano una comunicazione broadcast (a tutti). Il secondo bit segnala

se l'indirizzo è globale o locale (interno alla rete in cui si è). Quindi 6 Byte = 48 bit, 48 – 2 = 46. I 46 bit sono relativi all'indirizzo del destinatario: è l'**indirizzo MAC** (**Media Access Control**);

- **Source** (6 Byte): indirizzo del mittente del messaggio. È ancora un indirizzo MAC;
- **Type/Length** (2 Byte): questi 2 byte dipendono dal protocollo. Ethernet DIX usa il campo Type, Ethernet 802.3 usa il campo Length. Il primo segnala come interpretare i dati del frame, il secondo la lunghezza del campo dati. Il motivo per cui si ha questa ambiguità da risolvere (possono coesistere questi frame) è storico. L'IEEE ha deciso che per valori minori o uguali a 1536 (0x600) il campo è interpretabile come Length, maggiori a 1536 come Type;
- **Data** (0-1500 Byte): payload del frame;
- **Pad** (0-46 Byte): campo necessario per far rispettare la grandezza minima del frame, fissata a 64 Byte per poter controllare se si sono verificate collisioni. In una rete a 10 Mbps (la prima inventata), affinché una stazione si possa accorgere se si sono verificate collisioni, la lunghezza del frame deve essere tale da stare ancora spedendo il frame prima che il primo bit arrivi a destinazione (deve essere impiegato più tempo del tempo massimo di round-trip). Questa si è verificata essere 64 Byte (512 bit);
- **Checksum** (32 Byte)**:** rilevazione degli errori con CRC-32.

**Indirizzo MAC**: indirizzo definito MAC-48 per la sua lunghezza in bit (in futuro si passerà a EUI-64, quindi 64 bit). Quindi i bit sono 48, e vengono divisi in due blocchi. Il primo blocco da 24 bit (3 Byte) è usato per un **OUI** (**Organizationally Unique Identifier**), che è un indirizzo che viene assegnato dall'IEEE a un'azienda che vuole produrre dispositivi che si connettono alla rete. Assieme a questo indirizzo di 24 bit, le viene assegnato uno spazio di $2^{24}$ indirizzi univoci per quell'OUI, utilizzabile per i propri dispositivi. Il secondo blocco da 24 bit è quindi l'indirizzo vero è proprio del dispositivo. Assieme, i due blocchi, formano un indirizzo che è univoco fra tutti gli altri dispositivi esistenti nel mondo. Si stima che la fine degli indirizzi MAC si attesti attorno al 2100.

## CSMA/CD con binary exponential back off

Ethernet sfrutta il protocollo **1-persistent CSMA** per la trasmissione. Una stazione che vuole trasmettere resta in ascolto del canale e quando lo sente libero, trasmette. Se ci riesce, bene. Se non ci riesce, attende un tempo casuale (più specificatamente, un numero casuale di slot di tempo, ovvero il tempo di trasmissione di 512 bit – 64 Byte – la dimensione minima di un frame calcolata dal tempo massimo di round-trip) prima di ricominciare ad ascoltare il canale.

Questo tempo casuale è però generato tramite un sistema definito **truncated binary exponential back-off**:

- alla 1° collisione, attende 0 o 1 slot prima di ricominciare ad ascoltare;
- alla 2° collisione, attende 0, 1, 2 o 3 slot prima di ricominciare ad ascoltare;
- alla 3° collisione, attende da 0 a $2^3 - 1$ (= 7) slot prima di ricominciare ad ascoltare;
- …
- Alla *i*° collisione, attende da 0 a $2^i - 1$ slot prima di ricominciare ad ascoltare.

Però *i* non cresce sempre: *i* viene troncato (truncated) a 10, ovvero $2^{10} – 1 = 1023$ slot. Quindi dalla 10° collisione in poi attende al massimo 10 slot. Dopo la 16° collisione viene segnalato al mittente un errore. Ci si ferma a 1023 altrimenti in casi molto gravi di collisioni non ci si fermerebbe più ad aspettare (già 1023 è molto alto).

## Prestazioni di Ethernet

Diciamo che l'efficienza di Ethernet è:

$$\frac{T}{T + 2 \cdot \frac{roundtrip}{\alpha}}$$

in cui:

- $\alpha$ è la probabilità di trasmissione nel canale;
- $T$ è il tempo medio di trasmissione di un frame;
- $roundtrip$ è il tempo di andata e ritorno.

Se sostituissimo $T$ con $\frac{frame\ length}{bandwidth}$ otterremmo che l'efficienza è inversamente proporzionale al prodotto $bandwidth \cdot frame\ length$. Quindi più aumentiamo la banda, o più aumentiamo la lunghezza del cavo, più l'efficienza diminuisce!

La banda non è sacrificabile, la lunghezza del cavo sì. Soluzione introdotta: utilizzo di **switch** al posto degli **hub** (spiegati nel Capitolo 5).

### Fast Ethernet

La larghezza di banda a 10 Mbps iniziava a diventare poca, quindi si è dovuti passare a un nuovo standard. Sono stati formati due comitati, il primo ha prodotto lo standard **IEEE 802.3u Fast Ethernet**, il secondo è volato troppo in alto e ha formato lo standard **IEEE 802.12**.

Per implementare lo standard Fast Ethernet si sono impiegati nuovi cavi, riassunti nella tabella sottostante.

| ***Nome cavo*** | ***Tipologia cavo*** | ***Lunghezza massima*** |
|---|---|---|
| *100Base-T4* | UTP3 (doppino telefonico) | 100 m |
| *100Base-TX* | UTP5 (cavo "Ethernet") | 100 m |
| *100base-FX* | Fibra ottica | 2000 m |

### Gigabit Ethernet

Con l'evoluzione tecnologica ci si è spinti ancora più in là, sviluppando un protocollo, l'**IEEE 802.3z Gigabit Ethernet**, che aveva problemi da risolvere sempre più grandi.

Con una banda a 1 Gbps, il carrier-sense non è più impiegabile, però è a conoscenza degli switch, e questo gli permette di operare in modalità Point-to-Point. Altro problema: 1 ms di ritardo può far accumulare circa 2000 frame. Viene introdotto una modalità per cui si può mettere in pausa la trasmissione fino a che non si è gestito il problema.

Aumenta la velocità, diminuisce il cavo. Soluzione: vengono spediti blocchi di pacchetti invece di pacchetti singoli e si riescono a raggiungere 200 m di lunghezza massima per il cavo (considerato dal protocollo).

## LAN Wireless

Capitolo 4.4 della 5° edizione del libro "Reti di calcolatori" di A. S. Tanenbaum

L'implementazione delle reti locali senza fili è svolta dallo standard **IEEE 802.11**. Viene sfruttata la banda a **2,4 GHz**, che fa parte di una delle tre bande ISM e quindi può interferire con forni a microonde, telefoni senza filo, Bluetooth, ecc.

I primi standard implementati:

- **802.11a**: larghezza di banda fino a 54 Mbps;

- **802.11b**: larghezza di banda fino a 11 Mbps (viene sviluppato prima di 802.11a), ma ha un range 7 volte più ampio;
- **802.11g**: larghezza di banda fino a 54 Mbps.

La trasmissione dei dati (ovvero come i protocolli implementano il livello fisico) avviene:

- tramite infrarossi (ma hanno avuto poco successo);
- tramite **FHSS** (**Frequency Hopping Spread Spectrum**);
- tramite **OFDM** (**Orthogonal Frequency Division Multiplexing**) che utilizza **QAM** (**Quadrature Amplitude Multiplexing**), impiegato in 802.11a;
- tramite **DSSS** (**Direct Sequence Spread Spectrum**) e **HR-DSSS** (**High Rate DSSS**), che è un tipo di modulazione di fase, impiegato in 802.11b. È interessante perché utilizza codici di Walsh-Hadamard (gli stessi utilizzati per le missioni spaziali).

Ci sono due modalità operative:

- **DCF** (**Distributed Coordination Function**), senza controllo centrale;
- **PCF** (**Point Coordination Function**), con controllo centrale;

che possono coesistere (perché entrambe supportate da tutte le schede di rete.

DCF che può funzionare in modalità diverse:

- come CSMA/CD, usando il protocollo **non-persistent CSMA** con **truncated binary exponential back-off**;
- come CSMA/CA, usando un protocollo simile a MACAW (utilizzando frame RTS e CTS).

PCF usa modalità simili alle reti cellulari:

- la stazione base manda un frame “beacon” periodicamente in broadcast, per sincronizzarsi e ottenere informazioni sullo stato della trasmissione;
- è un sistema centralizzato, appunto simile a quello utilizzato dai cellulari.

Uno standard implementato successivamente:

- **802.11n**: larghezza di banda fino a 248 Mbps, ha un range outdoor di circa 250 m e indoor di circa 70 m grazie all’utilizzo di **MIMO** (**Multiple Input-Multiple Output**) implementato con **smart antennas** (più antenne che cooperano). I canali vengono aumentati da 20 a 40 MHz, restando sempre nella banda da 2,4 GHz ma usando anche quella da 5 GHz se necessario, impiegando **QAM** (tipicamente **QAM-64**).

# Commutazione a livello data link

Capitolo 4.8 della 5° edizione del libro “Reti di calcolatori” di A. S. Tanenbaum

La commutazione a livello data link è l’operazione che permette di realizzare un’unica grande rete locale a partire da reti locali già esistenti che si vogliono far cooperare. Per realizzare quest’obiettivo, vengono utilizzati i seguenti dispositivi:

- ripetitori;
- hub;
- bridge;
- switch.

**Ripetitore**: dispositivo che ha l'obiettivo di prendere il segnale di ingresso, amplificarlo e rispedirlo amplificato.

**Hub**: dispositivo che realizza connessioni fisiche, propagando il segnale da una parte all'altra. Può anche svolgere il lavoro del ripetitore. Un hub risulta poco complesso nel suo utilizzo, è di basso costo ed è affidabile nella sua semplicità, ma non risolve i problemi dei limiti della rete e inoltre estende solamente la rete che gestisce (non può unire reti diverse, ad esempio una da 10 Mbps e una da 100 Mbps).

I ripetitori e gli hub lavorando allo strato fisico, sono molto semplici ma molto limitati.

**Bridge**: dispositivo di livello più alto (data link), che interagisce con la struttura dei frame, permettendo operazioni impossibili ai due dispositivi precedenti.

**Switch**: dispositivo simile ad un bridge (di fatto i nomi bridge e switch sono intercambiabili) che viene però usato per connettere più computer o LAN, invece che reti più grandi.

Riferendoci assieme a questi due nuovi dispositivi (dato che la differenza oggi è sfumata), il loro obiettivo è quello di unire due o più reti, mantenendo però le loro distinzioni (cosa che non poteva fare uno hub) e di far comunicare i vari computer in maniera *quasi* istantanea, cosa ovviamente impossibile, ma è quasi raggiungibile (almeno dal punto di vista dell'utente) se questi dispositivi svolgono il loro lavoro molto rapidamente. Se c'è più di una stazione connessa, viene utilizzato un protocollo di tipo CSMA/CD per spedire i frame.

Gli switch/bridge operano al livello data link e non a livello fisico. Di conseguenza, non ci saranno più problemi di dimensione della rete e inoltre ci saranno domini di collisione diversi (i frame di una stessa rete collidono fra loro, non fra tutti quelli di tutte le reti connesse). Tutte queste operazioni si realizzano ispezionando il traffico e filtrando i pacchetti.

**Learning**: fase in cui uno switch/bridge impara la configurazione di rete e gestisce il corrispondente traffico. Essendo a livello data link, quello che deve fare è analizzare nei frame i campi Source e Destination, contenenti gli indirizzi MAC dei dispositivi. Del mittente sa già tutto (ovvero il suo indirizzo e anche la porta a cui è collegato), deve ora scoprire in che porta è situato il destinatario e inoltrare il frame, memorizzando tutte queste informazioni in una tabella al suo interno. Questa operazione è detta backward learning, perché viene costruita una tabella di hash analizzando i flussi dati. Quando un record viene aggiunto alla tabella, viene segnato il tempo di arrivo. Ogni qualvolta lo switch controlla la tabella per vedere dove spedire un frame, se trova il record corrispondente, aggiorna il tempo di arrivo. Per mantenere la tabella corretta, senza informazioni inutili, periodicamente un processo interno allo switch rimuove i record di dispositivi non più utilizzati da qualche minuto.

**Timeout**: c'è un timeout di **fading** quando si nota che la topologia della rete è cambiata.

**Spanning tree**: struttura creata dagli switch che permette di risolvere i problemi dei loop nelle reti, ovvero vengono rimossi i possibili grafi ciclici trovati e sostituiti con delle strutture ad albero. Questi grafi ciclici possono essere errori di creazione della rete fisica oppure ridondanze per evitare che le reti si scolleghino col guastarsi di un cavo. Un protocollo che svolge questa operazione è lo **IEEE 802.1D**. Nella creazione dello spanning tree viene sfruttato il fading per adattarsi ai cambiamenti della topologia di rete.

La creazione dello spanning tree tipicamente richiedeva sui 30 secondi per reti normali, con l'avvento di 802.1D si è riusciti a diminuire parecchio questo numero, portandolo a circa 6 secondi (ma esistono soluzioni più avanzate e più ottimizzate).

# Il livello di rete

La funzione principale del livello di rete è quella di inoltrare i pacchetti dal computer sorgente al computer destinazione. Nella maggior parte delle reti per raggiungere la meta i pacchetti compiono una sequenza di salti (**hop**). Se sorgente e destinazione non si trovano nella stessa rete, il routing è un problema. Gli algoritmi che scelgono i percorsi sono fondamentali a questo livello.

## Algoritmi di routing

Capitolo 5.2 della 5° edizione del libro “Reti di calcolatori” di A. S. Tanenbaum

**Algoritmo di routing**: parte del software del livello di rete che si preoccupa di scegliere lungo quale linea di uscita inoltrare i pacchetti in arrivo. Ne esistono di due gruppi principali:

- **algoritmi non-adaptive** (**non-adattivi**): le proprie decisioni non sono basate su misurazioni, stime del traffico o topologia di rete, ma sono decise staticamente; gli algoritmi di questo tipo si dice che fanno **routing statico**;
- **algoritmi adaptive** (**adattivi**): le proprie decisioni cambiano in base alla topologia di rete e al traffico corrente; gli algoritmi di questo tipo si dice che fanno **routing dinamico**.

La scelta del percorso in cui instradare un pacchetto può essere effettuata in più modi. Un buono modo è ad esempio scegliere il percorso minimo, ovvero trovare un percorso fra sorgente e destinazione che sia il più piccolo fra tutti i percorsi possibili, calcolato sommando le distanze di tutti i sotto-percorsi disponibili (il concetto è semplice, la realizzazione non molto). Un primo modo per implementare questo concetto è contare il numero il **hop** che si incontrano lungo un cammino. A ogni stazione incontrata corrisponde un hop. Potrebbe essere che percorsi più lunghi siano migliori di percorsi più corti, e questo può essere vero se si conoscono i tempi di percorrenza fra le stazioni (il percorso più breve potrebbe essere mediamente rallentato mentre quello più lungo più scorrevole).

In caso di routing statico, un buon modo per calcolare il cammino minimo (**shortest path**) è l’**algoritmo di Dijkstra**: ogni nodo/stazione della rete conterrà una tabella con la distanza da ogni altro nodo, permettendo a un pacchetto che deve essere instradato di scegliere il percorso più breve. Questo non va bene in caso di routing dinamico, non perché l’algoritmo non funzioni in questi casi, ma per altri motivi che si vedranno fra poco.

### Flooding

In caso di reti dinamiche, un buon modo per effettuare l’instradamento dei pacchetti è mediante il **flooding**.

Il flooding (inondazione) è un metodo molto semplice per effettuare l’operazione richiesta, ed è anche il più robusto fra tutti i tipi di algoritmi (è dimostrabile matematicamente). Quello che fa è semplicemente prendere il pacchetto ricevuto, inviarlo a tutte le linee di uscita fuori che a quella da cui il pacchetto è arrivato. Questa modalità ha sia degli svantaggi che dei vantaggi importanti.

Partendo dagli svantaggi, senza un controllo sui pacchetti, questi possono replicarsi all’infinito intasando la rete. Un metodo per ovviare a questo può essere inserire un contatore nel pacchetto, che viene decrementato ad ogni hop e quando è a zero viene scartato, oppure evitare che il router ritrasmetta pacchetti che ha già trasmesso. Ma ovviamente lo svantaggio principale è quello che ci si aspetta: genera troppi pacchetti.

Per quanto riguarda i vantaggi, non esiste instradamento migliore per quanto riguarda il broadcast (infatti per l'invio in broadcast gli algoritmi spesso usano il flooding) e soprattutto è robusto (utile in ambiti miltari). Ultimo, ma non meno importante, sceglie sempre il percorso più breve e genera il ritardo di trasmissione minore (senza considerare il tempo dovuto alla duplicazione dei pacchetti).

## Distance Vector Routing

È stato il routing implementato nella prima versione di ARPANET.

Ogni router conserva nella sua memoria una tabella (ovvero un vettore) che contiene la migliore distanza conosciuta verso gli altri router della rete. Le informazioni vengono scambiate fra i router in modo tale che siano sempre aggiornate.

Questo tipo di routing reagisce bene alle modifiche della rete in cui vengono aggiunte stazioni: in una rete con il percorso più lungo che è *N* salti, dopo *N* scambi tutti i router avranno le informazioni aggiornate e corrette. Ma se viene rimossa una stazione, o salta un collegamento, la situazione è tutt'altro che rosea: si presenta il problema del **count-to-infinity**.

Il **count-to-infinity**, tra l'altro ancora irrisolto, è il problema che rende fragile questo sistema di routing. Quando una stazione viene rimossa, a ogni scambio tutti i router aggiornano la propria distanza da quella stazione incrementandola unitariamente, a partire dalle stazioni distanti solo di un hop.

## Link State Routing

Questo nuovo algoritmo di routing, il Link State Routing, è stato il sostituto del Distance Vector Routing in ARPANET. Ne esistono due varianti per le reti di grandi dimensioni e per Internet.

Alla base di questo tipo di routing ci sono cinque passaggi:

1. **scoprire i propri vicini e i relativi indirizzi di rete**: vengono inviati pacchetti HELLO su ogni linea punto a punto, quelli che rispondono sono i router vicini;
2. **misurare la distanza o la metrica di costo di ogni vicino**: vengono inviati pacchetti ECHO ai quali i destinatari devono rispondere immediatamente in modo che il mittente calcoli il tempo necessario per raggiungere i propri vicini;
3. **costruire un pacchetto contenente tutte le informazioni raccolte**: viene creato un pacchetto con informazioni sull'identità del trasmittente, un numero di sequenza (di 32 bit, per eliminare la possibilità che avvengano numeri ripetitivi), l'Age (età) e una lista dei vicini con rispettivo ritardo misurato nel passaggio 2. Questo pacchetto può essere creato periodicamente oppure quando si interrompe una linea, un vicino si disconnette, si attiva o cambia proprietà;
4. **inviare tale pacchetto a tutti gli altri router e ricevere da loro i pacchetti**: viene inviato il pacchetto generato in broadcast mediante flooding, in cui il numero di sequenza viene incrementato per ogni nuovo pacchetto inviato e l'Age che viene decrementato a ogni hop che fa. Se un router riceve pacchetti duplicati, li scarta. Solo se riceve un pacchetto con un numero di sequenza maggiore a quello che ha già lo confronta con quello in suo possesso e aggiorna le informazioni. Si possono presentare problemi se un router si blocca e perde traccia dei numeri di sequenza, se il conteggio ricomincia da 0 (perché i suoi pacchetti verranno scartati) o se un numero di sequenza arriva danneggiato, specialmente se questo è maggiore di quello che dovrebbe essere, poiché farebbe scartare tutti i precedenti numeri;
5. **elaborare il percorso più breve verso tutti gli altri router**: viene impiegato l'algoritmo di Dijkstra.

## Routing gerarchico

La dimensione delle tabelle di routing cresce proporzionalmente alla grandezza della rete. Più la rete è grande, più elaborazione di dati deve fare il router e questo non è positivo se si vuole avere molta larghezza di banda.

Una soluzione può essere il routing gerarchico, in cui ogni router conosce tutti i dettagli relativi ai router nella stessa regione, ma delle strutture delle altre regioni non sa nulla. Si risparmia memoria, si delega una buona parte del lavoro, ma si paga in lunghezza dei percorsi.

## Routing broadcast

Quando occorre fare broadcasting, ovvero trasmettere a tutte le stazioni della rete contemporaneamente, invece di usare il flooding in una rete con in uso il Link State Routing, si può essere più intelligenti e usare un algoritmo detto **Reverse Path Forwarding**.

Il principio è lo stesso del flooding, ma vengono considerati solo i pacchetti provenienti dai cammini migliori.

# Algoritmi per il controllo della congestione

Capitolo 5.3 della 5° edizione del libro “Reti di calcolatori” di A. S. Tanenbaum

Quando in (una parte) della rete sono presenti troppi pacchetti, si verificano ritardi e perdite di pacchetti: situazione chiamata **congestione**. Essendo che la congestione avviene nella rete, chi lo sperimenta veramente è il livello di rete, che deve ridurre il carico del livello di trasporto immesso nella rete. Se la rete non è ben progettata, è possibile che questa collassi.

La congestione è uno dei problemi più importanti con cui ci si scontra per il miglioramento della qualità del servizio (**QoS**, **Quality of Service**). Ma com’è che si può verificare una congestione? Ad esempio, quando si tiene conto solo del percorso migliore per trasmettere e non il carico (conosciuto come **effetto seesaw**).

Ci sono alcune tecniche che permettono di evitare la congestione:

- **choke packet**: è il più semplice fra tutti i metodi. Se un router nota una congestione, seleziona un pacchetto congestionato, legge il campo Source e invia un **choke packet** (pacchetto di strozzamento) al mittente di quel pacchetto, che viene etichettato (viene impostato un flag nell’header) in modo tale che non vengano richiesti altri strozzamenti a quella stazione. Ovviamente, un router non deve inviare troppi choke packet, o congestionerà lui la rete. Ricevuto un choke packet, un host deve ridurre il proprio traffico del 50%;
- **choke hop-by-hop**: il metodo precedente funziona nel caso in cui la rete non è molto grande, ma potrebbe non risolvere il problema in caso di reti più grandi (il pacchetto potrebbe impiegare troppo per arrivare). Con **choke hop-by-hop**, viene chiesto non solo a chi trasmette di ridurre la trasmissione, ma anche ai router intermedi;
- **load shedding**: in caso i metodi precedenti non funzionino, i router possono ricorrere a tecniche drastiche, ovvero l’eliminazione dei pacchetti. Ovviamente, non vengono eliminati pacchetti casuali, ma viene eliminato ciò che viene ritenuto più opportuno. Ci sono due strategie:
    - **wine** (il vino più vecchio è migliore), in cui si tengono i pacchetti più vecchi (esempio: trasferimento di file);
    - **milk** (il latte fresco è migliore), in cui si tengono i pacchetti più nuovi (esempio: streaming video);

- **buffering**: riduce il **jitter** nel ricevente, ma non risolve la congestione;
- **leaky bucket**: garantisce un data rate ridotto costante, evitando i burst che creano congestioni);
- **token bucket**: genera ogni intervallo di tempo un token (gettone). I pacchetti in arrivo, per uscire dal secchio (bucket), devono bruciare un gettone che hanno acquisito entrando. I token possono accumularsi se non utilizzati. Il sistema a token permette un data rate costante, e in caso di poco traffico, un data rate maggiore.

# Qualità del servizio

Capitolo 5.4 della 5° edizione del libro “Reti di calcolatori” di A. S. Tanenbaum

La **QoS** (**Quality of Service**) è una serie di parametri che dettano la qualità del servizio offerto.

Nell’ambito delle reti, la QoS è data da quattro parametri principali:

- **reliability** (affidabilità, la capacità di gestire gli errori che si presentano);
- **bandwidth** (larghezza di banda);
- **delay** (ritardo);
- **jitter** (ritardo del secondo ordine, misura il **grado di variazione** del ritardo di arrivo dei pacchetti).

# Il livello di rete in Internet

Capitolo 5.6 della 5° edizione del libro “Reti di calcolatori” di A. S. Tanenbaum

Il collante che tiene unita Internet è il protocollo a livello di rete: **IP** (**Internet Protocol**). IP è “metà” TCP/IP (una scarsa metà), originata dalla scissione dell’originale protocollo **NCP** (**Network Control Protocol**) di ARPANET. NCP era un protocollo affidabile e che gestiva il controllo di flusso. Fu giudicato inadatto dal committente (Ministero della Difesa degli USA) e il risultato fu la divisione del protocollo in TCP e IP.

## Protocollo IP versione 4

I pacchetti di IP si chiamano **datagrammi**. Un datagramma **IPv4** è costituito da due parti, l’intestazione (**header**) e il corpo (**payload**).

I campi del header IP, in totale 20 Byte (la parte obbligatoria), sono i seguenti:

- **Version**: indica subito la versione del protocollo (nella mente del progettista c’era quindi la possibilità di realizzare più versioni del protocollo), in modo che il ricevente sappia subito come interpretare il datagramma;
- **IHL** (**IP Header Length**): lunghezza dell’header;
- **Differentiated services**: indica il tipo di servizio;
- **Total length**: lunghezza del datagramma;
- **Identification**: serve per identificare se i dati sono in più datagrammi;
- **DF** (**Don’t Fragment**): impone la non frammentazione dei dati;
- **MF** (**More Fragment**): segnala se ci sono altri frammenti oppure no;
- **Evil Bit**: “*se un pacchetto è malevolo questo campo è settato a 1*”;
- **Fragment offset**: numero del frammento;
- **TTL** (**Time To Live**): età massima del datagramma;
- **Protocollo**: indica quale protocollo deve interpretare i dati (ad esempio TCP o UDP);

- **Header checksum**: controllo dell'errore con somma in complemento a uno (molto scarsa);
- **Source address**: indirizzo IP sorgente;
- **Destination address**: indirizzo IP destinazione;
- **Options**: parte opzionale (40 Byte massimo, troppo pochi).

**Indirizzo IP**: è un indirizzo di 4 Byte (32 bit). I byte sono separati da un punto (**dot notation**).

Alcuni esempi:

- 87.69.205.3
- 192.168.1.1
- 127.0.0.1

Per l'assegnazione degli indirizzi c'è un'autorità centralizzata che ha questo compito. Gli indirizzi potrebbero essere assegnati a uno a uno, ma questo porterebbe ad avere nei router tabelle di routing gigantesche. La soluzione che si è trovata è avere una struttura gerarchica, e assegnare blocchi di indirizzi e non singoli indirizzi (quindi assegnando delle reti).

Questi blocchi sono detti **classi**. Il metodo di assegnazione è chiamato **metodo del classful addressing**.

| **Composizione indirizzo IP nel classful addressing** | | | |
|---|---|---|---|
| **CLASSE** | **IDENTIFICAZIONE (bit)** | **RETE (bit)** | **HOST (bit)** |
| A | 1 (0) | 7 | 24 |
| B | 2 (10) | 14 | 16 |
| C | 3 (110) | 21 | 8 |
| D | 4 (1110) | 28 | 0 |
| E | 4 (1111) | 28 | 0 |

Le classi vere e proprie sono le prime tre (A, B e C), mentre la D è usata per effettuare il multicasting e la E è stata riservata per usi futuri.

Con gli indirizzi di classe A, si possono avere 128 reti e indirizzamento a 3 Byte.

Con gli indirizzi di classe B, si possono avere 16384 reti e indirizzamento a 2 Byte.

Con gli indirizzi di classe C, si possono avere 2097152 e indirizzamento a 1 Byte.

Il problema degli indirizzi IP di cui si sente parlare, ovvero la loro fine vicina, è dovuto al fatto che molti hanno pensato di acquistare indirizzi di classe B, di fatto lasciandoli inutilizzati, quando potevano prenderli di classe C, acquistando magari più blocchi. Questo problema è stato risolto con **CIDR** in parte, ma ancor di più con **NAT**.

## CIDR

Con il **CIDR** (**Classless InterDomain Routing**) si va oltre il concetto delle classi. I blocchi di indirizzi diventano di lunghezza variabile ed è permesso ai router una gestione tramite **aggregate entries**: se due o più indirizzi hanno un prefisso comune (quindi è molto probabile che abbiano lo stesso instradamento) è più conveniente salvarli in un'unica entry, inserendo solo la parte diversa (e ovviamente indicando la parte comune una volta sola). Ad esempio:

- **85.32**.67.89
- **85.32**.56.36

hanno i primi due byte in comune, possono essere salvati in un'unica entry memorizzando solamente 67.89 e 56.36 oltre l'informazione comune 85.32.

Ovviamente questa operazione non è priva di errori, ci possono essere casi di indirizzi con un prefisso comune ma diverso instradamento richiesto: in questi casi vengono definite delle eccezioni di priorità maggiore nelle tabelle di routing.

## NAT

Il **NAT** (**Network Address Translation**) è la simulazione di un'intera rete usando un solo indirizzo IP. All'interno di questa rete, che altro non è che una LAN, c'è un indirizzamento (tramite indirizzi IP) che ha senso solo al suo interno e quando un pacchetto deve essere indirizzato verso l'esterno avviene una traduzione dell'indirizzo sorgente all'unico indirizzo IP reale disponibile della rete.

Negli indirizzi IP ci sono tre intervalli (uno per classe) che sono riservati per l'utilizzo del NAT. Questi intervalli sono:

- 10.0.0.0 (classe A, 16777216 host);
- 172.16.0.0 (classe B, 1048576 host);
- 192.168.0.0 (classe C, 65536 host).

È chiaro quindi come avviene la traduzione da indirizzo interno della rete a indirizzo unico, ma come avvenga la traduzione inversa non è chiaro. Per questa traduzione infatti è necessario introdurre i protocolli TCP e UDP.

## Protocollo IP versione 6

La vera soluzione alle limitazioni di IPv4 viene introdotta con IPv6.

È un protocollo che è stato standardizzato nel 1998, ma ancora fa fatica a essere adottato.

Le principali differenze sono la lunghezza dell'indirizzo, si passa da 4 Byte a 16 Byte (128 bit), potendo indirizzare $2^{128}$ dispositivi (fin troppi) e la riduzione dei campi dell'header, rimuovendo i campi IHL (perché l'header di IPv6 è di 40 Byte fissi), Total Length (sostituito da Payload length, aumentando la taglia del pacchetto da $2^{16}$ - 20 Byte a $2^{16}$ Byte), TTL (sostituito da Hop limit, è lo stesso campo con la stessa dimensione ma con un nome più corretto, in quanto TTL dovrebbe essere in secondi), Protocol (sostituito da Next header che indica il gestore del protocollo di cui segue l'header). Viene rimossa anche tutta la parte relativa alla frammentazione perché gestita diversamente (se un router non riesce ad interpretare un datagramma perché troppo grande, segnala un errore e il mittente sa che dovrà mandare datagrammi di un'altra dimensione a quel router, evitando di frammentare sempre al volo) e anche il checksum, rendendo IPv6 più inaffidabile di IPv4 per poter essere il più veloce possibile.

## Protocolli di controllo di Internet

Oltre a IP, a livello di rete esistono protocolli che gestiscono le operazioni di controllo. Questi sono **ICMP**, **DHCP** e **ARP**.

## ICMP

**ICMP** (**Internet Control Message Protocol**) è il protocollo di controllo di Internet che si occupa di gestire casi ed eventi inaspettati o imprevisti.

Tipi di messaggio ICMP:

- **DESTINATION UNREACHABLE**: è stato impossibile consegnare il pacchetto;

- **TIME EXCEEDED**: il TTL ha raggiunto lo 0;
- **PARAMETER PROBLEM**: campo dell'intestazione non valido;
- **SOURCE QUENCH**: pacchetto choke;
- **REDIRECT**: comunicazione a un router di aggiornare la propria tabella di routing;
- **ECHO REQUEST**: chiedo a una macchina se è attiva;
- **ECHO REPLY**: la macchina che ha ricevuto un ECHO REQUEST risponde;
- **TIMESTAMP REQUEST**: chiedo a una macchina se è attiva (registrando l'ora di invio);
- **TIMESTAMP REPLY**: la macchina che ha ricevuto un TIMESTAMP REQUEST risponde (registrando l'ora di invio).

## DHCP

**DHCP** (**Dynamic Host Control Protocol**) è il protocollo che gestisce la traduzione degli indirizzi MAC in indirizzi IP. Una macchina che vuole conoscere il proprio indirizzo IP spedisce un pacchetto DHCP DISCOVER: il gestore della rete (DHCP è un sistema centralizzato) risponderà con l'indirizzo.

Anche DHCP ha a che fare con macchine che si collegano e si scollegano in continuazione e per aggiornare i dati usa il fading, chiamato **leasing** (affitto, ovvero ogni indirizzo assegnato non è proprietà della macchina ma è solo in affitto).

## ARP

**ARP** (**Address Resolution Protocol**) è il protocollo che gestisce la traduzione degli indirizzi IP in indirizzi MAC. Permette l'utilizzo del livello data link per le comunicazioni nella stessa rete, evitando di dover passare per il router.

Il protocollo ARP è distribuito, ovvero ogni macchina dispone di una tabella di corrispondenze IP-MAC. Quando una macchina vuole inviare un messaggio a un'altra macchina con indirizzo *x*, manda un pacchetto ARP (broadcast) chiedendo chi è possiede tale indirizzo. La macchina all'indirizzo *x* risponderà con un ARP ACK e in piggyback l'indirizzo MAC.

Essendo i pacchetti inviati in broadcast, ogni qualvolta avviene uno scambio di pacchetti ARP gli host restano in ascolto e memorizzano queste informazioni nella loro tabella. Quando un host si accende, invia una richiesta ARP con il proprio indirizzo IP, per conoscere il proprio indirizzo MAC. Nessun altro dovrebbe rispondere ovviamente, a meno che non avvenga un conflitto di indirizzi IP, che in questo caso dovrebbe essere risolto dall'amministratore della rete.

## Routing in Internet

Dopo aver parlato dei vari protocolli utilizzati nel livello di rete, bisogna capire come le varie reti siano in comunicazione fra loro e che tipo di gerarchi sono presenti.

Internet si divide in più livelli, che sono bene o male sempre reti, grandi o piccole, connesse fra loro. C'è un livello particolare, che è quello degli **AS** (**Autonomous System**): gli AS sono reti sotto il controllo di un unico gestore (un esempio semplice: la rete di un ISP) in cui sono applicate le proprie regole (tra cui gli algoritmi di routing) e in cui vengono decise le regole di integrazione con gli altri AS.

Il routing a un livello superiore, quello fra gli AS, viene effettuato da un protocollo speciale: **BGP** (**Border Gateway Protocol**). Tramite questo protocollo, gli AS possono decidere alcune regole per l'instradamento dei pacchetti, come ad esempio:

- ogni pacchetto del governo americano non deve mai passare per le reti dell'Iraq;
- ogni pacchetto di un sito legato a Google non deve mai passare per le reti di Microsoft;

- ecc.

Come viene effettuato il routing? Con il Distance Vector Routing, solo che invece di tenere le distanze nelle tabelle tiene gli interi cammini.

A questo punto c'è un problema: come facciamo ad accertare che i pacchetti raggiungano l'altra parte del mondo in un tempo ragionevole? Supponiamo di avere pacchetti da 40-64 Byte e una ampiezza di banda a 100 Gbps, quanto tempo si potrà impiegare all'incirca per decidere dove instradare i pacchetti? Circa 3-5 ns, superiori ai tempi di accesso alle DRAM (tecnologia delle RAM nei computer): per questo motivo vengono utilizzati in questi router speciali delle SRAM e algoritmi di look-up veloci (ad esempio: **Lulea Scheme**, che comprime le informazioni per cercare più rapidamente).

# Il livello di trasporto

Nel livello di trasporto, Internet possiede due protocolli principali: TCP e UDP.

TCP è un protocollo orientato alla connessione e fa un po' di tutto: crea connessioni, è affidabile perché effettua le ritrasmissioni quando necessario, controlla il flusso dei dati e la congestione.

UDP è un protocollo non orientato alla connessione e non è affidabile, è semplicemente IP a cui viene aggiunto un concetto in più, che possiede anche TCP.

Le risorse di una macchina che vogliono interfacciarsi in rete hanno nello strato di trasporto una caratteristica in più: le **porte**, che in coppia con l'indirizzo IP formano le **socket**¸ che servono a identificare gli end-point all'interno dei computer sorgente e destinazione.

Ecco come il NAT può permettersi di capire come convertire l'indirizzo IP della rete in indirizzo IP locale. Sostanzialmente, il NAT fa un "multiplexing" dell'indirizzo IP utilizzando le porte (al posto dei canali) e quindi riesce a capire dove spedire il pacchetto in base alla Destination port presente al suo interno (notare come il principio della gerarchia dei livelli si rompe).

## Protocolli di trasporto di Internet: UDP

Capitolo 6.4 della 5° edizione del libro "Reti di calcolatori" di A. S. Tanenbaum

Partendo da **UDP** (**User Datagram Protocol**) si può subito dire che non è altro che IP a cui si aggiunge il concetto delle porte. UDP offre alle applicazioni la possibilità di inviare datagrammi senza dover stabilire una connessione (per esempio quando si ha bisogno di velocità più che di affidabilità).

UDP ha un header di 8 Byte ed è seguito da un payload. I campi dell'header sono i seguenti:

- **Source port** (2 Byte): porta da cui viene spedito il segmento UDP;
- **Destination port** (2 Byte): porta verso cui viene spedito il segmento UDP;
- **UDP length** (2 Byte): lunghezza del segmento UDP, includendo header e payload. Il valore minimo è 8 (8 Byte della lunghezza dell'header) e il massimo è 65515 che è il massimo della dimensione di un payload IP;
- **Checksum** (2 Byte): controllo dell'errore (opzionale).

Esempio d'uso di UDP: **DNS** (**Domain Name System**), ovvero il sistema che si occupa della traduzione degli URL in indirizzi IP. Il DNS ha un funzionamento gerarchico e quando non riesce ad associare immediatamente un URL a un indirizzo IP usa i resolver dei TLD (Top Level Domain).

## Protocolli di trasporto di Internet: TCP

Capitolo 6.5 della 5° edizione del libro "Reti di calcolatori" di A. S. Tanenbaum

**TCP** (**Transmission Control Protocol**) è stato sviluppato per fornire un flusso di byte **affidabile** end-to-end (**orientato alla connessione**) su una internetwork (in questo caso Internet) inaffidabile.

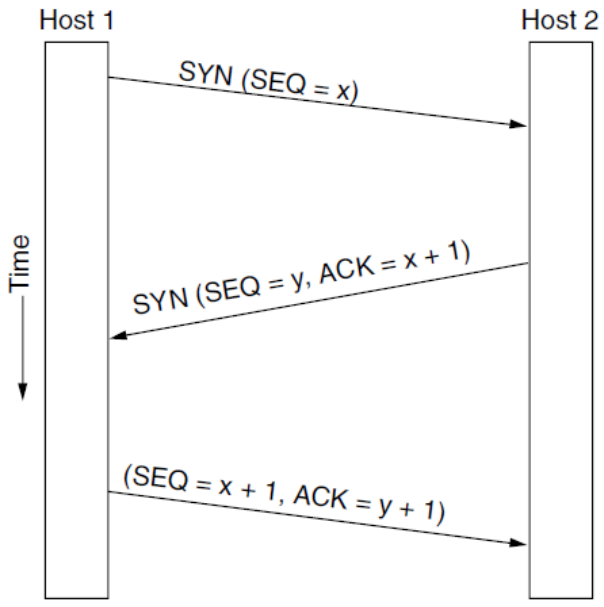
Le connessioni che instaura TCP sono **full-duplex** e **point-to-point**. Non sono supportate le comunicazioni multicast e broadcast.

TCP ha un header di 20 Byte (più eventuali campi opzionali) ed è seguito da un payload. I campi dell'header sono i seguenti:

- **Source port** (2 Byte): porta da cui viene spedito il segmento TCP;
- **Destination port** (2 Byte): porta verso cui viene spedito il segmento TCP;
- **Sequence number** (4 Byte);
- **Acknowledgement number** (4 Byte);
- **TCP header length** (4 bit): lunghezza dell'header (in *word*, ovvero blocchi da 32 bit);
- (4 bit): inutilizzati, in origine erano 6, e sono indice che TCP ha funzionato bene e non sono stati necessari grossi bug fix nel corso degli anni;
- **CWR** (1 bit): flag di segnalazione di congestione (aggiunto dopo);
- **ECE** (1 bit): flag di segnalazione di congestione (aggiunto dopo);
- **URG** (1 bit): flag di segnalazione che si usa il campo l'**Urgent pointer** (offset nel payload per i dati urgenti);
- **ACK** (1 bit): flag di segnalazione che il segmento sta facendo acknowledgement in piggyback;
- **PSH** (1 bit): flag di segnalazione della presenza di dati PUSH (ovvero di elaborare subito i dati e non di salvarli temporaneamente in un buffer);
- **RST** (1 bit): flag di segnalazione di una connessione confusa di cui si richiede il reset;
- **SYN** (1 bit): flag di sincronizzazione per iniziare una connessione;
- **FIN** (1 bit): flag di richiesta di terminazione di una connessione;
- **Windows size** (2 Byte): indica il numero di byte inviabili a partire da quello che ha ricevuto acknowledgement. TCP usa quindi un protocollo a finestra scorrevole ad ampiezza variabile, con ampiezza massima $2^{16} - 1$;
- **Checksum** (2 Byte): controllo dell'errore (somma in complemento a uno di header, dati, **pseudo-header**, infine si fa il complemento a uno del risultato)

Cos'è lo pseudo-header? È un'informazione aggiuntiva che fa parte del calcolo del checksum del protocollo (anche UDP la usa) ed è calcolato a partire dalle informazioni dell'header IP: questa cosa può sembrare banale, ma è una violazione della gerarchia dei protocolli (il livello di trasporto non dovrebbe sapere nulla di ciò che sta sotto di lui!).

## Instaurazione della connessione TCP




*Instaurazione di una connessione TCP con il Three-way handshake*

Una prima nota da fare è che in TCP tutto è regolato da timeout per evitare che si presentino problemi con duplicati che si disperdono nella rete e poi ritornano e fanno confusione nella comunicazione.

Per aprire una connessione TCP si deve effettuare il cosiddetto **Three-way handshake**. Si considerino Host 1 e Host 2 gli interlocutori della trasmissione, Host 1 chiede di aprire un canale di comunicazione e Host 2 accetta. Il procedimento, nel caso in cui non si verifichino errori, avviene nel seguente modo (come illustrato nella figura):

- Host 1 invia un segmento con i flag SYN=1 e ACK=0, indicando nel campo Sequence number un numero casuale *x* per effettuare questa connessione.
- Host 2 riceve la richiesta di connessione da Host 1 e risponde con un segmento con i flag SYN=1 e ACK=1, indicando nel campo Sequence number un numero casuale *y* e nel campo Acknowledgement number il numero *x*.
- Host 1 conferma a Host 2 di voler aprire una connessione e quindi invia un ulteriore acknowledgement di conferma a Host 2 con Sequence number e Acknowledgement number.

Come si può notare, Sequence e Acknowledgement number servono per evitare che vecchi tentativi di connessione possano arrivare a Host 2 e instaurare una connessione non più richiesta da Host 1, che senza questi numeri, appositamente casuali e grandi (32 bit), evitano. Se è necessario, si possono estendere i range di numeri di 32 bit aggiungendo anche dei timestamp (in particolare su linee veloci, in cui i numeri da 32 bit fanno presto a esaurire).

## Rilascio di una connessione TCP

Per rilasciare un'attiva connessione TCP i due host devono scambiarsi dei segmenti per poter chiudere in sicurezza e senza perdita di dati la comunicazione. I rilasci delle comunicazioni sono unidirezionali (se solo un host vuole disconnettersi lo può fare)

Il primo passo che deve fare un host che vuole interrompere una comunicazione è spedire un segmento con flag FIN=1 e ricevere successivamente un ACK. Se vogliono entrambi disconnettersi devono essere inviati quattro segmenti, eventualmente tre se si fanno corrispondere un ACK e una richiesta di disconnessione da un lato. Per evitare che si presenti il **problema dei due eserciti**, quando ci si vuole disconnettere si usa un timer: se la risposta a un segmento con FIN=1 non arriva entro il tempo di vita massimo di due pacchetti la connessione viene rilasciata unilateralmente.

# Sicurezza delle reti

## Crittografia

Capitolo 8.1 della 5° edizione del libro “Reti di calcolatori” di A. S. Tanenbaum

**Crittografia**: tecnica che permette di rendere un messaggio incomprensibile a chiunque non conosca il modo in cui è stato scritto (deriva da due parole greche antiche che significano “scrittura segreta”).

Si devono distinguere i termini **cifrario** e **codice**:

- **cifrario**: trasformazione di un messaggio carattere per carattere (o bit per bit);
- **codice**: trasformazione di un messaggio parola per parola.

Procedimento per la cifratura di un messaggio:

- è dato un testo in chiaro, il messaggio, detto **plaintext** (per semplicità detto ***P***);
- P deve essere trasformato da una funzione di **encoding** parametrica su una **chiave** (per semplicità la funzione è detta ***E***, la chiave ***K***) in un messaggio cifrato;
- il messaggio cifrato ***C***, detto **ciphertext** (o **cyphertext**), si ottiene mediante $\boldsymbol{C = E_K(P)}$.

Il procedimento inverso, la decifratura, si ottiene mediante una funzione di **decoding** parametrica sulla chiave ***K*** (per semplicità detta ***D***) usando $\boldsymbol{P = D_K(P)}$.

L’algoritmo di crittografia deve essere pubblico, la chiave deve essere privata. Questo semplice concetto è stato espresso la prima volta nel 1833 da Auguste Kerckhoff in “La Cryptographie Militaire”, di cui si elencano i principi base:

1. il sistema dovrebbe essere, in pratica (anche se non lo potrebbe essere in teoria), inviolabile;
2. **la progettazione di un sistema non dovrebbe richiedere segretezza, e compromettere il sistema non dovrebbe creare problemi ai corrispondenti (Principio di Kerckhoff)**;
3. la chiave dovrebbe essere memorizzabile senza dover essere scritta, e facilmente intercambiabile;
4. i crittogrammi dovrebbero essere trasmissibili tramite *telegrafo*;
5. l’apparato crittografico e i documenti dovrebbero essere portatili e gestibili da una sola persona;
6. il sistema dovrebbe essere facile, non richiedendo conoscenza di un gran numero di regole né tantomeno eccessive fatiche mentali.

Il contrario del principio di Kerckhoff è il principio della **security by obscurity**, che si rileva sempre molto inefficiente.

Il crittoanalista, ovvero chi tenta di intromettersi nella comunicazione cifrata e scoprire *P*, può dover risolvere problemi di tre tipi principali:

- **ciphertext-only**: ha a sua disposizione solo *C*;
- **known plaintext**: ha a sua disposizione *C*, ma anche alcune corrispondenze fra *C* e *P* (non il *C* attuale);

- **chosen plaintext**: ha a sua disposizione qualsiasi *P* da poter trasformare in *C*.

## Cifrari a sostituzione

Sono tipi di cifrari in cui ogni carattere (o bit) viene sostituito con un altro. Uno dei cifrari più antichi conosciuti è il **Cifrario di Cesare** la cui sua generalizzazione è: "spostare l'alfabeto del testo in chiaro di *k* lettere in avanti per cifrare; spostare l'alfabeto del testo cifrato di *k* lettere indietro per decifrare". Nel Cifrario di Cesare *k* = 3, quindi l'alfabeto subisce la seguente mutazione:

- Testo in chiaro: a b c d e f g h i j k l m n o p q r s t u v w x y z
- Testo cifrato: D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

È stato molto efficace per i Romani, ma in generale è un algoritmo molto scarso. Un buon miglioramento consiste nel far sostituire a ogni lettera una lettera casuale dell'alfabeto, come in questo esempio:

- Testo in chiaro: a b c d e f g h i j k l m n o p q r s t u v w x y z
- Testo cifrato: Q W E R T Y U I O P A S D F G H J K L Z X C V B N M

Questo tipo di algoritmo, chiamato **monoalphabetic substitution** (**sostituzione monoalfabetica**), è molto più efficiente, in quanto sono possibili 26! – 1 possibili combinazioni. Sono possibili sostituzioni anche di digrammi (due caratteri) o trigrammi (tre caratteri) e il principio resta uguale.

Il grosso limite di questo tipo di cifrario è che nonostante le lettere siano in posizione casuale, è cambiato solo il loro simbolo di rappresentazione, ma le proprietà statistiche (come le lettere, i digrammi, i trigrammi più frequenti) o la conoscenza della lingua o dell'argomento del messaggio cifrato possono aiutare a scartare molte di queste possibili combinazioni. Un modo per limitare questo tipo di problema è l'utilizzo di **lipogrammi**, ovvero testi in cui vengono rimosse alcune lettere (per un umano resta comprensibile il testo, per una macchina no).

## Cifrari a trasposizione

Il metodo dei **cifrari a trasposizione** (**transposition cipher**) riordinano le lettere ma non le mascherano. In questo cifrario la chiave è una parola che non ha lettere ripetute e che serve a numerare le colonne (la prima colonna è quella relativa alla lettera che viene prima nell'alfabeto. Se la chiave è lunga *n*, si scrive il testo spezzandolo in più righe di *n* caratteri fino alla fine del messaggio, aggiungendo caratteri di riempimento. Di seguito un esempio:

**M E G A B U C K**

7 4 5 1 2 8 3 6

p l e a s e t r

a n s f e r o n

e m i l l i o n

d o l l a r s t

o m y s w i s s

b a n k a c c o

u n t s i x t w

o t w o a b c d

Testo in chiaro: pleasetransferonemilliondollarstomyswissbankaccountsixtwotwo

Chiave: MEGABUCK

Testo cifrato: afllsksoselawaiatoossctclnmomantesilyntwrnntsowdpaedobuoeriricxb

Sono molto più faticosi da decifrare, ma lo sono comunque (ad esempio con il metodo dell'anagramma multiplo).

### Blocchi monouso

Il metodo dei **blocchi monouso** (**one-time pad**) è il metodo più sicuro per la cifratura di messaggi. Perché allora non è il metodo più usato? Per i suoi svantaggi che ne scoraggiano l'uso pratico.

Questo tipo di cifrario si realizza prendendo *P* e convertendolo in una stringa di bit (ad esempio convertendo ogni carattere nella sua rappresentazione ASCII), applicando infine lo XOR bit-a-bit con una chiave, anch'essa stringa di bit, che deve essere **casuale**, **lunga come il messaggio**, **usata una sola volta e poi buttata** (se viene riusata e poi scoperta, si rischia che un messaggio venga decifrato). Se i vincoli nella chiave vengono rispettati, questo metodo può essere utilizzato in tutta tranquillità e sicurezza. Un esempio di utilizzo di questo metodo sono le chiavette che generano codici univoci per accedere ai siti delle banche (chiamate per l'appunto chiavette OTP).

## Algoritmi a chiave simmetrica

Capitolo 8.2 della 5° edizione del libro "Reti di calcolatori" di A. S. Tanenbaum

Gli algoritmi a chiave simmetrica sono classi di algoritmi che usano la stessa chiave per cifrare e decifrare i messaggi.

Un esempio di algoritmi a chiave simmetrica sono i **block cipher** (**cifrari a blocco**), chiamati così perché prendono il messaggio in chiaro, lo dividono in blocchi di *n* bit che poi cifrano uno ad uno generando infine blocchi di *n* bit cifrati. Possono essere implementati in hardware, per aumentare la velocità, o in software. Ciò che fanno questi algoritmi è eseguire in sequenza permutazioni e sostituzioni, mediante:

- **P-Box** (**permutation box**) che effettuano una **permutazione** dei bit in ingresso, e sono molto veloci perché non fanno alcun calcolo;
- **S-Box** (**serial box**) che effettuano una **sostituzione** dei bit in ingresso con altri.

L'esecuzione a cascata di P-Box e S-Box forma un **cifrario prodotto**.

### DES

Il **DES** (**Data Encryption Standard**) è l'algoritmo standard nazionale in uso dal governo statunitense dal 1977 al 2001 per lo scambio di informazioni non riservate. Nella sua forma originale non è più sicuro, ma in una forma modificata è ancora utile.

DES è un cifrario a blocco con blocchi di 64 bit, chiave per cifrare di 56 bit (in origine 128 bit, poi ridotta perché giudicata troppo potente), con 19 stadi (di cui il primo e l'ultimo sono trasposizioni verso/da bit del messaggio).

Il DES ha la proprietà che $E_K(D_K(P)) = P$.

### Triplo DES

Nel 1979 ci si rese conto che la chiave di DES era troppo corta, ma venne trovato il modo di allungarla senza dover stravolgere l'algoritmo o cambiarlo completamente. Il principio del **Triplo**

**DES**, come dice il nome, sfrutta la proprietà del DES che $E_K(D_K(P)) = P$ e l'utilizzo di due chiavi e tre stadi.

Quindi l'algoritmo è lo stesso del DES, viene aggiunta solamente un'altra chiave $K_2$ oltre a $K$ (ora $K_1$) della stessa lunghezza e calcolato $C = E_{K1}(D_{K2}(E_{K1}(P)))$. La decifratura avviene nel modo inverso, ovvero $P = D_{K1}(E_{K2}(D_{K1}(C)))$. La motivazione dell'uso di una funzione *E-D-E* al posto di un *E-E-E* era il mantenimento della compatibilità con computer che non supportavano il Triplo DES, così due computer, uno supportante Triplo DES e l'altro no, potevano parlare usando due chiavi $K_1 = K_2$.

## AES

**AES** (**Advanced Encryption Standard**) è l'algoritmo sostitutivo di DES adottato dal governo statunitense per gli stessi scopi del precedente dal 2001.

L'algoritmo aveva come nome originale **Rijndael** (fusione del nome dei ricercatori belgi inventori dell'algoritmo, Vincent Rijmen e Joan Daemen) e fu scelto nel concorso indetto dal NIST (National Institute of Standards and Technology) per trovare un sostituto al DES.

Rijndael è un cifrario a blocco con blocchi e chiavi da 128 a 256 bit, scelti indipendentemente, con passi da 32 bit. AES specifica che i blocchi devono essere da 128 bit, le chiavi da 128, 192, 256 bit. Di fatto vengono implementati con 128 bit di blocco e chiavi da 128 bit o 256 bit.

## Blowfish e Twofish

**Blowfish** è un altro algoritmo di cifratura, inventato da Bruce Schneier nel 1993 (prima di AES) ancora oggi funzionante.

È un cifrario a blocco, con blocchi da 64 bit e chiavi da 32 bit a 448 bit. Ciò che lo rende meno attraente è che, a parità di chiave (di lunghezza), è più lento di AES.

**Twofish** è una versione avanzata di Blowfish, che cerca di colmarne le lacune. Varia nella dimensione del blocco (128 bit) e nelle chiavi (da 128 bit a 256 bit) che sono meno ma partono da valori più grandi. Inoltre, Twofish è finalmente più veloce di AES (nelle chiavi da 256 bit).

## Modalità di cifratura: ECB

Nonostante i cifrari a blocco descritti siano molto complessi e di fatto molto efficaci, alla fine non sono altro che dei complessi cifrari a sostituzione monoalfabetica con caratteri molto lunghi ("caratteri" da 64 bit per DES, da 128 bit per AES) poiché se si prende lo stesso testo in chiaro con la stessa chiave si ottiene lo stesso testo cifrato.

La modalità con cui operano AES e DES viene detta **ECB** (**Electronic Code Book**), in cui sostanzialmente ogni blocco è una pagina del libro, cifrata: è possibile non conoscere come la pagina sia cifrata, ma se si conosce più o meno il suo contenuto e può essere interessante, perché non appropriarsene? In pratica, è un modo semplice per disturbare il funzionamento dei cifrari a blocchi, modificando l'ordine dell'output.

## Modalità di cifratura: Cipher block chaining

Per evitare problemi di questa natura si utilizza il **cipher block chaining** in cui ogni blocco di testo in chiaro viene messo in XOR con il blocco cifrato precedente e di questo viene effettuata la cifratura. Il primo blocco, non avendo un precedente, viene messo in XOR con un **IV** (**Initialization Vector**), trasmesso in chiaro assieme al testo cifrato.

La sequenza di cifratura è quindi:

- $C_0 = E(P_0 \text{ XOR IV})$

- $C_1 = E(P_1 \text{ XOR } C_0)$
- …
- $C_i = E(P_i \text{ XOR } C_{i-1})$

La sequenza di cifratura avviene nello stesso modo, con *D* al posto di *E*. È quindi facile intuire che lo stesso testo in chiaro, in posizioni diverse, dà origine a blocchi cifrati diversi.

### Modalità di cifratura: Stream cipher

Se ci sono applicazioni per cui è inaccettabile che un errore di trasmissione possa rovinare l'intero blocco esiste un'ulteriore modalità: **stream cipher**.

In questa modalità viene cifrato IV con una chiave crittografica per ottenere un blocco in uscita, che viene poi cifrato per produrre il secondo blocco, e così per il terzo, quarto, ecc.

La sequenza di blocchi in uscita viene chiamata **keystream** (**flusso di chiavi**) e viene usato come blocco monouso applicato in XOR al testo in chiaro per ottenere il testo cifrato. IV è usato solo al primo passo e il keystream è indipendente dai dati e può essere calcolato in anticipo. Inoltre, gli errori in un bit nel testo cifrato non si ripercuoteranno sugli altri blocchi.

È fondamentale in stream cipher non riusare la stessa coppia (IV, chiave) o si rischia un **keystream reuse attack** (che permette di scoprire il contenuto di alcuni blocchi a partire da altri e poi dalla stessa chiave scoperta).

### Modalità di cifratura: Counter

Un ulteriore modalità di cifratura è data da **counter** che permette di accedere ai blocchi casualmente e non sequenzialmente (come permette ECB e non gli altri). L'obiettivo viene raggiunto utilizzando un IV e una chiave costante: l'IV viene cifrato con la chiave e si mette in XOR con il primo blocco, per il secondo blocco si cifra con la chiave IV+1 e si mette in XOR con il secondo blocco e così via. La decifrazione avviene in maniera inversa. Anche questa modalità è affetta dal problema del keystream reuse attack, quindi è ancora importante che la coppia (IV, chiave) non venga ripetuta.

## Algoritmi a chiave pubblica

Capitolo 8.3 della 5° edizione del libro "Reti di calcolatori" di A. S. Tanenbaum

Il problema più grande negli algoritmi a chiave simmetrica è dovuto allo scambio di chiavi, che deve essere per forza svolto, ma in un ambiente non sicuro. Bisogna trovare un modo per scambiare le chiavi in un ambiente **esterno** e **sicuro**.

L'idea è stata trovata da due ricercatori, Diffie e Hellman, che proposero un sistema con chiavi di cifratura e decifratura diverse, in cui la chiave di decifratura è difficile da derivare da quella di cifratura. La proposta aveva tre requisiti:

- ***D*(*E*(*P*)) = *P***, sapendo che *D* ed *E* non funzionano più allo stesso modo;
- *D* è difficilmente deducibile da *E*;
- *E* non può essere forzato da un attacco di tipo **chosen plaintext**.

Ogni utente avrà quindi a disposizione l'algoritmo di cifratura, pubblico, una **chiave pubblica** (pubblica perché tutti la possono e devono conoscere ), usata dagli altri per spedirgli messaggi cifrati che solo lui può decifrare avendo a disposizione la propria **chiave privata** (segreta).

## RSA

**RSA** (**Rivest**, **Shamir**, **Adleman**) è un algoritmo crittografico a chiave pubblica, che prende il nome dai suoi inventori, considerato molto robusto e ancora oggi difficilmente forzabile. È la base di molte applicazioni nel campo della sicurezza.

Il maggior svantaggio di questo algoritmo è la grandezza della chiave, che per essere considerata buona deve essere di almeno 1024 bit, che lo rende lento nel processo di cifratura/decifratura.

Il funzionamento di RSA si basa sul fatto che le chiavi private e pubbliche vengono generate a partire da due numeri primi molto grandi (di 1024 bit almeno) ***p*** e ***q***, e la comunicazione avviene scambiandosi ***p**q*** e le chiavi pubbliche. Se un attaccante riuscisse a fattorizzare ***p**q*** tutta la sicurezza sparirebbe, ma è un problema molto difficile da risolvere. Per complicare la cosa, spesso si richiede di ingrandire la chiave, inventando nuove versioni dell'algoritmo.

# Firme digitali

Capitolo 8.4 della 5° edizione del libro "Reti di calcolatori" di A. S. Tanenbaum

## Message digest

I **message digest** (***MD***) sono funzioni di tipo hash monodirezionale che prendono come input un testo di lunghezza arbitraria e restituiscono come output una stringa di lunghezza fissa. Gli MD hanno quattro proprietà importanti:

- dato *P*, calcolare *MD*(*P*) è facile;
- dato *MD*(*P*), è praticamente impossibile trovare *P*;
- dato *P*, non è possibile trovare *P'* tale che *MD*(*P*) = *MD*(*P'*);
- se *P* differisce anche solo di 1 bit, l'output diventa completamente diverso.

L'hash dovrebbe essere lungo almeno 128 bit per non trovare un *P'* tale che *MD*(*P*) = *MD*(*P'*), preferibilmente di più.

Alcuni esempi di message digest sono:

- **SHA-1** (**Secure Hash Algorithm 1**), che elabora i dati in blocchi da 512 bit e genera una stringa di 160 bit;
- **SHA-2** (**Secure Has Algorithm 2**), evoluzione di SHA-2;
- **MD5** (**Message Digest 5**), che è il quinto di una serie di message digest inventati da uno degli inventori di RSA, che elabora i dati in blocchi da 128 bit, oggi però in fase di abbandono poiché si sono trovate numerose falle.

Un esempio di attacco agli hash si chiama **preimage attack**, che consiste nel trovare un messaggio che genera un hash dato oppure due messaggi con lo stesso hash. Se un hash è sicuro e genera *n* bit, ci vogliono circa $2^n$ tentativi per trovare l'hash corrispondente.

## Attacco del compleanno

L'**attacco del compleanno** (**happy birthday attack**) si basa sul fatto che se si ha un message digest che genera un testo di $2^m$ bit, si suppone servano circa $2^m$ tentativi per trovare quello corretto. Sperimentalmente, succede che la probabilità è invece $2^{m/2}$.

È basato sull'idea di trovare la percentuale di probabilità che due persone in una stanza abbiano lo stesso giorno di nascita.

# Sicurezza delle comunicazioni

Capitolo 8.6 della 5° edizione del libro “Reti di calcolatori” di A. S. Tanenbaum

## IPsec

**IPsec** (**IP Security**) è un protocollo ideato per colmare i vuoti di sicurezza di IP. Alcuni servizi che può fornire sono la segretezza, l’integrità dei dati e protezione dagli attacchi di tipo ripetizione. IPsec rende IP un protocollo orientato alla connessione.

IPSec in modalità **tunnel** funziona incapsulando l’intero pacchetto IP, header compreso, in un nuovo pacchetto IP con un header completamente diverso. Questo “involucro” a IP in questo caso si chiama **ESP** (**Encapsulating Security Protocol**). Nell’header ESP sono presenti i seguenti campi:

- **Security parameters index** (32 bit): identificatore della connessione;
- **Sequence number** (32 bit): numero univoco del pacchetto di una connessione;
- **Initialization vector**: serve per la cifratura dei dati (non fa parte dell’header, ma è spesso dopo i precedenti numeri);
- **Authentication data (HMAC)** (32 bit): serve per autenticare l’header usando funzioni di hash, ma viene incluso in coda al payload (così può essere calcolato mentre i bit vengono trasmessi verso l’interfaccia di rete).

## Firewall

I **firewall** sono apparecchi hardware o software che filtrano il traffico di pacchetti in ingresso e in uscita in una rete: quelli che rispettano dei criteri decisi dall’amministratore di rete possono proseguire, gli altri vengono fermati. I criteri possono essere ad esempio il permesso/divieto di usare alcune porte TCP.

Generalmente i firewall sono di due grandi categorie:

- **stateless**: filtrano i pacchetti in base a come sono fatti (ad esempio: blocco di pacchetti da un certo IP). Hanno il vantaggio di svolgere i loro compiti molto rapidamente;
- **stateful**: filtrano i pacchetti in base al loro stato (ad esempio: in base alla sessione attiva) facendo una selezione più accurata. Sono più onerosi degli stateless, ma sono più efficaci.

I firewall hanno anche il compito di cercare di evitare attacchi come:

- **SYN attack**, in cui vengono inviati migliaia di pacchetti TCP con flag SYN=1, legittimi, per richiedere di aprire una connessione TCP con l’host vittima che dovrà allocare un buffer per ognuno e spedire pacchetti con flag SYN=1 e ACK=1, che non verranno considerati. Con tutti i buffer occupati, la vittima sarà fuori servizio;
- **DoS** (**Denial of Service**), attacchi generali che rendono la vittima non più in grado di operare, ma non vengono rubati dati;
- **DDoS** (**Distributed Denial of Service**), è un attacco DoS meno riconoscibile perché vengono controllate molte macchine “zombie” cui si fa fare richieste alla macchina vittima fino a che va fuori servizio.

## Sicurezza wireless

Le reti wireless hanno nello standard il requisito di prevenire che un nodo wireless legga o interferisca con i messaggi spediti tra altri due nodi wireless. L’attuale standard è **WPA2**, sostituto di **WPA**, a sua volta sostituto di **WEP**.

**WEP** (**Wired Equivalent Privacy**): prima generazione di protezione per le reti 802.11; cifrava i dati mettendoli in XOR con l'output di uno **stream cipher** denominato **RC4** (**Rivest Cipher 4**). La prima versione, WEP-40, aveva chiavi da 40 bit a cui andava aggiunto un IV lungo 24 bit. La dimensione massima della chiave fu poi incrementata fino a 128 bit, mentre l'IV venne lasciato com'era, con tutti i problemi che ne conseguirono: 24 bit significano $2^{24}$ = 16777216 possibili valori, finiti i quali andavano riutilizzati e quindi si era facilmente vulnerabili a keystream reuse attack (a 3000 pacchetti al secondo, in un'ora e mezza i valori dell'IV terminano).

**WPA** (**Wi-Fi Protected Access**): seconda generazione di protezioni per le reti 802.11; cifra i dati usando **TKIP** (**Temporal Key Integrity Protocol**) che è basato su RC4. La chiave è di 128 bit ed è nota come **WPA-PSK** (**WPA Pre Shared Key**), utilizzabile come passphrase dagli utenti. Il problema di questo protocollo nasce soprattutto dalla passphrase, che deve essere inventata dagli utenti, che raramente generano password lunghe e casuali (non riescono a ricordarle) e quindi le creano brevi e suscettibili di attacchi di tipo dictionary.

**WPA2** (**Wi-Fi Protected Access 2**): terza generazione di protezioni per le reti 802.11, derivato dallo standard **802.11i**; cifra i dati usando **CCMP** (**Counter mode with Cipher block chaining Message authentication code Protocol**). CCMP usa chiavi di 128 bit e blocchi di 128 bit.

# Protocolli di autenticazione

Capitolo 8.7 della 5° edizione del libro "Reti di calcolatori" di A. S. Tanenbaum

## Replay attack

L'attacco di tipo ripetizione (**replay attack**) è un attacco che avviene quando l'attaccante intercetta il traffico di una comunicazione, lo copia e lo ritrasmette. È un attacco che può essere risolto inserendo un timestamp all'interno dei pacchetti, che devono essere rifiutati dal destinatario se troppo vecchi, o un numero univoco che il destinatario possa verificare.

## DNS spoofing

Il **DNS spoofing** è una versione meno riconoscibile di **MITM** (**Man In The Middle**).

MITM è un tipo di attacco in cui l'attaccante si intromette nella comunicazione fra due host e fa da intermediario intercettando tutto il traffico e volendo modificandolo a suo piacimento.

Il principio del DNS spoofing è similare, ma l'attaccante non si mette fra i due host a intercettare il traffico, ne resta "fuori".

Quando il DNS deve fare il refresh dei suoi dati (il TTL del record DNS è 0) deve inviare con UDP a un altro server DNS di livello superiore una richiesta per conoscere i nuovi dati. L'attaccante procede quindi intromettendosi in questo punto, rispondendo al server DNS che ha fatto la richiesta che procederà ad aggiornare i suoi dati con quelli fasulli (potenzialmente l'indirizzo IP della macchina con cui viene fatto l'attacco).

A questo punto la vittima che si connette al sito web con il record appena modificato non si connetterà dove crede, ma al sito web dell'attaccante, senza accorgersi di nulla.