

# RETI DI CALCOLATORI

## Sommario

<b>LEZIONE 1 – INTRODUZIONE ALLE RETI .....</b>	4
1.1 Tipologie di Reti .....	4
1.2 Tecnologie di comunicazione .....	5
1.3 Topologie .....	6
1.4 Struttura di una rete .....	8
<b>LEZIONE 2 – MODELLI DI RETE .....</b>	9
2.1 Protocolli di comunicazione .....	9
2.2 Il modello ISO/OSI.....	11
2.3 Tipologie di Connessione .....	14
2.4 Affidabilità di un servizio .....	15
<b>LEZIONE 3 – MODELLO TCP/IP E CONFRONTO .....</b>	16
3.1 Modello TCP/IP .....	16
3.2 Livello Accesso Rete .....	16
3.3 Livello Internet.....	17
3.4 Livello Trasporto .....	17
3.5 Livello Applicazione .....	17
<b>LEZIONE 4 – LIVELLO FISICO E TEORIA DEI SEGNALI.....</b>	18
4.1 Basi della trasmissione.....	18
4.2 Mezzi di Trasmissione .....	18
4.3 Tipologia di segnale .....	19
4.4 Analisi Spettrale .....	19
4.4.1 Teorema di Fourier.....	19
4.4.2 Teorema di Nyquist .....	21
4.4.3 Rapporto Segnale/Rumore .....	21
4.4.4 Teorema di Shannon .....	21
4.5 Trasmissione .....	22
4.6 Mezzi trasmissivi.....	22
4.7 Topologie di reti in fibra ottica .....	24
4.8 Trasmissioni wireless .....	25
4.9 Classificazione onde elettromagnetiche .....	25
4.10 Trasmissione e modulazione.....	26

<b>LEZIONE 5 - SISTEMA TELEFONICO E MODULAZIONE .....</b>	27
<b>LEZIONE 6-7-8-9 – IL LIVELLO DATA LINK .....</b>	33
<b>6.1 Introduzione .....</b>	33
<b>6.2 Progettazione del livello Data Link .....</b>	33
<b>6.3 Operazioni del livello due .....</b>	34
<b>6.4 Suddivisione in frame .....</b>	34
<b>6.5 Rilevazione e correzione degli errori .....</b>	35
<b>6.6 Codici a correzione di errore.....</b>	36
<b>6.7 Codici a rilevazione d'errore.....</b>	40
6.8 Comunicazione ed ACK.....	41
6.9 Struttura di un frame .....	42
6.10 Gestione di sequenza di trasmissione e flusso .....	42
6.10.1 Protocollo Heaven.....	42
<b>6.10.2 Protocollo Stop and Wait.....</b>	43
6.10.3 Protocolli a finestra scorrevole .....	46
6.10.4 Protocolli go-back-n e selective repeat.....	49
6.11 Protocolli del livello Data Link.....	50
6.11.1 HDLC .....	50
6.11.2 SLIP .....	51
6.11.3 PPP .....	51
6.12 Il sottolivello MAC (Medium Access Control).....	52
6.12.1 Protocollo ALOHA .....	53
6.12.2 Protocolli CSMA (Carrier Sense Multiple Access) .....	56
6.12.3 Protocolli CSMA/CD (CSMA with Collision Detection) .....	57
6.12.4 Le reti ad anello .....	57
6.13 Lo standard 802 .....	59
6.13.1 IEEE 802.3 .....	60
6.13.2 Fast Ethernet.....	66
6.13.3 IEEE 802.5 .....	66
6.13.4 IEEE 802.2 .....	69
6.14 I Bridge.....	69
6.14.1 Standard IEEE per i bridge.....	71
6.14.2 Switch .....	72
<b>LEZIONE 10 – LIVELLO NETWORK .....</b>	74
10.1 Servizi offerti.....	74
10.2 Algoritmi di Routing .....	75

10.3 Principio di ottimalità.....	75
10.4 Classificazione algoritmi di routing .....	76
10.5 Algoritmi di routing statici .....	76
10.5.1 Shortest path routing.....	76
10.5.2 Flooding .....	76
10.5.3 Flow-based routing .....	77
10.6 Algoritmi dinamici.....	77
10.6.1 Distance vector routing.....	77
10.6.2 Link state routing .....	79
10.7 Routing gerarchico .....	80
10.8 Controllo della congestione .....	81
10.8.1 Approccio Traffic shaping (open loop) .....	81
10.8.2 Approccio Choke packet (closed loop).....	83
10.9 Internetworking .....	84
10.10 Internetwork routing .....	86
<b>LEZIONE 11 – IL LIVELLO NETWORK (PARTE 2) .....</b>	<b>87</b>
11.1 Il livello network in Internet.....	87
11.2 Header IP .....	87
11.3 Indirizzi IP.....	88
11.4 Routing IP.....	91
11.5 Subnet e Subnet Mask .....	92
11.6 Protocolli di Controllo .....	93
<b>ICMP (Internet Control Message Protocol, RFC 792) .....</b>	<b>93</b>
Interrogazioni ICMP .....	94
Comando ping.....	94
Comando traceroute/tracert .....	95
<b>ARP (Address Resolution Protocol, RFC 826) .....</b>	<b>95</b>
11.7 Esercizio Subnetting.....	96
<b>LEZIONE 12 – IL LIVELLO TRASPORTO TCP/UDP .....</b>	<b>100</b>
12.1 Protocollo TCP.....	100
12.2 Protocollo UDP.....	107
<b>LEZIONE 13 – IL LIVELLO APPLICATIVO.....</b>	<b>110</b>
DNS .....	110
Posta elettronica .....	115
FTP .....	115
<b>LEZIONE 14 – TELEFONIA CELLULARE .....</b>	<b>119</b>

LEZIONE 15 – NAT (Network Address Translation).....	121
LEZIONE 16 – Multimedia Network Application .....	125
Streaming audio video.....	125
Voice-and Video-over-IP .....	125
Streaming Audio Video Live.....	125
LEZIONE 17 – Streaming Stored Video.....	126
UDP Streaming.....	126
HTTP Streaming .....	126
Adaptive Streaming e DASH.....	127

## LEZIONE 1 – INTRODUZIONE ALLE RETI

Il primo argomento da introdurre quando si parla di reti di calcolatori è la differenza tra essi e i sistemi distribuiti.

**Un sistema distribuito** è un insieme di computer indipendenti che appare ai propri utenti come un singolo sistema coerente. Un classico esempio di sistema distribuito è il World Wide Web.

Altri esempi:

GRID Computing: risorse di calcolo distribuite su diverse aree geografiche

Cloud Computing: esecuzione remota di codice ed applicativi

Edge Computing: esecuzione di operazioni remote nel punto di acquisizione dei dati

**In una rete di calcolatori** mancano la coerenza, il modello ed il software tipico di un sistema distribuito. Gli utenti della rete vedono i singoli componenti di essi, un utente difatti per eseguire un programma che si trova su una specifica remota della rete deve collegarsi manualmente ad essa per eseguirlo.

### 1.1 Tipologie di Reti

Suddividiamo principalmente due tipologie di reti:

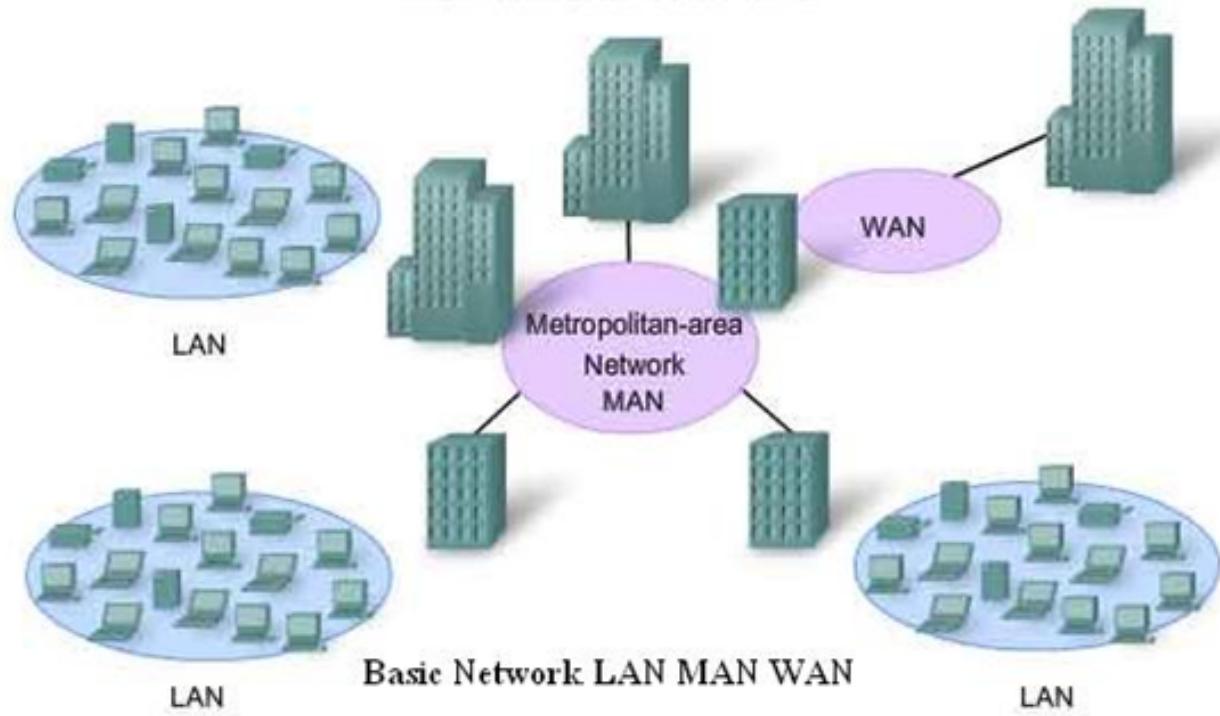
**Cablate:** (Rame, fibra ottica) da cui discendono le seguenti reti:

- LAN (Local Area Network) → rete locale/aziendale che va dai 10m – 1km
- MAN (Metropolitan Area Network) → rete metropolitana che va dai 100m – 10km
- WAN (Wide Area Network) → rete geografica che va dai 10km – 1000km

**Wireless:** (Radiofrequenze, infrarossi) da cui discendono le seguenti reti:

- WPAN (Wireless Personal Area Network) → rete di piccolissime distanze (cm). Rete tipica di NFC, Bluethoot, IR
- WLAN (Wireless Local Area Network) → come rete LAN, ma senza fili.
- WMAN (Wireless Metropolitan Area Network) → come rete MAN, ma senza fili.
- WWAN (Wireless WAN) → rete geografica wireless

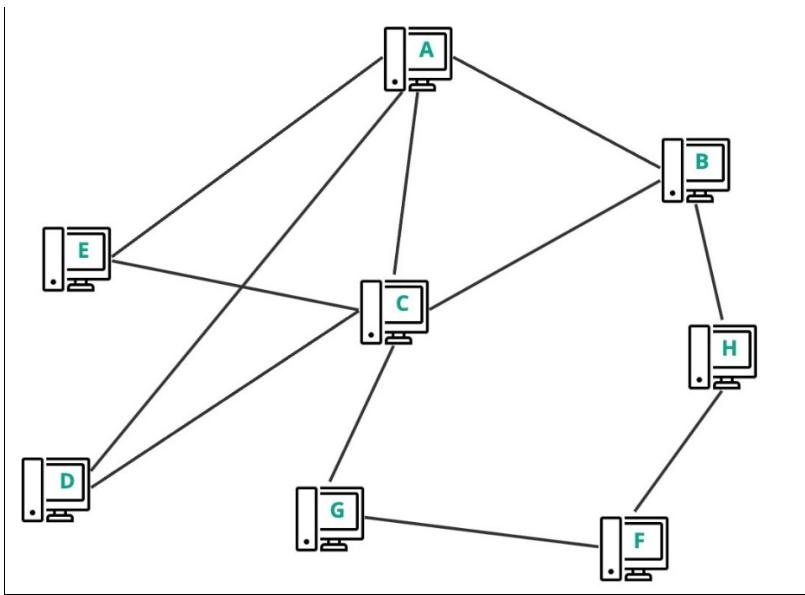
Diverse tipologie di reti possono essere interconnesse tra di loro formando reti sempre più grandi (Internet)



## 1.2 Tecnologie di comunicazione

Le reti di calcolatori dispongono di un'altra caratteristica principale: la tecnologia di trasmissione.  
Le tecnologie di trasmissione possiamo dividerle in due tipi:

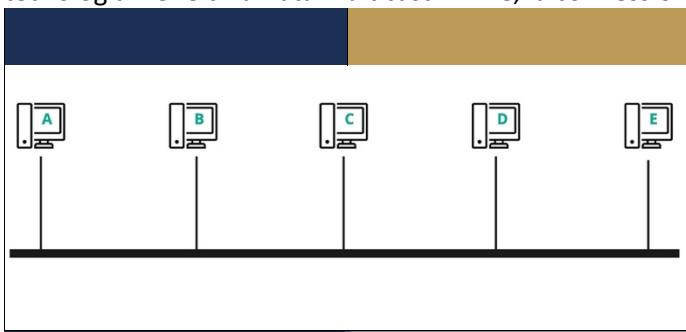
**1.Trasmissione punto a punto:** Ogni calcolatore deve connettersi direttamente ad un altro calcolatore. Se due calcolatori non sono direttamente collegati, è necessario creare un instradamento attraverso altri calcolatori. Bisogna sempre raggiungere un nodo nella rete.  
Risulta essere costoso per via delle numerose connessioni dedicate.



**2.Trasmissione broadcast:** sono dotate di un unico canale di comunicazione (bus). Ogni calcolatore ha un identificativo univoco (indirizzo di rete)

Il pacchetto viene inviato da una macchina a tutte le altre macchine, ma viene letto solamente dal destinatario esaminando il proprio indirizzo di rete.

Alcuni sistemi broadcast supportano la trasmissione a un sottoinsieme delle macchine, tale tecnologia viene chiamata multicast. Infine, la connessione di due o più reti è chiamata internetwork.



Non tutti i calcolatori possono trasmettere simultaneamente sul bus e quindi è necessario un sistema di regole che si dividono in:

**Regole statiche:** vengono prefissate e non possono cambiare nel tempo come, ad esempio, una politica di round robin con time slicing.

Ci saranno, ovviamente, spreco di tempo e risorse se un calcolatore non deve trasmettere.

**Regole dinamiche:** Di volta in volta si decide chi può utilizzare il mezzo di trasmissione.

Possono essere centralizzate (un'unità centrale decide chi può iniziare la trasmissione) o distribuite (un calcolatore può decidere se trasmettere in base allo stato del mezzo).

Possibili collisioni.

### 1.3 Topologie

La topologia è un concetto diverso dalla tipologia.

La topologia definisce come gli apparati di rete sono collegati tra di loro. Ogni elemento connesso nella rete è detto nodo e l'informazione scambiata è detta pacchetto.

Suddividiamo topologia logica (come i dati vengono scambiati tra i nodi) e topologia fisica (dislocazione fisica dei nodi. Ogni topologia risulterà essere un grafo)

Esistono varie topologie fisiche tra cui:

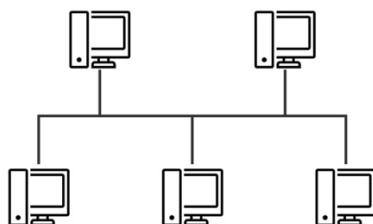
#### Rete a bus:

Le informazioni viaggiano su un unico canale.

Tutti i nodi possono leggere le informazioni in viaggio. In particolare, un nodo NON destinatario riceve il pacchetto, ma lo scarta. Un nodo destinatario lo legge.

**PRO:** Semplice da realizzare e da estendere.

**CONTRO:** Velocità ridotte.



#### Rete ad anello:

Le informazioni viaggiano su un unico canale.

Ci sono due modalità:

**Unidirezionale:** i pacchetti sono trasmessi in senso orario o antiorario.

**Bidirezionale:** i pacchetti sono trasmessi in entrambe le direzioni

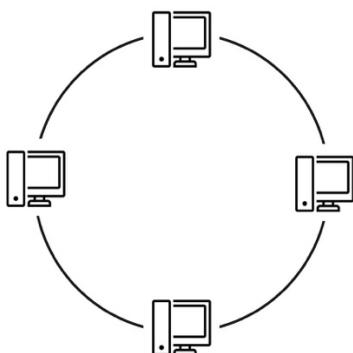
Nella rete ad anello, un nodo NON destinatario inoltra al successivo il pacchetto.

Un nodo destinatario lo legge e blocca l'inoltro.

Se il pacchetto torna al mittente, la comunicazione si interrompe ( destinatario non trovato)

**PRO:** Semplice da estendere e veloce

**CONTRO:** Bassa tolleranza ai guasti.



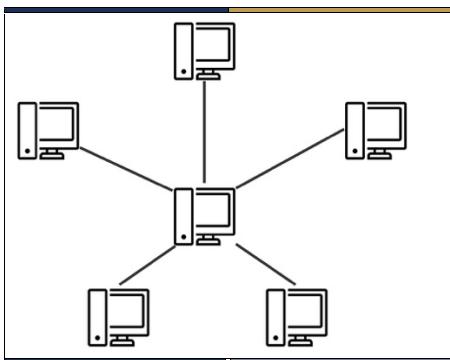
#### Rete a stella:

Esiste un nodo centrale che gestisce la comunicazione.

Le informazioni vengono inviate ad un nodo centrale che funge da "router" e indirizza il pacchetto verso il destinatario.

**PRO:** Semplice da realizzare, buone velocità

**CONTRO:** Tolleranza ai guasti parziale.

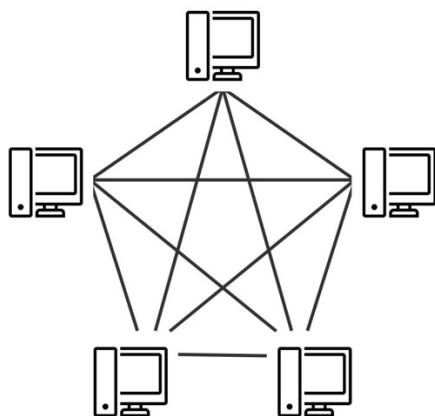


### Rete a maglia (mesh)

Si tratta di una rete in cui tutti i nodi sono collegati tra loro e ciascuno di essi ne riesce a raggiungere un altro attraverso un solo passaggio. Nel caso in cui uno dei cavi dovesse rompersi sarebbe possibile, comunque, l'arrivo a destinazione dei pacchetti. Si viene dunque a formare una maglia con percorsi multipli tra i nodi.

Tutti i nodi possono leggere le informazioni in viaggio.

Un nodo non destinatario riceve il pacchetto, ma lo scarta, mentre il nodo destinatario lo legge.



**PRO:** Massima velocità e tolleranza ai guasti.

**CONTRO:** Costo elevato e difficile realizzazione.

## 1.4 Struttura di una rete

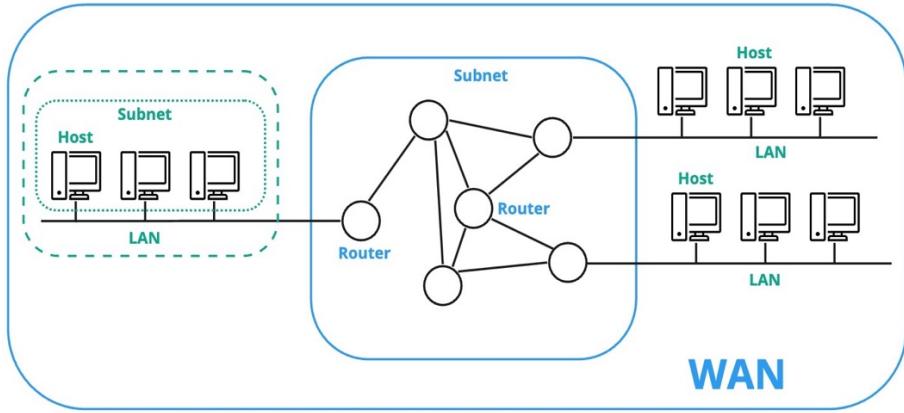
Una rete è strutturata dai seguenti componenti:

**Host** (unità di calcolo)

**Subnet** (Sottorete composta da tutti i nodi di una singola rete)

**Mezzo di trasmissione**

**Dispositivi di rete** che si occupano di instradare la comunicazione nella sottorete o verso le reti esterne → Modem, Gateway, Router, Switch ecc cc.



Anche se esistono regole standard per le reti queste sono eterogenee tra di loro, una rete può comunicare con un'altra di diverso tipo

Possano inoltre comunicare reti basate su mezzi di trasmissione differente

Cablata <→ Satellite

Radio <→ Cablata

Satellite <→ Radio

**Internetwork** - collega reti di diversa tipologia utilizzando dei router multiproocollo detti gateway

- ➔ LAN <→ MAN <→ WAN
- ➔ I gateway effettuano sia il routing (instradamento) sia la conversione di protocollo.

## LEZIONE 2 – MODELLI DI RETE

### 2.1 Protocolli di comunicazione

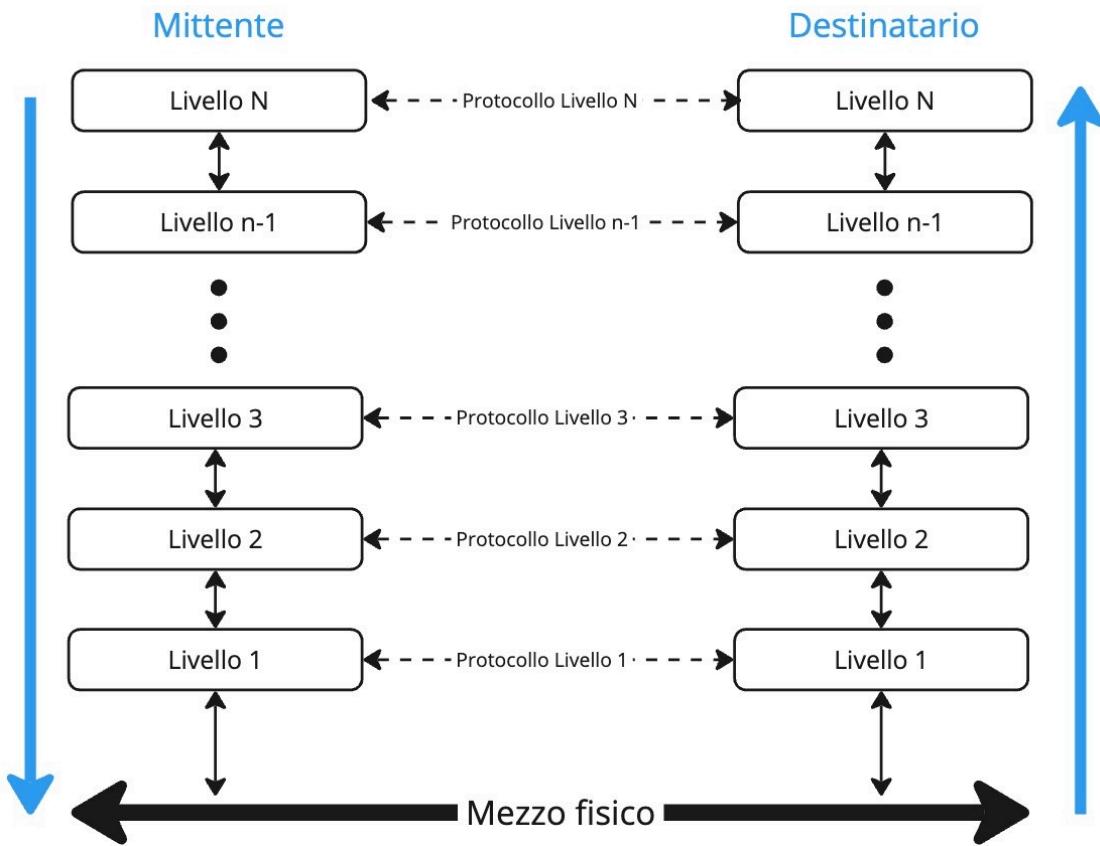
Visto che la rete è eterogenea (diverse tipologie e tecniche di comunicazione e scambio dati) è necessario definire un insieme di regole specifiche. Tali regole sono dette protocolli.

**Def:** Insieme coordinato di regole che consente a due interlocutori (**un utente e un calcolatore elettronico, due utenti oppure due calcolatori**) di scambiarsi rapidamente e univocamente dati e messaggi.

Una rete principalmente è organizzata a livelli (o strati) dove ogni livello ha una propria responsabilità nella rete.

Ogni elemento di un livello **n** comunica con lo strato successivo **n+1 (service user)** e quello precedente **n-1 (service provider)** mediante interfacce di comunicazione dette **SAP** (un livello NON può comunicare con livelli non adiacenti)

Ogni livello interagisce con il suo corrispettivo sull'host mittente/destinatario attraverso un protocollo. L'insieme dei protocolli di una singola architettura è detto **Stack di protocolli**.

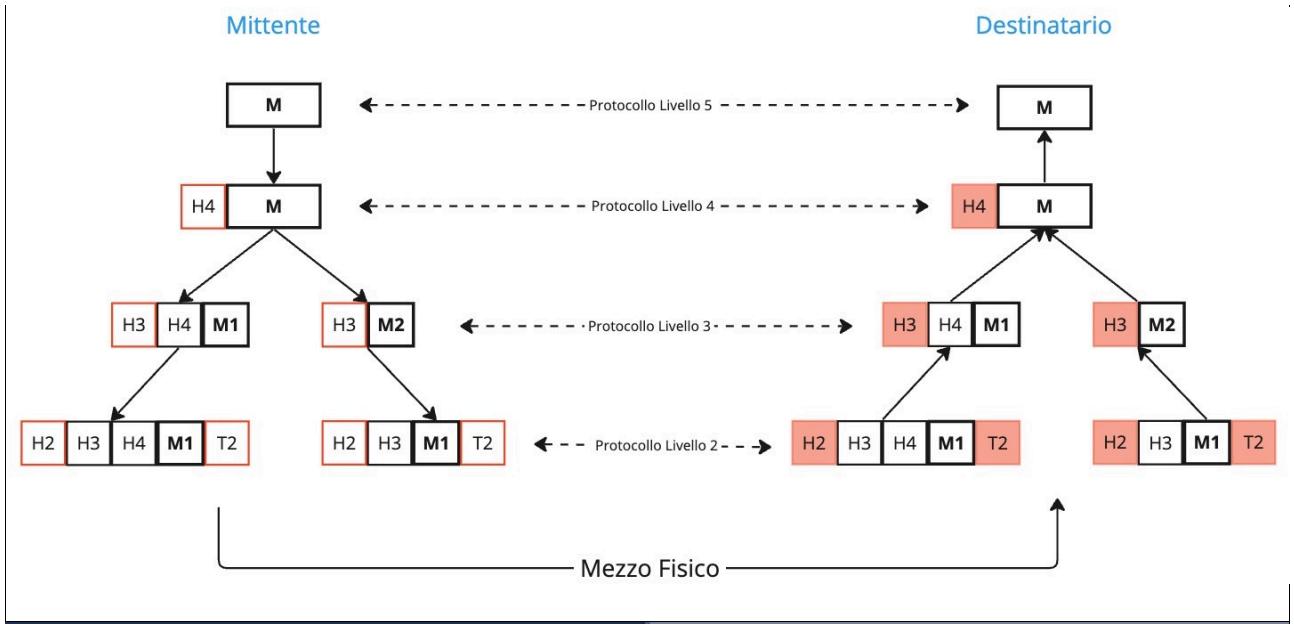


#### COME AVVIENE LA COMUNICAZIONE TRA LIVELLI ?

Facciamo un esempio con un'architettura a 5 livelli ( $N=5$ )

- 1) Un messaggio M parte dal livello più alto ( $n=5$ ).
- 2) Il messaggio viene inviato al livello  $n-1$  (4)
- 3) Il livello 4 inserisce un header con informazioni di controllo (H4).
- 4) Si passa il pacchetto al livello  $n-1$  (3)
  - Esiste una dimensione massima per ogni pacchetto
  - Il pacchetto deve essere scomposto in pacchetti più piccoli ( $M_1, M_2$ ).
  - Nuove informazioni di controllo vengono aggiunte ad ogni porzione ( $H_3$ ) N.B. Le informazioni di controllo precedenti non vengono suddivise!
- 5) Il pacchetto viene inviato al livello  $n-1$  (2). Vengono aggiunte informazioni di controllo ( $H_2$ ) ed informazioni dette trailer per determinare il termine del pacchetto ( $T_2$ ).
- 6) Il messaggio può essere inviato tramite mezzo fisico.
- 7) Si risale l'albero verso il livello più grande
- 8) Ad ogni livello  $n+1$  vengono lette le informazioni di controllo.

N.B. Le informazioni di controllo sono utilizzate dai protocolli dello stesso livello



Ogni livello ha un determinato compito, es:

- **Indirizzamento**,
- **Controllo errori di trasmissione**
- **Frammentazione**
- **Controllo degli errori**
- **Trasferimento di dati (Simplex, Half-Duplex, Full-Duplex)**
- **Multiplexing**

Le operazioni che un livello effettua per utilizzare un servizio sono dette **primitive** e coinvolgono il livello n e i suoi adiacenti.

Ogni primitiva ha associata una corrispettiva risposta con il livello da raggiungere.

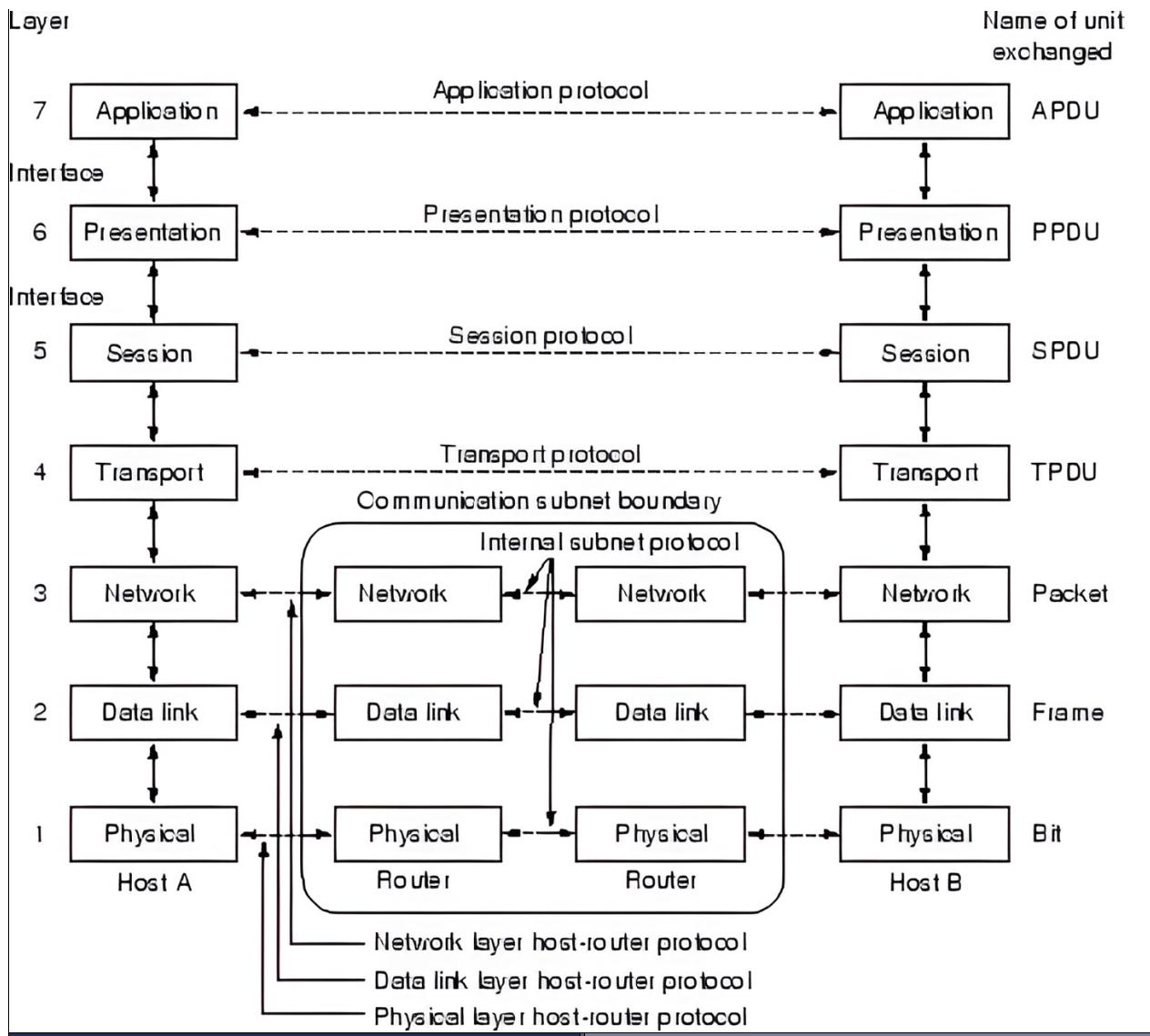
## 2.2 Il modello ISO/OSI

Il **modello ISO/OSI** è il modello di riferimento per le architetture di rete, difatti è chiamato standard de iure (standard di diritto poiché è nato dopo lo standard TCP/IP → de facto) e si tratta solo di uno schema concettuale che i produttori di hw di rete devono seguire.

E' un modello composto da 7 livelli (3 fisici, 4 applicativi).

Quando il mittente spedisce verso il destinatario si tratta di encapsulamento poiché verranno aggiunte informazioni.

Quando il destinatario riceve il messaggio si tratta di decapsulamento poiché ricostruisce il messaggio.



#### SINTESI DEL FUNZIONAMENTO DEI LIVELLI:

#	Livello	Definizione
7	Applicazione	Interfaccia tra il sistema di comunicazione e le applicazioni
6	Presentazione	Formatta e trasforma i dati in base la loro rappresentazione locale. Fornisce anche la cifratura/decifratura dei dati
5	Session	Si occupa delle sessioni di comunicazione, dall'inizializzazione alla chiusura
4	Trasporto	Invio e ricezione dei dati. Controllo e correzione (se possibile) degli errori
3	Rete	Creazione dei pacchetti, indirizzamento ed instradamento degli stessi ad alto livello (astrazione)
2	Data Link	Definizione del frame e dell'indirizzamento in funzione del mezzo fisico

**Livello Fisico:** Il livello fisico si occupa della trasmissione dei dati grezzi (bit) su un canale di comunicazione. Specifica le caratteristiche meccaniche, elettriche e procedurali dell'apparato di connessione.

Inoltre, specifica le caratteristiche del mezzo fisico tra cui le tensioni scelte, durata di un singolo bit, tipo di trasmissione ecc.

Problemi tipici di questo livello sono i tempi di trasmissione del segnale, come avviene l'inizio e la fine della comunicazione o se la comunicazione può avvenire simultaneamente in entrambe le direzioni.

BIT → FISICO(Mezzo, Segnale e trasmissione binaria)

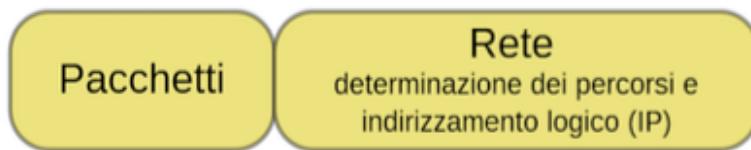
**Livello Data Link:** Il livello data link si occupa di trasformare i pacchetti ricevuti dal livello Network in frame che vengono inviati in sequenza al livello fisico.

Al termine del framing i dati diventano parte di un **nuovo pacchetto, dotato di un'intestazione (header) e di una coda (tail)**, che hanno la funzione anche di sequenza di controllo. Per ogni pacchetto ricevuto **il destinatario trasmette al mittente un segnale di ACK (Acknowledgment)**, ovvero di conferma di ricevuta. In questo modo il mittente è in grado di **capire quali pacchetti siano o meno arrivati a destinazione**. Nel caso di pacchetti corrotti, incompleti, persi o mal trasmessi, il mittente deve occuparsi della loro ritrasmissione.

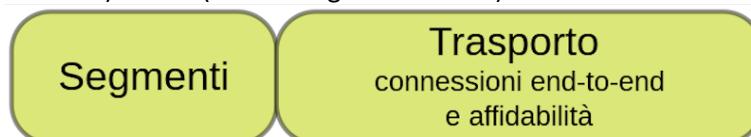
Il livello 2 **si occupa anche del controllo di flusso dei dati** e di tutta una serie di interventi correttivi nel caso in cui venga registrato uno sbilanciamento della velocità di trasmissione mediante un sottolivello chiamato MAC (Media Access Control).



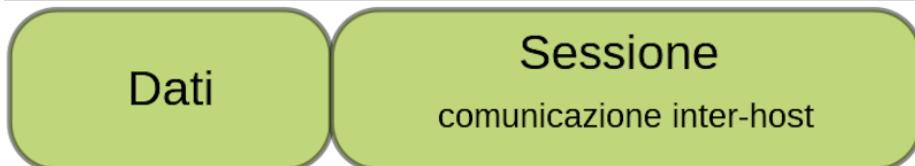
**Livello di Rete (Network Layer):** Questo livello si occupa del routing dei dati attraverso una rete composta da più collegamenti e nodi. I principali compiti includono l'instradamento dei pacchetti, la gestione degli indirizzi IP e il controllo del traffico per evitare congestioni. Il protocollo IP (Internet Protocol) è uno degli esempi più noti di protocollo di livello di rete.



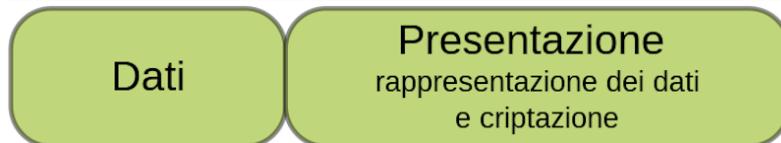
**Livello di Trasporto (Transport Layer):** Il livello di trasporto si concentra sulla trasmissione end-to-end dei dati tra i dispositivi. Fornisce servizi come la segmentazione dei dati in pacchetti più piccoli, il controllo degli errori e il controllo del flusso. Due protocolli chiave a questo livello sono **TCP** (Transmission Control Protocol) e **UDP** (User Datagram Protocol).



**Livello di Sessione (Session Layer):** Questo livello è responsabile dell'apertura, della gestione e della chiusura delle sessioni di comunicazione tra dispositivi. Aiuta a stabilire e mantenere il dialogo tra le applicazioni su entrambi i lati della comunicazione. Può includere funzionalità come il controllo della sincronizzazione e la gestione delle sessioni.



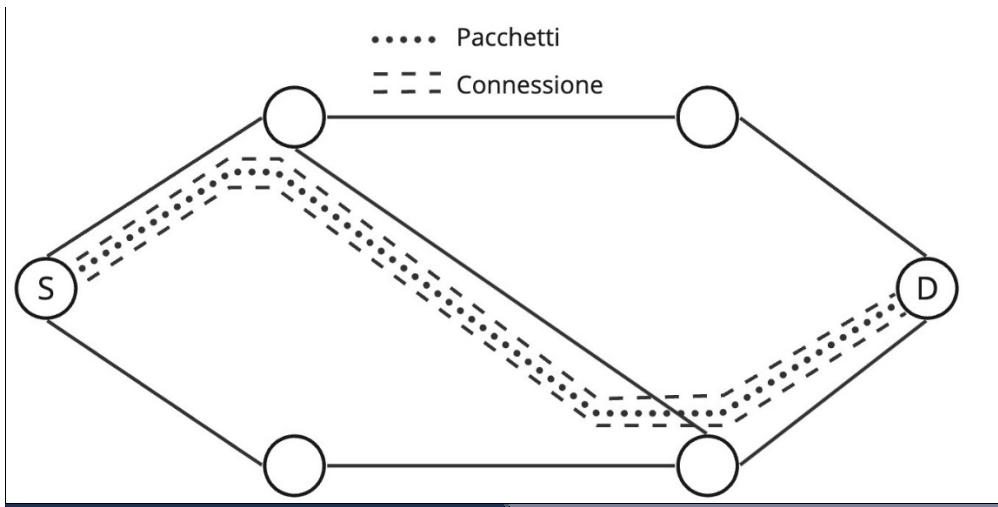
**Livello di Presentazione (Presentation Layer):** Il livello di presentazione si occupa della traduzione, dell'elaborazione e della conversione dei dati in un formato comune comprensibile da entrambi i dispositivi. Questo livello può gestire la crittografia per la sicurezza dei dati, la compressione per ridurre la larghezza di banda necessaria e la conversione dei formati dati.



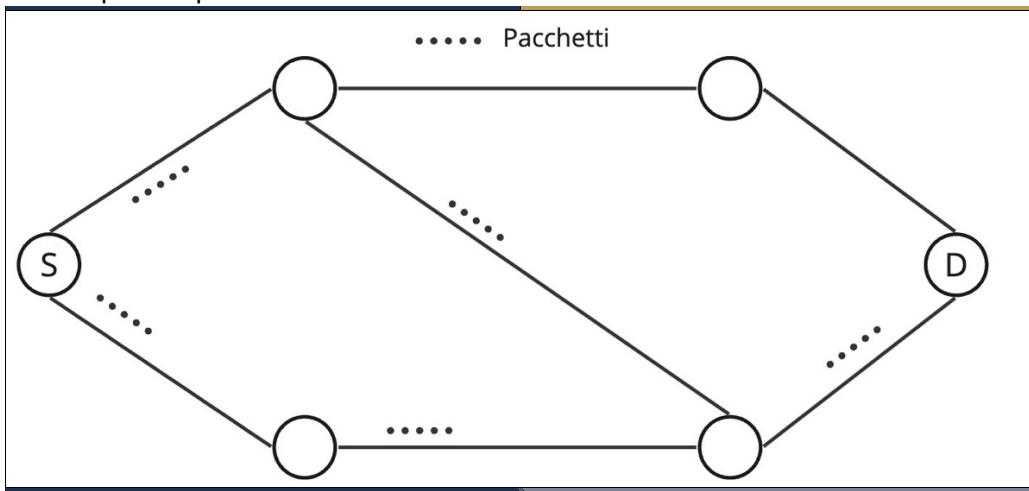
**Livello di Applicazione (Application Layer):** Questo è il livello più alto e fornisce servizi direttamente alle applicazioni utente. Include una vasta gamma di protocolli e servizi che consentono alle applicazioni di comunicare attraverso la rete. Esempi di protocolli di livello di applicazione includono HTTP per il web, SMTP per la posta elettronica, FTP per il trasferimento di file e molti altri.

### 2.3 Tipologie di Connessione

**Connection-Oriented:** Si stabilisce la connessione, si stabilisce un percorso (instradamento), si effettua la comunicazione inviando i pacchetti ed infine si rilascia la connessione.  
Il percorso è definito a priori.



**Connectionless:** I pacchetti vengono inviati sulla rete senza un percorso predefinito e potrebbero arrivare in ordine sparso o perdersi.



## 2.4 Affidabilità di un servizio

**ACK:** Sistema di notifica di azioni tra hosts

**RELIABLE (Affidabile):** I dati devono essere tutti consegnati al destinatario e per ogni pacchetto ricevuto viene inviato un ack al mittente. E' affidabile, ma lento.

**UNRELIABLE (Non Affidabile):** Non c'è nessuna garanzia che i dati vengano consegnati. E' veloce, ma poco affidabile.

E' possibile andare a combinare connessione ed affidabilità per ottenere servizi:

**Reliable Connection-Oriented:** Altamente affidabile

**Reliable Connectionless:** I dati saranno ricevuti tutti, ma non nell'ordine desiderato. Vengono utilizzati ACK. Anche chiamato acknowledged datagram service

**Unreliable Connection-Oriented:** Possibili perdite di dati

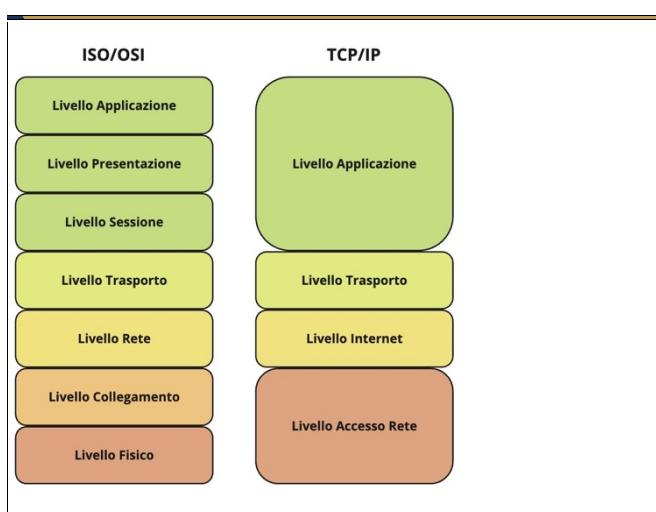
**Unreliable Connectionless:** Altamente inaffidabile con possibili perdite di dati non ordinati. Anche chiamato datagram service

## LEZIONE 3 – MODELLO TCP/IP E CONFRONTO

### 3.1 Modello TCP/IP

Il modello TCP/IP è un modello di comunicazione di rete che è stato sviluppato dal Dipartimento della Difesa degli Stati Uniti ed è ampiamente utilizzato per la progettazione, l'implementazione e la gestione delle reti di computer. È stato creato prima del modello OSI (Open Systems Interconnection) ed è ancora comunemente utilizzato come modello di riferimento per la comprensione delle reti di computer (**standard de facto**). Il modello TCP/IP contiene protocolli multipli oltre a quelli da cui prende il nome.

Il modello TCP/IP è diviso in quattro livelli, ciascuno dei quali svolge un ruolo specifico nella trasmissione dei dati attraverso una rete.



Come si può notare, nel modello TCP/IP:

- Il livello **Applicazione** comprende il livello **Presentazione** e **Sessione** del modello ISO/OSI
- Viene ridefinito il livello **Rete** in livello **Internet**
- Il livello **Accesso Rete** comprende sia il livello **Collegamento** (Data Link) sia quello **Fisico**

### 3.2 Livello Accesso Rete

In inglese **host-to-network**.

Questo è il livello più basso del modello e si occupa della trasmissione dei dati su un singolo collegamento fisico. Include protocolli e tecnologie per la gestione dell'accesso al mezzo di trasmissione, la rilevazione degli errori e la trasmissione affidabile dei dati all'interno dello stesso segmento di rete. Alcuni esempi di protocolli di livello di collegamento includono Ethernet e Wi-Fi.

Non specifica come questo debba funzionare nel dettaglio e lascia la possibilità di utilizzare i dispositivi di rete con propri protocolli ( Nel caso di comunicazione con dispositivi differenti l'apparato di rete provvederà alla conversione multiprotocollo ).

### 3.3 Livello Internet

Questo livello gestisce il routing dei dati tra diversi segmenti di rete. È responsabile della consegna dei pacchetti da un nodo a un altro attraverso una serie di reti interconnesse. Il protocollo principale di questo livello è l'Internet Protocol (IP), che assegna indirizzi IP unici a ciascun dispositivo nella rete e determina come i pacchetti di dati vengono instradati da un dispositivo all'altro.

### 3.4 Livello Trasporto

Questo livello si occupa del trasferimento dei dati end-to-end tra dispositivi. È responsabile del controllo del flusso, della segmentazione dei dati in pacchetti più piccoli e della gestione degli errori. Due dei protocolli più noti a questo livello sono il Transmission Control Protocol (TCP), che fornisce una comunicazione affidabile, e il User Datagram Protocol (UDP), che è più veloce ma meno affidabile.

#### TCP:

- Reliable Connection-Oriented
- Mittente - Frammenta il messaggio in pacchetti
- Destinatario - I pacchetti vengono riuniti per ricostruire il messaggio

#### UDP:

- Unreliable Connectionless
- Perdita di pacchetti

### 3.5 Livello Applicazione

Questo è il livello più alto e contiene le applicazioni che consentono agli utenti di interagire con la rete. Include una vasta gamma di protocolli e servizi, come HTTP per il web, SMTP/POP/IMAP per la posta elettronica, FTP per il trasferimento di file, DNS per il mapping degli indirizzi IP e molti altri. Questo livello si interfaccia direttamente con le applicazioni utente e fornisce loro l'accesso ai dati di rete.

Il modello TCP/IP nasce come modello di riferimento e non si fa troppa distinzione tra protocolli, servizi ed interfacce.

## LEZIONE 4 – LIVELLO FISICO E TEORIA DEI SEGNALI

La trasmissione delle informazioni parte dal livello fisico, ovvero dal collegamento e lo scambio di dati tramite mezzi fisici, che possono essere cavi, wireless e satellite.

Le connessioni possono essere di vario tipo: connessioni che possono essere utilizzate in entrambe le direzioni nello stesso momento sono dette **full-duplex**, connessioni che possono essere utilizzate in entrambe le direzioni, ma una direzione alla volta, sono dette **half-duplex**, connessioni che possono essere utilizzate in una sola direzione sono dette **simplex**.

La larghezza di banda (bandwidth), ovvero la capacità di trasmissione di un mezzo, si misura in Hz.

### 4.1 Basi della trasmissione

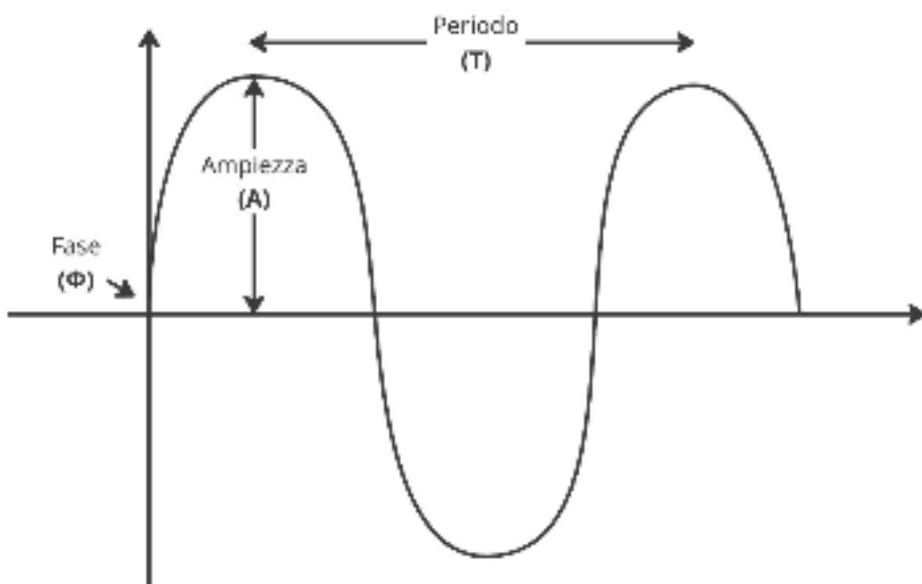
Un segnale sinusoidale è una forma d'onda periodica e regolare che si ripete in un periodo di tempo  $t$

L'informazione può essere trasmessa a distanza variando opportunamente una qualche caratteristica fisica del segnale scelto per la trasmissione.

Il segnale, inoltre, si propaga attraverso un qualche mezzo di trasmissione con una certa velocità  $t$  per un intervallo di tempo  $\Delta t$

Il segnale ha le seguenti caratteristiche:

- **Aampiezza (A)**: Differenza fra il valore massimo ed il valore minimo
- **Periodo (T)**: Quantità di tempo prima della ripetizione del segnale
- **Frequenza (F)**: Numero di oscillazioni sul periodo T ( $1/T$ )
- **Fase ( $\Phi$ )**: Scostamento dall'origine



### 4.2 Mezzi di Trasmissione

Il segnale può essere trasmesso principalmente attraverso tre diversi fenomeni fisici associati ad un determinato mezzo di trasmissione

- **Corrente elettrica** → Cavi in rame
- **Luce** → Fibra ottica
- **Onde elettromagnetiche** → Wireless (una combinazione di campo elettrico e campo magnetico variabili, che propagandosi nello spazio è in grado di indurre a distanza una corrente elettrica in un dispositivo ricevente (antenna).

#### 4.3 Tipologia di segnale

Il segnale si divide in due categorie:

- **Analogico**: il valore del segnale può variare **gradualmente** in un intervallo costituito da un **numero infinito** di possibili valori;
- **Digitale**: il segnale varia **bruscamente** assumendo in ogni istante solo uno di un **insieme finito di valori**.

I fenomeni naturali sono tutti esclusivamente analogici.

A causa dell'interazione con il mezzo trasmissivo, la **forma del segnale a destinazione non sarà mai esattamente quella di partenza**.

#### 4.4 Analisi Spettrale

##### 4.4.1 Teorema di Fourier

Una funzione  **$g(t)$** , definita in un intervallo finito  $T$ , può essere considerata come una funzione periodica di periodo  $T$ , e quindi espressa come somma di un numero infinito di funzioni sinusoidali

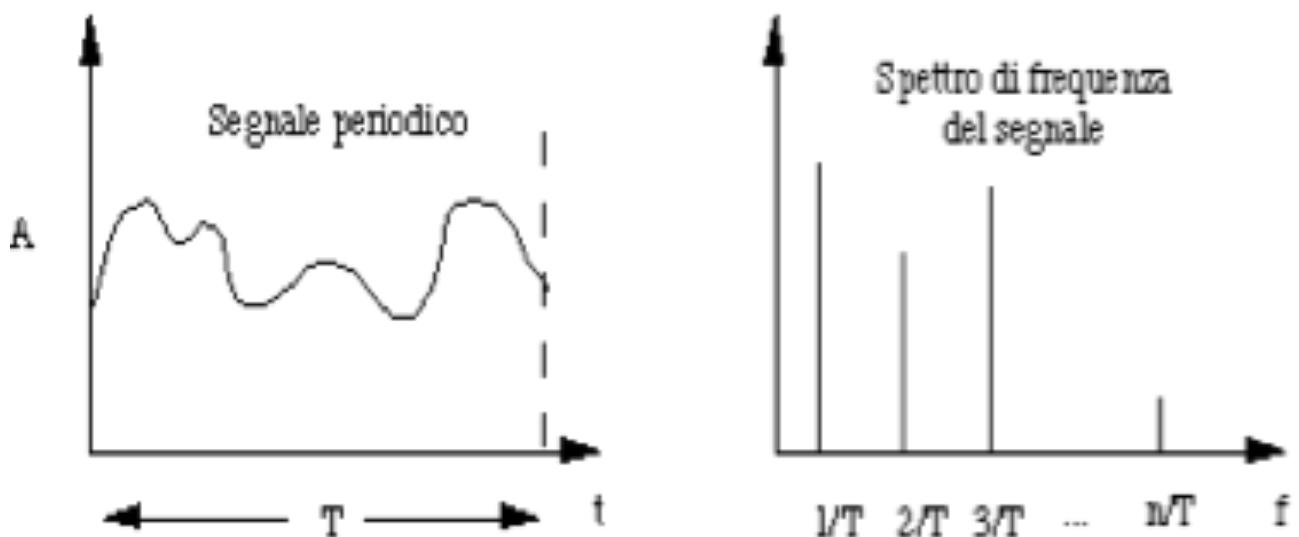
$$g(t) = \frac{1}{2}c + \sum_{n=1}^{\infty} a_n \sin(2\pi n f t) + \sum_{n=1}^{\infty} b_n \cos(2\pi n f t)$$

Tale funzione può essere vista anche in base alla frequenza fondamentale  $f = 1/T$ .

**$a_n$**  e  **$b_n$**  sono le ampiezze dell'ennesima armonica che ha frequenza  $n*f$

**$c$**  è detto coefficiente di Fourier

Un qualunque segnale  **$g(t)$** , di durata  $T$ , può essere rappresentato dal suo **spettro di frequenze** (la sua scomposizione in sinusoidi).



- L'analisi spettrale porta a vedere il segnale rappresentato da un insieme di frequenze
  - Queste frequenze rappresentano le sue sinusoidi
  - La scomposizione in frequenze è detto spettro di frequenze
  - L'intervallo delle frequenze di tutte le sinusoidi è detto **frequency band**
- 
- Minore è la durata del periodo  $T$ , maggiore sarà il valore della frequenza fondamentale
  - Quanto più velocemente varia  $g(t)$ , tanto più numerose sono le armoniche che la descrivono

Dato un mezzo fisico di comunicazione abbiamo che le seguenti definizioni:

- **Banda passante** - intervallo di frequenze che il mezzo fisico è in grado di trasmettere senza alterarle oltre certi limiti
- **Attenuazione e Ritardo** - Alterazioni del segnale. L'attenuazione dipende dalla frequenza del segnale ed è proporzionale alla distanza percorsa
- **Ampiezza di banda** - quantità di informazioni che è possibile trasmettere in un determinato lasso di tempo. (Possono esserci delle attenuazioni sulle bande tramite l'utilizzo di filtri (es. passa-basso, passa-banda))

**Nella trasmissione in un mezzo trasmissivo:**

**L'attenuazione subita dal segnale dipende dalla frequenza del segnale ed è proporzionale alla distanza percorsa;**

**se la banda passante è inferiore alla banda di frequenza, il segnale viene distorto (privato di alcune armoniche).**

**Se un numero sufficiente di armoniche arriva a destinazione, il segnale è comunque utilizzabile.**

Un segnale, anche se distorto, può essere ricostruito se un numero sufficiente di armoniche giunge a destinazione

#### 4.4.2 Teorema di Nyquist

Un segnale analogico di banda  $h$  può essere completamente ricostruito mediante una campionatura effettuata  $2 * h$  volte al secondo

**In ambito binario:** Se ciascun campione può assumere uno di  $n$  valori distinti, **il segnale risulta completamente rappresentato** con  $2^h * (\lg n)$  bit per ogni secondo ( $\lg \rightarrow \log$  in base 2)

Tale valore è chiamato anche **bit rate**

**Baud rate:** Velocità di comunicazione (numero di volte in un secondo in cui è possibile cambiare il valore)

#### 4.4.3 Rapporto Segnale/Rumore

Non esiste un canale di comunicazione perfetto, per cui è sempre possibile che il segnale sia distorto. Si utilizza il rapporto segnale/rumore (signal to noise ratio)

$\text{SNR} = 10 * \log(\text{S/N})$  ( $\log \rightarrow \log$  in base 10) e si misura in decibeli (db)

Rapporto S/N	Decibel
2	3
10	10
100	20
1000	30

*Il teorema di Nyquist è valido per canali totalmente privi di disturbi* (purtroppo non è realistico).

#### 4.4.4 Teorema di Shannon

Il teorema di Shannon riguarda la capacità di trasmissione di un canale di comunicazione. Esso stabilisce il limite superiore della quantità di informazione che può essere trasmessa attraverso un canale di comunicazione, soggetto a rumore o interferenze, a una data velocità e con una data qualità di trasmissione.

In termini più precisi, il teorema di Shannon afferma che la capacità massima  $C$  di trasmissione di un canale di comunicazione (max data rate), in bit al secondo, è data dalla seguente formula:

$$C = h * \log_2 \left( 1 + \frac{S}{N} \right)$$

Su un canale con banda **3 kHz e S/N=30dB** (tipici di una normale linea telefonica) massimo si può arrivare a **30.000 bps**.

### In generale:

- più alto è il numero di **bps** da trasmettere, più ampia deve essere la banda passante;
- a parità di mezzo utilizzato, minore è la lunghezza del canale di trasmissione tanto più è alto il numero di **bps** che si possono trasmettere (attenuazioni e sfasamenti restano accettabili);
- la trasmissione digitale è più critica di quella analogica (genera frequenze più alte), ma può essere più facilmente "rigenerata" lungo il percorso (in quella analogica ogni amplificazione introduce distorsione, che si somma a quella degli stadi precedenti).

## 4.5 Trasmissione

Trasmissione **baseband** (trasmissione in banda base): la singola trasmissione impegnava l'intera banda passante.

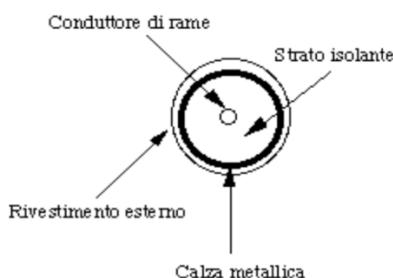
Trasmissione **broadband**: attraverso opportune tecniche di multiplazione si effettuano contemporaneamente più trasmissioni distinte (nella telefonia con broadband si indicano trasmissioni che impegnano più di 4 kHz)

## 4.6 Mezzi trasmittivi

**Doppino intrecciato**: coppia di conduttori in rame intrecciati in forma elicoidale (**L'intreccio riduce i disturbi causati dalle interferenze**)

- usato, in particolare, per le connessioni terminali del sistema telefonico (da casa dell'utente alla centrale più vicina).
- larghezza di banda - dipende dalla distanza (100 -1000 Mb/s a 100 metri).

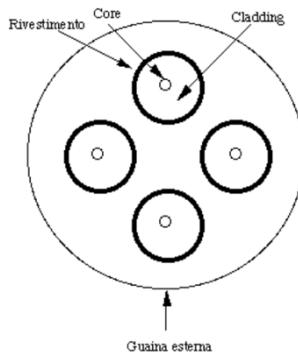
**Cavo coassiale**: conduttore centrale in rame circondato da uno strato isolante all'esterno del quale vi è una calza metallica, il tutto rivestito in plastica. Il miglior isolamento, rispetto al doppino, consente maggiori velocità di trasmissione e distanze superiori.



- Usato in passato nel sistema telefonico per le tratte a lunga distanza, è ormai sostituito quasi ovunque dalla fibra ottica (oggi è usato per la TV via cavo ed in vecchie LAN)

**Fibra ottica:** sottilissimo cilindro centrale in vetro (**core**), circondato da uno strato esterno (**cladding**) di vetro avente un diverso indice di rifrazione e da una guaina protettiva.

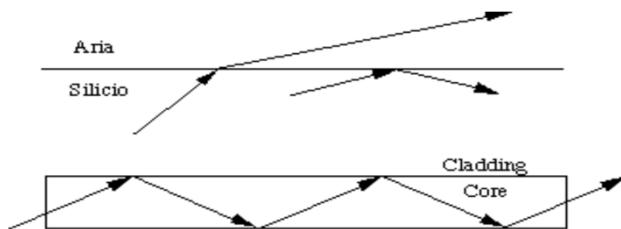
In genere più coppie sono contenute insieme in una stessa guaina esterna.



Un sistema di trasmissione ottico ha tre componenti

1. **sorgente luminosa**
  1. converte un segnale elettrico in impulsi luminosi (può essere un LED o un laser)
2. **mezzo di trasmissione**
  1. fibra ottica
3. **fotodiodo ricevitore**
  1. converte gli impulsi luminosi in segnali elettrici (l'ordine dei Gbps deriva dal tempo di risposta dei fotodiodi che è dell'ordine del nsec)

*Quando un raggio di luce attraversa il confine fra il core ed il cladding subisce una deviazione.*



La deviazione dipende dagli **indici di rifrazione** dei due materiali ed è tale che **per certi angoli di incidenza, il raggio resta intrappolato all'interno del core.**

Le fibre ottiche si dividono in:

- **multimodali:** (**core di 50 micron**) raggi diversi con diversi angoli (**mode**) possono contemporaneamente propagarsi nella stessa fibra;
- **monomodali:** (**core di 8-10 micron**) la luce di un singolo raggio avanza nella fibra che si comporta come una **guida d'onda**. Sono più costose ma garantiscono distanze maggiori (fino a 30 km).

Nelle fibre ottiche un impulso luminoso rappresenta un **1**, mentre la sua assenza rappresenta uno **0**.

- Le attuali fibre consentirebbero velocità di trasmissione di 50 Tbps ad un bassissimo tasso d'errore
- Mancando al momento sistemi di conversione luminoso/elettrico in grado di operare a tali velocità, la pratica corrente limita l'uso delle fibre a qualche Gbps

La **bassa attenuazione** nella trasmissione in fibra è dovuta:

- alla **particolare trasparenza** delle fibre ottiche (se il mare fosse fatto di questo vetro si vedrebbe il fondo...);
- all'utilizzo di **tre particolari bande (finestre)** per la trasmissione (tutte tra **infrarosso e UV**:  $0.85\mu$ ,  $1.30\mu$ ,  $1.55\mu$  – perdita di meno del 5% per Km-), larghe da 25.000 GHz a 30.000 Ghz ciascuna.

#### 4.7 Topologie di reti in fibra ottica

**Anello:** concatenando più fibre ottiche si crea un anello. L'interfaccia del singolo sistema può essere passiva (fa passare l'impulso luminoso nell'anello) o attiva (converte l'impulso luminoso in elettrico, lo amplifica e lo riconverte in luce);

**Stella passiva:** l'impulso, inviato da un trasmettitore, arriva in un cilindro di vetro al quale sono attaccate (fuse) tutte le fibre ottiche. Si realizza così una rete broadcast.

**Vantaggi** delle fibre ottiche rispetto al rame:

- **banda:** due fibre sono più capaci di 1000 doppini;
- **peso:** 100 kg/km contro 8.000 kg/km;
- **totale insensibilità a disturbi elettromagnetici;**
- **difficili intrusioni.**

**Svantaggi** delle fibre ottiche rispetto al rame:

- **costo delle giunzioni;**
- **comunicazione unidirezionale**

## 4.8 Trasmissioni wireless

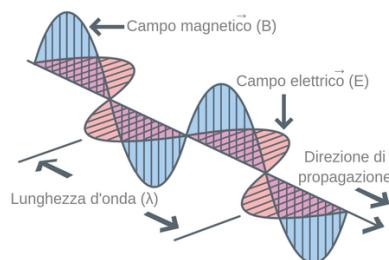
Si utilizzano le onde elettromagnetiche che viaggiano nel vuoto alla velocità della luce ( $c=3 \cdot 10^8$  m/s).

In particolare:

Un'onda elettromagnetica monocromatica (cioè con una ben definita frequenza e lunghezza d'onda) è costituita da un campo elettrico (E) e un campo magnetico (B) mutuamente perpendicolari che oscillano in fase fra loro perpendicolarmente alla direzione di propagazione

**Caratteristiche di un'onda elettromagnetica:**

- **Frequenza f:** numero di oscillazioni al secondo. Si misura in Hertz (Hz)
- **Lunghezza d'onda  $\lambda$ :** distanza tra due massimi o minimi consecutivi.
- **La velocità c è data dalla relazione  $c = f * \lambda$**



La lunghezza d'onda decresce al crescere della frequenza.  $\lambda=c/f$

## 4.9 Classificazione onde elettromagnetiche

norme	$f$ in Hz	$\lambda$ in m	indicazione	esempi
bassa frequenza	$10^{-1}$ $10^0$ $10^1$ $10^2$ $10^3$ $10^4$ $10^5$ $10^6$ $10^7$ $10^8$ $10^9$ $10^{10}$ $10^{11}$ $10^{12}$			Oscillazioni di terremoti, maree, ponti, torri, grattacieli, pendoli di orologio
BF	3 Hz 30 Hz 300 Hz 3 kHz 30 kHz 300 kHz 3 MHz 30 MHz 300 MHz 3 GHz 30 GHz 300 GHz	$10^8$ $10^7$ $10^6$ $10^5$ $10^4$ $10^3$ $10^2$ $10^1$ $10^0$ $10^{-1}$ $10^{-2}$ $10^{-3}$	onde miriamiche onde chilometriche onde ettorimetriche onde decametriche onde metriche onde decimetriche onde centimetriche onde millimetriche	telescrittori telefoni radio televisione ponti radio, radar
Very Low Frequencies VLF	3 kHz	$10^5$	onde miriamiche	televisori
Low Frequencies LF	30 kHz	$10^4$	onde chilometriche	telefoni
Medium Frequencies MF	300 kHz	$10^3$	onde ettorimetriche	radio
High Frequencies HF	3 MHz	$10^2$	onde decametriche	televisione
Very High Frequencies VHF	30 MHz	$10^1$	onde metriche	radar
Ultra High Frequencies UHF	300 MHz	$10^0$	onde decimetriche	radar
Super High Frequencies SHF	3 GHz	$10^{-1}$	onde centimetriche	radar
Extremely High Frequencies EHF	30 GHz 300 GHz	$10^{-2}$ $10^{-3}$	onde millimetriche	gamma infrarossi, luce e raggi x

- **Onde radio a bassa frequenza**

Sono le onde VLF, LF, MF.

Si propagano in tutte le direzioni e passano attraverso gli edifici percorrendo lunghe distanze.

Sono soggette ad interferenze elettromagnetiche.

- **Onde radio ad alta frequenza**

Sono le onde HF, VHF.

Con queste frequenze inizia ad aversi una propagazione in linea retta con lambda (lungh. Onda) dell'ordine del cm/mm.

Rimbalzano gli ostacoli e tendono essere assorbite dal suolo.

Vengono impiegate per le trasmissioni direzionali, le onde che colpiscono la ionosfera vengono rifratte e rispedite sulla terra.

Sono soggette ad interferenze elettromagnetiche.

- **Microonde**

Sono onde con frequenza intorno ai 100Mhz e si propagano in linea retta.

A causa della curvatura della terra, la trasmissione oltre certe distanze richiede l'uso di ripetitori (la distanza fra i ripetitori cresce come la radice quadrata dell'altezza delle torri: per torri alte 100 m i ripetitori possono essere distanti 80 Km).

Usate per trasmissioni a grande distanza e come bande per applicazioni industriali-scientifiche-mediche.

A causa della dispersione nello spazio alcune onde possono essere rifratte, e quindi arrivare in ritardo e fuori fase (Multipath Fading: distruzione del segnale causata dall'arrivo fuori fase di onde rifratte).

Per bande fino a circa 8 Ghz, le onde vengono assorbite dall'acqua (pioggia)

- **Onde infrarosse e millimetriche**

Usate per comunicazioni su piccole distanze (telecomandi televisori)

Sono relativamente direzionali.

Non passano attraverso corpi solidi

Usate in ambienti chiusi

Non interferiscono con quelle impiegate nei locali attigui (usando il proprio telecomando non si cambia il canale al televisore del vicino).

#### 4.10 Trasmissione e modulazione

Le informazioni si trasmettono tramite modulazione

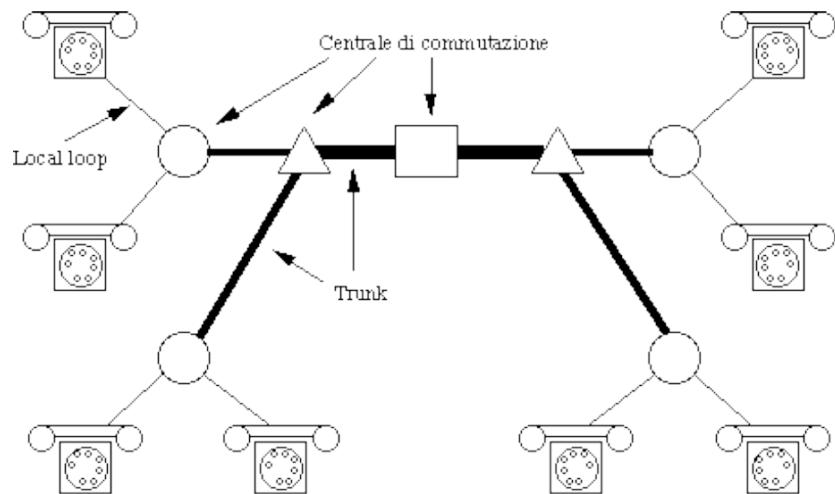
Si possono modulare

- Ampiezza (A)
- Frequenza (f)
- Fase ( $\Phi$ )

## LEZIONE 5- SISTEMA TELEFONICO E MODULAZIONE

Il sistema telefonico (**rete pubblica telefonica commutata**) è nato ed evolutosi per la fonia, poco adatto per la trasmissione dati

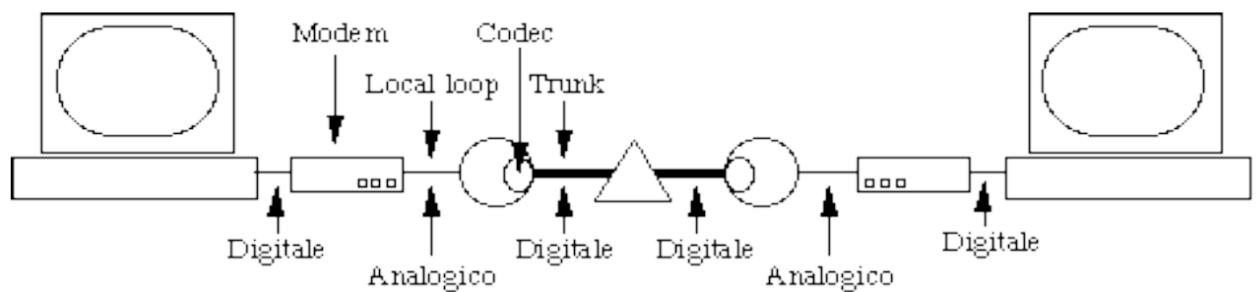
I sistemi telefonici attuali seguono una **gerarchia multilivello** con elevata ridondanza



**Local loop:** collega il telefono alla più vicina **centrale di commutazione**. Il collegamento è realizzato principalmente da un **doppino** lungo da 1 a 10km e trasporta un segnale analogico che occupa una **banda di 3 kHz** → detto anche **ultimo miglio**

**Centrali di commutazioni:** quasi sempre digitali perché, oltre a data rate più alti, è più facile sia ricostruire il segnale senza introdurre errori e mescolare voce, dati, video.

**Trunk:** collegano le **centrali di commutazioni**. I collegamenti sono realizzati tipicamente in **fibra ottica**



**Modem:** effettua la trasformazione del segnale **digitale/analogico** in trasmissione e **analogico/digitale** in ricezione, necessarie per trasmettere dati sul **local loop** a 3 kHz

**Codec:** effettua le trasformazioni **analogico/digitale** nella prima centralina di commutazione e **digitale/analogico** nell'ultima

Le trasmissioni sui **local loop** sono soggette ad **attenuazione e distorsione**.

Poiché l'ampio spettro di frequenze generato dalle onde quadre contrasta con la banda ridotta del local loop, **la trasmissione digitale sui local loop può avvenire solo a bassissime velocità**.

Per trasmettere un segnale digitale sul local loop, si modula un segnale sinusoidale, detto **portante**, la cui frequenza è compresa tra 1 e 2 kHz (la banda del local loop è di 3 kHz)

Un segnale può essere modulato in:

- **Aampiezza** → si varia l'ampiezza
- **Frequenza** → si varia la frequenza
- **Fase** → si varia la fase (cioè lo scostamento rispetto al segnale originale)

Un ulteriore aumento delle prestazioni si può ottenere effettuando una **compressione dei dati** prima di trasmetterli.

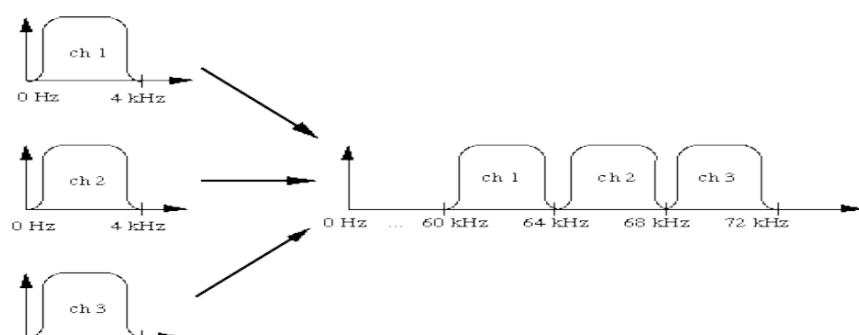
Per la compressione dei dati, esistono due standard: **V.42 bis (de iure)** emesso da ITU e **MNP 5** emesso da Microm Network (**de facto**)

Maggiori velocità sono oggi possibili con nuove tecnologie quali ISDN, ADSL, ATM...

I trunk trasmettono migliaia di informazioni simultaneamente.

La capacità di trasmettere migliaia di informazioni simultaneamente prende il nome di **multiplexing TDM, FDM e WDM**.

- **Frequency Division Multiplexing (FDM):** adatto alla gestione di segnali analogici.  
E' una tecnica di multiplexing utilizzata nelle telecomunicazioni per trasmettere più segnali attraverso un unico canale di comunicazione.  
L'idea alla base dell'FDM è quella di suddividere la banda disponibile del canale di comunicazione in diverse sotto-bande di frequenza più strette e trasmettere segnali diversi su ciascuna di queste sotto-bande. Queste sotto-bande possono sovrapporsi parzialmente senza interferire tra loro, grazie alla differente gamma di frequenze utilizzate.  
Lo spettro di frequenza è suddiviso in bande più piccole, e ogni comunicazione ha l'esclusivo uso di una di esse.



- **Wavelength Division Multiplexing (WDM):**  
è una variante della FDM, impiegata per le fibre ottiche.

I raggi di bande differenti, provenienti da più fibre, convergono in un prisma che li combina inoltrandoli in un'unica fibra. All'altro capo, attraverso un prisma o un sintonizzatore ottico (basato sugli interferometri Fabry-Perot o Mach-Zehnder), il raggio viene diviso tra le fibre destinate (la divisione avviene selezionando, per ciascuna delle fibre, la lunghezza d'onda del raggio che in essa si intende inoltrare).

- **Time Division Multiplexing (TDM):** adatto per la gestione di dati in forma digitale.

I bit provenienti da diverse connessioni vengono a turno prelevati ed inviati su un'unica connessione ad alta velocità:

Questa tecnica con il tempo si è confermata essere molto affidabile se non fosse per il fatto che l'ITU non riuscì a creare uno standard internazionale facendo sì che nel mondo venissero utilizzare una serie di modalità incompatibili tra di loro.

Nord America e Giappone usano la portante **T1**, essa è composta da 24 canali vocali uniti in multiplexing, ognuno dei quali, a turno, inserisce 8 bit nel flusso in uscita ogni 125 us.

Al di fuori del Nord America e Giappone, al posto della T1 viene usata la portante **E1**, composta da 32 canali (30 dati e 2 controllo) con valori ad 8 bit ogni 125 us.

Con il passare degli anni, per creare collegamenti sempre più performanti si è deciso di:

- **T2:** collegare 4 canali T1, ognuno dei quali garantiva 1.544 Mbps, per formare un unico canale che garantisse velocità di 6,312 Mbps (non 6,176 perché c'è l'aggiunta di bit di controllo).
- **T3:** collegare 7 canali T2, ognuno dei quali garantiva 6,312 Mbps, per formare un unico canale che garantisse velocità di 44,736 Mbps.
- **T4:** collegare 6 canali T3, ognuno dei quali garantiva 44,736 Mbps, per formare un unico canale che garantisse velocità di 274,176 Mbps.

La differenza tra i sistemi richiede costose apparecchiature di conversione ai confini fra un l'uno e l'altro.

- **SDH (Synchronous Digital Hierarchy):** gerarchia unificata a livello mondiale, introdotta dal CCITT verso la metà degli anni '80.
- **SONET (Synchronous Optical NETwork):** nato negli USA, si basa sul canale STS-1 (810 byte ogni 125 µs: 51.84 Mbps) non presente in SDH.

## Riapplicando TDM

Carrier	Caratteristiche	Velocità
T2	4 canali T1	6.312 Mbps
T3	6 canali T2	44.736 Mbps
T4	6 canali T3	274.176 Mbps
E2	4 canali E1	8.848 Mbps
E3	4 canali E2	34.304 Mbps
E4	4 canali E3	139.264 Mbps
E5	4 canali E4	565.148 Mbps

Poiché IL **TDM** può essere usato solo nei segnali digitali e l'ultimo miglio (local loop) produce segnali analogici, è necessario fare una conversione nella centrale locale, dove i segnali vengono raggruppati per essere spediti lungo i **trunk**.

I segnali analogici sono digitalizzati da un dispositivo chiamato **codec**. Il codec estrae 8000 campioni al secondo (uno ogni 125 micro secondi) perché secondo il teorema di Nyquist sono sufficienti a catturare tutta l'informazione da un canale telefonico a 4 kHz. Ad un tasso di campionamento più basso l'informazione andrebbe persa, viceversa, non si otterrebbe alcuna informazione aggiuntiva.

Questa tecnica è chiamata **PCM** (Pulse Code Modulation, modulazione codificata di impulsi) ed è il cuore dei sistemi telefonici moderni. Di conseguenza, quasi tutti gli intervalli di tempo all'interno di un sistema telefonico sono multipli di 125ms.

All'altro capo della linea il segnale verrà riconvertito in analogico dal codec anche se non sarà del tutto uguale a quello di partenza.

Finora abbiamo analizzato il sistema telefonico dal punto di vista dell'impianto esterno (ultimo miglio e i trunk), andiamo ora ad analizzare l'impianto interno, quello composto quindi dai commutatori. Attualmente le reti usano due tecniche di commutazione:

**Commutazione di circuito:** tecnica su cui si basa il sistema telefonico tradizionale, tale tecnica, concettualmente, quando una persona avvia una telefonata, l'apparecchiatura di commutazione cerca di creare un percorso fisico completo tra il chiamante ed il chiamato. Agli inizi della telefonia era l'operatore manualmente a creare questo percorso oggi ci sono dispositivi di commutazione automatica.

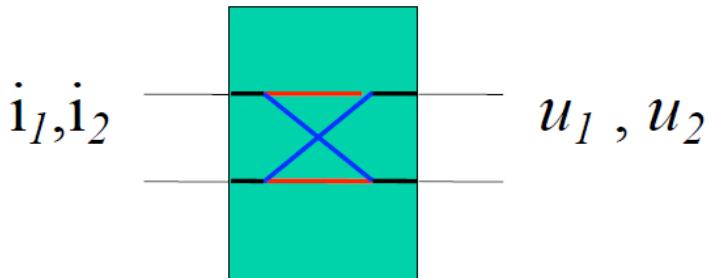
**Commutazione di pacchetto:** con questa tecnica la comunicazione tra i due utenti avviene mediante l'invio di pacchetti, questo fa sì che si hanno notevoli benefici, tra cui, non avere la necessità di un percorso fisico dedicato, poiché ogni pacchetto segue una strada diversa, seguendo la disponibilità della banda dei percorsi

disponibili, inoltre il pacchetto appena generato viene inviato, senza la necessità di creare prima un percorso. Di seguito vediamo le differenze principali tra le due tecniche.

I dispositivi di commutazione sono:

- **Switch**

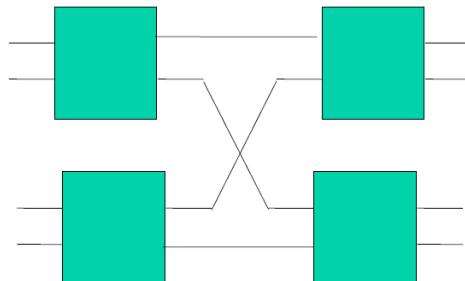
dispositivo a semiconduttori con due ingressi ( $i_1, i_2$ ), due uscite ( $u_1, u_2$ ) e due possibili stati:



*i1 è connesso a u1 e i2 è connesso a u2;*

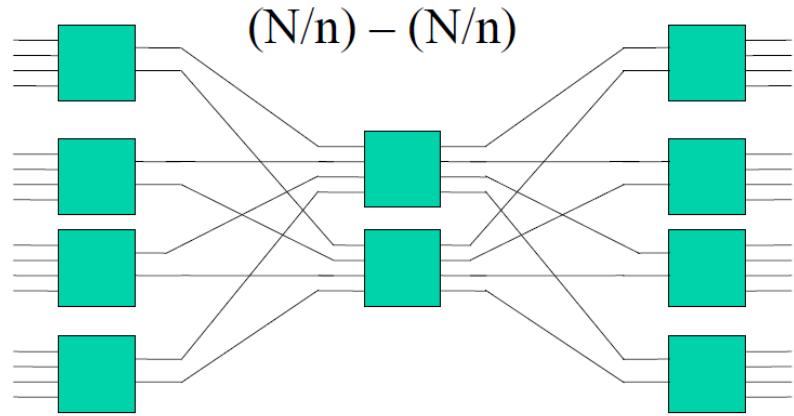
*i1 è connesso a u2 e i2 è connesso a u1.*

- **Crossbar:** N ingressi, N uscite ed  $N^2$  punti di incrocio, realizzati ciascuno con uno switch.



- **Switch multilivello:** (multistage switch): utilizzo di più crossbar di piccole dimensioni (ad es.  $2 \times 2$ ), organizzati in gruppi su più livelli successivi

$$(N/n) - k \quad k - (N/n)$$



$$2*(N/n)*(n*k) + k * (N/n)^2 \text{ punti di incrocio.}$$

## LEZIONE 6-7-8-9 – IL LIVELLO DATA LINK

### 6.1 Introduzione

Nel livello data link discuteremo degli algoritmi usati per ottenere una comunicazione affidabile ed efficiente tra unità d'informazione chiamate frame fra due macchine adiacenti, ovvero due macchine collegate da un canale di comunicazione che agisce concettualmente come un cavo.

### 6.2 Progettazione del livello Data Link

Il livello Data Link fa uso del servizio messo a disposizione dal livello fisico per inviare e ricevere bit su un canale di comunicazione.

Ha molte funzioni, tra cui:

- Fornire un'interfaccia di servizio ben definita per il livello tre
- Gestire gli errori di trasmissione
- Regolare il flusso dati in modo che i buffer dei dispositivi lenti non vengano inondati dai trasmittenti veloci

Il livello Data Link può essere progettato in modo da offrire una varietà di servizi che cambiano da protocollo a protocollo, come ad esempio:

- Servizio senza conferma (**unacknowledged**) senza connessione
- Servizio con conferma (**acknowledged**) senza connessione
- Servizio con conferma orientato alla connessione

Il servizio di tipo **unacknowledged** senza connessione consiste nell'avere una macchina sorgente che invia i frame senza preoccuparsi dell'arrivo effettivo di quest'ultimi.

In particolare, il destinatario non invierà al mittente una conferma.

Se un frame viene perso a causa del rumore sulla linea, non viene fatto alcun tentativo di rilevare la perdita o di correggerla nel livello data link. L'uso di questa classe di servizio è appropriato quando la frequenza degli errori di trasmissione è molto bassa oppure quando si deve utilizzare un traffico real-time, come la trasmissione vocale, per le quali gli effetti del ritardo sono peggiori di quelli derivanti dalla trasmissione di dati sbagliati.

Il servizio **acknowledged** senza connessione, continua a non utilizzare alcun tipo di connessione logica, però quando il ricevente riceve un frame, spedisce al mittente una conferma chiamata **acknowledgment (ACK)**.

In questo modo il mittente riesce a sapere se il frame è arrivato a destinazione in modo corretto oppure no.

Se non è arrivato in uno specifico intervallo temporale, il frame può essere rispedito. Questa classe di servizio è utilizzata per i canali di trasmissione poco affidabili come ad esempio il WI-FI.

L'ACK è da considerarsi come un'ottimizzazione e mai come un requisito obbligatorio.

Il servizio **acknowledged orientato alla connessione** è un tipo di servizio dove il mittente e destinatario stabiliscono una connessione prima di iniziare a trasferire i dati.

Ogni frame trasferito è numerato e il livello data link garantisce che venga effettivamente ricevuto una sola volta e nell'ordine corretto, quindi si tratta di un servizio che consente il trasferimento affidabile di un flusso di bit.

E' utilizzato per collegamenti lunghi e inaffidabili come i canali satellitari e linee telefoniche a lunga distanza.

Se, in queste situazioni, si utilizzasse semplicemente un servizio con ACK senza connessione, è possibile che i frame di ACK vadano persi e ciò causerebbe un re-invio di frame duplicati che andrebbe solamente a sprecare la banda del canale di comunicazione.

Con l'ack sorgono alcuni problemi:

**Un frame può anche sparire del tutto, per cui il mittente rimane bloccato in attesa di un ack che non arriverà mai** (Il mittente stabilisce un time-out per la ricezione dell'ack. Se questo non arriva in tempo, ritrasmette di nuovo il frame).

**Se sparisce l'ack, il destinatario può trovarsi due (o più) copie dello stesso frame** (Il mittente inserisce un numero di sequenza all'interno di ogni frame dati).

Un altro aspetto importante è il **controllo del flusso**.

Si invia un feedback dal destinatario al mittente per impedire che il mittente spedisca dati più velocemente di quanto il destinatario sia in grado di ricevere.

Meccanismi basati sull'esplicita autorizzazione:

**il destinatario informa il mittente riguardo il numero di frame che può ricevere.**

### 6.3 Operazioni del livello due

In trasmissione:

- suddivide il flusso di bit, che arriva dal livello tre, in una serie di frame;
- calcola una funzione (**checksum**) per ciascun frame;
- inserisce il checksum nel frame;
- consegna il frame al livello uno, il quale lo spedirà come sequenza di bit.

In ricezione:

- riceve una sequenza di bit dal livello uno;
- ricostruisce da essa un frame dopo l'altro;
- per ciascun frame ricalcola il checksum;
- se il checksum ricalcolato è diverso da quello contenuto nel frame, il frame viene scartato.

### 6.4 Suddivisione in frame

L'approccio del livello data link è quello di suddividere il flusso di bit in una serie discreta di frame, calcolare un valore a lunghezza fissa, chiamato **checksum**, per ogni frame e includere il checksum nel frame quando viene trasmesso. Quando un frame arriva a destinazione, il checksum viene ricalcolato e se differisce, il livello data link sa che c'è stato un errore e prende provvedimenti, per esempio scartando il frame o restituendo un messaggio di errore.

Suddividere il flusso di bit in frame è un'operazione che viene chiamata **framing** ed è possibile effettuarla attraverso 4 metodi:

- **Conteggio dei byte**
- **Flag byte con byte stuffing**
- **Flag bit con bit stuffing**

- **Violazioni della codifica del livello fisico**

Il metodo **conteggio dei byte** consiste nell'usare un campo nell'header per specificare il numero di byte del frame. Quando il livello data link della destinazione legge tale numero, sa quanti byte seguiranno e quindi dove si troverà la fine del frame.

Tuttavia questo metodo non è efficiente, poiché l'informazione viaggia nel frame che è soggetto a rumore e quindi il valore contenuto nell'header potrebbe cambiare, ottenendo letture di frame inconsistenti.

Il secondo metodo chiamato **Flag byte con byte stuffing** consiste nell'inserire byte speciali all'inizio e alla fine di ogni frame. La maggior parte dei protocolli utilizza un byte speciale che prende il nome di **flag byte (DLE – Data Link Escape)**, per delimitare sia l'inizio che la fine dei frame.

Quindi, due flag byte consecutivi indicano la fine e l'inizio di un frame.

**DLE STX** (Start of TeXt): inizio frame.

**DLE ETX** (End of TeXt): fine frame.

Questo metodo presenta tuttavia un problema, in quanto può accadere che nella trasmissione dati compaia naturalmente il flag byte, specialmente in quelli binari, quali immagini, video, audio ecc, interferendo così con le operazioni di framing.

La soluzione è quella di inserire un byte di escape (**un altro DLE**) subito prima di ogni occorrenza nei dati. Il livello data link di destinazione provvederà a rimuovere i byte di escape prima di passarli al livello rete. Questa tecnica è chiamata **byte stuffing**

Il terzo metodo aggira il limite imposto dal byte stuffing di usare byte ad 8 bit. L'operazione di framing può essere effettuata anche a livello di bit in modo che i frame possano contenere un numero arbitrario di bit. Ad esempio, ogni volta che si incontrano 5 bit **consecutivi** ad 1, si inserisce uno 0 aggiuntivo per delimitare il frame.

Anche in questo caso è presente lo stuffing.

L'ultimo metodo sfrutta il concetto che il livello fisico per conservare l'integrità dei dati inserisce ridondanza, quindi nell'invio di 4 bit di dati spesso si usano pacchetti da 5 bit, tenendo così 16 delle 32 combinazioni libere, questa tecnica usa proprio quelle combinazioni libere per inserire l'inizio e la fine del frame.

## 6.5 Rilevazione e correzione degli errori

Molti fattori possono provocare errori, soprattutto sul local loop e nelle trasmissioni wireless (mentre sono piuttosto rari nei mezzi più moderni quali le fibre ottiche).

Gli errori sono dovuti in generale a **rumore di fondo, disturbi improvvisi** (ad es. fulmini), **interferenze** (ad es. motori elettrici).

Per trattare questi errori il livello data link aggiunge informazione ridondanti ai dati che devono essere spediti.

Una strategia consiste nell'includere informazioni ridondanti in un numero sufficiente da permettere al destinatario di dedurre i dati realmente trasmessi, l'altra consiste nell'introdurre una ridondanza di informazioni per permettere al destinatario di dedurre che c'è stato un errore (ma non quale).

La prima strategia usa **codici a correzione d'errore** (*error-correction code*), l'altra utilizza **codici a rilevazione d'errore** (*error-detecting code*). L'uso di codici a correzione d'errore viene spesso indicato come **FEC** (*forward error correction – correzione d'errore in anticipo*)

## 6.6 Codici a correzione di errore

Tra i codici a correzione d'errore troviamo il **codice di Hamming**.

Un frame (a parte i delimitatori) sarà composto di

**$n = m + r$  bit dove  $m$  = bit del messaggio vero e proprio,  $r$  = redundant bit o check bit → bit per il controllo dell'errore.**

Per introdurre il codice di Hamming è importante stabilire cos'è la **distanza di Hamming**.

La **distanza di Hamming** è semplicemente il numero di bit in cui differiscono due configurazioni binarie.

Chiameremo ognuna di queste configurazioni **Codeword (parola codice → sequenza di  $m+r$  bit)** e l'insieme di tutte le configurazioni **Codice (code → insieme prefissato di codeword)**

Nell'ambito tecnico, tale distanza è calcolata attraverso un'operazione bitwise (bit a bit) di OR esclusivo (**XOR**) e poi si contano i bit ad 1 di tale sequenza.

Es:  $D(10001 . 000011) = 2$

Quando consideriamo l'insieme di tutte le codeword, quindi il codice, possiamo stabilire il concetto di **distanza minima di Hamming** di un codice, ovvero la minor distanza considerando due sue parole qualsiasi.

La distanza tra le parole di un codice gioca un ruolo fondamentale nel meccanismo di rilevazione degli errori. Un codice più è ridondante e più è efficace per rilevare errori. Se un codice non è ridondante, non ci sono possibilità di rilevazione degli errori.

ES: Supponiamo di avere la seguente codifica:

A 000  
B 001  
C 010  
D 011  
E 100  
F 101  
G 110  
H 111

Se vogliamo inviare A → 000 , ma arriva B → 001 , quindi c'è stato un errore sul singolo bit finale, **il ricevente non è in grado di rilevare se c'è stato un errore poiché la sequenza 001 è una codeword valida.**

Se invece intervallassimo le parole codice come nel seguente caso:

A 000  
001  
010  
100  
B 011  
111  
101  
110  
C ....

Qualunque bit corrotto in una codifica valida porterebbe ad un errore poiché la distanza di Hamming di questo codice è 2.

In generale, un codice per la rilevazione di modifiche **su  $d$  bit**, deve avere almeno

**$D_{MIN} = d+1$** .

Di conseguenza se ho un codice con  **$D_{MIN}=2$** , sarò in grado di rilevare errori sul singolo bit e così via.

I codici di questo tipo sono detti **SEDC (Single Error Detection Code)** ed hanno tutti  **$D_{MIN}$**  almeno pari a 2.

Hamming, ha dato anche un ulteriore apporto per poter, non solo rilevare l'errore, ma anche individuare dove è presente l'errore e correggerlo.

Questo avviene aggiungendo un numero di bit sufficienti di parità e ciascun bit di parità sarà la parità di alcuni bit e l'algoritmo sarà in grado di determinare anche se un bit di parità si è sporcato.

E' importante notare che il numero di bit di controllo va scelto secondo la seguente formula:

$$2^r \geq (m + r + 1)$$

Dove  $r$  sono i bit di controllo ed  $m$  i bit del messaggio originale.

Dato un messaggio M da inviare, il procedimento da applicare per ottenere il frame MS effettivamente trasmesso, consiste nel sistemare i bit su di una griglia di 12 posizioni, dove le posizioni in cui l'indice è potenza di 2 sono occupate dai bit di controllo, mentre tutte le altre ospitano ordinatamente i bit del messaggio M.

Sia  $M = 10110110$

con:

$m_1 = 1$   $m_2 = 0$   $m_3 = 1$   $m_4 = 1$   $m_5 = 0$   $m_6 = 1$   $m_7 = 1$   $m_8 = 0$

La stringa da trasmettere sarà la seguente:

pos 1	pos 2	pos 3	pos 4	pos 5	pos 6	pos 7	pos 8	pos 9	pos 10	pos 11	pos 12
0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100
<b>h<sub>1</sub></b>	<b>h<sub>2</sub></b>	$m_1$	<b>h<sub>3</sub></b>	$m_2$	$m_3$	$m_4$	<b>h<sub>4</sub></b>	$m_5$	$m_6$	$m_7$	$m_8$
$2^0=1$	$2^1=2$		$2^2=4$				$2^3=8$				

Osservando la figura si può notare che per ogni byte da spedire il codice di Hamming utilizza 4 bit di controllo: **h<sub>1</sub> h<sub>2</sub> h<sub>3</sub> h<sub>4</sub>** nelle posizioni 1,2, 4 e 8; il codice è quindi completato dai bit del messaggio.

Nella seconda riga della stessa figura sono mostrati i valori binari delle posizioni occupabili dai bit; si noti che tutte le posizioni occupate dai bit controllori hanno un solo 1 in tale valore binario, mentre tutte le posizioni occupate dai bit controllati, i bit del messaggio, hanno almeno due 1.

Vediamo ora come si determinano i valori dei bit di controllo.

Il bit  $h_1$  si trova in posizione 1 (valore binario 0001) della stringa del codice Hamming. **Esso controlla tutti i bit che nel valore binario della posizione occupata hanno un 1 in quarta posizione incominciando da sinistra**, ovvero:

$h_1$  controlla:

$m_1$ , posizione 0011  
 $m_2$ , posizione 0101  
 $m_4$ , posizione 0111  
 $m_5$ , posizione 1001  
 $m_7$ , posizione 1011

Il bit  $h_2$  si trova in posizione 2 (valore binario 0010) della stringa del codice Hamming. **Esso controlla tutti i bit che nel valore binario della posizione occupata hanno un 1 in terza posizione incominciando da sinistra**, ovvero:

$h_2$  controlla

$m_1$ , posizione 0011  
 $m_3$ , posizione 0110  
 $m_4$ , posizione 0111  
 $m_6$ , posizione 1010  
 $m_7$ , posizione 1011

Il bit  $h_3$  si trova in posizione 4 (valore binario 0100) della stringa del codice Hamming. **Esso controlla tutti i bit che nel valore binario della posizione occupata hanno un 1 in seconda posizione incominciando da sinistra**, ovvero:

$h_3$  controlla  
 $m_2$ , posizione 0101  
 $m_3$ , posizione 0110  
 $m_4$ , posizione 0111  
 $m_8$ , posizione 1100

Il bit  $h_4$  si trova in posizione 8 (valore binario 1000) della stringa del codice Hamming. **Esso controlla tutti i bit che nel valore binario della posizione occupata hanno un 1 in prima posizione incominciando da sinistra**, ovvero:

$h_4$  controlla  
 $m_5$ , posizione 1001  
 $m_6$ , posizione 1010  
 $m_7$ , posizione 1011  
 $m_8$ , posizione 1100

Nel codice così costruito ogni bit del messaggio da spedire, M, è controllato dal almeno due bit di controllo. **I valori dei bit di controllo si ottengono mediante un'operazione di XOR tra tutti i bit controllati**, per cui avremo che:

$$\begin{aligned}
 h_1 &= m_1 \oplus m_2 \oplus m_4 \oplus m_5 \oplus m_7 = 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 = 1 \\
 h_2 &= m_1 \oplus m_3 \oplus m_4 \oplus m_6 \oplus m_7 = 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 = 1 \\
 h_3 &= m_2 \oplus m_3 \oplus m_4 \oplus m_8 = 0 \oplus 1 \oplus 1 \oplus 0 = 0 \\
 h_4 &= m_5 \oplus m_6 \oplus m_7 \oplus m_8 = 0 \oplus 1 \oplus 1 \oplus 0 = 0
 \end{aligned}$$

Si ricava, quindi, il seguente codice Hamming:

$MS = 111001100110$

dove in neretto sono individuati i bit di controllo.

L'entità che riceve il frame verifica la sua bontà **mediante un controllo che si basa sull'XOR tra tutti i bit controllati con il proprio bit controllore determinando le somme  $S_i$** , così come mostrato negli esempi che seguono. Dal valore delle somme  $S_i$  dipende la correttezza del messaggio; i casi possibili sono due; ovvero:

1. messaggio corretto  $\rightarrow$  tutti gli  $S_i$  sono uguali a 0;
2. messaggio errato  $\rightarrow$  esiste almeno un  $S_i$  uguale a 1. In tal caso i valori delle quattro somme  $S_4S_3S_2S_1$  indicano la posizione del bit errato.

Allo scopo vediamo tre esempi in cui si verificano i seguenti eventi:

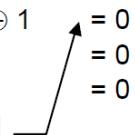
1. frame arrivato in modo esatto;
2. frame arrivato in modo errato perché è cambiato un bit del messaggio (bit controllato);
- 3: frame arrivato in modo errato perché è cambiato un bit controllore.

### Esempio1: Frame arrivato in modo esatto

L'entità che riceve il frame determina le somme  $S_i$  che seguono:

$$\begin{aligned}
 S_1 &= h_1 \oplus m_1 \oplus m_2 \oplus m_4 \oplus m_5 \oplus m_7 = 1 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 = 0 \\
 S_2 &= h_2 \oplus m_0 \oplus m_2 \oplus m_3 \oplus m_5 \oplus m_6 = 0 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 1 = 0 \\
 S_3 &= h_3 \oplus m_1 \oplus m_2 \oplus m_3 \oplus m_7 = 0 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 0 \\
 S_4 &= h_4 \oplus m_4 \oplus m_5 \oplus m_6 \oplus m_7 = 0 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 0
 \end{aligned}$$

indica messaggio giusto

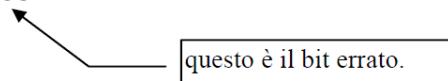


Come volevasi dimostrare: tutti gli  $S_i$  sono uguali a 0, per cui il frame è arrivato in modo corretto.

## Esempio2: Frame arrivato in modo errato, cambia un bit controllato

Supponiamo che il frame arrivi in modo errato; per esempio,  $m_7$  inverta il suo stato da 1 a 0; otterremmo che l'entità ricevente leggerà il seguente messaggio:

MS = 111001100100



L'entità che riceve verifica l'esattezza del messaggio calcolando i valori di  $S_1, S_2, S_3$  e  $S_4$ , ovvero:

$$S_1 = h_1 \oplus m_1 \oplus m_2 \oplus m_4 \oplus m_5 \oplus m_7 = 1 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 0 = 1$$

$$S_2 = h_2 \oplus m_1 \oplus m_3 \oplus m_4 \oplus m_6 \oplus m_7 = 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 1$$

$$S_3 = h_3 \oplus m_2 \oplus m_3 \oplus m_4 \oplus m_8 = 0 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 0$$

$$S_4 = h_4 \oplus m_5 \oplus m_6 \oplus m_7 \oplus m_8 = 0 \oplus 0 \oplus 1 \oplus 0 \oplus 0 = 1$$

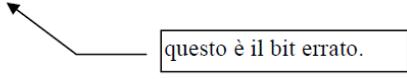
Il bit errato è quindi quello che si trova nella posizione:

$$S_4S_3S_2S_1 = (1011)_2, = (11)_{10}$$

## Esempio3: Frame arrivato in modo errato: cambia un bit controllore

Vediamo un ulteriore esempio, e supponiamo che cambi proprio un bit di controllo, per esempio  $h_4$  (da 0 a 1) posto nell'ottava posizione del codice Hamming, ottenendo in output la seguente stringa:

MS = 111001110110



L'entità che riceve, verifica l'esattezza del messaggio calcolando i valori di  $S_0, S_1, S_2$  e  $S_3$ , ovvero:

$$S_1 = h_1 \oplus m_1 \oplus m_2 \oplus m_4 \oplus m_5 \oplus m_7 = 1 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 = 0$$

$$S_2 = h_2 \oplus m_1 \oplus m_3 \oplus m_4 \oplus m_6 \oplus m_7 = 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 = 0$$

$$S_3 = h_3 \oplus m_2 \oplus m_3 \oplus m_4 \oplus m_8 = 0 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 0$$

$$S_4 = h_4 \oplus m_5 \oplus m_6 \oplus m_7 \oplus m_8 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 1$$

Il bit errato è quindi quello che si trova nella posizione:

$$S_4S_3S_2S_1 = (1000)_2, = (8)_{10}$$

(l'ottava posizione) del codice Hamming.

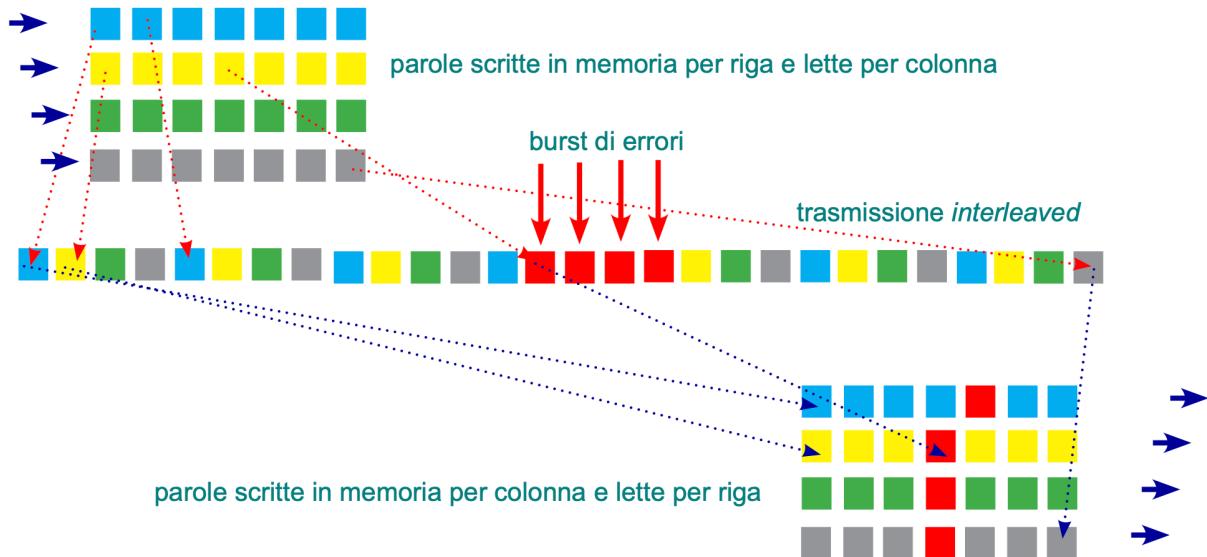
In quella posizione si trova proprio  $h_4$ , per cui, per ottenere il byte corretto, basterà trascurare il bit sbagliato, visto che era un bit di controllo e non un bit del messaggio.

Il byte giusto sarà, quindi, quello che si otterrà eliminando tutti i bit di controllo; ovvero: 10110110

Tecnica per **correggere burst di errori** (gruppi contigui di errori) di lunghezza massima prefissata k:

1. Si accumulano K codeword, riga per riga;
2. I codeword si trasmettono per colonne (*interleaved*);
3. Quando arrivano si riassemblano per righe.

Poiché un burst di k errori comporta un singolo errore in ciascuna delle K codeword; correggendo ciascuna codeword ricostruiamo l'intero burst.



Tecnica per rilevare ***burst di errori*** di lunghezza massima k:

1. Si accumulano k codeword di n bit, secondo una matrice di K righe e n colonne;
2. Si aggiunge il parity bit per ciascuna riga formando una colonna aggiuntiva;
3. I dati vengono trasmessi per colonne;
4. Ricevuto l'intero blocco, si controllano tutti i bit di parità e nel caso di errori si richiede la ***ritrasmissione del blocco***.

I codici correttori di errore, ad eccezione delle trasmissioni simplex dove non è possibile inviare al mittente una richiesta di ritrasmissione, sono usati molto raramente.

**E' più efficiente limitarsi a rilevare gli errori, e ritrasmettere saltuariamente i dati errati, piuttosto che impiegare un codice (più dispendioso in termini di ridondanza) per la correzione degli errori.**

## 6.7 Codici a rilevazione d'errore

Tra i codici a rilevazione d'errore troviamo:

- **Bit di parità**
- **Checksum**
- **CRC (Cyclic Redundant Check)**

Il **bit di parità** viene scelto in modo che il numero di 1 nella parola di codice sia dispari (o pari). Questa operazione equivale a calcolare il bit di parità (disparità) come somma modulo 2 o lo **XOR** dei bit di dati. Per esempio, se viene inviata la sequenza 1011010 in parità dispari, viene aggiunto un bit ad 1 per ottenere 10110101 in modo che il numero di bit a 1 sia dispari. Un codice con un singolo bit di parità ha distanza pari a 2 ed è in grado di rilevare errori sul singolo bit poiché se ci fossero errori su 2 bit si otterrebbe una codeword valida.

Il **checksum** è spesso usato per indicare un gruppo di bit di controllo associati al messaggio, qualunque sia il modo di calcolarli. Un gruppo di bit di parità è un checksum, tuttavia esistono metodi per calcolare il checksum basati sulla somma progressiva di gruppi di bit del messaggio.

Il checksum verrà posizionato alla fine del messaggio (frame).

Il ricevente eseguirà la stessa operazione di somma e confronterà il risultato.

Se il risultato è identico al checksum, non ci saranno errori nella trasmissione.

Se cambiano 2 bit di un byte, il controllo di parità non se ne accorgerebbe (dist di hamming 2), mentre il checksum si e quindi, a differenza di hamming, è impiegabile anche per errori di tipo **burst**.

Il checksum viene utilizzato anche nel protocollo IP.

Il **CRC** è il più potente codice a rilevazione d'errore ed è noto anche come **codifica polinomiale**.

Le codifiche polinomiali si basano sul fatto di trattare sequenze di bit come dei polinomi a coefficienti che possono assumere solo i valori 0 ed 1 naturalmente.

Un frame di  $k$  bit viene visto come un polinomio con  $k$  coefficienti che variano da  $x^{k-1}$  a  $x^0$  di grado  $k-1$ .

L'aritmetica è gestita in modulo 2. Non ci sono riporti per le addizioni o prestiti per le sottrazioni. Sia l'addizione che la sottrazione sono identiche all' OR esclusivo.

Quando si utilizza una codifica polinomiale, sorgente e destinazione devono mettersi d'accordo in anticipo su un cosiddetto **polinomio generatore  $G(x)$** . Il generatore deve avere i bit di ordine più alto e basso uguali ad 1.

Per poter calcolare il **checksum** di un frame di  $m$  bit corrispondente al polinomio  $M(x)$  (**dividendo**), il frame deve essere più lungo del polinomio generatore (**divisore**). L'idea è quella di aggiungere un checksum alla fine del frame in modo che il polinomio rappresentato dal frame + checksum sia divisibile per  $G(x)$ . Quando la destinazione riceve il frame con checksum prova a dividerlo per  $G(x)$ . Se c'è resto vuol dire che c'è stato un errore di trasmissione.

I passi per calcolare il checksum sono i seguenti:

1. Sia  $r$  il grado di  $G(x)$ , si aggiungano  $r$  bit con valore 0 dopo la parte di ordine più basso al polinomio  $M(x)$  in modo da contenere  $m+r$  bit
2. Si divida tale sequenza per la sequenza corrispondente a  $G(x)$  usando la divisione modulo 2
3. Si sottragga il resto (che può contenere al più  $r$  bit) dalla sequenza corrispondente di  $M(x)$   $m+r$  bit. Il risultato è il frame con checksum pronto per la trasmissione.

In ogni divisione, se si sottrare il resto dal dividendo, quello che resta è sempre divisibile per il divisore.

## 6.8 Comunicazione ed ACK

Nei livelli fisico, data link e network, ci sono processi (HW o SW) indipendenti, che comunicano fra loro, ad esempio scambiandosi messaggi;

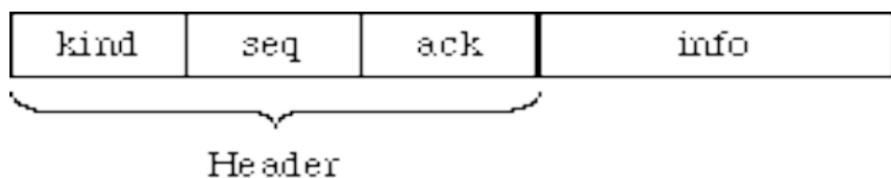
Quando il SW del livello data link riceve un pacchetto dal livello network, lo incapsula tra un **header** ed un **trailer**, calcola (con un apposito HW del livello data link) il **checksum** e i **delimitatori** ed infine passa il frame al livello sottostante, che trasmette il tutto.

All'inizio della trasmissione la destinazione non ha niente da fare, deve solamente aspettare che succeda qualcosa. Per cui il livello data link aspetta che accada qualcosa con la chiamata ad una procedura `wait_for_event(&event)` che termina solo quando è successo qualcosa.

Quando un frame arriva alla destinazione, l'HW calcola il checksum. Se il checksum non è corretto, il livello data link viene informato (*event = cksum\_err*). Se il frame è arrivato senza problemi il SW del livello datalink è altresì informato (*event = frame\_arrival*) e lo acquisisce dal livello sottostante.

Successivamente ispeziona le informazioni di controllo contenute nell'header/trailer e, se tutto va bene, passa la parte informativa (dati) al livello di rete. In nessuna circostanza viene passato l'header di un frame al livello di rete. Questo accade per mantenere separati i protocolli data link e di rete. Se il livello di rete non conosce alcun dettaglio implementativo del protocollo data link o del formato dei frame, questi ultimi possono essere cambiati senza che ci sia anche una modifica nel livello di rete.

## 6.9 Struttura di un frame



**kind**: tipo di frame (dati, controllo, ecc.)

**seq**: numero progressivo del frame

**ack** : informazioni legate all'acknowledgement

**info**: pacchetto di livello network completo.

## 6.10 Gestione di sequenza di trasmissione e flusso

### 6.10.1 Protocollo Heaven

Questo protocollo, per canale simplex (i frame viaggiano **in una sola direzione**), è molto semplice ed è basato sulle ipotesi (**non realistiche**) che:

- I frame dati vengono trasmessi in una sola direzione;
- Le peer entity di livello network sono sempre pronte (non devono mai attendere per inviare o ricevere al/dal livello data link);
- Si ignora il tempo di elaborazione del livello data link (**tempo nullo**);
- C'è spazio infinito per il buffering nel ricevitore;
- Il canale fisico non fa mai errori.

Il protocollo consiste di due procedure, relative rispettivamente al mittente e al destinatario.

**Mittente** (loop infinito):

1. Attende un pacchetto dal livello network;

2. Costruisce un frame dati;
3. Passa il frame al livello fisico;
4. Torna ad 1).

**Destinatario** (loop infinito):

1. Attende l'arrivo di un frame da livello fisico;
2. Estrae il pacchetto;
3. Lo passa al livello network;
4. Torna ad 1).

### 6.10.2 Protocollo Stop and Wait

Rilasciamo l'ipotesi (poco realistica) che esista un buffer infinito nel ricevitore. Tutte le altre rimangono. La novità è che il mittente deve essere opportunamente rallentato. Ciò però non si può fare con ritardi prefissati: sarebbe troppo gravoso, perché questi dovrebbero essere calibrati sul caso pessimo, che non sempre si verificherà.

La soluzione consiste nell'invio, da parte del destinatario, di una esplicita autorizzazione all'invio del prossimo frame. Questo tipo di protocolli, nei quali il mittente attende un OK dal destinatario, si chiamano **stop and wait** su un canale almeno half-duplex.

Il mittente invia il prossimo frame solo se esplicitamente autorizzato dal destinatario.

**Mittente** (loop infinito):

1. Attende l'arrivo di un pacchetto dal livello network;
2. Costruisce un frame dati;
3. Passa il frame al livello fisico;
4. Attende l'ack (ok);
5. Torna ad 1.

**Destinatario** (loop infinito):

1. Attende l'arrivo di un frame dati da livello fisico;
2. Estrae il pacchetto;
3. Consegna il pacchetto al livello network;
4. Invia un frame di ack al mittente;
5. Torna ad 1.

Assumiamo ora che il canale possa fare errori. I frame (dati o di ack) possono essere danneggiati o persi completamente. Se un frame arriva danneggiato, l'HW di controllo del checksum se ne accorge e informa il SW di livello data link; se il frame si perde del tutto, ovviamente, la cosa non si rileva.

Una possibile soluzione è la seguente:

- **mittente**: quando invia un frame dati, fa anche partire un timer. Alla scadenza del timer se l'ack per il frame trasmesso non è ancora arrivato, reinvia il frame.
- **destinatario**: per ogni frame dati arrivato senza errori invia un ack.

Questo semplice schema in realtà non basta, infatti può verificarsi la seguente sequenza di eventi:

1. Il frame dati x arriva bene;
2. L'ack del frame x si perde completamente (gli errori non discriminano tra frame dati e frame di ack);
3. Il frame dati x viene inviato di nuovo e arriva bene;
4. Il livello network di destinazione riceve due copie di x, errore!

E' necessario che il destinatario possa riconoscere gli eventuali doppioni. Ciò si ottiene sfruttando il campo **seq dell'header**, dove il mittente mette il numero di sequenza del frame dati inviato.

Notiamo che basta un bit per il numero di sequenza, poiché l'unica ambiguità in ricezione è tra un frame ed il suo immediato successore (**siamo ancora in stop and wait**): infatti, fino a che un frame non viene confermato, è sempre lui ad essere ritrasmesso, altrimenti è il suo successore.

Dunque, sia mittente che destinatario useranno, come valori per i numeri di sequenza, la successione ...01010101...

Il mittente trasmette i frame dati alternando zero ed uno nel campo **seq** e passa a trasmettere il prossimo frame **solo quando riceve l'ack di quello precedente**.

Il destinatario invia un frame di ack per tutti quelli ricevuti senza errori, **ma passa al livello network solo quelli con il numero di sequenza atteso**.

Per quanto riguarda le possibilità che i frame dati arrivino rovinati o non arrivino affatto, un meccanismo basato su timer va bene, bisogna però che esso sia regolato in modo da permettere sicuramente l'arrivo dell'ack, pena la ritrasmissione errata di un duplicato del frame, che può creare grossi guai.

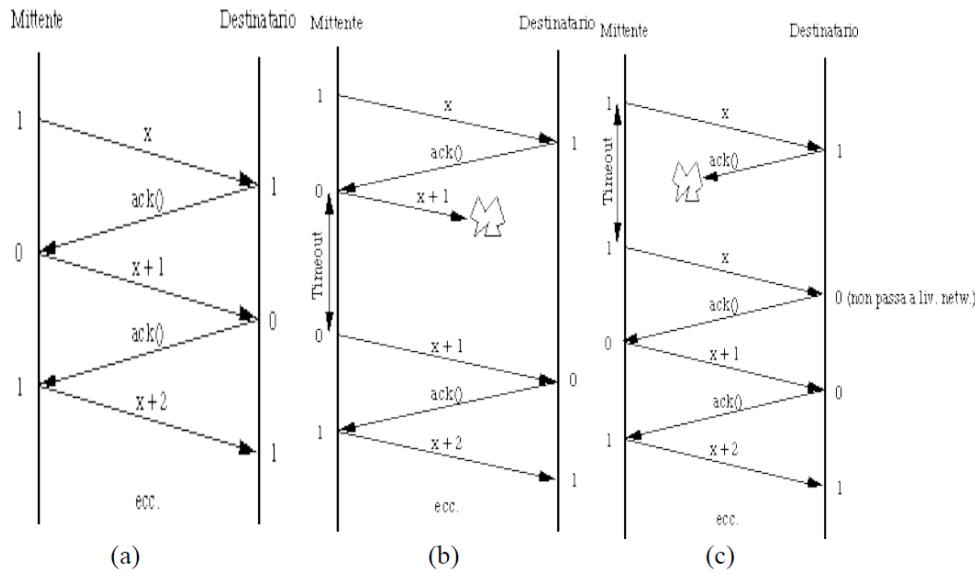
Protocolli come questo, in cui il mittente aspetta un ack di conferma prima di trasmettere il prossimo frame, si chiamano **PAR** (Positive Ack with Retransmission) o **ARQ** (Automatic Repeat Request).

**Mittente** (loop infinito; [seq] rappresenta il campo seq di un frame):

- 0)  $n\_seq = 0;$
- 1)  $n\_seq = 1 - n\_seq;$
- 2) attende un pacchetto dal livello network;
- 3) costruisce frame dati e copia  $n\_seq$  in [seq];
- 4) passa il frame dati al livello fisico;
- 5) resetta il timer;
- 6) attende un evento:
  - \* timer scaduto: torna a 4)
  - \* arriva frame di ack (vuoto) non valido: torna a 4)
  - \* arriva frame di ack (vuoto) valido: torna ad 1)

**Destinatario** (loop infinito; [seq] rappresenta il campo seq di un frame):

- 0)  $n\_exp = 1;$
- 1) attende evento;
  - \* arriva frame dati valido da livello fisico:
    - 2) se ( $[seq] == n\_exp$ )
      - 2.1) estrae pacchetto
      - 2.2) lo consegna al livello network
      - 2.3)  $n\_exp = 1 - n\_exp$
    - 3) invia frame di ack (vuoto)
    - 4) torna ad 1)
- \* arriva frame non valido: torna ad 1)



(a) Normale funzionamento; Perdita (o danneggiamento) (b) di un frame (c)di un ack

Un problema importante è legato, come abbiamo già accennato, alla lunghezza del timeout. Se esso è troppo breve, può succedere questo:

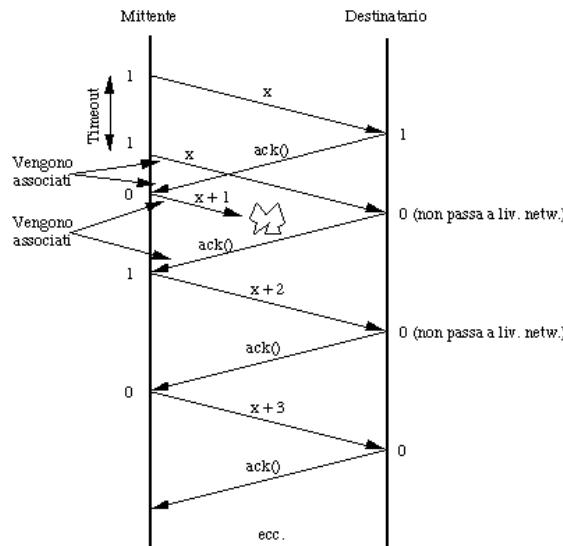


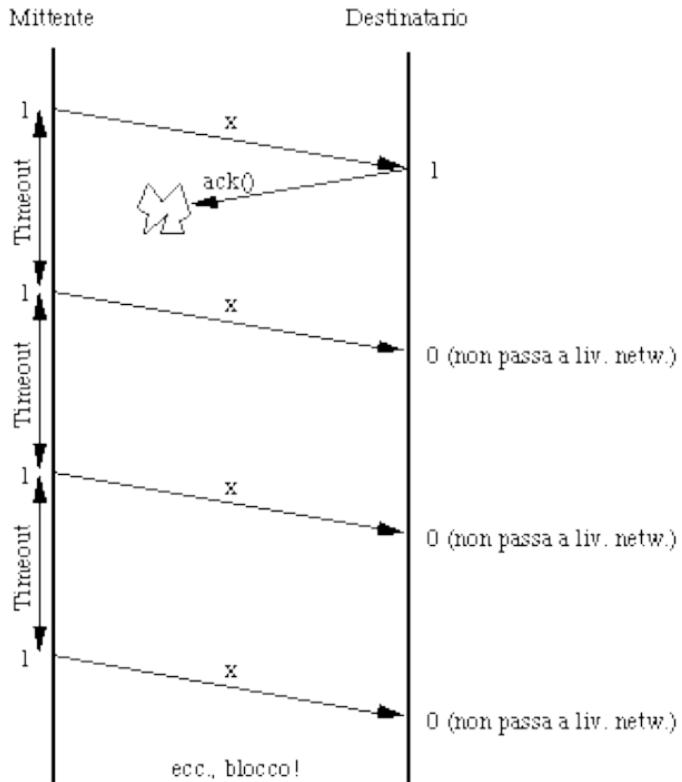
Figura 3-5: Timeout troppo ridotto

Se  $(x+1)$  si perde, l'ack inviato per la seconda copia di  $x$  potrebbe essere scambiato per l'ack di  $x+1$  e quindi non ci si accorgerebbe che quest'ultimo è perso.

Ovviamente  $(x+2)$ , sebbene ricevuto regolarmente non sarà passato al livello rete perché il numero di sequenza non è corretto, e quindi  $(x+2)$  viene scambiato per una copia di  $x$ .

Ovviamente il problema non lo si risolve se si invia l'Ack per i soli frame non duplicati.

Infatti la perdita di ack causerebbe un blocco dal lato mittente che non ricevendo l'ack continuerebbe ad inviare ininterrottamente il frame per il quale non avrebbe mai riscontro.



### 6.10.3 Protocolli a finestra scorrevole

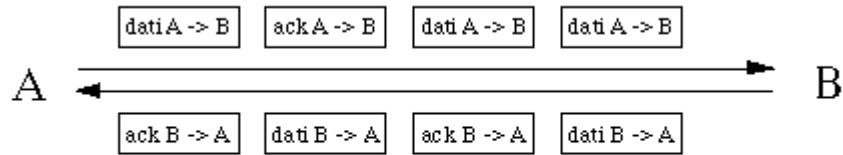
#### 6.10.3.1 Preambolo

Nei casi precedenti i frame dati viaggiano in una sola direzione e i frame di ack nella direzione contraria, quindi esistono dei frame che viaggiano in entrambe le direzioni.

Dunque, volendo stabilire una comunicazione dati bidirezionale è necessario disporre di due circuiti, il che ovviamente è uno spreco di risorse. Un'idea migliore è usare un solo circuito, nel quale far convivere tutte le esigenze.

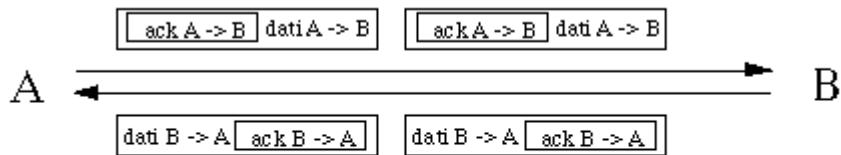
Supponendo che la comunicazione sia fra A e B, si avrà che:

- nella direzione da A a B viaggiano i frame dati inviati da A a B e i frame di ack inviati da A a B (in risposta ai frame dati inviati da B ad A);
- nella direzione da B a A viaggiano i frame dati inviati da B a A e i frame di ack inviati da B a A (in risposta ai frame dati inviati da A a B);
- il campo **kind** serve a distinguere fra i due tipi di frame, dati e di ack, che viaggiano nella stessa direzione.



**Figura 3-7:** Comunicazione dati bidirezionale su unico canale

Una miglioria può essere effettuata utilizzando la tecnica del **Piggybacking** cioè:  
quando B deve inviare un ack ad A, lo inserisce (in un apposito campo) nel prossimo frame dati che B deve inviare ad A.



**Figura 3-8:** Piggybacking

Il campo ack serve proprio a questo scopo, infatti è il campo in cui viene trasportato, se c'è, un ack.  
Questa tecnica consente un notevole risparmio di:

- **banda utilizzata;**
- **uso di cpu.**

Infatti, con questa tecnica le informazioni di ack non richiedono la costruzione di un apposito frame (e quindi il tempo necessario alla creazione ed al riempimento della struttura, al calcolo del checksum, ecc.) né la sua trasmissione (e quindi l'uso di banda).

Tuttavia, non è possibile aspettare l'ack all'infinito. Mittente e Destinatario stabiliscono un tempo di attesa limite, trascorso il quale si crea un apposito frame di ack.

I protocolli che vedremo ora appartengono alla classe dei protocolli sliding window (finestra scorrevole), sono **full-duplex** (per i dati), sfruttano il **piggybacking** e sono più robusti di quelli precedenti.

Differiscono fra loro per efficienza, complessità e capacità dei buffer. Alcuni aspetti però **sono comuni** a tutti gli algoritmi:

#### Operazioni del mittente:

1. Ogni frame inviato ha un numero di sequenza, da 0 a  $2^{n-1}$  (il campo seq è costituito da n bit);
2. Ad ogni istante il mittente mantiene una **finestra scorrevole sugli indici dei frame**, e **solo quelli entro la finestra possono essere trasmessi**. I numeri di sequenza entro la finestra rappresentano frame da spedire o spediti, ma non ancora confermati:
  - a. quando arriva dal livello network un pacchetto, un nuovo indice **entra** nella finestra;
  - b. quando arriva un ack, il corrispondente indice **esce** dalla finestra;
  - c. i frame dentro la finestra devono essere mantenuti in memoria per la possibile ritrasmissione; se il buffer è pieno, il livello data link deve costringere il livello network a sospendere la consegna di pacchetti;

Analogamente, il **destinatario**:

1. Mantiene una finestra corrispondente agli indici dei frame che possono essere accettati:
  - a. se arriva un frame il cui indice è fuori dalla finestra, il frame viene scartato (e non si invia il relativo ack);
  - b. se arriva un frame il cui indice è dentro la finestra, il frame viene accettato; viene spedito il relativo ack; la finestra viene spostata in avanti;

Si noti che la finestra del destinatario rimane sempre della stessa dimensione, e se essa è pari a 1 il livello accetta i frame solo nell'ordine giusto altrimenti uscirebbero fuori dalla finestra (ma per dimensioni maggiori di 1 questo non è più detto).

Le finestre di mittente e destinatario non devono necessariamente avere uguali dimensioni, né uguali limiti inferiori o superiori.



**Figura 3-9:** Finestra scorrevole sugli indici dei frame

In questi protocolli il livello data link ha più libertà nell'ordine di trasmissione, fermo restando che:

- i pacchetti devono essere riconsegnati al livello network nello stesso ordine di partenza;
- il canale fisico è **wire-like**, cioè consegna i frame nell'ordine di partenza.

#### 6.10.3.2 Protocolli a finestra scorrevole di un bit

In questo protocollo sia mittente che destinatario usano una finestra scorrevole di dimensione uno. Di fatto questo è un protocollo stop-and-wait.

Non c'è differenza di comportamento fra mittente e destinatario, entrambi usano lo stesso codice (questo vale anche per i protocolli successivi).

Il funzionamento, molto semplice, è il seguente:

- il **mittente**, quando invia un frame, fa partire un timer:
  - se prima che scada il timer arriva un ack con lo stesso numero di sequenza del frame che si sta cercando di trasmettere, si avanza la finestra e si passa a trasmettere il frame successivo;
  - se arriva un ack diverso o scade il timer, si ritrasmette il frame;
- il **destinatario** invece:
  - quando arriva un frame corretto, senza errori, invia un ack col corrispondente numero di sequenza;
  - se il frame non è un duplicato lo passa al livello network e avanza la finestra.

Qui sta la principale novità rispetto al protocollo stop and wait: l'ack è etichettato col numero di sequenza del frame a cui si riferisce. I valori dell'etichetta possono solo essere 0 e 1, come nel protocollo stop and wait.

Il peggio che può succedere è la ritrasmissione inutile di qualche frame, ma questo protocollo è sicuro.

#### 6.10.4 Protocolli go-back-n e selective repeat

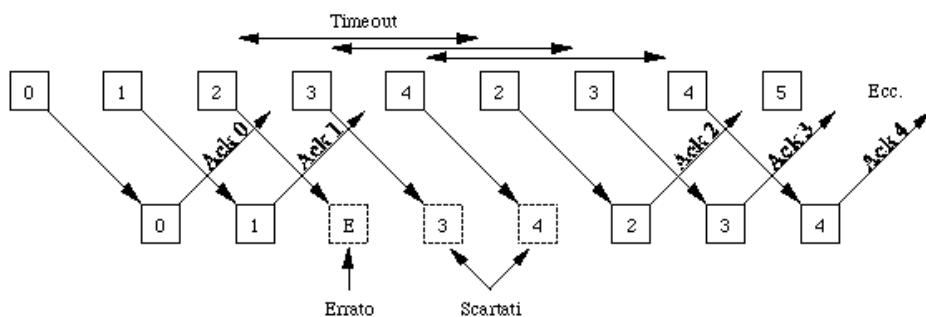
Se il tempo di andata e ritorno del segnale (**round-trip time**) è alto, come ad esempio nel caso dei canali satellitari nei quali è tipicamente pari a 500 + 500 msec, c'è un'enorme inefficienza coi protocolli stop-and-wait, perché si sta quasi sempre fermi aspettando l'ack.

Per migliorare le cose, si può consentire l'invio di un certo numero di frame anche senza aver ricevuto l'ack del primo. Questa tecnica va sotto il nome di **pipelining**.

Ciò però pone un serio problema, perché se un frame nel mezzo della sequenza subisce alterazioni, il mittente invia molti altri frame prima di riceverne notizia.

Il primo approccio al problema è quello del protocollo **go-back-n**:

- Se arriva un frame danneggiato o con un numero di sequenza non progressivo, il destinatario **ignora** tale frame e tutti i successivi, non inviando i relativi ack. Ciò corrisponde ad una finestra di dimensione uno nel ricevitore, che quindi accetta i frame solo nell'ordine giusto;
- **il mittente ad un certo punto va in time-out sul frame sbagliato, e poi su tutti quelli successivi (scartati dal destinatario)**, e quindi provvede a ritrasmettere la sequenza di frame che inizia con quello per il quale si è verificato il time-out.

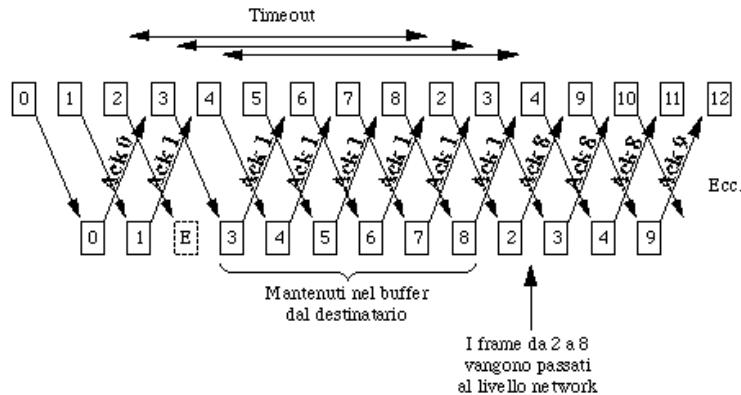


**Figura 3-10:** Funzionamento del protocollo go-back-n

Si noti che il mittente deve mantenere in un apposito buffer tutti i frame non confermati per poterli eventualmente ritrasmettere. Se il buffer si riempie, il mittente deve bloccare il livello network fino a che non si ricrea dello spazio. Inoltre, **vi è spreco di banda se il tasso d'errore è alto e/o il time-out è lungo**

Il secondo approccio è più efficiente, ed è chiamato **selective repeat**:

- **il destinatario mantiene nel suo buffer tutti i frame ricevuti successivamente ad un eventuale frame rovinato; non appena questo arriva nuovamente (senza errori), esso e tutti i successivi frame contigui che il destinatario ha mantenuto nel buffer vengono consegnati al livello network;**
- per ogni frame arrivato bene, il destinatario invia un ack col numero più alto della sequenza completa arrivata fino a quel momento;
- quando si verifica un timeout, il mittente rispedisce il frame corrispondente.



**Figura 3-11:** Funzionamento del protocollo selective repeat

Considerazioni sul **selective repeat**:

- mittente e destinatario devono entrambi gestire un buffer per mantenervi i frame:
  - non confermati (mittente);
  - successivi ad un errore (destinatario);
- vi è un basso spreco di banda, che si può ulteriormente diminuire mandando un **NACK** (Negative ACKnowledgement) quando:
  - arriva un frame danneggiato;
  - arriva un frame diverso da quello atteso (ciò può indicare l'avvenuta perdita del frame precedente).

Infine, si noti che per entrambi i precedenti protocolli:

- è necessaria la gestione di **timer multipli** (uno per ogni frame inviato e non confermato);
- il ricevente, per inviare gli ack, usa il **piggybacking** se possibile, altrimenti invia un apposito frame.

## 6.11 Protocolli del livello Data Link

I protocolli data link più diffusi oggi sono discendenti del protocollo **SDLC (Synchronous Data Link Control)**, nato nell'ambito dell'architettura SNA. Nel seguito verranno illustrate brevemente le caratteristiche di tre diffusi protocolli: **HDLC** (standard ISO), **SLIP** (architettura TCP/IP) e **PPP** (suo successore).

### 6.11.1 HDLC

**HDLC** è un protocollo **bit oriented**, e quindi usa la tecnica del **bit stuffing**.

Il formato del frame HDLC è illustrato nella figura seguente.

Bit:	8	8	8	$\geq 0$	16	8
	01111110	Address	Control	Dati	Checksum	01111110

**Figura 3-12:** Frame HDLC

<b>Address</b>	utilizzato nelle linee multipunto, dove identifica i diversi terminali (il protocollo offre funzioni per il dialogo fra un concentratore e diversi terminali)
<b>Control</b>	contiene numeri di sequenza, ack, ecc.
<b>Dati</b>	contiene i dati da trasportare
<b>Checksum</b>	è calcolata con CRC-CCITT

#### Caratteristiche:

- usa una finestra scorrevole con numeri di sequenza a 3 bit, contenuti dentro un campo **Seq** situato all'interno del campo Control;
- utilizza il campo **Next**, anch'esso contenuto in Control, per il piggybacking degli ack;
- ha tre tipi di frame (identificati dai primi due bit di Control):
  - Information**, per la trasmissione dati;
  - Supervisory**, per comandare diverse modalità di ritrasmissione;
  - Unnumbered** (non ha il numero di sequenza), per finalità di controllo o per trasportare il traffico di connessioni non affidabili.

#### 6.11.2 SLIP

E' nato nel 1984 ed è il più vecchio protocollo di livello data link dell'Internet Protocol Suite.

Molto semplice, nacque per collegare via modem macchine Sun ad Internet. Usa **character stuffing**.

Ha diverse limitazioni:

- non c'è controllo degli errori;
- supporta solo IP, e per di più solo indirizzi statici;
- non è uno standard ufficiale di Internet.

#### 6.11.3 PPP

E' adatto sia a connessioni telefoniche che a linee router-router.

Fornisce le seguenti funzionalità:

- framing**;
- rilevamento degli errori**;
- un protocollo di controllo per attivare, testare e disattivare la linea (**LCP, Link Control Protocol**, RFC [1570](#));
- supporto di molteplici protocolli di livello network;
- una famiglia di protocolli per negoziare opzioni di livello network (**NCP, Network Control Protocol**):
  - per ogni livello network supportato c'è un differente NCP;
  - ad esempio, nel caso di IP, NCP viene usato per negoziare un indirizzo IP dinamico;

Il traffico derivante (nelle fasi iniziali e finali della connessione) dall'uso dei protocolli LCP e NCP viene trasportato dentro i frame PPP.

Il protocollo è modellato su HDLC, ma con alcune differenze:

- è character-oriented anziché bit-oriented, e utilizza il character stuffing (quindi i frame sono costituiti da un numero intero di byte);
- c'è un campo apposito per il supporto multiprotocollo offerto al livello network.

Il formato del frame PPP è il seguente.

Byte:	1	1	1	1	Variabile	2 oppure 4	1
	Flag 01111110	Address 11111111	Control 00000011	Protocol	Dati	Checksum	Flag 01111110

**Figura 3-13: Frame PPP**

<b>Flag</b>	come in HDLC
<b>Address</b>	sempre 11111111: di fatto non ci sono indirizzi, in quanto non c'è più l'idea di gestire linee multipunto
<b>Control</b>	il default (00000011) indica un unnumbered frame, quindi relativo ad un servizio non affidabile
<b>Protocol</b>	indica il protocollo relativo al pacchetto che si trova nel payload (LCP, NCP, IP, IPX, Appletalk, ecc.)
<b>Payload</b>	è di lunghezza variabile e negoziabile, il default è 1500 byte
<b>Checksum</b>	normalmente è di due byte (quattro sono negoziabili)

## 6.12 Il sottolivello MAC (Medium Access Control)

Il sottolivello **MAC (Medium Access Control)** è un sottolivello del livello data link che si occupa di implementare i meccanismi per la gestione delle linee multi-accesso o ad accesso casuale.

Come già chiarito, le reti sono divise in due categorie: **punto a punto** e **broadcast**.

Nelle reti broadcast il problema principale è decidere quale elaboratore (detto anche **stazione**) ha diritto di usare il mezzo trasmisivo quando c'è competizione. Si deve evitare che molte stazioni trasmettano contemporaneamente, perché i relativi segnali si disturberebbero a vicenda.

I protocolli per decidere chi è il prossimo a trasmettere su un canale broadcast (detto anche **multiaccess channel** o **random access channel**) appartengono ad un sottolivello del livello data link, detto **sottolivello MAC**.

Essi sono usati soprattutto nelle LAN, ma anche nelle parti di WAN basate su satelliti.

Il problema principale è come allocare il canale ai vari utenti in competizione. Ci sono due meccanismi fondamentali:

- **allocazione statica**, che viene decisa in anticipo;
- **allocazione dinamica**, che si adatta alle esigenze di ogni momento.

L'**allocazione statica** prevede la suddivisione del canale fra gli N utenti, ciascuno dei quali riceve di conseguenza una frazione della banda totale. Si può fare, ad esempio, con tecniche quali FDM, allocando a ciascun utente una banda di frequenze distinta da quella degli altri utenti. Ciò va bene se il numero di utenti non varia rapidamente e se tutti trasmettono con un data rate più o meno costante, però in genere comporta vari problemi (svantaggi):

- si verifica uno spreco di banda quando uno o più utenti non trasmettono;

- poiché il traffico è in generale molto ***bursty***, i picchi che si verificano non possono essere gestiti solamente con la sotto-banda allocata.

Viceversa, l'**allocazione dinamica** cerca di adeguarsi alle esigenze trasmissive, in modo da soddisfarle al meglio. Ci sono alcune assunzioni da fare:

1. **modello a stazioni**: ci sono N stazioni indipendenti, ognuna delle quali genera nuovi frame per la trasmissione. La probabilità di generare un frame in un intervallo di tempo T è uguale a  $pT$ , dove p è una costante e rappresenta il tasso d'arrivo dei nuovi frame. Quando un frame è generato, la stazione si blocca finché esso non è trasmesso;
2. **singolo canale**: un singolo canale, e null'altro, è disponibile per le comunicazioni; tutte le stazioni vi possono trasmettere e da esso possono ricevere, e tutte sono ad uguale livello;
3. **collisioni**: se due frame vengono trasmessi contemporaneamente, si sovrappongono ed il segnale risultante è rovinato (si verifica collisione):
  - tutte le stazioni possono rilevare la collisione;
  - i frame devono essere ritrasmessi;
  - non ci sono altri tipi di errori;
4. **tempo**: può essere gestito in due modi:
  - **continuous time**: la trasmissione di un frame può iniziare in un qualunque istante;
  - **slotted time**: il tempo è diviso in intervalli discreti (**slot**). Uno slot può contenere 0, 1 oppure più di un frame. Ciò corrisponde ad uno slot vuoto, ad uno slot con un frame e ad uno slot in cui si verifica una collisione. La trasmissione può iniziare solo all'inizio di uno slot;
5. **ascolto del canale**: ci sono due possibilità,
  - **carrier sense** (tipico delle LAN): le stazioni, prima di trasmettere, ascoltano il canale; se è occupato non cercano di trasmettere;
  - **no carrier sense** (tipico dei canali via satellite, nei quali vi è un elevato round trip time): le stazioni non ascoltano, trasmettono senz'altro; si preoccupano dopo di vedere se c'è stata una collisione.

### 6.12.1 Protocollo ALOHA

Nacque negli anni '70 per collegare tra loro, tramite radio al suolo, gli elaboratori sparsi nelle isole Hawaii. Esistono due versioni, **Pure Aloha** e **Slotted Aloha**.

Nel **Pure Aloha** le stazioni trasmettono quando vogliono, però durante la trasmissione ascoltano il canale e confrontano ciò che ricevono con ciò che hanno spedito.

Dunque, se si verifica una collisione se ne accorgono, e in tal caso, dopo aver lasciato passare una quantità di tempo casuale, ritrasmettono il frame. La scelta di attendere per una quantità di tempo casuale discende dal fatto che altrimenti una collisione ne ricrea infinite altre.

Efficienza dello schema Aloha puro:

Detto **frame time** il tempo necessario per trasmettere un frame e  $S$  è il numero medio di frame per frame time definito da una distribuzione, se  $S > 1$ , vengono inviati più frame di quanti il canale può gestirne, e si avranno quasi sempre collisioni, mentre se  $0 < S < 1$ , il numero di collisioni può essere più o meno accettabile, ma richiedono comunque delle ritrasmissioni.

Supponiamo che la distribuzione di tutti i frame (vecchi e nuovi) sia anch'essa una distribuzione di Poisson, con valor medio pari a  $G$  frame per frame time.

A basso carico ci aspettiamo poche collisioni, quindi  $G \approx S$ . Ad alto carico invece avremo più collisioni, per cui  $G$  sarà maggiore di  $S$ .

In ogni caso, sotto qualunque condizione di carico, il **throughput** (cioé la **quantità di pacchetti che arrivano a destinazione**) è uguale al carico offerto moltiplicato per la probabilità che la trasmissione abbia successo, ossia:

$$\text{Throughput} = G * P(0)$$

dove  $P(0)$  è la probabilità che un frame non soffra collisioni.

Per calcolare il throughput effettivo, e quindi l'efficienza, ottenibile col protocollo Pure Aloha, si devono fare due considerazioni.

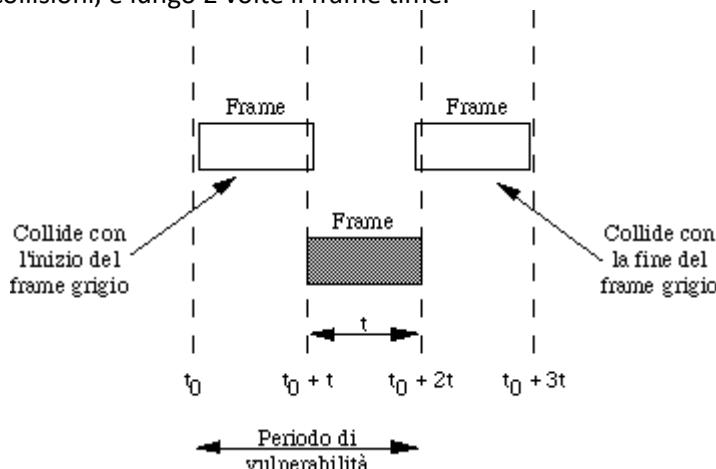
La prima è che la probabilità di generare  $k$  frame durante un intervallo di tempo pari ad un frame time è data, per la distribuzione di Poisson sopra definita (avente, si ricordi, valor medio pari a  $G$  frame per frame time) dalla relazione:

$$P(k) = \frac{G^k e^{-G}}{k!}$$

Dunque, la probabilità che si generino zero frame in un intervallo di tempo pari ad un frame time è pari a

$$P(0) = e^{-G}.$$

La seconda considerazione è che il **periodo di vulnerabilità** di un frame, cioé l'intervallo di tempo nel quale esso è a rischio di collisioni, è lungo 2 volte il frame time.



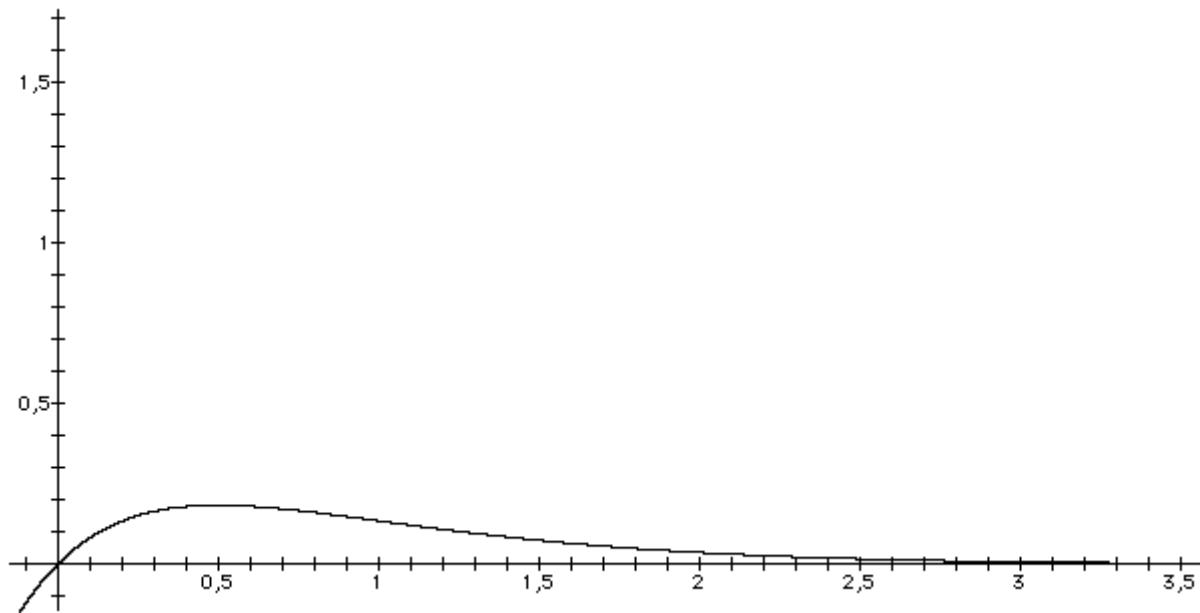
In tale periodo vengono generati mediamente  $2G$  frame. Di conseguenza, la probabilità che non si generino nuovi frame per tutto il periodo di vulnerabilità di un frame è:

$$P(0) = e^{-2G}$$

Utilizzando tale probabilità nella relazione vista sopra per il throughput, otteniamo la stima del throughput raggiungibile col protocollo Pure Aloha, che è

$$\text{Throughput} = Ge^{-2G}$$

ed ha la seguente forma:



I massimo throughput è 0,184, cioè meno del 20% (due frame su 10 slot) in corrispondenza di un carico G pari a 0,5 frame per frame time.

Un modo per aumentare l'efficienza di Aloha (Roberts, 1972) consiste nel dividere il tempo in intervalli discreti, ciascuno corrispondente ad un frame time. Ovviamente gli utenti devono essere d'accordo nel confine fra gli intervalli, e ciò può essere fatto facendo emettere da una attrezzatura speciale un breve segnale all'inizio di ogni intervallo.

Le stazioni non possono iniziare a trasmettere quando vogliono, ma solo all'inizio dell'intervalle.

Questo protocollo, che prende il nome di Slotted Aloha, dimezza il periodo di vulnerabilità che diviene uguale ad un solo frame time.

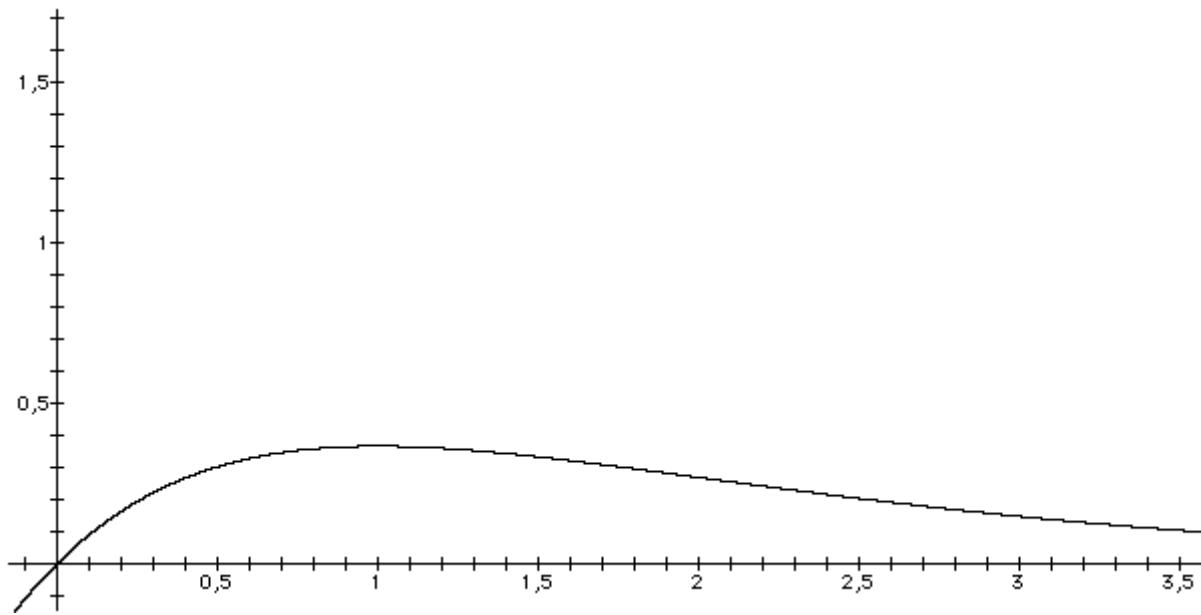
In tale periodo vengono generati mediamente G frame, per cui la probabilità che non si generino nuovi frame per tutto il periodo di vulnerabilità di un frame è:

$$P(0) = e^{-G}$$

Utilizzando tale probabilità nella relazione vista precedentemente per il throughput, otteniamo la stima del throughput raggiungibile col protocollo Slotted Aloha, che è:

$$\text{Throughput} = Ge^{-G}$$

ed ha la seguente forma:



### 6.12.2 Protocolli CSMA (Carrier Sense Multiple Access)

Nelle reti locali le stazioni possono ascoltare il canale e regolarsi di conseguenza, ottenendo un'efficienza molto più alta. I protocolli nei quali le stazioni ascoltano il canale prima di iniziare a trasmettere si dicono **carrier sense**.

Ci sono vari tipi di protocolli carrier sense:

- **1-persistent**
  - Quando una stazione deve trasmettere, ascolta il canale:
    - se è occupato, aspetta finché si libera e quindi trasmette;
    - se è libero, trasmette (con probabilità 1, da cui il nome).
  - Se avviene una collisione, la stazione aspetta un tempo random e riprova tutto da capo.
  - Problemi:
    - una stazione A trasmette, e prima che il suo segnale arrivi a B anche B inizia a trasmettere, dunque si verifica una collisione. Più alto è il tempo di propagazione fra A e B e più grave è il fenomeno;
    - A e B ascoltano contemporaneamente durante la trasmissione di C, e non appena quest'ultima termina iniziano entrambe a trasmettere: anche in questo caso si verifica una collisione.
- **Nonpersistent**
  - Quando una stazione deve trasmettere, ascolta il canale:
    - se è occupato, invece di trasmettere non appena si libera come in 1-persistent la stazione aspetta comunque un tempo random e ripete tutto il procedimento da capo;
    - se è libero, si comporta come in 1-persistent.
  - Intuitivamente, ci si aspettano maggiori ritardi prima di riuscire a trasmettere un frame e meno collisioni rispetto a 1-persistent.
- **P-persistent** (si applica a canali slotted)
  - Quando una stazione deve trasmettere, ascolta il canale:
    - se è occupato, aspetta il prossimo slot e ricomincia da capo;
    - se è libero:
      - con probabilità p trasmette subito;
      - con probabilità 1 - p aspetta il prossimo slot; se anch'esso è libero, riapplica tale procedimento;

- Il processo si ripete finché:
  - il frame è trasmesso, oppure qualcun altro ha iniziato a trasmettere. In questo caso la stazione si comporta come in una collisione: aspetta un tempo random e ricomincia da capo.
- Intuitivamente, al diminuire di  $p$  ci si aspettano crescenti ritardi prima di riuscire a trasmettere un frame ed una progressiva diminuzione delle collisioni.

#### 6.12.3 Protocolli CSMA/CD (CSMA with Collision Detection)

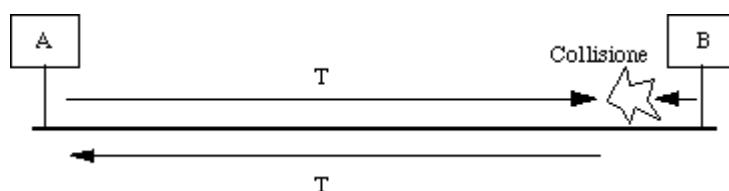
Un ulteriore miglioramento si ha se le stazioni interrompono la loro trasmissione non appena rilevano una collisione, invece di portarla a termine.

Rilevare la collisione è un processo analogico: si ascolta il canale durante la propria trasmissione, e se la potenza del segnale ricevuto è superiore a quella trasmessa si scopre la collisione.

Quando si verifica una collisione, la stazione aspetta una quantità casuale di tempo e riprova a trasmettere.

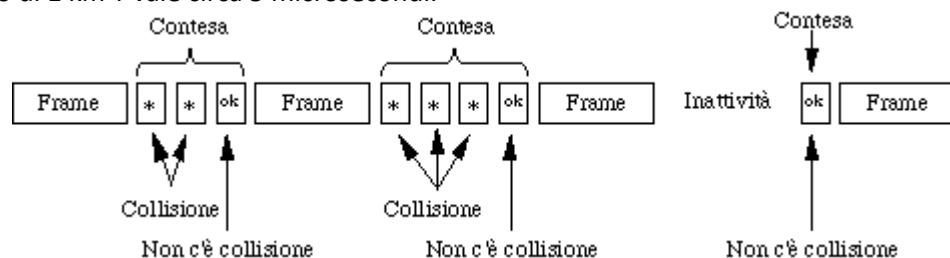
Posto uguale a  $T$  il tempo di propagazione del segnale da un capo all'altro della rete, è necessario che trascorra un tempo pari a  $2T$  perché una stazione possa essere sicura di rilevare una collisione.

Infatti, se una stazione A posta ad una estremità della rete inizia a trasmettere al tempo  $t_0$ , il suo segnale arriva a B (posta all'altra estremità della rete) dopo al tempo  $t_0 + T$ ; se un attimo prima di tale istante anche B inizia a trasmettere, la collisione conseguente viene rilevata da B quasi immediatamente, ma impiega una ulteriore quantità  $T$  di tempo per giungere ad A, che la può quindi rilevare solo un attimo prima dell'istante  $t_0 + 2T$ . (**RTT – Round Trip Time**)



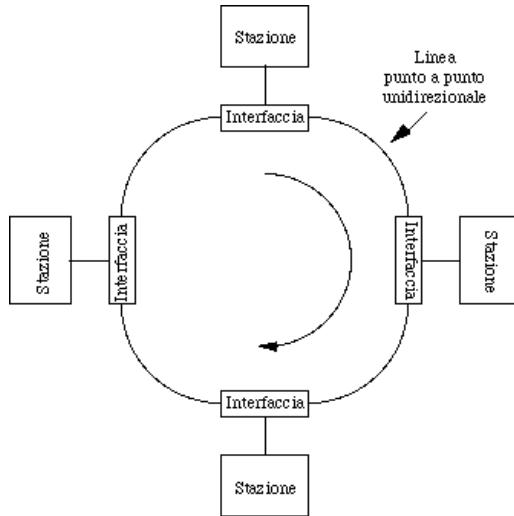
Il modello concettuale che si utilizza è il seguente:

- vi è un'alternanza di periodi di **contesa**, di **trasmissione** e di **inattività**;
- il periodo di contesa è modellato come uno Slotted Aloha con slot di durata  $2T$ : a titolo di esempio, per un cavo di 1 km  $T$  vale circa 5 microsecondi.



#### 6.12.4 Le reti ad anello

Una **rete ad anello** consiste di una collezione di interfacce di rete, collegate a coppie da linee punto a punto:

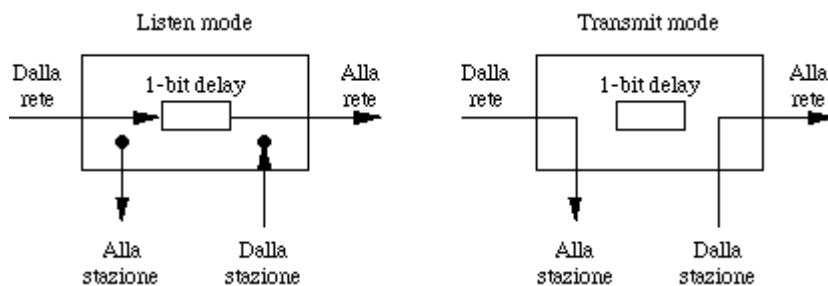


Le reti ad anello hanno diverse attrattive:

- non sono reti basate su un mezzo trasmisivo broadcast;
- non c'è una significativa componente analogica per la rilevazione delle collisioni (che non possono verificarsi);
- l'anello è intrinsecamente equo.

Ogni bit che arriva all'interfaccia è copiato in un buffer interno, poi rigenerato e ritrasmesso sul ring. Può essere modificato prima di essere ritrasmesso.

L'interfaccia di rete può operare in due diverse modalità, ***listen mode*** e ***transmit mode***:



In ***listen mode*** i bit in ingresso vengono copiati nel buffer interno (dove possono essere anche modificati) e quindi ritrasmessi con un ritardo di un bit (***1-bit delay***).

In ***transmit mode*** l'anello è aperto, e i bit in arrivo vengono rimossi; nuovi bit vengono trasmessi sull'anello.

Una speciale configurazione binaria, detta ***token (gettone)*** circola in continuazione se nessuno vuole trasmettere.

Quando una stazione vuole trasmettere, deve:

1. aspettare che arrivi il token (in listen mode);
2. rimuoverlo dal ring (in listen mode, vedremo come);
3. trasmettere i dati (in transmit mode);
4. rigenerare il token (in transmit mode);
5. rimettersi in listen mode.

Poiché c'è un solo token, questo meccanismo risolve senza conflitti il problema dell'accesso al mezzo.

**Time Holding Token (THT):** tempo massimo entro il quale, una volta preso il token, si deve completare la trasmissione (schedulazione round-robin delle trasmissioni).

Alcune considerazioni sono degne di nota:

- il token deve essere contenuto per intero sull'anello, il che non è così ovvio come sembra
- un frame, invece, non è necessario che ci stia tutto sull'anello (che in trasmissione è aperto), quindi non ci sono limiti alla dimensione dei frame;
- in genere esiste un tempo massimo entro il quale, una volta preso il token, si deve completare la trasmissione; ciò permette di ottenere una schedulazione round-robin delle trasmissioni;
- quando tutte le stazioni hanno qualcosa da trasmettere, l'efficienza si avvicina al 100%;
- viceversa, quando non c'è traffico, una stazione deve attendere un pò più che in CSMA/CD per trasmettere (mediamente dovrà attendere un tempo pari a quello di attraversamento di mezzo anello, per ricevere il token).

La velocità di propagazione del segnale nel rame è circa 200 metri per microsecondo. Con un data rate (ad esempio) di 1 Mbps, si genera un bit al microsecondo. Dunque, un bit è lungo in tal caso circa 200 metri, per cui per contenere 10 bit un anello dovrebbe essere lungo almeno 2 km.

In realtà sul ring trovano posto:

- x bit sull'anello, in funzione della sua lunghezza totale;
- y bit nei buffer delle interfacce delle y stazioni presenti (1 bit delay).

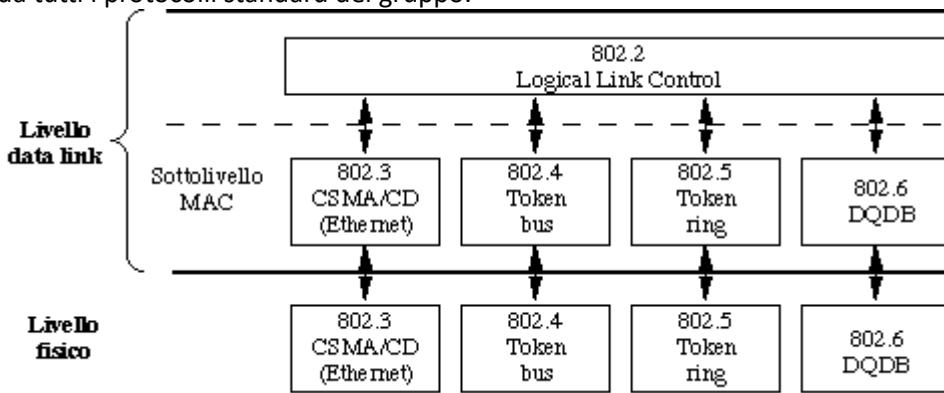
In definitiva, è necessario che  $x + y$  sia maggiore del numero di bit del token. Ciò significa che, a seconda delle caratteristiche dimensionali della rete in questione, può essere necessario ricavare un ritardo addizionale, sotto forma di buffer aggiuntivi, in una stazione (che ha un ruolo particolare, quello di **monitor dell'anello**).

## 6.13 Lo standard 802

IEEE ha prodotto diversi standard per le LAN, collettivamente noti come **IEEE 802**. Essi includono gli standard per:

- **Specifiche generali** del progetto (802.1);
- **Logical Link Control, LLC** (802.2);
- **CSMA/CD** (802.3);
- **token bus** (802.4, destinato a LAN per automazione industriale);
- **token ring** (802.5);
- **DQDB** (802.6, destinato alle MAN).

I vari standard differiscono a livello fisico e nel sottolivello MAC, ma sono compatibili a livello data link. Ciò è ottenuto separando dal resto, attraverso l'apposito standard LLC, la parte superiore del livello data link, che viene usata da tutti i protocolli standard del gruppo.



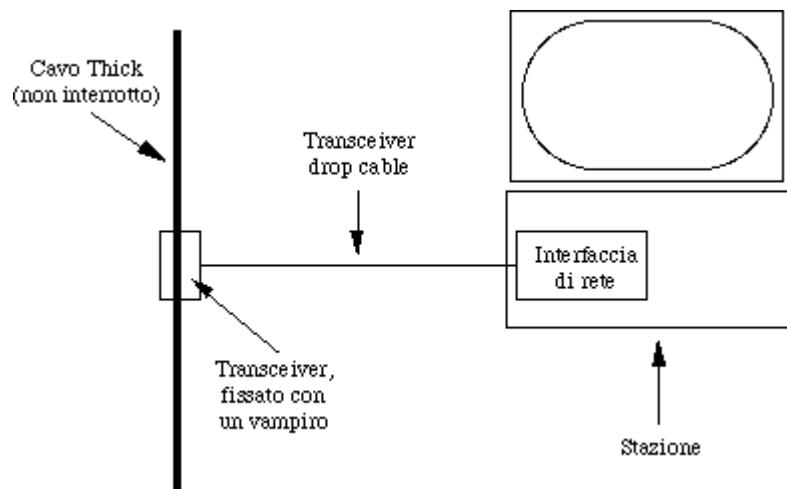
### 6.13.1 IEEE 802.3

E' lo standard per un protocollo CSMA/CD, di tipo 1-persistent, funzionante a 10Mbps. 802.3 è l'evoluzione dello standard **Ethernet**, proposto da Xerox, DEC e INTEL sulla base dell'esperienza maturata con Aloha prima e nei laboratori Xerox PARC poi.

802.3 e Ethernet hanno alcune differenze, ma sono largamente compatibili.

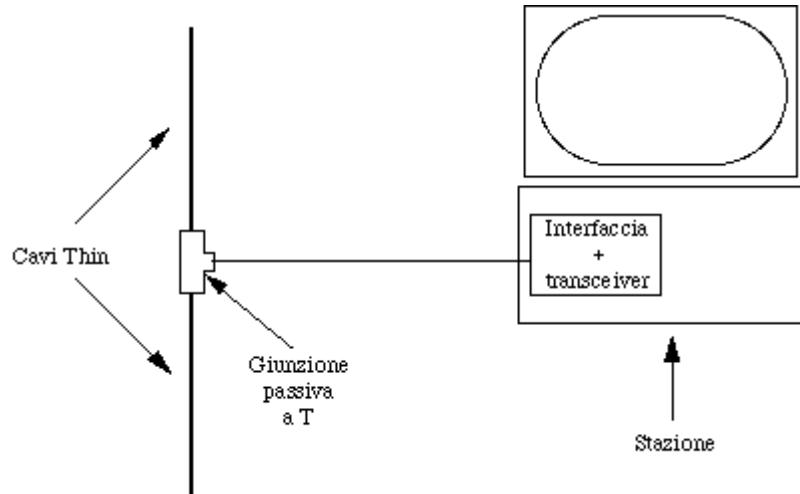
Sono previsti vari cablaggi:

- **Thick ethernet:** è il primo storicamente; consiste di un cavo coassiale spesso (lo standard suggerisce il colore giallo per la guaina esterna).
  - Ufficialmente si chiama **10Base5**, ossia:
    - 10 Mbps;
    - Baseband signaling;
    - 500 metri di lunghezza massima.
  - Possono essere installate 100 macchine su un segmento.
  - Ogni stazione contiene un'**interfaccia di rete** (detta anche **scheda ethernet**) che:
    - incapsula i dati del livello superiore;
    - gestisce il protocollo MAC;
    - codifica i dati da trasmettere;
    - in ricezione decapsula i dati, e li consegna al livello superiore (o lo informa dell'errore).
  - All'interfaccia di rete viene collegata una estremità di un corto cavo (pochi metri), detto **transceiver drop cable**, all'altra estremità del quale è connesso un **transceiver** che si aggancia, con un dispositivo detto **vampiro**, al cavo thick (che non viene interrotto).
  - Il transceiver contiene la circuiteria analogica per l'ascolto del canale e la rilevazione delle collisioni. Quando c'è una collisione, il transceiver informa l'interfaccia ed invia sulla rete uno speciale segnale di 32 bit (**jamming sequence**) per avvisare le altre stazioni, che così scartano quanto già ricevuto.



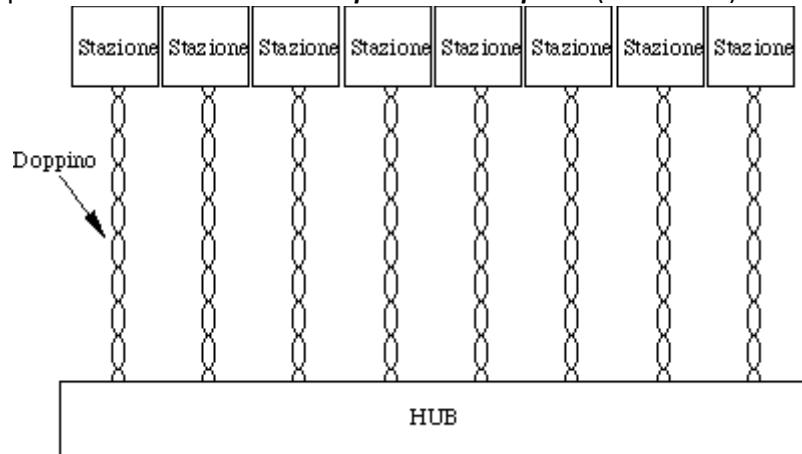
- **Thin ethernet:** è un cavo coassiale più sottile, e si piega più facilmente.
  - Ufficialmente si chiama **10Base2**, ossia:
    - 10 Mbps;
    - Baseband signaling;
    - 200 metri di lunghezza massima per un singolo segmento.
  - Possono essere installate 30 macchine su un segmento.

- Di norma l'interfaccia di rete contiene anche il transceiver.
- L'allaccio di una stazione alla rete avviene con una giunzione a T, alla quale sono collegati il cavo che porta alla stazione e due cavi thin che costituiscono una porzione del segmento. Le varie stazioni sono collegate in cascata (**daisy-chain**) sul segmento.



- **Doppino telefonico:**

- Lo standard **10BaseT** (twisted) prevede il collegamento fra una sola coppia di stazioni.
- La lunghezza massima è 100 metri (150 se il doppino è di classe 5).
- Per connettere più di due stazioni serve un **ripetitore multiporta** (detto **HUB**).



Un ripetitore è un dispositivo che opera a livello uno (fisico): riceve il segnale da un segmento, lo amplifica e lo ritrasmette su tutti gli altri segmenti. I ripetitori possono essere usati anche per aumentare la lunghezza complessiva della rete.

Comunque, sono in vigore delle regole generali stabilite dallo standard:

- la lunghezza massima dell'intera rete, fra qualunque coppia di stazioni, non deve superare i 2,5 km;
- fra qualunque coppia di stazioni non devono trovarsi più di quattro ripetitori;
- possono esservi al massimo 1024 stazioni sulla rete.

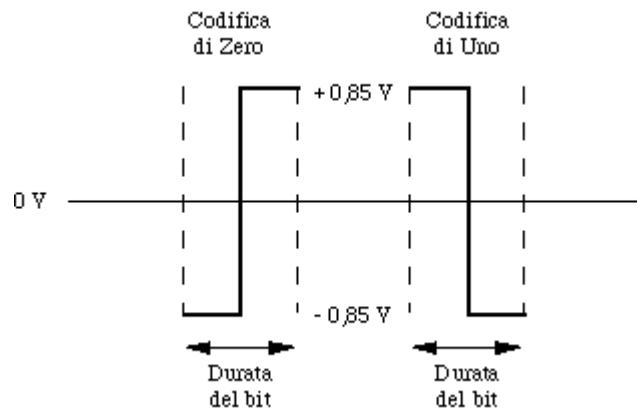
In 802.3 non si usa una codifica diretta dei dati (ad esempio, zero volt per lo zero e cinque volt per l'uno), perché sarebbe difficile rilevare le collisioni. Inoltre, si vuole delimitare con facilità l'inizio e la fine di ogni singolo bit.

Si usa una codifica, detta **Manchester**, che prevede una transizione del valore del segnale nel mezzo di ogni bit, zero o uno che sia.

In questa codifica, ogni periodo di bit è diviso in due fasi: alta e bassa.

La rappresentazione dei bit è definita dalla transizione di segnale al centro del periodo di bit.

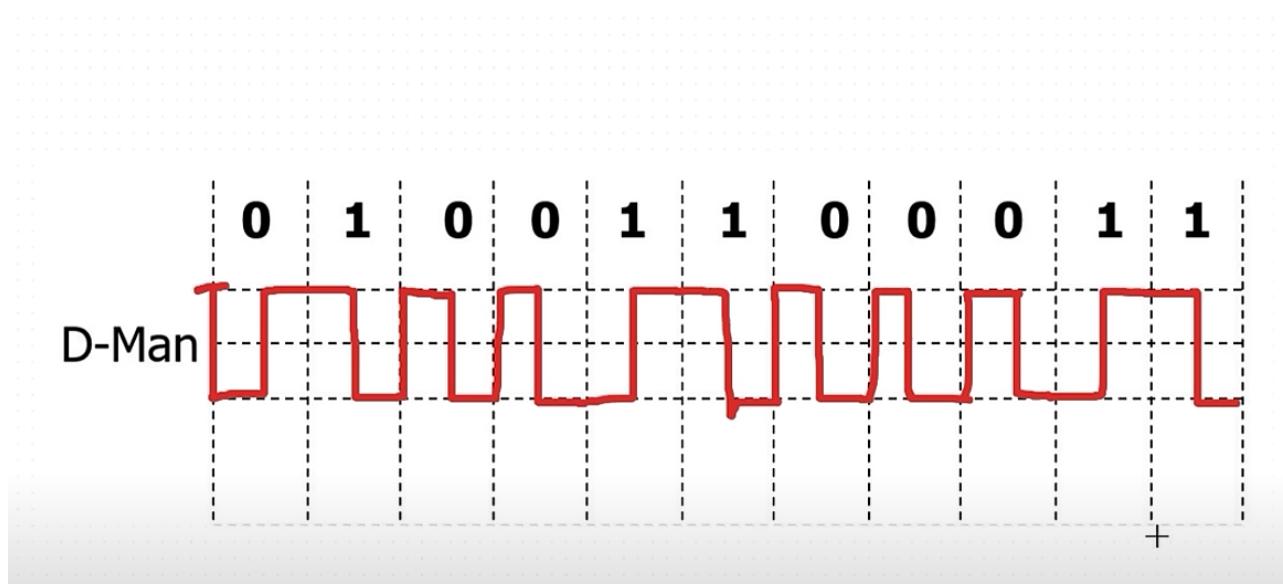
Per rappresentare un "1", viene utilizzata una transizione di segnale da alto a basso al centro del periodo di bit. Per rappresentare uno "0", viene utilizzata una transizione da basso ad alto.



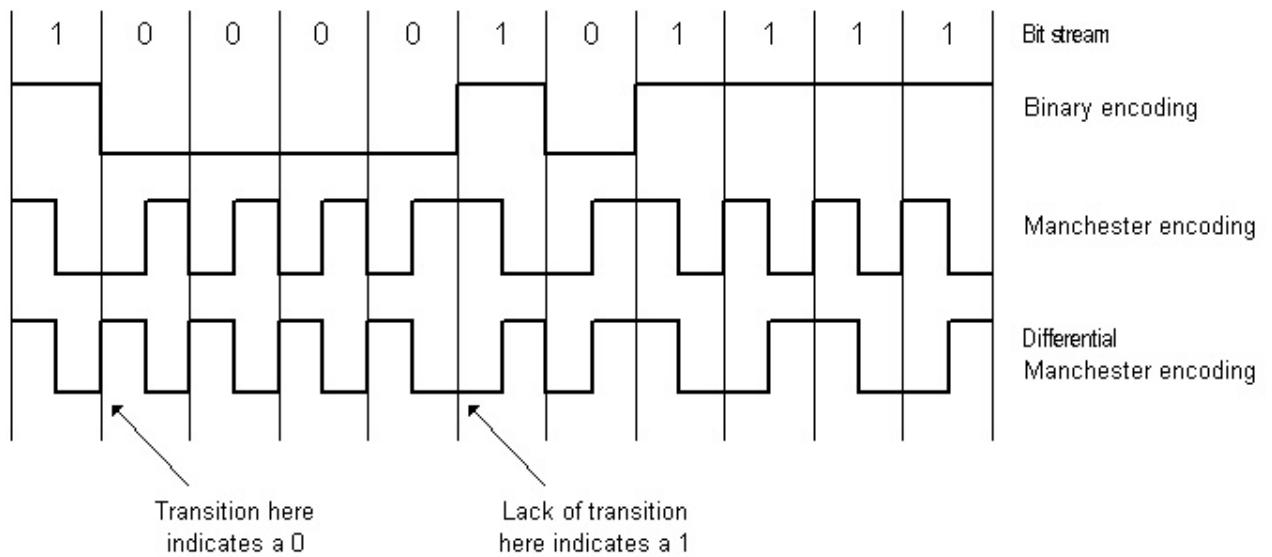
#### **Differential Manchester Encoding:**

La codifica differenziale, a differenza della codifica Manchester, si basa sulla differenza tra il bit attuale e il bit precedente.

Abbiamo un'inversione (rispetto al bit precedente) ogni volta che troviamo un 1, mentre non abbiamo un'inversione quando troviamo uno 0.



## Data Link Bit Encoding



Nei modelli ***Carrier Sensitive***, che fanno uso della codifica ***Manchester***, l'assenza di portante può essere codificata con segnale nullo

Fra i vantaggi di tale codifica:

- facilità di sincronizzazione fra mittente e destinatario;
- il codice trasmissivo è ***bilanciato***, cioè vi è uguale energia per lo zero e per l'uno, e quindi la trasmissione di dati, anche se genera diverse quantità di zeri e uni, non produce componenti in corrente continua, molto dannose perché ostacolano la trasmissione dei segnali;
- è facile rilevare le collisioni.

Si noti però che tale codifica richiede, a parità di velocità di trasmissione, una banda doppia rispetto alla codifica diretta (ogni bit richiede la trasmissione di due valori distinti).

**La struttura di un frame 802.3 è la seguente:**

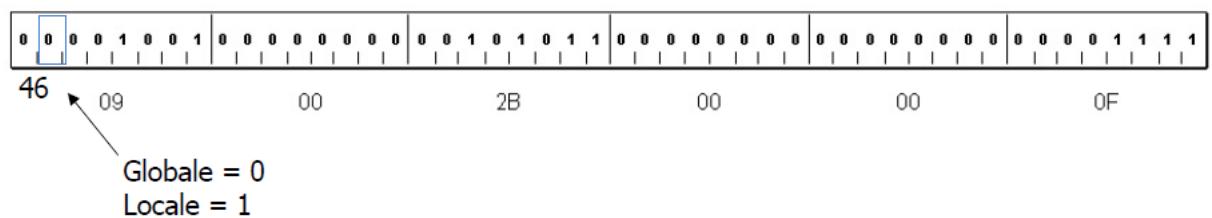
Byte:	7	1	2 opp. 6	2 opp. 6	2	0 - 1500	0 - 46	4
Preamble	Start of frame	Indirizzo destinaz.	Indirizzo sorgente	Lunghezza dei dati	Dati	Pad	Checksum	

I campi del frame hanno le seguenti funzioni:

<b>Preamble</b>	7 byte tutti uguali a 10101010. Producono, a 10 Mbps, un'onda quadra a 10 Mhz per 5,6 microsecondi, che consente al ricevitore di sincronizzare il suo clock con quello del trasmittitore.
<b>Start of frame</b>	un byte delimitatore, uguale a 10101011.
<b>Indirizzi</b>	gli indirizzi usati sono sempre a 6 byte, e sono univoci a livello mondiale (sono cablati dentro l'interfaccia). Il bit 47 (Individual, Group) definisce

	se il frame è indirizzato ad una singola stazione (unicast) o a un gruppo di stazioni (multicast). Un indirizzo composto da tutti 1 è riservato per il broadcast (il frame è ricevuto da tutte le stazioni)
<b>Lunghezza dei dati</b>	indica quanti byte ci sono nel campo dati (da 0 a 1500).
<b>Dati</b>	contiene il payload del livello superiore.
<b>Pad</b>	Se il frame (esclusi preambolo e delimiter) è più corto di 64 byte, con questo campo lo si porta alla lunghezza di 64 byte, vedremo poi perché.
<b>Checksum</b>	è un codice CRC come quelli già visti.

Indirizzi Ethernet:



Il bit 46 distingue gli indirizzi locali da quelli globali.

Gli indirizzi globali sono assegnati dalla IEEE per assicurare l'unicità degli indirizzi.  
Sono disponibili  $2^{46} \approx 7 \times 10^{13}$  indirizzi globali.

Tutte le stazioni ricevono il frame e lo accettano se l'indirizzo destinazione è compatibile con quello a loro assegnato.

- Se la trasmissione è **unicast** solo la stazione con l'indirizzo specificato nel campo destinazione del frame accetta il pacchetto. Le altre stazioni lo scartano
  - Il riconoscimento dell'indirizzo è a livello hardware
- \* Se l'interfaccia è configurata in modo promiscuo, accetta tutti i pacchetti

Nessun livello MAC garantisce un servizio affidabile. Ciò è dettato dal fatto che, visto il bassissimo tasso d'errore delle LAN, si preferisce un protocollo datagram ad alte prestazioni.

Vediamo ora perché esiste un limite minimo di 64 byte per la lunghezza di un frame.

Abbiamo già visto che, perché una collisione possa essere certamente rilevata da chi trasmette, deve passare un tempo non inferiore a due volte il tempo di attraversamento dell'intera rete.

Nel caso di IEEE 802.3, che prevede 2,5 km di lunghezza massima totale e l'interposizione di un massimo di quattro ripetitori, si ha che il tempo massimo di attraversamento dell'intera rete moltiplicato per due è pari a 57,6 microsecondi.

Ora, è essenziale che la collisione venga rilevata durante la trasmissione e non dopo, altrimenti il mittente dedurrà erroneamente che la sua trasmissione è andata a buon fine.

Dunque, la trasmissione di un frame non deve durare meno di 57,6 microsecondi, che sono il tempo necessario per trasmettere (a 10 Mbps) proprio 72 byte (e cioè 576 bit, ciascuno dei quali viene trasmesso in un decimo di microsecondo). Dunque, il frame non può essere costituito da meno di 72 byte, 8 dei quali sono costituiti dal preamble e dal delimitatore, e 64 dal resto del frame.

Si noti che se si vuole aumentare la velocità di un certo fattore, diciamo 10, si deve diminuire di 10 volte la lunghezza massima ammessa per la rete o aumentare di 10 volte la lunghezza minima del frame. Vedremo nel seguito come viene risolto il problema per il protocollo **Fast Ethernet** (100 Mbps).

Il protocollo 802.3 è un CSMA/CD di tipo 1-persistent:

- prima di trasmettere, la stazione aspetta che il canale sia libero;
- appena è libero inizia a trasmettere;
- se c'è una collisione, la circuiteria contenuta nel transceiver invia una sequenza di jamming di 32 bit, per avvisare le altre stazioni;
- se la trasmissione non riesce, la stazione attende una quantità di tempo casuale e poi riprova.

La quantità di tempo che si lascia passare è regolata da un apposito algoritmo, il **binary backoff exponential algorithm**:

- dopo una collisione, il tempo si considera discretizzato (slotted) con uno **slot time** pari a 51,2 microsecondi (corrispondenti al tempo di trasmissione di 512 bit, ossia 64 byte, pari alla lunghezza minima di un frame senza contare il preamble ed il delimiter);
- il tempo di attesa prima della prossima ritrasmissione è un multiplo intero dello slot time, e viene scelto a caso in un intervallo i cui estremi dipendono da quante collisioni sono avvenute; (va a premiare chi ha fatto meno collisioni)
- dopo n collisioni, il numero r di slot time da lasciar passare è scelto a caso nell'intervallo  $0 \leq r \leq 2^k - 1$ , con  $k = \min(n, 10)$ ;
- dopo 16 collisioni si rinuncia (invia un messaggio di errore al livello superiore).

Un numero alto di slot di attesa diminuisce la probabilità che due stazioni collidano di nuovo ma introduce un ritardo medio elevato, viceversa, un numero basso di slot di attesa rende improbabile la risoluzione della collisione quando molte stazioni collidono.

Le prestazioni osservate sono molto buone, migliori di quelle stimabili in via teorica. Peraltro, queste ultime sono fortemente influenzate dal modello di traffico che si assume.

Di solito lo si assume poissoniano, ma in realtà è bursty e per di più ***self similar***, ossia il suo andamento su un lungo periodo è simile a quello su un breve periodo, ricordando in questo le caratteristiche dei frattali.

La pratica ha mostrato che 802.3:

- può sopportare un carico medio del 30% (3 Mbps) con picchi del 60% (6 Mbps);
- sotto carico medio:
  - il 2-3% dei pacchetti ha una collisione;
  - qualche pacchetto su 10.000 ha più di una collisione.

#### 6.13.2 Fast Ethernet

Questo standard (803.2u), approvato nel 1995, prevede l'aumento di velocità di un fattore 10, da 10 Mbps a 100 Mbps.

A seconda del mezzo, abbiamo:

- Doppino cat3 (100BaseT4)
  - si usano quattro doppini Cat.3 fra l'hub ed ogni stazione:
    - uno viene usato sempre per il traffico dall'hub alla stazione;
    - uno viene usato sempre per il traffico dalla stazione all'hub;
    - 2 vengono usati di volta in volta nella direzione della trasmissione in corso;
  - la codifica è **8B6T**, cioè 8 bit vengono codificati con 6 **trit** (che hanno valore 0, 1 o 2);
  - la velocità di segnalazione è 25 Mhz (solo 25% in più di quella dello standard 802.3, che è di 20 Mhz);
  - si inviano 3 trit sui 3 doppini contemporaneamente a 25 Mhz, ossia 6 trit alla frequenza di 12,5 Mhz. Poiché 6 trit convogliano 8 bit, di fatto si inviano 8 bit a 12,5 Mhz, ottenendo così i 100 Mbps.
- Doppino cat5 (100BaseT)
  - velocità di segnalazione 124 Mhz;
  - codifica **4B5B** (4 bit codificati con 5 bit, introducendo ridondanza);
  - a seconda del tipo di hub:
    - **hub tradizionale**: la lunghezza massima di un ramo è 100 metri, quindi il diametro della rete è 200 metri (contro i 2,5 km di 802.3).
    - **switched hub**: ogni ramo è un dominio di collisione separato, e quindi (poiché su esso vi è una sola stazione) non esiste più il problema delle collisioni, ma rimane il limite di 100 metri per i limiti di banda passante del doppino.
- Fibra ottica (100BaseFX) 802.3u
  - velocità di segnalazione 125 Mhz;
  - codifica 4B5B;
  - obbligatorio switched hub;
  - lunghezza rami fino a 2 km (con uno switched hub non c'è il problema delle collisioni, ed inoltre come sappiamo la fibra regge velocità dell'ordine dei Gbps a distanze anche superiori).

#### 6.13.3 IEEE 802.5

Nel 1972 IBM scelse l'anello per la sua architettura di LAN, a cui diede il nome di ***Token Ring***.

Successivamente, IEEE ha definito lo standard IEEE 802.5 sulla base di tale architettura.

Le differenze principali sono che la rete IBM prevede velocità di 4 Mbps e 16 Mbps, mentre 802.5 prevede oltre ad esse anche la velocità di 1 Mbps

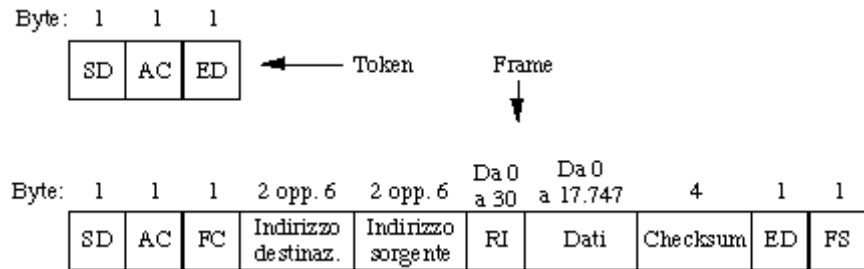
Il cablaggio più diffuso è basato su doppino telefonico:

- schermato (STP);
- non schermato (UTP):
  - categoria 3, 4 o 5 per 4 Mbps;
  - categoria 4 o 5 per 16 Mbps.

Normalmente il cablaggio è fatto utilizzando un **wire center**, che ha la possibilità di isolare parti dell'anello guaste: se manca corrente su un lobo il corrispondente relais si chiude automaticamente.

Viene usata la codifica **Differential Manchester Encoding**.

La struttura del token e del frame di 802.5 è la seguente:



I campi del frame hanno le seguenti funzioni:

<b>SD, ED</b>	Starting e ending delimiter: contengono all'interno due coppie di bit codificati con i valori high-high e low-low, in violazione della codifica Manchester. Si usano high-high e low-low accoppiati per non introdurre uno sbilanciamento del codice.
<b>AC</b>	Access control, serve per il controllo dell'accesso. E' costituito di 8 bit: PPPTMRRR <ul style="list-style-type: none"> <li>• i tre bit P indicano la <b>priorità attuale</b>;</li> <li>• il bit M serve per il controllo di <b>frame orfani</b>: il monitor lo setta ad 1 al passaggio del frame, e se lo ritrova ad uno al passaggio successivo il frame è orfano e viene tolto dall'anello;</li> <li>• il bit T, detto <b>token bit</b>, identifica un token (se vale 0) o un frame (se vale 1);</li> <li>• i tre bit R indicano la priorità richiesta.</li> </ul>
<b>FC</b>	Frame control, distingue frame contenenti dati da frame con funzioni di controllo.
<b>Indirizzi</b>	come 802.3.
<b>RI</b>	Routing information, contiene (se c'è) le informazioni necessarie al source routing (vedremo più avanti).
<b>Dati</b>	contiene il payload del livello superiore.
<b>Checksum</b>	è un codice CRC come quelli già visti.
<b>FS</b>	Frame status, serve per sapere cosa è successo del frame. Contiene, fra l'altro, due bit, A e C, gestiti come segue:

	<ul style="list-style-type: none"> <li>• bit A: viene messo ad 1 (dal destinatario) quando il frame gli arriva;</li> <li>• bit C: viene messo ad 1 (dal destinatario) quando il frame gli arriva ed il destinatario lo copia al suo interno.</li> </ul>
--	---

## Funzionamento di 802.5

Quando il token circola e una stazione vuole trasmettere, essa, che è in listen mode, opera come segue:

- aspetta che arrivi il token;
- quando il token arriva:
  - lascia passare SD;
  - lascia passare i bit PPP di AC;
  - quando ha nel buffer il token bit T:
    - lo cambia in uno, trasformando il token in un frame;
    - invia il bit T modificato sul ring;
    - si mette immediatamente in transmit mode;
    - invia il resto del frame;
- quando il frame è trasmesso:
  - se non ha esaurito il **THT (Token holding time)** può trasmettere un altro frame;
  - altrimenti rigenera un nuovo token e lo trasmette;
  - appena trasmesso l'ultimo bit del token si rimette immediatamente in listen mode.

Ogni ring ha una stazione con un ruolo speciale, il **monitor** (ogni stazione è in grado di diventare il monitor).

Se una stazione si accorge che non c'è monitor, trasmette un frame di controllo Claim Token. Se il frame ritorna, essa diviene il monitor.

Il monitor annuncia periodicamente la sua presenza con un frame di controllo Active Monitor Present.

Il monitor viene designato all'avvio dell'anello. I suoi compiti principali sono:

- rigenerare il token se esso si perde;
- ripulire il ring dai resti di frame danneggiati;
- ripulire il ring dai frame orfani.

## Confronto tra 802.3 ed 802.5

Vantaggi di 802.3:

- ha un'enorme diffusione;
- esibisce un buon funzionamento a dispetto della teoria.

Svantaggi di 802.3

- ha sostanziose componenti analogiche (per il rilevamento delle collisioni);
- il funzionamento peggiora con forte carico.

Vantaggi di 802.5:

- è totalmente digitale;
- va molto bene sotto forte carico.

Svantaggi di 802.5

- c'è ritardo anche senza carico (per avere il token);
- ha bisogno di un monitor (e se è "malato", cioè malfunzionante, e nessuno se ne accorge?).

In definitiva, nessuna delle due può essere giudicata la migliore in assoluto.

#### 6.13.4 IEEE 802.2

Questo standard, chiamato **Logical Link Control (LLC)**, definisce la parte superiore del livello data link in modo indipendente dai vari sottolivelli MAC.

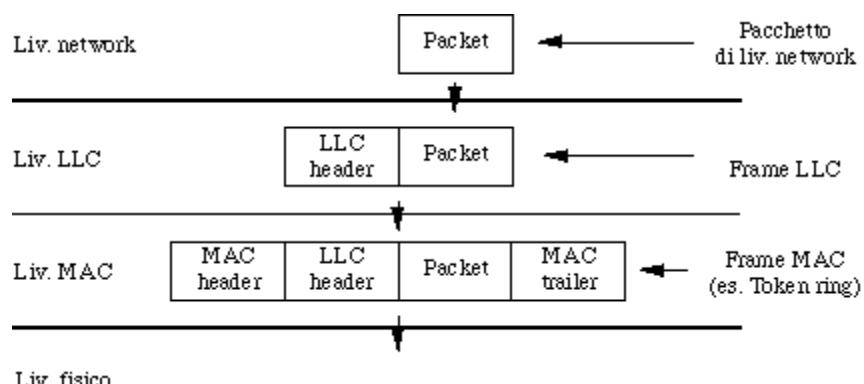
Ha due funzioni principali:

- fornire al livello network un'interfaccia unica, nascondendo le differenze fra i vari sottolivelli MAC;
- fornire, se è richiesto dal livello superiore, un servizio più sofisticato di quello offerto dai vari sottolivelli MAC (che, ricordiamo, offrono solo servizi datagram). Esso infatti offre:
  - servizi datagram;
  - servizi datagram confermati;
  - servizi affidabili orientati alla connessione.

Il frame LLC è modellato ispirandosi a HDLC, con indirizzi di mittente e destinatario, numeri di sequenze, numeri di ack (questi ultimi due omessi per i servizi datagram), ecc.

Gli indirizzi LLC sono lunghi un byte e servono sostanzialmente ad indicare quale protocollo di livello superiore deve ricevere il pacchetto di livello tre; in questo modo LLC offre un supporto multiprotocollo al livello superiore.

Il frame LLC viene imbustato, in trasmissione, in un frame dell'opportuno sottolivello MAC. Il processo inverso ha luogo in ricezione.



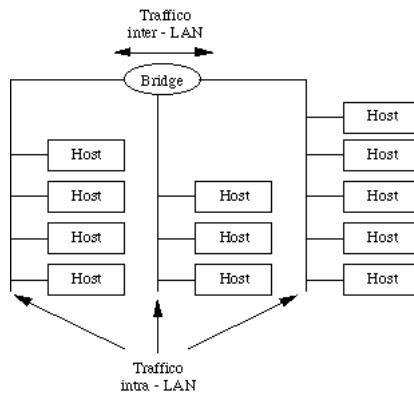
#### 6.14 | Bridge

Molto spesso c'è la necessità di connettere fra di loro LAN distinte, per molte ragioni:

- due LAN di tipo differente (ad esempio una Ethernet ed una Token ring), che non si possono semplicemente collegare l'una con l'altra, contengono host che vogliono dialogare fra loro;
- si vuole una LAN la cui lunghezza superi i limiti massimi consentiti (ad esempio, 2,5 km per Ethernet);
- si desidera, nel caso di una LAN contenente molti host, suddividerla in molteplici LAN interconnesse. Questo per tenere separato il traffico generato nelle sue parti, in modo da avere un traffico totale molto superiore a quello possibile su una singola LAN.

Due o più LAN possono essere interconnesse con dispositivi detti **bridge**, che operano a livello data link.

Ciò significa che la loro operatività è basata esclusivamente sulle informazioni contenute nelle buste di livello due, mentre non vengono prese affatto in considerazione quelle di livello tre. Questa è la caratteristica fondamentale che li differenzia dai **router**, che invece agiscono a livello tre.



In questo esempio il traffico totale (se è tutto confinato entro le singole LAN) può arrivare a tre volte quello di una singola LAN. Solo il traffico fra host di LAN diverse attraversa il bridge.

I bridge si occupano di instradare il traffico da una LAN all'altra. E' importante sottolineare che, anche se l'instradamento di per se è una funzione tipica del livello tre, qui avviene sulla base dei soli indirizzi di livello due, quindi il bridge appartiene in tutto e per tutto al livello data link.

Il funzionamento di un bridge, che ha tante interfacce di rete quante sono le LAN alle quali è fisicamente collegato, è il seguente:

- quando una delle interfacce di rete del bridge riceve un frame MAC, lo passa al relativo software di livello MAC che toglie la busta MAC;
- il resto viene passato dal livello MAC al software di livello LLC del bridge, nel quale, sulla base dell'indirizzo di destinazione, si decide a quale LAN inviarlo:
  - se la destinazione si trova sulla LAN di provenienza il frame viene scartato;
  - altrimenti, il frame LLC viene passato al livello MAC competente per la LAN di destinazione, che lo imbusta in un frame MAC e provvede ad inviarlo su tale LAN, secondo le regole di quest'ultima.

Si noti che un bridge è ben diverso da un ripetitore, che copia pedissequamente tutto ciò che riceve da una linea su tutte le altre. Il bridge infatti acquisisce un frame, lo analizza, lo ricostruisce e lo instrada, quindi può anche essere configurato in modo da **filtrare** (cioè non far passare) alcuni tipi di traffico. Ciò tipicamente avviene in funzione dell'indirizzo LLC, che identifica il protocollo di livello superiore, o sulla base dell'indirizzo MAC del mittente o del destinatario.

I bridge progettati per interconnettere LAN di tipo diverso devono risolvere vari problemi legati alle diverse regole in vigore su tali LAN, tra cui:

- formati dei frame differenti;
- data rate differenti;
- massima lunghezza di frame differente: è fuori questione spezzare un frame in questo livello, dato che tutti i protocolli si aspettano che il frame o arrivi per intero o non arrivi affatto; ad esempio, nello standard 802 i frame troppo grandi devono essere scartati;
- funzioni previste da un tipo di LAN ma non dall'altra: ad esempio, il concetto di priorità ed i bit A e C presenti in 802.5 non hanno un equivalente in 802.3.

### 6.14.1 Standard IEEE per i bridge

Ci sono due tipi di bridge standardizzati da IEEE:

- **transparent bridge** (promossi dai comitati 802.3 e 802.4)
- **source-routing bridge** (scelti dal comitato 802.5)

Il **transparent bridge** (IEEE 802.1 part D) può essere installato e diventare operativo in modo totalmente trasparente, senza richiedere niente altro che la connessione fisica e l'accensione.

Il meccanismo è il seguente:

- Dal momento in cui il bridge viene attivato, esamina tutti i frame che gli arrivano dalle varie LAN, e sulla base di questi costruisce progressivamente le sue tabelle di instradamento. Infatti, ogni frame ricevuto consente al bridge di sapere su quale LAN si trova la stazione che lo ha inviato.
- Ogni frame che arriva al bridge viene ritrasmesso:
  - se il bridge ha nelle sue tabelle di instradamento l'indirizzo del destinatario, invia il frame sulla corrispondente LAN;
  - altrimenti il frame viene inviato a tutte le LAN tranne quella di provenienza, con una tecnica detta **flooding** (che vedremo meglio più avanti);
  - man mano che il bridge aumenta la sua conoscenza degli indirizzi delle varie macchine, la ritrasmissione diventa sempre più selettiva (e quindi più efficiente).
- Le tabelle vengono aggiornate ogni qualche minuto, rimuovendo gli indirizzi che non si sono fatti vivi nell'ultimo periodo (così, se una macchina si sposta, entro pochi minuti viene di nuovo indirizzata correttamente) Questa tecnica si chiama **backward learning**.
- Se ci sono maglie nella topologia di connessione delle LAN, i bridge si costruiscono di essa uno **spanning tree**, che poi utilizzano per l'instradamento, al fine di evitare la generazione di un infinito numero di duplicati durante il flooding.

Il **source-routing bridge** (nato per le reti 802.5) è progettato invece per ottenere l'instradamento più efficiente possibile, anche a scapito della trasparenza.

L'idea di base è che il mittente indichi esplicitamente il cammino (espresso come sequenza di bridge e reti) che il frame deve percorrere. L'amministratore di sistema deve assegnare numeri di identificazione distinti ad ogni rete e ad ogni bridge, operazione che deve essere fatta manualmente.

Tali informazioni sono incluse in un apposito campo **RI (Routing Information)** del frame 802.5, e la loro eventuale presenza è indicata dal valore 1 del bit più significativo dell'indirizzo sorgente (che, essendo sempre relativo a un indirizzo singolo e mai di gruppo o broadcast, originariamente è sempre zero). Il bridge esamina solo i frame che hanno tale bit a uno.

E' ovvio che ogni host deve avere il quadro della topologia delle connessioni, memorizzato in un'apposita struttura dati. Per costruirla e mantenerla, il meccanismo usato è il seguente:

- quando un host deve spedire un frame ma non conosce il cammino da seguire per raggiungere la destinazione, invia un **discovery frame**, chiedendo tale informazione;
- il discovery frame viene inviato in flooding da ogni bridge a tutti gli altri, e quindi raggiunge tutti gli host. In questa fase, ogni bridge scrive nel discovery frame il suo ID, che si aggiunge a quello dei bridge precedentemente incontrati. Quando un discovery frame arriva alla destinazione, contiene tutto il cammino percorso;
- quando l'host di destinazione riceve un discovery frame, lo invia indietro al mittente;
- il mittente, sulla base del primo discovery frame che ritorna (considerando il relativo cammino quello più conveniente) aggiorna le sue tabelle e può mandare il frame che voleva spedire originariamente.

Un vantaggio di questo schema di funzionamento è che si trova sempre il cammino ottimo; uno svantaggio è l'esplosione del numero di discovery frame.

Dopo un periodo in cui entrambi gli standard sopra descritti erano abbastanza diffusi, oggi praticamente tutti i bridge costruiti sono di tipo transparent, ed al più offrono la funzionalità source-routing come un'opzione supplementare.

#### 6.14.2 Switch

Stessa modalità di funzionamento del bridge. Ha un numero di porte  $> 2$  (8, 12, 24, ...)

Ogni porta può essere collegata a un segmento della rete o a una stazione singola.

Contiene una tabella (**SAT = Source Address Table**) dove mappa le porte con gli indirizzi MAC.

La sua tecnologia può essere basata su:

- **Shared Memory:** Memorizza i pacchetti in una memoria comune a tutte le porte. Invia i pacchetti in memoria alla porta destinazione
- **Matrix:** Utilizza una matrice di commutazione. In base all'indirizzo e al contenuto della tabella viene attivata la connessione necessaria.
- **Bus-Architecture:** Ha un BUS interno condiviso ad alta velocità. La comunicazione interna usa TDMA

Inoltre, ci sono diverse tipologie di switch:

- **Cut-through-switching:** Il frame è subito reindirizzato sulla porta corretta
- **Store-and-forward:** Il frame è letto completamente dallo switch; viene controllato il CRC; in caso di errore il frame è scartato; permette di filtrare il traffico
- **Port-based-switching:** Ad ogni porta corrisponde un solo indirizzo Ethernet
- **Segment-based-switching:** Ad ogni porta corrispondono più indirizzi (ad esempio è collegato un HUB)

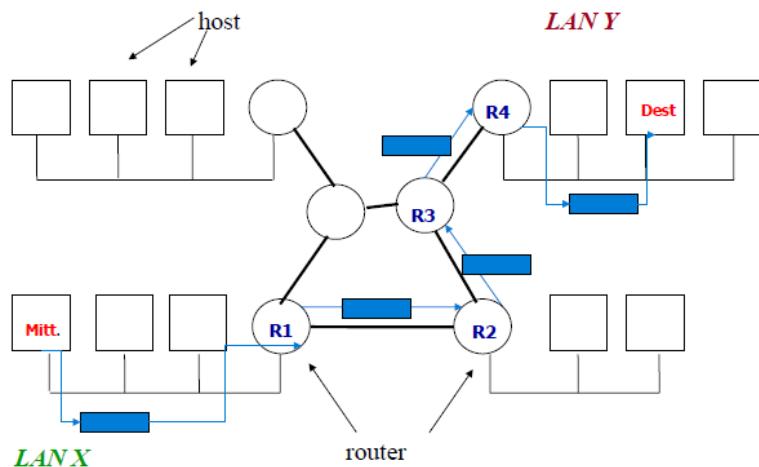
#### Qual è la differenza tra router e switch?

Uno switch agisce nel livello data link e non ha alcuna idea sul contenuto del pacchetto, è in grado solo di leggere l'header data link. Mentre un router modifica un pacchetto rimuovendo header e trailer e passando il contenuto del payload a un software di routing per scegliere una linea di output. Il router non conosce l'indirizzo del frame e non sa nemmeno se il pacchetto arriva da una LAN o una linea punto-a-punto.



## LEZIONE 10 – LIVELLO NETWORK

Il livello network è incaricato di muovere i pacchetti dalla sorgente fino alla destinazione finale, ottenuto con salti (**hop**) da un **router** all'altro attraversando reti intermedie.



Ciò è molto diverso dal compito del livello data link, che è di muovere informazioni solo da un capo all'altro di un singolo canale di comunicazione wire-like.

Le incombenze principali di questo livello sono:

- conoscere la topologia della rete;
- scegliere di volta in volta il cammino migliore (**routing**);
- gestire il flusso dei dati e le congestioni (**flow control** e **congestion control**);
- gestire le problematiche derivanti dalla presenza di più reti eterogenee (**internetworking**).

Nel progetto e nella realizzazione del livello network di una architettura di rete si devono prendere decisioni importanti in merito a:

- **servizi offerti** al livello transport;
- **organizzazione interna** della subnet di comunicazione.

Il livello Network si deve preoccupare oltre alla trasmissione dei pacchetti anche all'interfaccia che fornisce al livello trasporto, deve cioè mascherare tutta la topologia della rete al livello trasporto, e decidere se implementare una connessione di tipo connectionless, detta **datagram network** o connection-oriented, detta **VC (Virtual Circuit)**

### 10.1 Servizi offerti

In merito ai servizi offerti al livello superiore, ci sono (come abbiamo già accennato) due tipologie fondamentali di servizi:

- servizi connection-oriented;
- servizi connectionless.

In proposito, ci sono due scuole di pensiero:

- fautori dei servizi connection-oriented (compagnie telefoniche);
- fautori dei servizi connectionless (Internet Community).

**Connection-oriented** (circuiti virtuali): viene stabilito un circuito virtuale fra sorgente e destinazione  
L'ID di tale circuito viene trasmesso con il pacchetto, e tutti i router, lungo il cammino, inoltrano il pacchetto in base ad esso.

**Connectionless**: i router di volta in volta stabiliscono (in base all'indirizzo del nodo destinatario) su quale linea far proseguire la trasmissione.

La scelta si basa su apposite tabelle di instradamento (**routing table**) in cui, per ogni possibile destinazione, viene indicata la linea d'uscita.

## 10.2 Algoritmi di Routing

La funzione principale del livello network è di instradare i pacchetti sulla subnet, tipicamente facendo fare loro molti **hop** (letteralmente, salti) da un router ad un altro.

Un **algoritmo di routing** è quella parte del software di livello network che decide su quale linea di uscita instradare un pacchetto che è arrivato.

Mentre nelle subnet **connection-oriented** l'**algoritmo di routing** viene applicato solo nella fase di setup (**session routing**), in quelle **connectionless** viene applicato ex novo per ogni pacchetto.

Un algoritmo di routing deve essere:

- di **facile implementazione**;
- **robusto** (deve essere in grado di operare anche in caso di cadute di linee e/o router);
- **convergente** (e possibilmente in fretta);
- **ottimale** (scegliere il percorso ottimo).

## 10.3 Principio di ottimalità

E' possibile fare una considerazione generale sull'ottimalità dei cammini, indipendentemente dallo specifico algoritmo adottato per selezionarli.

Il **principio di ottimalità** dice che se il router j è nel cammino ottimo fra i e k, allora anche il cammino ottimo fra j e k è sulla stessa strada:

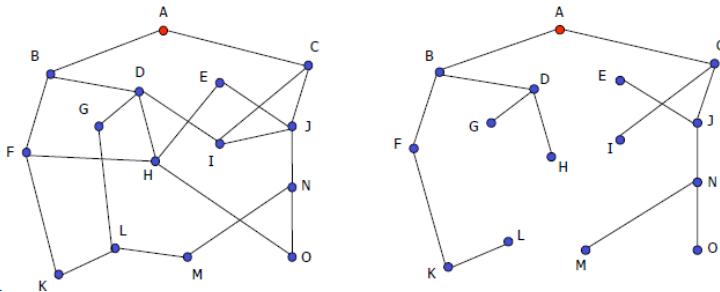


Se così non fosse, ci sarebbe un altro cammino (ad es. quello tratteggiato in figura) fra j e k migliore di quello che è parte del cammino ottimo fra i e k, ma allora ci sarebbe anche un cammino fra i e k migliore di quello ottimo.

Una diretta conseguenza è che l'insieme dei cammini ottimi da tutti i router a uno specifico router di destinazione costituiscono un **albero**, detto **sink tree** per quel router.

In sostanza, gli algoritmi di routing cercano e trovano i sink tree relativi a tutti i possibili router di destinazione, e quindi instradano i pacchetti esclusivamente lungo tali sink tree.

Il **Sink tree** per un router **A** è l'**albero** costituito dall'insieme dei cammini ottimi da tutti i router ad **A**.



Gli algoritmi di routing individuano i sink tree relativi a tutti i possibili router destinazione, ed instradano esclusivamente attraverso i cammini previsti dal sink-tree, instradando i pacchetti sempre per il cammino ottimo.

## 10.4 Classificazione algoritmi di routing

Gli algoritmi di routing possono essere:

- **statici:** Sono detti anche **algoritmi non adattivi** poiché sono eseguiti solamente all'avvio della rete, e le decisioni di routing a cui essi pervengono sono poi applicate senza più essere modificate. I router instradano i pacchetti secondo i circuiti virtuali ad essi inizialmente associati;
- **dinamici:** Sono detti anche **algoritmi adattivi** poiché ogni router sceglie la porta di uscita in base all'indirizzo del destinatario, al traffico, alla topologia della rete, ecc.

## 10.5 Algoritmi di routing statici

### 10.5.1 Shortest path routing

L'idea dello **Shortest path routing** è semplice: un host di gestione della rete mantiene un grafo che rappresenta la subnet:

- i nodi rappresentano i router;
- gli archi rappresentano le linee punto-punto.

All'avvio della rete (o quando ci sono variazioni permanenti della topologia) l'algoritmo:

- applica al grafo un algoritmo per il calcolo del **cammino minimo** fra ogni coppia di nodi; ad esempio, il noto algoritmo di Dijkstra ('59) può essere usato;
- invia tali informazioni a tutti i router.

Il cammino minimo cambia in funzione delle grandezze che si desidera minimizzare (numero di hop, lunghezza dei collegamenti, ...).

### 10.5.2 Flooding

La tecnica del **flooding** consiste nell'inviare ogni pacchetto su tutte le linee eccetto quella da cui è arrivato.

In linea di principio il flooding può essere usato come algoritmo di routing (ogni pacchetto inviato arriva a tutti i router) ma presenta l'inconveniente di generare un numero enorme (teoricamente infinito!) di pacchetti.

Ci sono delle tecniche per limitare il traffico generato:

- inserire in ogni pacchetto un **contatore** che viene decrementato ad ogni hop. Quando il contatore arriva a zero, il pacchetto viene scartato. Un appropriato valore iniziale può essere il diametro della subnet;
- inserire la coppia (**source router ID, sequence number**) in ogni pacchetto. Ogni router esamina tali informazioni e ne tiene traccia, e quando le vede per la seconda volta scarta il pacchetto;
- **selective flooding**: i pacchetti vengono duplicati solo sulle linee che vanno all'incirca nella giusta direzione (per questo si devono mantenere apposite tabelle a bordo).

### 10.5.3 Flow-based routing

Questo algoritmo è basato sull'idea di:

- calcolare in anticipo il traffico atteso su ogni linea;
- da questi calcoli derivare una stima del ritardo medio atteso per ciascuna linea;
- basare su tali informazioni le decisioni di routing.

Sceglie il cammino che minimizza il ritardo medio.

Per applicare l'algoritmo è necessario conoscere:

- *la topologia della rete*;
- *la matrice del traffico  $T(i,j)$*  (*l'elemento  $t(i,j)$  indica il traffico stimato fra il router  $i$  ed il router  $j$* );
- *la matrice delle capacità  $B(i,j)$ , espresse in bps* (*l'elemento  $b(i,j)$  indica la capacità della linea point to point che collega il router  $i$  al router  $j$* ).

Il **Flow-based routing** calcola il **ritardo medio dell'intera rete** effettuando la somma pesata dei ritardi delle singole linee (il peso di ogni linea è dato dal traffico su quella linea diviso il traffico totale sulla rete).

**Fissato un percorso tra mittente e destinatario:**

- *si calcola il traffico che incide su ogni linea (dato alla somma di tutti i  $t(i,j)$  instradati su quella linea);*
- *si calcola il ritardo di ogni linea;*
- *si calcola il ritardo medio dell'intero percorso;*
- *si ripete il procedimento per tutti i possibili percorsi, scegliendo alla fine quello che ha il minimo ritardo medio*

### 10.6 Algoritmi dinamici

Nelle moderne reti si usano algoritmi dinamici, che si adattano automaticamente ai cambiamenti della rete. Questi algoritmi non sono eseguiti solo all'avvio della rete, ma rimangono in esecuzione sui router durante il normale funzionamento della rete.

#### 10.6.1 Distance vector routing

Ogni router mantiene una tabella (**vector**) contenente un elemento per ogni altro router. Ogni elemento della tabella contiene:

- la distanza (numero di hop, ritardo, ecc.) che lo separa dal router in oggetto;
- la linea in uscita da usare per arrivarci;

Per i suoi vicini immediati il router stima direttamente la distanza dei collegamenti corrispondenti, mandando speciali **pacchetti ECHO** e misurando quanto tempo ci mette la risposta a tornare.

A intervalli regolari ogni router manda la sua tabella a tutti i vicini, e riceve quelle dei vicini.

Quando un router riceve le nuove informazioni, calcola una nuova tabella scegliendo, fra tutte, la concatenazione migliore

**se stesso -> router immediato -> router remoto di destinazione**

per ogni destinazione.

Ovviamente, la migliore è la concatenazione che produce la minore somma di:

- distanza fra il router stesso ed un suo vicino immediato (viene dalla misurazione diretta);
- distanza fra quel vicino immediato ed il router remoto di destinazione (viene dalla tabella ricevuta dal vicino immediato).

L'algoritmo **distance vector routing** funziona piuttosto bene, ma è molto lento nel reagire quando un collegamento va giù. Ciò è legato al fatto che i router non conoscono la topologia della rete.

Infatti, consideriamo questo esempio:

A	B	C	D	E	<- Router
*	-----*	-----*	-----*	*	<- Collegamenti (topologia lineare)
	1	2	3	4	<- Distanze da A

Se ora cade la linea fra A e B, dopo uno scambio succede questo:

A	B	C	D	E	<- Router
*	*-----*	-----*	-----*	*	<- Collegamenti
	3	2	3	4	<- Distanze da A (dopo uno scambio)

Ciò perché B, non ricevendo risposta da A, crede di poterci arrivare via C, che ha distanza due da A. Col proseguire degli scambi, si ha la seguente evoluzione:

A	B	C	D	E	<- Router
*	-----*	-----*	-----*	*	<- Collegamenti
	3	4	3	4	<- Distanze da A (dopo due scambi)
	5	4	5	4	<- Distanze da A (dopo tre scambi)
	5	6	5	6	<- Distanze da A (dopo quattro scambi)

A lungo andare, tutti i router vedono lentamente aumentare sempre più la distanza per arrivare ad A. Questo è il problema del **count-to-infinity**.

Se la distanza rappresenta il numero di hop si può porre come limite il diametro della rete; nel caso invece rappresentasse il tempo, i cammini che a causa della congestione dovessero presentare un forte ritardo, verrebbero erroneamente considerati interrotti.

Sono state proposte molte soluzioni al problema count-to-infinity, ma nessuna veramente efficace.

Nonostante ciò, il distance vector routing era l'algoritmo di routing di ARPANET ed è usato anche in Internet col nome di **RIP (Routing Internet Protocol)**, e nelle prime versioni di DECnet e IPX.

#### 10.6.2 Link state routing

Soprattutto a causa della lentezza di convergenza del distance vector routing, si è cercato un approccio diverso, che ha dato origine al **link state routing**.

L'idea è questa:

- ogni router tiene sott'occhio lo stato dei collegamenti fra se e i suoi vicini immediati (misurando il ritardo di ogni linea) e distribuisce tali informazioni a tutti gli altri;
- sulla base di tali informazioni, ogni router ricostruisce localmente la topologia completa dell'intera rete e calcola il cammino minimo fra se e tutti gli altri.

Quando il router si avvia, invia un **pacchetto HELLO** su tutte le linee in uscita. In risposta riceve dai vicini i loro indirizzi (univoci in tutta la rete).

Successivamente, invia vari pacchetti ECHO, misurando così il tempo di arrivo della risposta e, mediando su vari pacchetti, si deriva il ritardo della linea.

Si costruisce un pacchetto con:

- identità del mittente;
- numero di sequenza del pacchetto;
- età del pacchetto;
- lista dei vicini con i relativi ritardi.

La costruzione e l'invio di tali pacchetti si verifica tipicamente:

- a intervalli regolari;
- quando accade un evento significativo (es.: una linea va giù o torna su).

La distribuzione dei pacchetti è la parte più delicata, perché errori in questa fase possono portare qualche router ad avere idee sbagliate sulla topologia, con conseguenti malfunzionamenti.

Di base si usa il **flooding**, inserendo nei pacchetti le coppie (source router ID, sequence number) per eliminare i duplicati. Tutti i pacchetti sono confermati. Inoltre, per evitare che pacchetti vaganti (per qualche errore) girino per sempre, l'età del pacchetto viene decrementata ogni secondo e, quando arriva a zero, il pacchetto viene scartato.

Quindi, ricapitolando, il **link state routing**:

1. Scopre i propri vicini e i loro indirizzi di rete spedendo uno speciale pacchetto di **HELLO** su ogni linea
2. Misura il ritardo o il costo per ognuno dei suoi vicini con speciali pacchetti ECHO
3. Costruisce un pacchetto contenente tutto quello che ha appena scoperto, più l'identità del router ed un numero di sequenza a 32 bit (pacchetto LSP)
4. Via **flooding** inserendo nei pacchetti le coppie (source router ID, sequence number) spedisce questo pacchetto a tutti i router
5. Ogni router ha una mappa della rete costruita proprio grazie allo scambio dei pacchetti LSP
6. Calcola il cammino minimo per ogni altro router utilizzando l'algoritmo di Dijkstra

Il link state routing è molto usato attualmente:

- in Internet **OSPF (Open Shortest Path First)** è basato su tale principio e si avvia ad essere l'algoritmo più utilizzato;

- un altro importante esempio è **IS-IS (Intermediate System-Intermediate System)**, progettato per DECnet e poi adottato da OSI. La sua principale caratteristica è di poter gestire indirizzi di diverse architetture (OSI, IP, IPX) per cui può essere usato in reti miste o multiprotocollo.

## 10.7 Routing gerarchico

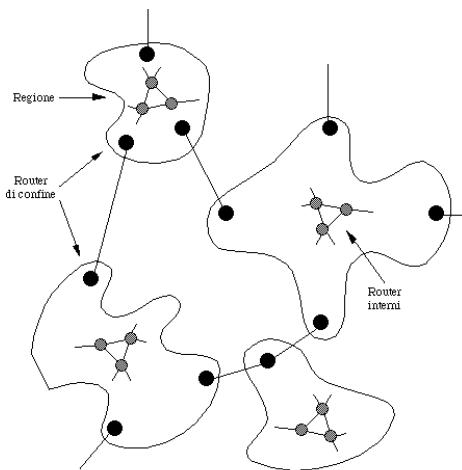
Quando la rete cresce fino contenere decine di migliaia di nodi, diventa troppo gravoso mantenere in ogni router la completa topologia. Il routing va quindi impostato in modo gerarchico, come succede nei sistemi telefonici.

La rete viene divisa in **zone** (spesso dette **regioni**):

- all'interno di una regione vale quanto visto finora, cioè i router (detti **router interni**) sanno come arrivare a tutti gli altri router della regione;
- viceversa, quando un router interno deve spedire qualcosa a un router di un'altra regione sa soltanto che deve farlo pervenire a un particolare router della propria regione, detto **router di confine**.
- il router di confine sa a quale altro router di confine deve inviare i dati perché arrivino alla regione di destinazione.

Di conseguenza, ci sono due livelli di routing:

- un primo livello di routing all'interno di ogni regione;
- un secondo livello di routing fra tutti i router di confine.



I **router interni** mantengono nelle loro tabelle di routing:

- una entrata per ogni altro router interno, con la relativa linea da usare per arrivarci;
- una entrata per ogni altra regione, con l'indicazione del relativo router di confine e della linea da usare per arrivarci.

I **router di confine**, invece, mantengono:

- una entrata per ogni altra regione, con l'indicazione del prossimo router di confine da contattare e della linea da usare per arrivarci.

**Tabelle per il routing gerarchico:**

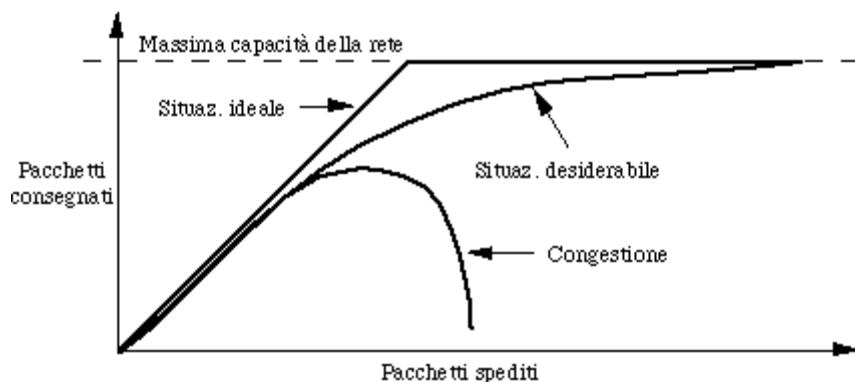
- I **router interni** mantengono una riga per ogni altro router interno, con la relativa linea da usare per arrivarci e una riga per ogni regione, con l'indicazione del relativo router di confine e della linea da usare per arrivarci.

- I **router di confine** mantengono una riga per ogni regione, con l'indicazione del router di confine da contattare e della linea da usare per arrivarcì.

Non è detto che due livelli siano sufficienti. In tal caso il discorso si ripete su più livelli.

## 10.8 Controllo della congestione

Quando troppi pacchetti sono presenti in una parte della subnet, si verifica una **congestione** che degrada le prestazioni. Ciò dipende dal fatto che, quando un router non riesce a gestire tutti i pacchetti che gli arrivano, comincia a perderli, e ciò causa delle ritrasmissioni che aggravano ancor più la congestione.



La congestione in un router può derivare da diversi fattori:

- **troppi pochi buffer nel router;**
- **processore troppo lento nel router;**
- **linea di trasmissione troppo lenta** (si allunga la coda nel router di partenza).

Inoltre, la congestione in un router tende a propagarsi ai suoi vicini che gli inviano dati. Infatti, quando tale router è costretto a scartare i pacchetti che riceve non li conferma più, e quindi i router che li hanno spediti devono mantenerli nei propri buffer, aggravando così anche la propria situazione.

Il **controllo della congestione** è un problema globale di tutta la rete, ed è ben diverso dal problema del controllo di flusso nei livelli data link, network (nel caso dei servizi connection oriented) e trasporto, che invece riguarda una singola connessione sorgente-destinazione.

Ci sono due approcci al problema della congestione:

- **open loop** (senza controreazione);
- **closed loop** (con controreazione)

Nell'approccio **open loop** si fissano parametri per la rete (*velocità di trasmissione, ampiezza dei buffer..*) in modo che la congestione non si verifichi, ma poi **non** effettua azioni correttive.

Nell'approccio **closed loop** controlla la situazione della rete, intraprendendo le azioni opportune quando necessario.

### 10.8.1 Approccio Traffic shaping (open loop)

In questo approccio, di tipo open loop, l'idea è di forzare la trasmissione dei pacchetti a un ritmo piuttosto regolare, onde limitare la possibilità di congestioni.

Verdiamo tre tecniche per implementare il traffic shaping:

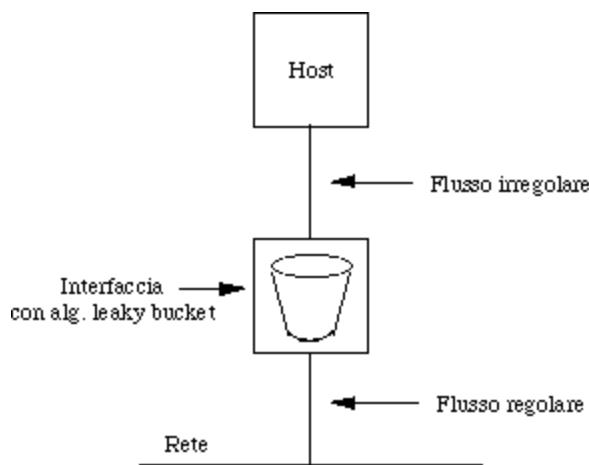
- **leaky bucket;**
- **token bucket;**
- **flow specification.**

#### Algoritmo Leaky bucket (secchio che perde)

L'idea è semplice, e trova un'analogia reale in un secchio che viene riempito da un rubinetto (che può essere continuamente manovrato in modo da risultare più o meno aperto) e riversa l'acqua che contiene attraverso un forellino sul fondo, a ritmo costante. Se viene immessa troppa acqua, essa fuoriesce dal bordo superiore del secchio e si perde.

Sull'host si realizza (nell'interfaccia di rete o in software) un leaky bucket, che è autorizzato a riversare sulla rete pacchetti con un fissato data rate (diciamo  $b$  bps) e che mantiene, nei suoi buffer, quelli accodati per la trasmissione.

Se l'host genera più pacchetti di quelli che possono essere contenuti nei buffer, essi si perdono.



In questo modo l'host può anche produrre un traffico **bursty** senza creare problemi sulla rete, finché il data rate medio non supera i  $b$  bps, oltre si cominciano a perdere pacchetti.

#### Algoritmo token bucket (secchio di gettoni)

E' una tecnica per consentire un grado di irregolarità controllato anche nel flusso che esce sulla rete.

Essenzialmente, si accumula un credito trasmisivo con un certo data rate (fino ad un massimo consentito) quando non si trasmette nulla.

Quando poi c'è da trasmettere, lo si fa sfruttando tutto il credito disponibile per trasmettere, fino all'esaurimento di tale credito, alla massima velocità consentita dalla linea.

Il secchio contiene dei **token**, che si creano con una cadenza prefissata (ad esempio uno ogni millisecondo) fino a che il loro numero raggiunge un valore  $M$  prefissato, che corrisponde all'aver riempito il secchio di token.

Per poter trasmettere un pacchetto (o una certa quantità di byte), deve essere disponibile un token.

#### Flow specification

Il traffic shaping è molto efficace se tutti (sorgente, subnet e destinazione) si accordano in merito.

Un modo di ottenere tale accordo consiste nello specificare:

- le caratteristiche del traffico che si vuole inviare (data rate, grado di burstiness, ecc.);
- la qualità del servizio (ritardo massimo, frazione di pacchetti che si può perdere, ecc.).

Tale accordo si chiama ***flow specification*** e consiste in una struttura dati che descrive le grandezze in questione.

**Sorgente, subnet e destinatario** si accordano di conseguenza per la trasmissione.

Questo accordo viene preso prima di trasmettere, e può essere fatto sia in subnet connesse (e allora si riferisce al circuito virtuale) che in subnet non connesse (e allora si riferisce alla sequenza di pacchetti che sarà trasmessa).

In generale nelle reti connesse è più facile il controllo della congestione, perché le risorse per una connessione sono allocate in anticipo.

Quindi, nel caso di circuiti virtuali, per evitare la congestione è possibile negare l'attivazione di nuovi circuiti virtuali ove non vi siano sufficienti risorse per gestirli. Questa tecnica va sotto il nome di ***admission control***.

#### 10.8.2 Approccio Choke packet (closed loop)

Quando il grado di utilizzo di una linea in uscita si avvicina ad una soglia prefissata, il router con un choke packet (**to choke: soffocare**) invita l'Host d'origine a diminuire il flusso.

Quando l'host sorgente riceve il choke packet diminuisce il flusso (tipicamente lo dimezza) e ignora i successivi choke packet per un tempo prefissato, perché tipicamente ne arriveranno molti in sequenza. Trascorso tale tempo prefissato, l'host si rimette in attesa di altri choke packet. Se ne arrivano altri, riduce ancora il flusso. Altrimenti, aumenta di nuovo il flusso.

#### Hop-by-hop choke packet

L'unico problema della tecnica precedente è la lentezza di reazione, perché l'host che produce i pacchetti ci mette un certo tempo a ricevere i choke packet ed a diminuire di conseguenza il ritmo della trasmissione. Per migliorare le cose si può costringere ogni router sul percorso, appena riceve tali pacchetti, a rallentare subito il ritmo. In tal caso si parla di ***hop-by-hop choke packet***.

Questa tecnica rende molto più veloce il sollievo del router che ha per primo i problemi di congestione, ma richiede più spazio di buffer nei router sul percorso dall'host originario a quel router.

La **qualità del servizio (QoS, Quality of Service)**, è altresì importante a seconda del tipo di trasmissione. In particolare per assicurare una QoS bisogna rispondere a 4 problematiche:

- Cosa l'applicazione richiede dalla rete
- Come regolare il traffico che entra nella rete
- Come predisporre abbastanza risorse ai router per garantire le prestazioni
- Se la rete può o meno accettare altro traffico

Nessun meccanismo da solo può affrontare tutte allo stesso momento, pertanto viene implementato un diverso range di tecniche sia al livello network che al livello trasporto.

Il **packet scheduling** indica l'ordine in cui i pacchetti che arrivano vengono immessi sulle linee in uscita, il sistema più semplice è ovviamente di tipo FIFO, ma soffre del problema che un mittente che invia tanti dati potrebbe essere privilegiato, per questo motivo è possibile implementare il **Fair Queuing** dove i pacchetti

vengono divisi in diverse code scansionate in maniera round-robin, oppure una versione migliorata detta **WFQ (Weighted Fair Queueing)** dove le code ora hanno diverso peso. Ciò viene fatto per evitare che i mittenti che inviano pacchetti di dimensioni più elevate ottengano una quantità di banda maggiore. I sistemi di QoS possono essere costruiti come **servizi integrati o servizi differenziati**. I servizi integrati sono costruiti creando delle linee nel momento in cui devono essere effettuate trasmissioni unicast o multicast, ma sono difficili da implementare in quanto dovrebbe essere instaurata una nuova linea per milioni di utenti. I servizi differenziati dividono invece i pacchetti in classi di diversa priorità, e il router divide le code di uscita in diverse code divise in priorità diverse.

## 10.9 Internetworking

Come sappiamo, esistono diverse architetture di rete, ciascuna caratterizzata dalle scelte effettuate in molti settori fra i quali ricordiamo:

- i servizi offerti dai vari livelli (connection oriented o no, reliable o no, ecc.);
- le modalità di indirizzamento;
- la dimensione massima dei pacchetti.

Per connettere fra loro reti eterogenee si devono superare problemi non banali, tra i quali:

- difformità nei servizi offerti (ad esempio, un servizio connected viene offerto solo su una delle reti);
- difformità nei formati dei pacchetti e degli indirizzi;
- difformità, nelle subnet, dei meccanismi di controllo dell'errore e della congestione;
- difformità nella dimensione massima dei pacchetti.

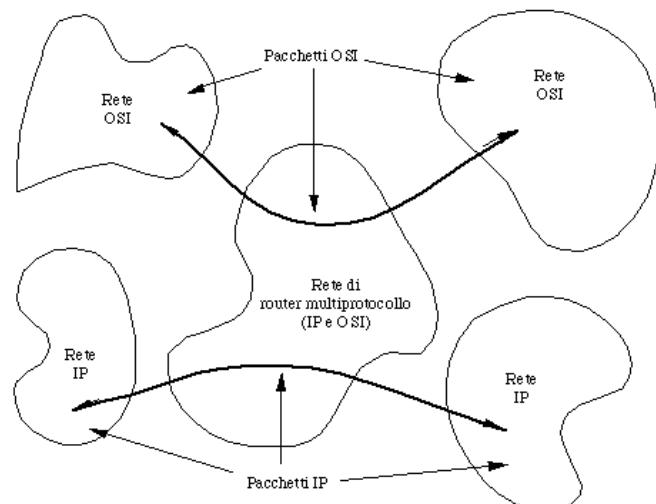
A causa di tali problemi, in generale (a meno che le architetture di rete non siano molto simili) non si usa questo approccio generale, ma altre tecniche più specifiche.

Tali tecniche sono basate sull'uso di particolari attrezature, che operano a livelli diversi:

- i bridge, che abbiamo già visto e che operano a livello data link;
- i **router multiprotocollo**: sono dei router in grado di gestire contemporaneamente più pile di protocolli.

### Reti di router multiprotocollo

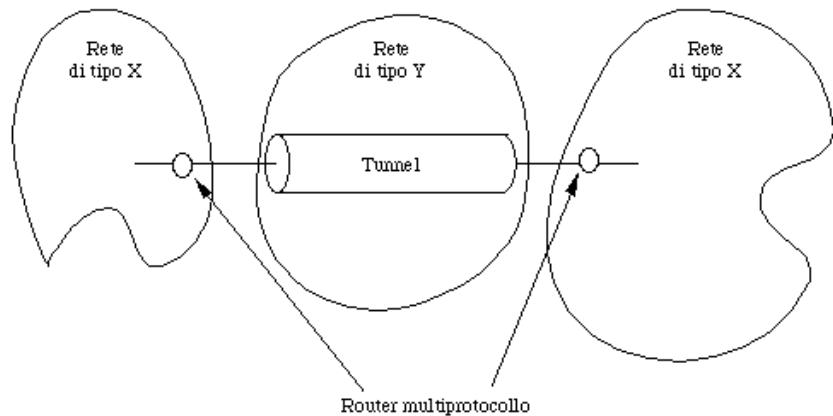
Mediante ricorso a tali apparecchiature diventa possibile, per esempio, impostare una internetwork costituita di reti eterogenee. Ogni rete può comunicare con le altre reti a lei conformi attraverso una porzione di rete costituita di router multiprotocollo.



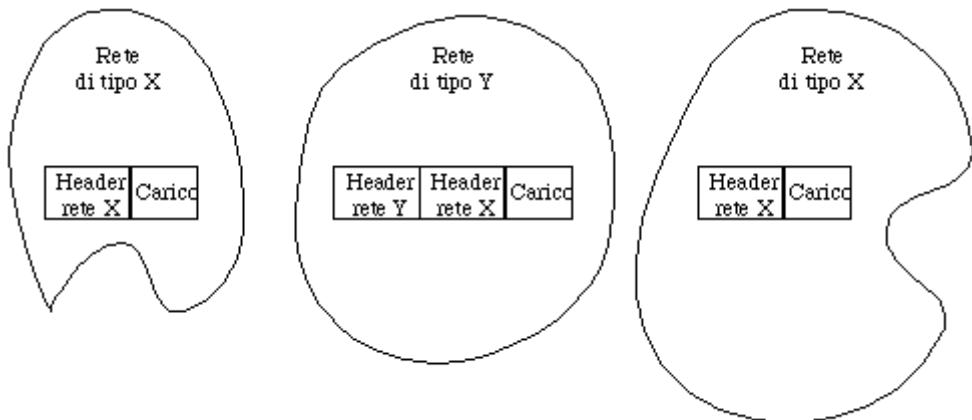
Le reti OSI possono parlare fra loro, e così quelle IP. Nella subnet costituita dai router multiprotocollo circolano pacchetti di entrambe le architetture, che vengono instradati secondo le regole di competenza dell'architettura di cui fanno parte.

### Tunneling

Un'altra tecnica che risolve un problema analogo è il **tunneling**, che si utilizza per mettere in comunicazione due reti uguali attraverso una rete diversa.



In questo caso la rete di tipo Y non è dotata di router multiprotocollo. Invece, un router designato in ciascuna delle due reti di tipo X è multiprotocollo e incapsula i pacchetti delle reti di tipo X dentro pacchetti di tipo Y, consegnandoli poi alla rete di tipo Y. Si noti che in questi pacchetti ci sono due buste di livello network:



### Frammentazione

Un diverso problema che talvolta si presenta è legato al fatto che in generale è possibile che i pacchetti in arrivo da una rete siano troppo grandi per transitare su un'altra.

In questo caso è necessario che il livello network della rete di origine (e di quella di destinazione) prevedano meccanismi di **spezzettamento** del pacchetto in **frammenti** prima di consegnarli alla rete di transito, e di **ricomposizione** dei frammenti appena essi giungono dalla rete di transito in quella di destinazione (come vedremo, il protocollo IP è fornito di questa funzionalità).

### Circuiti virtuali concatenati

Se tutte le reti interessate offrono servizi connessi nel livello network, è possibile costruire un circuito virtuale che si estende attraverso più reti eterogenee come **concatenazione di circuiti virtuali** che attraversano ciascuno una delle reti. Ai confini fra ogni rete ci sono dei router multiprotocollo che:

- creano la porzione di circuito virtuale che attraversa la rete di competenza, arrivando fino ad un router multiprotocollo situato all'altra estremità della rete;
- instradano successivamente i pacchetti lungo tale circuito virtuale.

## 10.10 Internetwork routing

In generale, in una internetwork le singole reti componenti sono entità autonome e vengono chiamate **AS (Autonomous System)**.

In questo caso, il routing complessivo è a due livelli:

- un primo livello è costituito dall'**Interior Gateway Protocol (IGP)**. Questo termine identifica l'algoritmo di routing usato da un AS al proprio interno. Naturalmente, diversi AS possono utilizzare diversi IGP. Come abbiamo già visto, un IGP può anche essere gerarchico, in particolare quando le dimensioni dell'AS sono considerevoli;
- un secondo livello è dato dall'**Exterior Gateway Protocol (EGP)**, che è l'algoritmo che si usa per gestire il routing fra diversi AS. Tipicamente, in questo scenario EGP è l'algoritmo di routing che riguarda i router multiprotocollo. L'aspetto più interessante relativo a EGP è che esso deve spesso tener conto di specifiche leggi nazionali (ad esempio, divieto di far transitare dati sul suolo di una nazione ostile), per cui le decisioni di routing devono adattarsi a tali direttive.

## LEZIONE 11 – IL LIVELLO NETWORK (PARTE 2)

### 11.1 Il livello network in Internet

Internet (una rete di reti) è una collezione di AS connessi gli uni con gli altri. Non esiste una struttura rigida, ma comunque si possono distinguere alcune componenti:

- **backbone principali** (linee ad alta velocità);
- reti regionali (USA);
- reti nazionali (Europa e resto del mondo);
- reti locali.

Ciò che tiene tutto insieme è il protocollo di livello network dell'architettura TCP/IP, e cioè **IP (Internet Protocol, RFC 791)**.

IP è un protocollo datagram, quindi non connesso e non affidabile (**best effort**), che opera come segue:

- riceve i dati dal livello transport e li incapsula in pacchetti di dimensione massima pari a 64 Kbyte (normalmente circa 1.500 byte);
- instrada i pacchetti sulla subnet, eventualmente frammentandoli lungo il viaggio;

a destinazione:

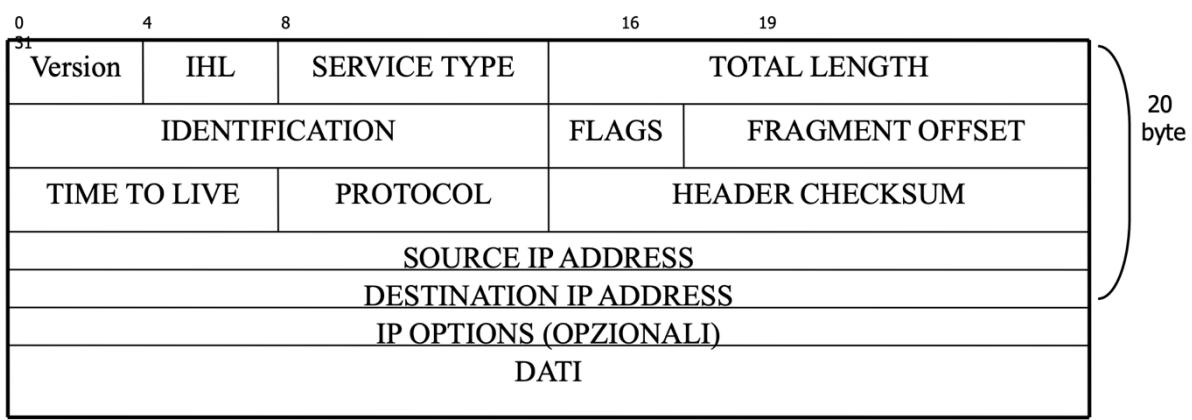
- riassembra (se necessario) i frammenti in pacchetti;
- estraе da questi i dati del livello transport;
- consegna al livello transport i dati nell'ordine in cui sono arrivati (che non è necessariamente quello in cui sono partiti)

I **datagrammi** contengono l'indirizzo del destinatario e ogni datagramma viene spedito/gestito indipendentemente. Essendo un protocollo non affidabile, la consegna non è garantita.

### 11.2 Header IP

Un pacchetto IP è costituito da un **header** e da una parte dati.

L'header ha una parte fissa di 20 byte e una parte, opzionale, di lunghezza variabile.



I campi dell'header hanno le seguenti funzioni:

Version	il numero di versione del protocollo (oggi è 4).
---------	--

<b>IHL</b>	lunghezza dell'header in parole di 32 bit (minimo 5, massimo 15).
<b>Type of service (3+3+2 bit)</b>	3 bit per la precedenza, 3 bit per il tipo di servizio in termini di affidabilità, velocità e ritardo (può influenzare le scelte di routing), 2 inutilizzati
<b>Total length</b>	lunghezza del pacchetto (inclusi dati), massimo 65.535 byte. ( $2^{16}-1$ )
<b>Identification</b>	identifica il datagram a cui appartiene il frammento. Tutti i frammenti di uno stesso pacchetto hanno lo stesso valore.
<b>FLAGS (1+1+1 bit)</b>	il primo è inutilizzato, il secondo ( <b>Don't Fragment</b> ) ordina ai router di non frammentare, il terzo indica se ci sono ( <b>More Fragments</b> ) o meno altri frammenti
<b>Fragment offset (13 bit)</b>	indice del frammento nel pacchetto. Massimo 8192 frammenti
<b>Time to live</b>	contatore (inizializzato a 255) utilizzato per limitare la vita dei pacchetti. Viene decrementato di uno a ogni hop (o ad ogni secondo). Quando arriva a zero, il pacchetto viene scartato e si invia una notifica al mittente
<b>Protocol</b>	codice del protocollo di livello transport a cui consegnare i dati (es. TCP o UDP)
<b>Header checksum</b>	checksum di controllo del solo header: <ul style="list-style-type: none"> <li>• si sommano (in complemento ad uno) le parole a 16 bit dello header, considerando il checksum a zero;</li> <li>• si complementa ad uno il risultato;</li> <li>• viene ricalcolato ad ogni hop (time to live cambia).</li> </ul>
<b>Source e destination address</b>	indirizzi di mittente e destinatario.
<b>Options</b>	opzioni, solo cinque sono definite oggi: <ul style="list-style-type: none"> <li>• <b>security</b>: quanto è segreto il pacchetto;</li> <li>• <b>strict source routing</b>: specifica il percorso da seguire, dalla sorgente alla destinazione con una sequenza di indirizzi IP;</li> <li>• <b>loose source routing</b>: fornisce una lista di router che non devono essere saltati;</li> <li>• <b>record route</b>: forza i router ad aggiungere i loro IP nel campo opzione del pacchetto (max 9 router)</li> <li>• <b>timestamp</b>: ogni router sul percorso oltre all'IP aggiunge anche un timestamp a 32 bit</li> </ul>

### 11.3 Indirizzi IP

L'indirizzo IP a 32 bit identifica univocamente un dispositivo (host o router) sulla rete.  
Un indirizzo IP è formato da 32 bit e codifica due cose:

- **network number**, cioè il numero assegnato alla rete IP (detta **network**) su cui si trova l'elaboratore; in questo contesto una network è caratterizzata dal fatto di essere costituita da un unico canale di comunicazione cui sono connessi tutti gli host della network stessa (e quindi, ad esempio, una LAN oppure una linea punto-punto fra due router);
- **host number**, cioè il numero assegnato all'elaboratore.

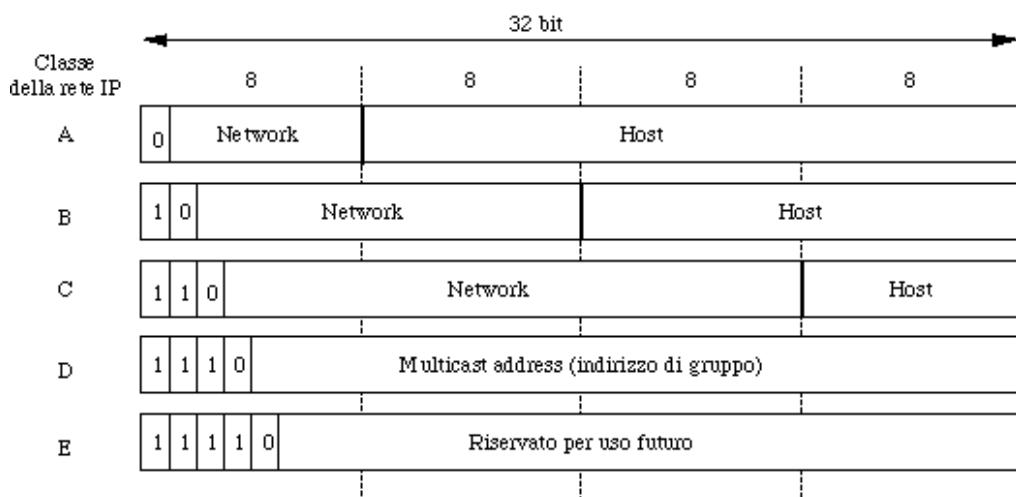
La combinazione è unica: non possono esistere nell'intera rete Internet due indirizzi IP uguali.

Si noti che solitamente si ritiene che ogni host sulla rete abbia un singolo indirizzo IP. In realtà gli indirizzi sono assegnati alle interfacce di rete, quindi:

- se un host ha un'unica interfaccia di rete (come è il caso di un PC in LAN) allora ha un unico indirizzo IP;
- se un host ha X interfacce di rete (come è il caso di un router connesso ad una LAN e ad X-1 linee punto-punto) ha X indirizzi.

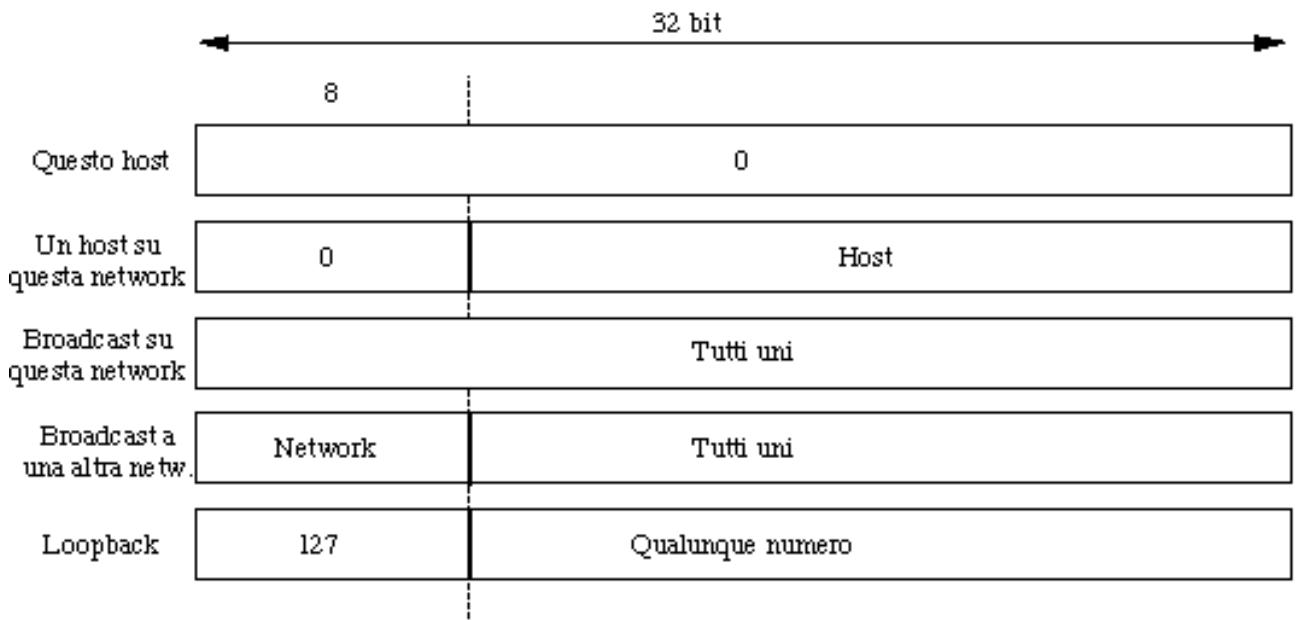
Gli indirizzi IP sono assegnati da autorità nazionali (**NIC, Network Information Center**) coordinate a livello mondiale.

I formati possibili degli indirizzi sono i seguenti:



<i>classe</i>	<i>bit nel prefisso</i>	<i>numero massimo di reti</i>	<i>bit nel suffisso</i>	<i>numero massimo di host per rete</i>
<i>A</i>	7	128	24	16777216
<i>B</i>	14	16384	16	65536
<i>C</i>	21	2097152	8	256

**Indirizzi speciali:**



## Indirizzi speciali

prefisso	suffisso	indirizzo	scopo
tutti 0	tutti 0	<b>questo computer</b>	utilizzato in fase di boot
rete	tutti 0	<b>la rete</b>	identificare una rete
rete	tutti 1	<b>broadcast diretto</b>	bcast su una rete remota
tutti 1	tutti 1	<b>broadcast locale</b>	broadcast sulla propria rete
127	qualsiasi	<b>loopback</b>	Test

Ricapitolando, poiché alcune configurazioni binarie per gli indirizzi sono impegnate per gli indirizzi speciali, possono esistere:

- 126 network di classe A, le quali possono contenere 16 milioni di host ciascuna;
- 16382 network di classe B, con circa 64.000 host ciascuna;
- 2 milioni di network di classe C, con 254 host ciascuna.

Gli indirizzi sono usualmente espressi nella **dotted decimal notation**, cioè i valori dei singoli byte sono espressi in decimale e sono separati da un punto, come nell'indirizzo:

141.192.140.37

In tale notazione, è possibile rappresentare separatamente il network number e l'host number. Per distinguerli, il primo è seguito da un punto. Ad esempio, nel caso dell'indirizzo IP precedente (che è relativo ad una network di tipo B), si ha:

- il network number è 141.192. (notare il punto finale);
- l'host number è 140.37 (non c'è il punto finale).

### Indirizzi privati:

Gli indirizzi privati sono indirizzi riservati per reti (private) non connesse ad Internet

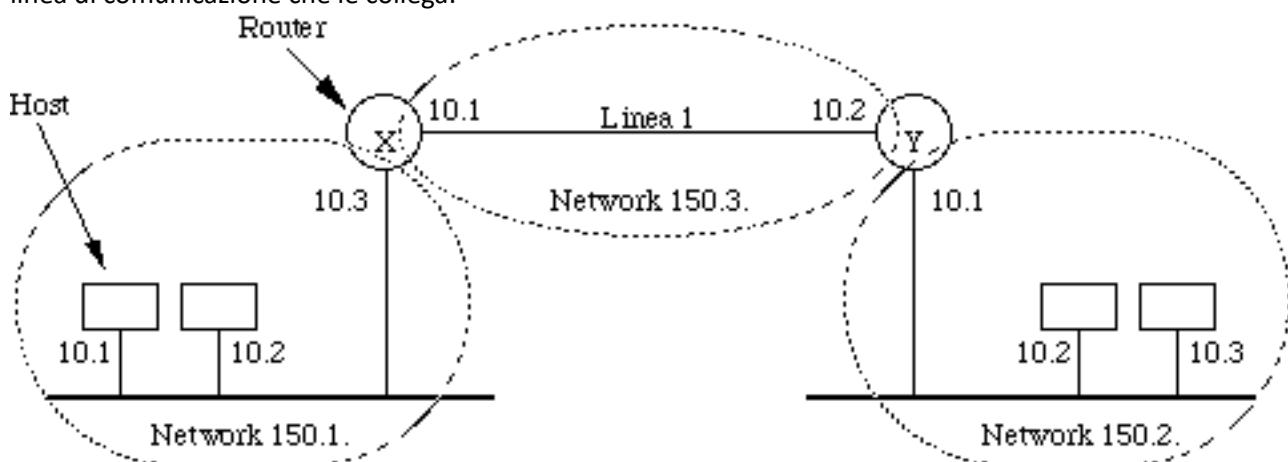
classe	rete	numero reti
A	<b>10.0.0</b>	<b>1</b>
B	Da <b>172.16</b> a <b>172.31</b>	<b>16</b>
C	Da <b>192.168.0</b> a <b>192.168.255</b>	<b>256</b>

La suddivisione in Classi, e la crescita esplosiva di Internet, ha causato un rapido consumo degli indirizzi IPv4.

Difatti, sono stati introdotti gli indirizzi IPv6 (indirizzi non più a 32 bit, ma a 128 bit)

### 11.4 Routing IP

Il collegamento fra due router non avviene direttamente, ma attraverso una network (in realtà di solito una **subnet**, come vedremo poi) che li collega, e che di fatto è costituita dalle due interfacce di rete e dalla linea di comunicazione che le collega:



Ogni router possiede una tabella (costruita e mantenuta dinamicamente dall'algoritmo di routing in esercizio) che contiene elementi del tipo:

1. **(this network number, host number)** per ciascun host della network a cui il router è direttamente collegato;
2. **(network number, 0)** per ciascuna network lontana di cui il router conosce l'esistenza.

Associate a tali elementi ci sono le informazioni sull'interfaccia di rete da usare per instradare il pacchetto e, nel caso delle linee punto punto, l'indirizzo del router che si trova dall'altra parte.

Inoltre, viene mantenuto l'indirizzo di un **default router** (e la corrispondente linea seriale da usare per raggiungerlo) a cui inviare tutti i pacchetti destinati a network sconosciute.

Nell'esempio della figura precedente, il router X si comporta come segue:

- se arriva da fuori un pacchetto destinato a 150.1.10.1 il router, attraverso un elemento di tipo 1, scopre che deve inviarlo sulla LAN;
- se arriva dalla LAN un pacchetto per 150.2.10.3 e il router ha un elemento di tipo 2 quale (150.2., 0), allora invia il pacchetto (al router Y) sulla linea 1.

## 11.5 Subnet e Subnet Mask

Al fine di economizzare nel numero di network da usare (utilizzando al meglio quelle che possono contenere migliaia o milioni di host) una network può essere divisa in varie **subnet**, ciascuna contenente i suoi host.

Questo è un fatto privato della network che viene suddivisa, e non ha bisogno di essere comunicato all'esterno.

Il meccanismo usato è di considerare, nell'indirizzo IP originario, l'host number come una coppia di valori: un **subnet number** e l'host number. Ad esempio se ho un indirizzo di rete del tipo 200.100.50.0\24 (classe C) e volessi veicolare 8 sottoreti, devo "prendere" tanti bit dall'host number in modo da veicolare 8 sottoreti, ovvero 3 bit. Di conseguenza, l'host number avrà i primi 3 bit che identificano la sottorete e i restanti 5 che identificano l'host di quella sottorete.

Tutto ciò avviene sulla base di una maschera di bit, detta **subnet mask**, che deve essere unica per tutta la network e che delimita la parte di host number che viene usata come subnet number.

La subnet mask è una maschera che ha i bit ad 1 in corrispondenza del **net\_id** e 0 nell'**host\_id**.

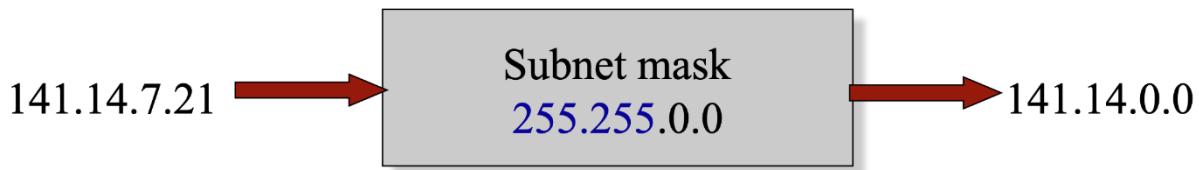
A seconda dell'ampiezza del campo dedicato alla subnet, si possono ottenere molte subnet contenenti ciascuna pochi host oppure poche subnet che però potranno contenere molti host.

Nei router si aggiungono elementi del tipo:

- **(this network number, this subnet number, host number)** per gli host delle subnet a cui il router è collegato direttamente;
- **(this network number, subnet number, 0)** per le altre subnet a cui il router non è collegato direttamente;

con le relative informazioni sulle interfacce di rete da usare.

**ES: 141.14.0.0 (classe B) → mask (default) = 255.255.0.0**



**11111111.11111111.00000000.00000000** mask  
**10001101.00001110.00000111.00010101** IP host

---

**10001101.00001110.00000000.00000000** IP sottorete

Fare l'AND bit a bit dell'indirizzo IP con la Maschera, corrisponde ad ottenere l'indirizzo di rete della sottorete.

La subnet mask, quindi, permette di verificare se l'IP del destinatario appartiene o meno alla rete del mittente.

Se l'host del destinatario non è nella stessa sottorete, il pacchetto deve essere instradato verso il router (gateway).

## 11.6 Protocolli di Controllo

Assieme a IP esistono diversi protocolli per il controllo del funzionamento della subnet.

### ICMP (Internet Control Message Protocol, RFC 792)

ICMP è un protocollo del livello **network** per la segnalazione di eventi relativi allo stato della rete. Viene utilizzato, per esempio, quando un host o un router devono informare la sorgente di un datagram circa eventi relativi al trasferimento del datagram. ICMP utilizza IP come se fosse un protocollo di livello trasporto, incapsulando il **messaggio ICMP**, che è individuato da un **codice numerico** composto da un **tipo** e da un eventuale **sottotipo** (code), in un datagram IP (costituisce il campo dati). ICMP è comunque parte integrante di IP e deve essere implementato da ogni modulo IP ed **opera con messaggi di richiesta e risposta**.

Per comprendere meglio il funzionamento del protocollo, conviene dare un'occhiata alla struttura dell'ICMP o meglio del suo **header**. Esso si collega direttamente con l'header dell'IP, nel quale viene indicato dal numero 1 (ICMPv4 se viene usato lo standard IPv4) o 58 (ICMPv6 se viene usato lo standard IPv6) del campo *Protocollo*. L'header dell'Internet Control Message Protocol non è molto ampio e ha la forma seguente:

	<b>Bit 0–7</b>	<b>Bit 8–15</b>	<b>Bit 16–23</b>	<b>Bit 24–31</b>
<b>0</b>	Tipo	Codice		Checksum
<b>32</b>	Dati dell'header			

Il primo campo di 8 bit (*Tipo: messaggio di errore o richiesta di informazioni*) indica la categoria a cui appartiene il messaggio ICMP. Il campo seguente (*Codice*), anch'esso di 8 bit, fornisce una descrizione ulteriore del messaggio. Facendo un esempio, un messaggio ICMP di tipo 3 specifica che **la destinazione del pacchetto di dati non è raggiungibile**, mentre il codice indica se a non essere disponibile sia la rete di destinazione (0), l'host desiderato (1) o la porta (3). L'ultimo campo di 16 bit (*Checksum*) ha la funzione di verificare la correttezza dei dati forniti. La sua struttura è analoga a quella dei checksum degli altri protocolli standard (IP, UDP, TCP).

Per ultimi ci sono i dati ICMP, che sono **creati e strutturati diversamente** in base al tipo di messaggio e all'istanza in funzione. In questo campo vengono spesso menzionati di nuovo l'header dell'IP e i primi 64 bit del pacchetto di dati responsabile del messaggio di errore o dello stato della query. Nel cosiddetto **ICMP Tunneling** questo campo viene utilizzato in modo improprio per inviare dati utente al di là del firewall o per stabilire un canale di comunicazione tra due computer.

### TIPI ICMP

Tipo	Nome	Utilizzo
0	Echo reply	Risposta da un host attivo sulla rete
3	Destination unreachable	Diagnostica sui problemi inerenti la trasmissione di un datagram; prevede diversi sottotipi
4	Source quench	Segnala che un router intermedio non ha spazio per mettere in coda il datagram
5	Redirect	Emesso da un router intermedio, segnala il router cui inviare i datagram seguenti
8	Echo request	Verifica della presenza di un host sulla rete
11	Time exceeded	Segnala che il campo TTL del datagram è esaurito senza che sia stata raggiunta la destinazione
12	Parameter problem	Indica che si è verificato un problema nell'elaborazione dell'header IP
13	Timestamp request	Utilizzato per il debugging e la misura di prestazioni ), richiede un timestamp
14	Timestamp reply	Il messaggio di risposta ad una richiesta di timestamp
...	.....	.....

### Interrogazioni ICMP

Il protocollo ICMP è di grande importanza per la comunicazione nelle reti IP, nelle quali viene utilizzato in particolare dai router, ma non solo: server e client fanno uso dei messaggi legati al protocollo Internet e ricevono in questo modo **importanti informazioni di rete**.

### Comando ping

Una situazione tipica di impiego è il cosiddetto **ping test**, che può essere eseguito utilizzando l'omonimo comando disponibile in tutti i sistemi operativi. L'utilizzo di questo strumento è il modo più semplice per controllare la disponibilità di un host nella rete. A questo scopo, ping invia un pacchetto ICMP di tipo **Echo request**, al quale il destinatario risponde con un messaggio consistente in un pacchetto di tipo **Echo reply**.

Nel caso in cui il sistema a cui è destinato il ping non sia raggiungibile, l'ultimo nodo di rete disponibile invia un pacchetto di risposta di tipo **Destination unreachable** (cioè destinazione non raggiungibile).

### Comando traceroute/tracert

Il comando traceroute è un altro esempio d'uso del protocollo ICMP. Permette di tracciare il percorso compiuto dai pacchetti IP da una sorgente ad una destinazione. In pratica elenca i router attraversati nel percorso tra la sorgente e la destinazione. Per determinare gli indirizzi e i nomi dei router il programma traceroute invia messaggi ICMP inseriti in pacchetti IP con TTL (Time To Live) via via crescente.

Il primo pacchetto avrà TTL 1 e si fermerà al primo router. Il secondo pacchetto avrà TTL 2 e si fermerà al secondo router e così via. Riceve i messaggi di errore (TTL scaduto, tipo 11, codice 0) dai router e costruisce il percorso tra sorgente e destinazione. Il timer associato ad ogni pacchetto IP permette di calcolare il tempo trascorso per il raggiungimento dell'ennesimo router.

#### Esempio

Immaginiamo di voler tracciare il percorso verso il server web di Google. Utilizzeremo il comando **traceroute** da una finestra di terminale. Supponiamo che l'indirizzo IP del server di Google sia 172.217.17.142.

**traceroute** 172.217.17.142

1. **Invio dei pacchetti ICMP: traceroute** invia pacchetti ICMP Echo Request (ping) a 172.217.17.142, con un limite di tempo iniziale (time-to-live, TTL) di 1.
2. **Router 1:** Il primo router lungo il percorso riceve il pacchetto, ma il TTL è insufficiente, quindi il router restituisce un messaggio ICMP "Time Exceeded." Questo ci dice che il pacchetto è passato attraverso il primo router.
3. **Incremento del TTL:** traceroute invia un secondo pacchetto con TTL incrementato a 2. Ora, il secondo router lungo il percorso riceve il pacchetto e restituisce anch'esso un messaggio "Time Exceeded."
4. **Ripetizione del processo:** Questo processo viene ripetuto con TTL crescenti fino a raggiungere il server di destinazione.
5. **Destinazione raggiunta:** Infine, quando il pacchetto raggiunge il server di Google, riceveremo risposte ICMP Echo Reply.

I risultati mostreranno l'indirizzo IP di ciascun router intermedio e il tempo impiegato per raggiungerlo. Questo ci dà un'idea del percorso che i pacchetti seguono attraverso la rete per raggiungere la destinazione desiderata.

### ARP (Address Resolution Protocol, RFC 826)

Il protocollo **ARP** serve per derivare, dall'indirizzo IP dell'host di destinazione, l'indirizzo di livello data link (fisico) necessario per inviare il frame che incapsulerà il pacchetto destinato all'host di cui all'indirizzo IP. Esso opera appoggiandosi direttamente sul livello data link e non su IP:

- viene inviata a tutte le stazioni della LAN, in data link broadcast, una richiesta del tipo: "chi ha l'indirizzo IP uguale a 151.100.17.102 ?"
- solo l'host che ha quell'indirizzo IP risponde, inserendo nella risposta il proprio indirizzo data link (MAC);
- per evitare la risoluzione continua IP → MAC, quando riceve la risposta, l'host la mantiene in memoria per circa 15 minuti (cache ARP).

Se l'indirizzo IP è relativo ad un'altra network:

- la soluzione più semplice è mandare il pacchetto ARP come prima, configurando però il router in modo che risponda alle richieste ARP relative ad altre reti fornendo il proprio indirizzo ethernet (**proxy ARP**); il router farà poi da tramite nella conversazione IP fra il mittente e il destinatario, inviando di volta in volta all'uno i pacchetti IP che gli giungono dall'altro;
- alternativamente, si configura l'host impostando al suo interno l'indirizzo ethernet di default (quello del router) a cui mandare i pacchetti IP per le altre reti; anche in questo caso il router deve fare da tramite nella conversazione IP fra il mittente e il destinatario.

### DHCP (Dynamic Host Configuration Protocol)

DHCP viene utilizzato per assegnare automaticamente gli indirizzi IP alle macchine che lo richiedono, attraverso un server di configurazione. Questo server potrebbe essere situato sulla stessa rete o su un'altra rete.

Quando una macchina viene attivata, essa contiene solo il proprio indirizzo MAC incluso nel NIC, invia quindi un broadcast di richiesta di IP, attraverso un pacchetto **DHCP DISCOVER**, quando questo pacchetto arriva al server DHCP, questo alloca un indirizzo IP libero e invia all'host un pacchetto **DHCP OFFER** con tale indirizzo. Per evitare che gli indirizzi rimangano bloccati per sempre anche quando una macchina non li utilizza più, si usa una tecnica chiamata **leasing**, ovvero appena prima che questo *lease* scada, l'host deve inviare un pacchetto al server DHCP chiedendo un rinnovo. Se il pacchetto non viene inviato o la richiesta è rifiutata l'indirizzo IP viene ritirato

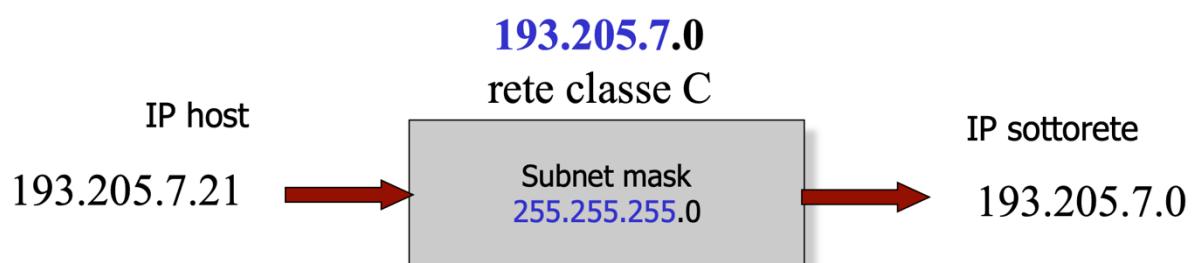
### RARP (Reverse Address Resolution Protocol, RFC 903)

Il protocollo **RARP** risolve il problema inverso, cioè consente di trovare quale indirizzo IP corrisponda a un determinato indirizzo data link.

Esso è utile nel caso di stazioni senza disco, che al momento dell'avvio caricano l'immagine del codice binario del sistema operativo da un server.

Il vantaggio che si ottiene è che una sola immagine binaria va bene per tutte le stazioni: ogni stazione personalizza poi l'immagine binaria con la determinazione del proprio indirizzo IP mediante una richiesta RARP, nella quale invia il proprio indirizzo data link (che è cablato nell'interfaccia di rete).

## 11.7 Esercizio Subnetting



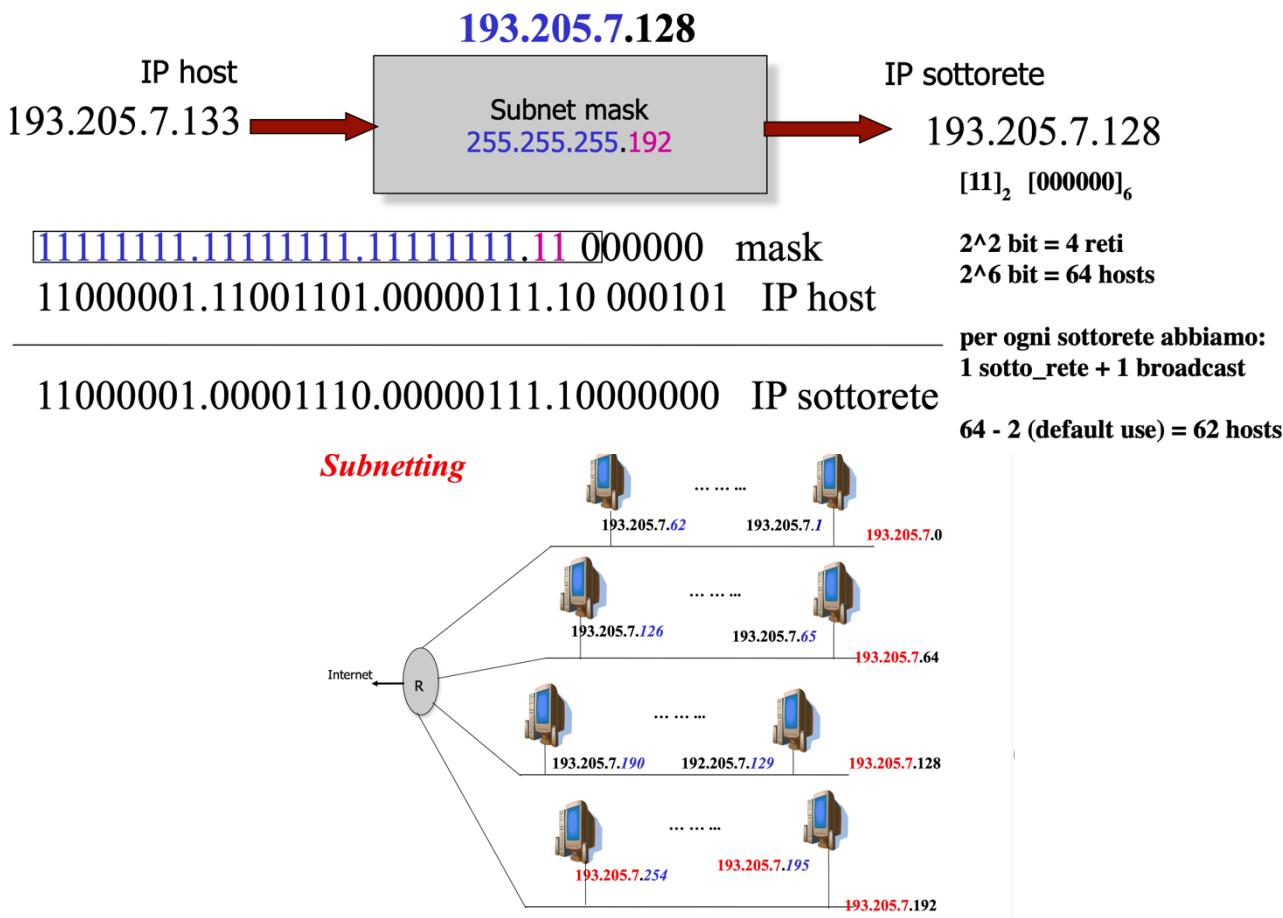
Supponiamo di voler creare, a partire dall'indirizzo di rete 193.205.7.0, 4 sottoreti.

Servono esattamente 2 bit, di conseguenza la **subnet mask sarà di 26 bit ad 1**

**11111111.11111111.11111111.110000** → rimangono 6 bit ( $2^6 = 64$  host)

Quindi, attraverso questa subnet mask, saranno possibili 4 sottoreti, ognuna di 64 host

Di questi 64, per ogni sottorete, vanno tolti i due indirizzi speciali che identificano l'indirizzo di sottorete e l'indirizzo di broadcast. Per tale motivo, gli host veicolabili saranno 62 (per ogni sottorete).



Per descrivere facilmente una subnet è possibile agire sulla scrittura dell'indirizzo ip.

Usiamo il numero di bit della maschera di rete per determinare la subnet:

193.205.7.0 ha una **4 sottoreti** con netmask 11111111.11111111.11111111.110000

Possiamo riscrivere gli indirizzi come:

193.205.7.0/26,  
193.205.7.64/26,  
193.205.7.127/26,  
192.205.7.192/26

### Variable Length Subnet Mask (VLSM)

Il subnetting **VLSM**, acronimo di **Variable Length Subnet Masking**, è una tecnica utilizzata nella progettazione delle reti per creare subnet di dimensioni variabili all'interno di una rete più ampia. Questa tecnica consente di ottimizzare l'utilizzo degli indirizzi IP, soprattutto in scenari in cui sono disponibili poche risorse di indirizzamento.

In sostanza, il subnetting VLSM permette di suddividere una rete in sottoreti di dimensioni diverse, utilizzando maschere di sottorete diverse per ciascuna subnet. Ciò consente di allocare indirizzi IP in modo più efficiente, in quanto consente di assegnare blocchi di indirizzi di dimensioni appropriate alle diverse parti della rete in base alle loro esigenze specifiche.

E' possibile dividere una subnet in più subnet. Creiamo diverse maschere di rete per ogni subnet.

**Esempio:**

Supponiamo di avere un router che collega 3 LAN di cui:

**LAN B** necessita 50 host;

**LAN A** necessita 100 host;

**LAN C** necessita 10 host;

Supponiamo di avere un unico indirizzo di rete di classe C: **193.100.1.0/24**

Con il subnetting classico a lunghezza fissa, per indirizzare 3 sottoreti, dobbiamo utilizzare 2 bit della parte host dell'indirizzo poiché con 2 bit siamo capaci di indirizzare 4 sottoreti.

Della parte host rimarrebbero 6 bit, per cui siamo capaci di indirizzare  $2^6 - 2$  host, ovvero 62 host. Di conseguenza non potremmo indirizzare correttamente la **LAN A**.

In aiuto viene il VLSM che andrà anche a minimizzare lo spreco degli indirizzi IP.

Ricapitolando:

**LAN A** → 100 **host** → necessitiamo di 7 bit

**LAN B** → 50 host → necessitiamo di 6 bit

**LAN C** → 10 host → necessitiamo di 4 bit

Si prende l'indirizzo di rete riportando la parte host in binario:

**193.100.1.00000000**

Per effettuare il subnetting VLSM della LAN A, abbiamo detto che occorrono 7 bit per la parte **host**; di conseguenza rimane un solo bit da dedicare al **network (subnet)**. Con un bit dedicato al network, possiamo indirizzare 2 subnet, ognuna di  $2^7 - 2$  host.

**193.100.1.0000000 /25**

**193.100.1.0111111 /25**

**193.100.1.1000000 /25**

**193.100.1.1111111 /25**

Quindi, per la **LAN A**, abbiamo (scegliamo il range con gli indirizzi più piccoli):

**Subnet mask: 255.255.255.128**

**Indirizzo di network: 193.100.1.0 /25**

**Indirizzo di broadcast: 193.100.1.127 /25**

Per effettuare il subnetting VLSM della LAN B, utilizziamo la sottorete ricavata dal subnetting della LAN A che era in più, andando a modificare la lunghezza della subnet mask.

Per la LAN B, servono 6 bit per gli **host**, di conseguenza rimangono 2 bit da dedicare al **network, di cui il 1 bit rimane fisso** perché “preso” dall'altra subnet ricavata precedentemente (è come se la subnet mask precedente diventasse la nuova maschera di default e quindi il bit non cambia), con il quale riusciamo a ricavare 2 sottoreti.

**193.100.1.1000000 /26**

**193.100.1.1011111 /26**

193.100.1.**11000000** /26

193.100.1.**11111111** /26

Quindi, per la **LAN B**, abbiamo (scegliamo il range con gli indirizzi più piccoli):

**Subnet mask: 255.255.255.192**

**Indirizzo di network: 193.100.1.128 /26**

**Indirizzo di broadcast: 193.100.1.191 /26**

Per la LAN C il meccanismo è il medesimo. Servono esattamente 4 bit da dedicare agli **host**. Di conseguenza, rimangono 4 bit da dedicare al **network**, di cui 2 rimangono fissi perché presi dall'altra subnet.

193.100.1.**11000000** /28

193.100.1.**11001111** /28

Potremmo calcolare altre 3 subnet, tuttavia ci fermiamo alla prima poiché non ci sono altre LAN da indirizzare.

Quindi, per la **LAN C**, abbiamo (scegliamo il range con gli indirizzi più piccoli):

**Subnet mask: 255.255.255.240**

**Indirizzo di network: 193.100.1.192 /28**

**Indirizzo di broadcast: 193.100.1.223 /28**

## LEZIONE 12 – IL LIVELLO TRASPORTO TCP/UDP

L'utente, attraverso chiamate a funzioni, richiede il servizio di trasporto **con connessione o senza connessione**. Il trasporto con connessione fornisce un **canale affidabile** su cui scrivere e leggere dati. Ad esempio nel modello client/server uno scenario può essere:

Il **server** attiva un processo che rimane in attesa di una richiesta di connessione (**LISTEN**)

Il **client** invia una richiesta di connessione (**CONNECT**).

Aperta la connessione, Server e Client si scambiano dati con le primitive **SEND** e **RECEIVE**.

Si chiude la connessione con la procedura **DISCONNECT**.

I protocolli del livello di Trasporto devono:

- Poter definire la **modalità di indirizzamento** a livello trasporto (su uno stesso host possono essere disponibili più connessioni)
- Gestire il controllo degli errori, i numeri di sequenza e il controllo di flusso tra end system collegati attraverso una rete.

I problemi del livello trasporto sono simili a quelli del livello data link, ma il canale fisico è in questo caso l'intera sottorete di comunicazione.

Lungo la sottorete di comunicazione i pacchetti possono essere trattenuti nei router e consegnati dopo diversi secondi, possono seguire strade diverse ed arrivare in ordine diverso da quello di trasmissione).

**TCP e UDP** sono due Protocolli di trasporto definiti su IP

**Transmission Control Protocol (TCP)**: protocollo di trasporto orientato alla connessione.

E' definito in RFC 793, RFC 1122 e RFC 1323 ed è progettato per fornire un **flusso affidabile end-to-end** su una internet inaffidabile

**User Data Protocol (UDP)**: protocollo senza connessione.

E' descritto in RFC 768 e permette di inviare datagram IP **senza stabilire una connessione**.

### 12.1 Protocollo TCP

**TCP** o Transmission Control Protocol è un protocollo che ha le seguenti caratteristiche:

- **Servizio connesso**: Il protocollo TCP crea un canale logico (connessione logica) tra due host. Ciò implica che il TCP lavora in unicast (1:1, un host deve potersi collegare con un altro host a livello logico). La connessione viene instaurata in prima istanza attraverso una tecnica chiamata Three-Way-Handshake e rilasciano la connessione attraverso una tecnica chiamata Four-Way-End-Connection.
- **Controllo dell'errore**: Ogni singolo segmento veicolato dal TCP è sottoposto al controllo dell'errore (impostato nell'header TCP).
- **Affidabile**: Assegna un numero di sequenza ad ogni byte trasmesso, attendendo una conferma di avvenuta ricezione (ACK). Il TCP ricevente, quando restituisce un ACK al mittente, invia anche il **numero massimo di sequenza** che può ricevere nella prossima trasmissione.
- **Ordine dei segmenti**: Il protocollo TCP è in grado di riordinare in automatico i segmenti TCP. Di conseguenza i segmenti arrivano sempre ordinati.
- **Controllo del flusso**: L'host mittente e destinatario si mettono d'accordo per far sì che un host non inondi di segmenti l'altro host che non è in grado di gestire.

#### In trasmissione:

- Riceve un flusso dati dall'applicazione
- Organizza i dati in unità lunghe massimo 64KB
- Spedisce le unità come datagram IP

#### In ricezione

- Riceve i datagram IP
- Ricostruisce e consegna all'applicazione la sequenza corretta del flusso di byte originale

Poiché è un protocollo affidabile, il TCP deve ritrasmettere i datagram non ricevuti e riordinare quelli arrivati in ordine sbagliato.

**Le comunicazioni TCP richiedono una esplicita connessione fra un socket della macchina mittente ed un socket della macchina ricevente.**

**Le connessioni sono identificate con gli identificatori dei socket dei due lati (socket1, socket2).**

Una socket è caratterizzata dalla coppia INDIRIZZO DELL'HOST + numero di porta (a 16 bit)

**Una volta attivato, un socket è utilizzato come un file.**

**Sono disponibili primitive nei linguaggi di programmazione per aprire e usare socket (C, Java...)**

Le porte attive definiscono i servizi TCP disponibili.

Per connettersi ad un servizio specifico occorre conoscere il numero di porta su cui gira il processo (server) attivato per quel servizio. Le porte inferiori a 256 sono dette **porte ben note (well-known ports)** e sono normalmente utilizzate per servizi standard. In Unix la lista dei servizi e delle porte è nel file **/etc/services**.

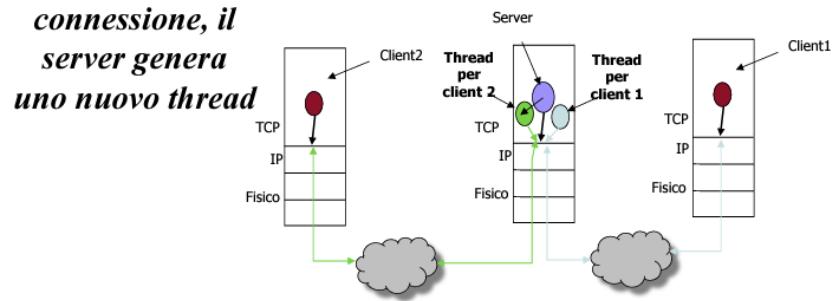
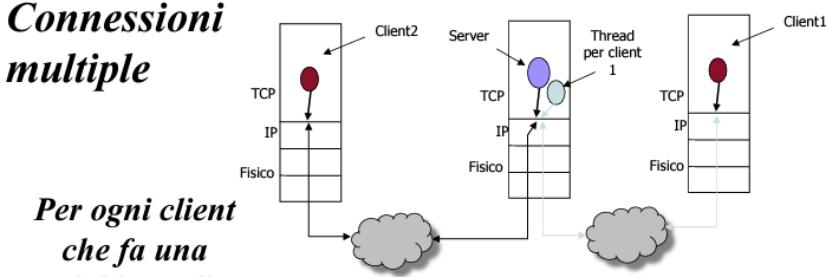
In genere, il client, per le connessioni, utilizza dei **numeri di porta effimeri**, ovvero dei numeri di porta elevati e scelti in modo tale da essere unici sull'host.

Ad esempio la coppia di porte assegnate ad una connessione http potrebbe essere la seguente:

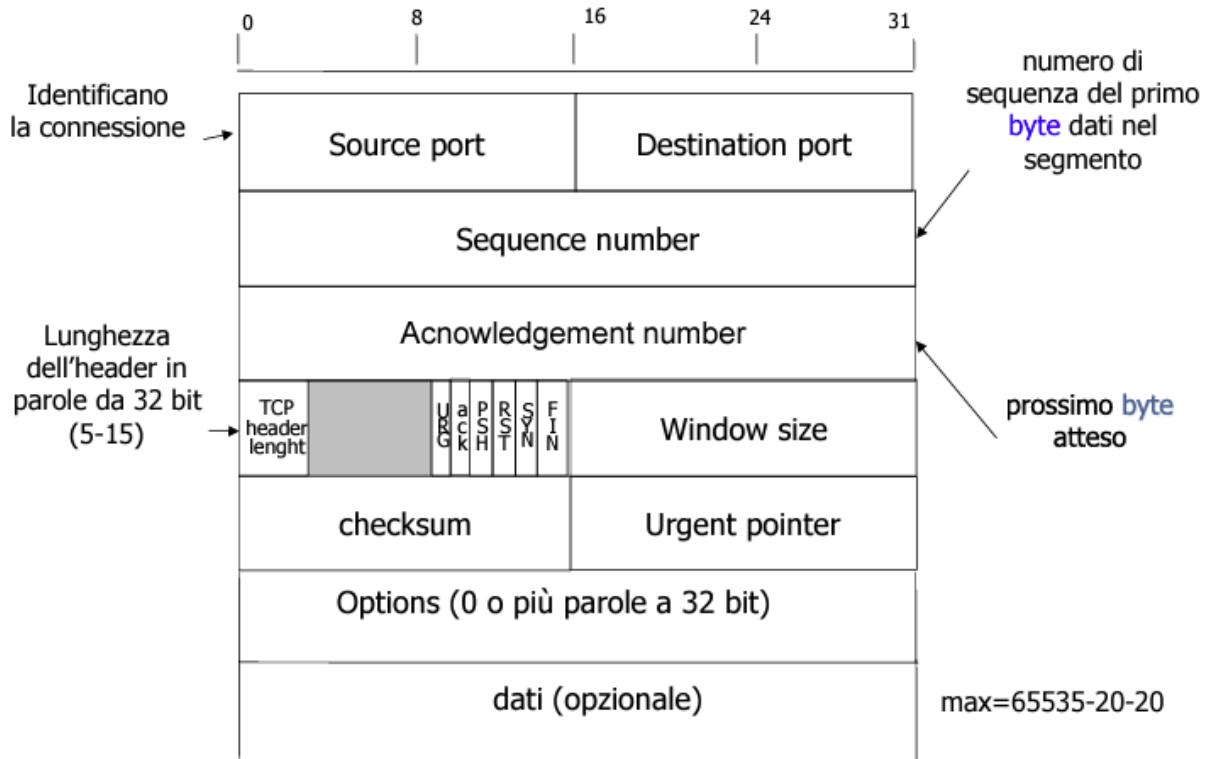
```
server port 80  
client port 23443
```

Le connessioni sono punto-punto e full-duplex.

## Connessioni multiple



Ogni segmento TCP ha un header di 20 byte più eventuali parti opzionali seguito da 0 o più byte di dati



**Nel segmento TCP sono presenti 6 bit di flag**

### **URG**

Se URG assume valore 1, *l'Urgent Pointer indica la posizione, a partire dal numero di sequenza attuale, di dati urgenti* (es. pressione di CTRL-C per interrompere il programma remoto)

### **ACK**

*Indica se il campo Acknowledgement number è valido*

### **PSH**

*Indica dati di tipo PUSH ovvero si richiede di consegnare subito i dati senza bufferizzarli*

### **○ RST**

*Richiesta di reinizializzazione di una connessione diventata instabile.* Viene anche usato per rifiutare un segmento non valido o l'apertura di una connessione

### **○ SYN**

*Viene utilizzato per creare connessioni.* La richiesta di connessione è caratterizzata da SYN=1 e ACK=0 (**CONNECTION REQUEST**). La risposta di connessione contiene un ack e quindi ha SYN=1 e ACK=1 (**CONNECTION ACCEPTED**)

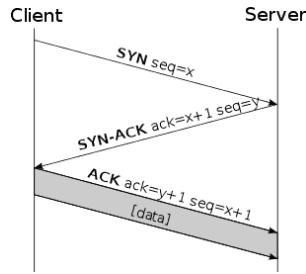
### **○ FIN**

*Viene utilizzato per chiudere una connessione* (il mittente non ha altri dati da spedire)

Per l'apertura di una connessione si utilizza un protocollo 3-way handshake:

#### **Funzionamento**

- Un client invia un pacchetto dati SYN su una rete IP a un server sulla stessa rete o a una rete esterna. L'obiettivo di questo pacchetto è di chiedere se il server è disponibile per nuove connessioni.
- Il server di destinazione deve disporre di porte aperte in grado di accettare e avviare nuove connessioni. Quando il server riceve il pacchetto SYN dal nodo client, risponde e restituisce una ricevuta di conferma, il pacchetto ACK o SYN/ACK.
- Il client riceve il SYN/ACK dal server e risponde con un pacchetto ACK.
- Al termine di questo processo, viene creata la connessione e l'host e il server sono in grado di comunicare.



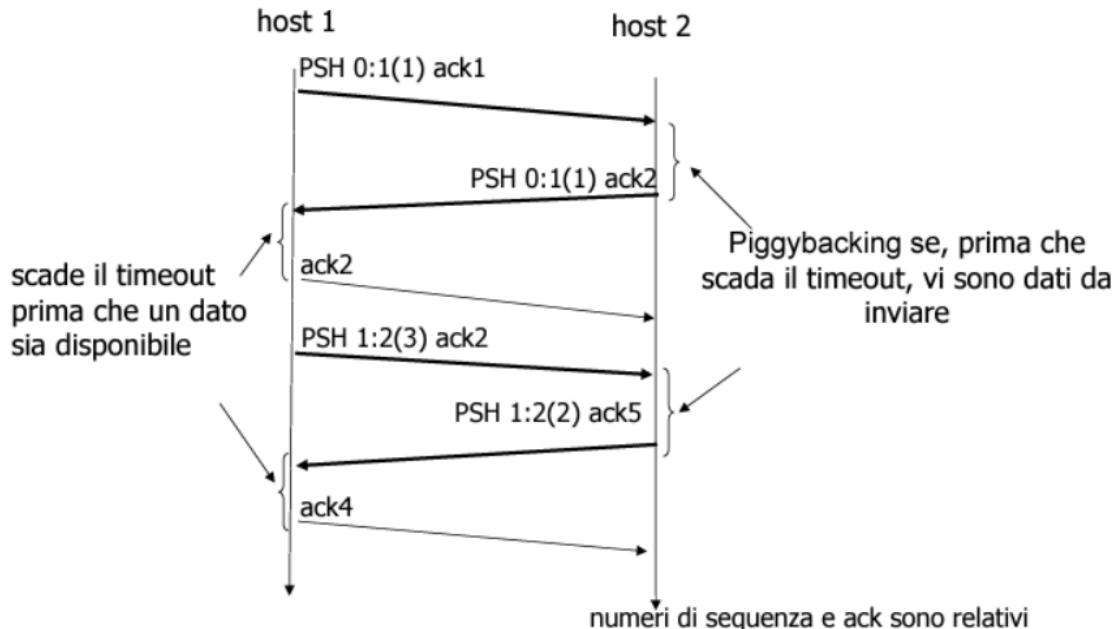
1. **A invia un segmento SYN a B** – il flag SYN è impostato a 1 e il campo Sequence number contiene il valore x che specifica l'Initial Sequence Number di A;
2. **B invia un segmento SYN/ACK ad A** – i flag SYN e ACK sono impostati a 1, il campo Sequence number contiene il valore y che specifica l'Initial Sequence Number di B e il campo Acknowledgment number contiene il valore x+1 confermando la ricezione del ISN di A;
3. **A invia un segmento ACK a B** – il flag ACK è impostato a 1 e il campo Acknowledgment number contiene il valore y+1 confermando la ricezione del ISN di B.

Il terzo segmento non sarebbe, idealmente, necessario per l'apertura della connessione in quanto già dopo la ricezione da parte di A del secondo segmento, entrambi gli host hanno espresso la loro disponibilità all'apertura della connessione. Tuttavia, esso risulta necessario al fine di permettere anche all'host B una stima del timeout iniziale, come tempo intercorso tra l'invio di un segmento e la ricezione del corrispondente ACK.

Se il TCP riscontra l'assenza del processo che deve essere in attesa sulla porta destinazione, manda un segmento di rifiuto della connessione (**RST**)

## Ack ritardati

- Normalmente il TCP non invia un ack istantaneamente ma ritarda l'invio sperando di avere dati da spedire (**piggyback**)
- Molte implementazioni usano un ritardo di 200ms



### Politiche di trasmissione:

L'idea di fondo è la seguente: la dimensione delle finestre scorrevoli non è strettamente legata agli ack (come invece di solito avviene), ma viene continuamente adattata mediante un dialogo fra destinazione e sorgente.

In particolare, quando la destinazione invia un ack di conferma, dice anche quanti ulteriori byte possono essere spediti.

Nell'esempio che segue, le peer entity si sono preventivamente accordate su un buffer di 4K a destinazione.

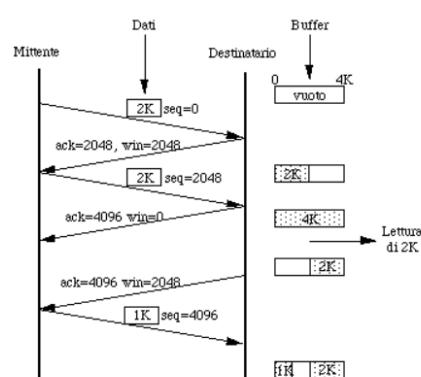


Figura 6-16: Esempio di controllo del flusso TCP

Anche se riceve win=0, il mittente può comunque inviare:

- dati urgenti;
- richieste di reinvio dell'ultimo ack spedito (per evitare il deadlock se esso si è perso).

### Gestione degli errori:

Il mittente ha un timeout di attesa dell'ack per i pacchetti spediti

Se il timeout scade il pacchetto viene ritrasmesso.

Permette di gestire la perdita di pacchetti (non arrivati o arrivati con errori)

TCP utilizza un timeout di attesa dell'ack dopo di che provvede alla ritrasmessione dei dati.

Il problema è determinare il valore del timeout migliore (i ritardi possono essere molto variabili nel tempo sulla rete), poiché, se il timeout è troppo piccolo si fanno ritrasmissioni inutili, mentre se il timeout è troppo elevato si avranno ritardi di trasmissione.

Vengono utilizzati algoritmi di stima del migliore timeout basato sulla misura del **Round-Trip-Time (RTT)**

### Controllo della congestione:

Il TCP adatta la velocità di trasmissione alla capacità della rete.

I pacchetti possono essere rifiutati dai router se si riempiono i buffer di trasmissione/ricezione.

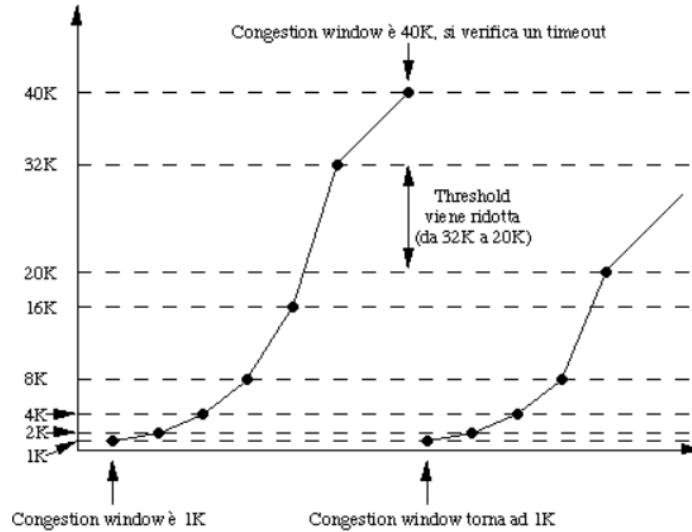
Per i pacchetti distrutti, non si riceve l'ack e quindi vengono ritrasmessi. La ritrasmessione può essere un fattore potenziale per incrementare ulteriormente la congestione.

Si utilizza una finestra di congestione per il mittente (**congestion window**) che è la minima fra quella di congestione (definita dalla rete) e quella di ricezione (definita dallo stato del buffer del ricevente)

La congestion window viene gestita in questo modo:

- il valore iniziale è pari alla dimensione del massimo segmento usato nella connessione;
- ogni volta che un ack torna indietro in tempo la finestra si raddoppia, fino a un valore **threshold**, inizialmente pari a 64 Kbyte, dopodiché aumenta linearmente di 1 segmento alla volta;
- quando si verifica un timeout per un segmento:
  - il valore di threshold viene impostato alla metà della dimensione della congestion window;
  - la dimensione della congestion window viene impostata alla dimensione del massimo segmento usato nella connessione.

Vediamo ora un esempio, con segmenti di dimensione 1 Kbyte, threshold a 32 Kbyte e congestion window arrivata a 40 Kbyte:



**Figura 6-17:** Esempio di controllo della congestione TCP

## 12.2 Protocollo UDP

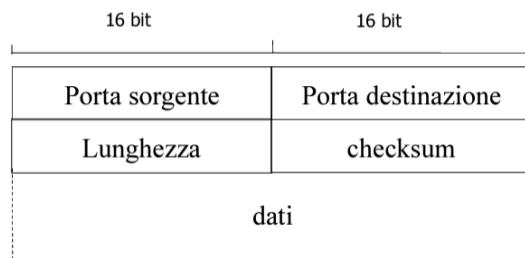
**UDP o User Datagram Protocol** è un protocollo senza connessione che si trova nel livello di trasporto del modello TCP/IP. Non stabilisce una connessione né controlla se l'host ricevente è pronto per ricevere o meno; invia semplicemente i dati direttamente. UDP viene utilizzato per trasferire i dati a una velocità maggiore. È meno affidabile e quindi utilizzato per la trasmissione di dati come file audio e video, VoIP

UDP non garantisce la consegna dei dati, né ritrasmette i pacchetti persi. È solo un protocollo wrapper che facilita l'applicazione nell'accesso all'IP.

In genere, quindi UDP:

- Non garantisce affidabilità di consegna
- Se il datagram eccede la MTU (Maximum Transfer Unit) della rete, esso viene frammentato
- Richiede meno overhead di una connessione TCP ( header/connesione/ack)

### **Header UDP**



Le porte UDP sono indipendenti da quelle TCP.

La lunghezza in byte comprende sia i dati che l'header e un pacchetto UDP può contenere fino a 65507 byte = 65535-20(IP)-8 (UDP).

Il checksum comprende anche uno pseudo-header che contiene le informazioni IP (indirizzi IP, tipo di protocollo, dimensione)

Ulteriori caratteristiche dell'UDP:

- **Servizio di consegna con impegno (best effort):** i segmenti UDP possono essere persi, duplicati, consegnati senza ordine.
- **Servizio senza connessione:** non vi è handshaking tra mittente e destinatario del segmento UDP.
- Ogni segmento UDP è trattato in modo indipendente dagli altri

A differenza di TCP, UDP:

- Non introduce ritardo per instaurare la connessione
- Non mantiene lo stato di connessione (non deve gestire buffer di invio/ricezione, parametri per il controllo della congestione, numeri di sequenza, etc.)
- Produce un minore overhead (header di 20 byte per TCP e 8 per UDP)
- Non controlla la congestione (il controllo della congeszione effettuato dal TCP può avere un severo impatto su applicazioni real-time)

#### Elenco applicazioni:

<i>Applicazione</i>	<i>Prot. strato applicativo</i>	<i>Prot. trasporto sottostante</i>
Posta Elettronica	SMTP	TCP
Accesso Terminale Remoto	Telnet	TCP
Web	HTTP	TCP
Trasferimento File	FTP	TCP
File Server Remoto	NFS	solitamente UDP
Multimedia Streaming	proprietario	solitamente UDP
Telefonia Internet	proprietario	solitamente UDP
Gestione della rete	SNMP	solitamente UDP
Protocollo di Routing	RIP	solitamente UDP
Traduzione dei nomi	DNS	solitamente UDP

Posta elettronica, Telnet, Web, FTP: per questi servizi è necessario il servizio di trasferimento dati affidabile fornito da TCP.

RIP per gli aggiornamenti periodici delle tabelle usa UDP, eventuali informazioni perse saranno sostituite da quelle più aggiornate.

**DNS usa UDP:** si evitano i ritardi dovuti all'instaurazione della connessione.

Telefonia su Internet usa l'UDP perché tollera la perdita di dati ma richiede un tasso di trasmissione costante (non controlla la congestione).

Applicazioni multimediali usano l'UDP perché il TCP non consente il multicast (problema della mancanza del controllo di congestione in UDP).

Il checksum UDP risulta indispensabile per la rilevazione di errori laddove il livello 2 adottato non fornisce tale servizio.

Il checksum a livello IP estende il suo controllo al solo header IP, quindi non consente di rilevare errori nei dati.

**L'UDP non opera alcun recupero dell'errore** (il segmento errato viene scartato o consegnato all'applicazione segnalando che presenta errori)

Checksum UDP: calcolato usando il complemento ad 1 della somma di tutti i campi dello pseudo-header e del segmento UDP.

Esempio: 3 parole da 16 bit l'una

0110011001100110	0110011001100110
0101010101010101	0101010101010101
000011100001111	<u>000011100001111</u>
	1100101011001010
complemento ad 1	0011010100110101

campo checksum nel segmento UDP = 0011010100110101.

Il destinatario calcola il checksum del segmento UDP ricevuto (senza calcolare complemento a 1)

Se checksum dati + checksum UDP = 1111111111111111 → assenza di errore

Se checksum dati + checksum UDP ≠ 1111111111111111 → errore

## LEZIONE 13 – IL LIVELLO APPLICATIVO

Nel modello ISO/OSI il livello **Application** è sopra il livello Transport (primo posto), ed è dove viene svolto il lavoro utile per l'utente.

Questo è il livello più alto e fornisce servizi direttamente alle applicazioni utente. Include una vasta gamma di protocolli e servizi che consentono alle applicazioni di comunicare attraverso la rete. Esempi di protocolli di livello di applicazione includono:

- DNS (Domain Name System) per la traduzione dei nomi di dominio in indirizzi IP;
- SNMP (Simple Network Management Protocol, RFC 1157) per la gestione della rete;
- FTP (File Transfer Protocol, RFC 959) per il trasferimento di file;
- SMTP e POP3 (Simple Mail Transfer Protocol, RFC 821, e Post Office Protocol, RFC 1225) per la posta elettronica;
- HTTP (HyperText Transfer Protocol, RFC 1945) alla base del World Wide Web (WWW)

### DNS

La gestione degli indirizzi IP in forma numerica è una pretesa inaccettabile dal lato utente.

Per tale ragione agli indirizzi IP sono di solito associati dei nomi.

La **trasformazione di un indirizzo in un nome** può avvenire in due modi:

- Tramite un elenco indirizzo-nome contenuto nel file **/etc/hosts**
- Utilizzando il **servizio DNS** (lavora al livello 7 della pila ISO-OSI)

Uno scenario d'esempio potrebbe essere quello in cui ricerchiamo un particolare indirizzo nel browser(es: [www.google.com](http://www.google.com)). Il browser controllerà in prima istanza se nella sua memoria già c'è la corrispondenza [www.google.com](http://www.google.com) indirizzo IP, se non la trova andrà a controllare il file **hosts** della macchina. Qualora non venga trovato neanche nel file hosts, verrà contattato un **server DNS** (primario o secondario (nel caso in cui il primario sia irraggiungibile) per la risoluzione dell'indirizzo.

Solitamente, nel caso in cui il client non abbia un indirizzo IP statico, ma è assegnato automaticamente dal DHCP, verrà contattato in prima istanza il server DNS dell'ISP (Internet Service Provider)

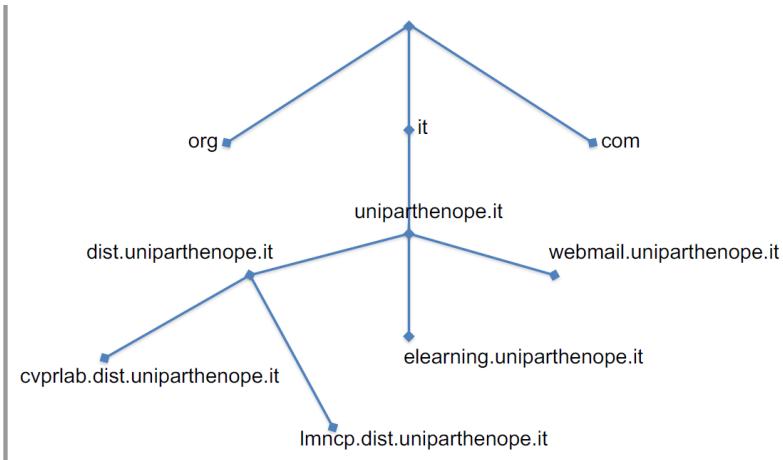
L'utilizzo del DNS impone l'utilizzo della convenzione dei **nomi di dominio**.

Tale convenzione è rappresentabile da una **struttura ad albero**, dove la **radice dell'albero** è il dominio principale, rappresentato da un punto singolo (solitamente è sottinteso)

Ogni **nodo** dell'albero è un **dominio**.

Il **nome di un dominio** si ottiene effettuando l'unione dei nomi dei nodi attraversati per giungere alla radice, separando tali nomi l'un dall'altro con un punto.

I **nodi terminali** dell'albero corrispondono agli **host** di rete.



I nomi di dominio utilizzati in Internet si ottengono attraverso una fase chiamata registrazione.

La registrazione avviene facendo una richiesta all'autorità competente per la zona a cui appartiene il nome. Generalmente si registrano nomi di secondo livello, per cui ci si rivolge all'**autorità di registrazione (RA)** competente per il dominio di primo livello.

La RA di gerarchia più alta di tutte è l'Internet Corporation For Assigned Names and Numbers (**ICANN**), ovvero un'organizzazione no-profit responsabile della gestione e del coordinamento del DNS a livello mondiale, che:

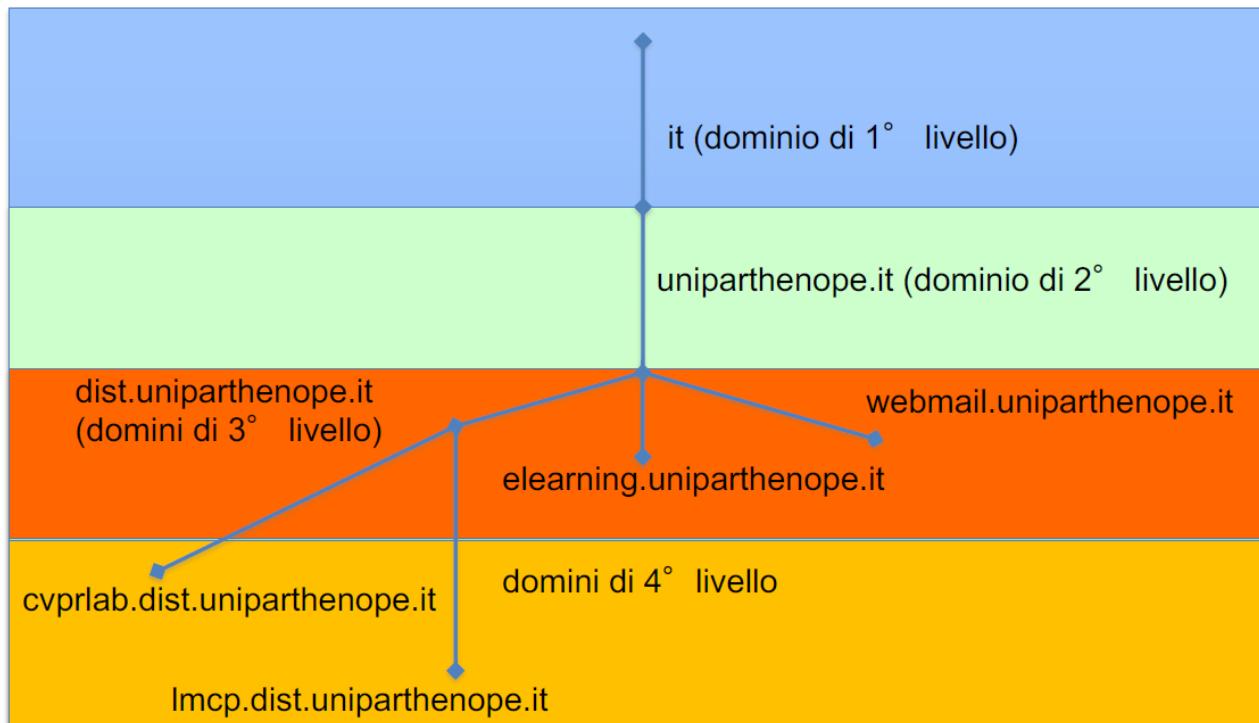
- è competente per la zona relativa alla radice dell'albero dei domini
- demandando la gestione dei domini di primo livello ad opportune RA, le Top Level Domain RA (**TLD RA**)
- delega **13 named authorities** ubicate in diverse parti del mondo, con funzione di **root server**
- ciascuno dei root server dell'ICANN contiene uno stesso database di informazioni nomi-indirizzi, relativo alle TLD RA.

Per visualizzare le **informazioni sui domini** come l'azienda a cui è intestato il dominio, il dominio di validità, i nomi DNS e/o gli indirizzi IP degli host che fungono da server autoritativi è possibile utilizzare il **protocollo NICKNAME (WHOIS)**.

Il DNS è un **servizio distribuito**, ovvero un **insieme di server** di nomi (host su cui è in esecuzione il servizio), ciascuno con una propria **zona di competenza**.

Una **richiesta di traduzione** nome-indirizzo è inviata al server di zona; se tale informazione non è disponibile, invia la richiesta e recupera l'informazione da un altro server DNS.

Le **zone di competenza si sovrappongono all'albero dei domini**, in modo da garantire che una qualsiasi richiesta possa essere soddisfatta grazie all'interrogazione di una opportuna catena di server



Ogni zona organizza le informazioni di sua competenza in **record di risorsa** che definiscono l'**associazione** tra un **nome** di dominio ed un'altra **informazione**, in base al **tipo di record**.

## Record risorsa

Nome	TTL	Classe	Tipo	Dati
------	-----	--------	------	------

- Nome: nome di dominio
  - stringhe alfanumeriche
  - iniziano con una lettera
  - hanno max 63 caratteri
  - separate da un punto
  - non vi è distinzione tra maiuscole e minuscole
- Tempo di vita: tempo di validità in minuti del record per un server caching-only
  - un unsigned int (32 bit) che esprime (di default 86400 = 1 giorno)
- Classe: individua il tipo di protocollo
  - l'unico valore previsto è IN (Internet)
- Tipo: tipo di informazione associata al nome di dominio
- Dati: informazione associata al nome di dominio

# Tipi e dati di un record di risorsa

TIPO	VAL	SIGNIFICATO	DATI
A	1	L'indirizzo di un host	Un indirizzo IP
CNAME	5	Il nome canonico: specifica un alias per un host	Un nome di dominio
HINFO	13	La CPU ed il SO utilizzati dal computer host	Un commento
MX	15	Un server di mail per il dominio.	Un numero seguito da un nome di dominio
NS	2	Un server autoritativo per il dominio	Un nome host
PTR	12	Un puntatore a un'altra parte dello spazio dei nomi	
SOA	6	L'inizio di una zona di autorità	Parametri di config. relativi alla zona
WKS	11	Servizi di rete certamente in esecuzione su un dato host	Un indirizzo IP, il prot. trasp. ed i servizi

Per risolvere un nome in un indirizzo ci sono due modalità:

## 1. Risoluzione diretta (query iterativa):

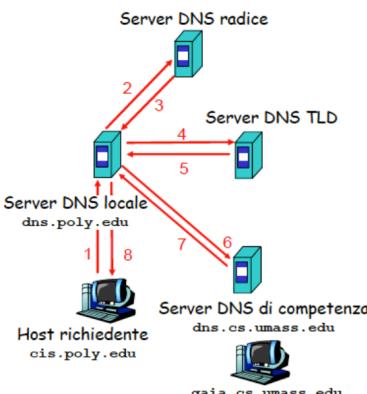
Supponiamo che l'host "PIPRO" voglia risolvere l'indirizzo [www.pluto.com](http://www.pluto.com)

L'host gira la richiesta di traduzione al server locale DNS, definito nella configurazione dell'host stesso (se l'IP è assegnato dal DHCP, il server locale sarà quello dell'ISP).

Il server locale contatta un **root server** (13 nel mondo di 12 organizzazioni, ma ci sono le repliche) che si occupa di andare a risolvere la prima parte dell'indirizzo (**com.**). Il root server risponderà con l'indirizzo del **server TLD** da contattare.

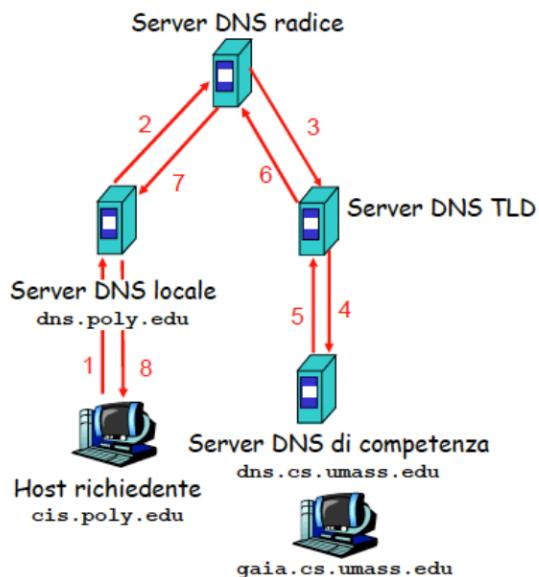
Il server DNS locale contatterà tale **server TLD** che ha la funzione di sapere tutto quello che serve per un determinato dominio (nel nostro caso **com.**) e quindi conosce chi ha registrato tale dominio. Infatti il server TLD risponderà con l'indirizzo dell'**authoritative server**, il quale, essendo responsabile della registrazione del sito web, conosce l'indirizzo IP di [www.pluto.com](http://www.pluto.com).

A questo punto il server DNS locale memorizza la corrispondenza e restituisce la risposta all'host PIPRO.



## 2. Risoluzione diretta (query ricorsiva):

Il server locale affida al server contattato il compito di tradurre il nome. Ogni server che non è in grado di rispondere si rivolge esso stesso ad un altro server ed attende la risposta. La risposta torna al server locale seguendo lo stesso percorso.



### 3. Risoluzione indiretta:

Il DNS fornisce anche un meccanismo per la traduzione di un indirizzo IP in un nome di dominio (risoluzione inversa).

Questo meccanismo, vista la natura degli indirizzi IP, non può sfruttare l'approccio gerarchico usato nella ricerca di un IP a partire da un nome di domini, infatti necessita di uno **spazio dei nomi specifico**.

Il dominio utilizzato è **in-addr.arpa**, per il quale gli indirizzi IP sono rappresentati al contrario (es: 192.168.1.5 diviene 5.1.168.192), per omogeneità con la codifica dei nomi di dominio

Quindi, in sostanza, L'interrogazione di un servizio DNS corrisponde all'interrogazione di una base di dati distribuita, in cui il risultato è il record desiderato.

La base di dati del DNS ha due scopi

1. trovare l'indirizzo numerico associato ad un nome
2. trovare il nome a partire da un indirizzo numerico

Esistono diverse modalità per interrogare il servizio direttamente.

Il programma classico per l'interrogazione del DNS è **nslookup**, disponibile sia su sistemi di tipo Unix che Windows.

Su alcuni sistemi Linux, ad nslookup sono preferiti programmi quali **host** e **dig**, e l'implementazione di nslookup può essere solo parziale.

Nell'implementazione completa **nslookup** prevede il comando **ls**, che permette di ottenere l'elenco di tutti i record presenti sul server DNS, ma generalmente viene impedito.

Esempi:

- nslookup google.com [server dns]
- nslookup -type=mx google.com [server dns]
- nslookup -type=ns google.com [server dns]
- nslookup -type=any google.com [server dns]

## Posta elettronica

Le **email** o **electronic mail** funzionano attraverso una semplice architettura: l'utente utilizza sul proprio computer un programma detto **User Agent**, questo programma interagisce con un **message transfer agent** per inviare o ricevere le mail alla propria **mailbox**. L'operazione di invio di una mail è detta **mail submission**.

Una email è formata da un messaggio composto da **header** e **body**, racchiuso in un **envelope**. L'header contiene le informazioni del mittente e l'oggetto del messaggio, l'envelope invece indica le informazioni del destinatario.

Inizialmente le mail consistevano in semplici segmenti di testo in ASCII, questo sistema molto velocemente è diventato obsoleto, e con il tempo è stato sviluppato **MIME (Multipurpose Internet Mail Extensions)**, che è costruito sopra allo standard originale (ovvero se MIME non è supportato il messaggio viene inviato come ASCII normale). MIME permette la trasmissioni di dati binari, file multimediali e consente ulteriori metodi di encoding.

Il trasferimento effettivo della mail avviene tramite il protocollo **SMTP (Simple Mail Transfer Protocol)**, che funziona con un processo dedicato alla gestione della mail in ascolto sulla porta 25 del computer. SMTP è un semplice protocollo ASCII, e i comandi sono formati da 4 caratteri. Il problema dell' SMTP è che non supporta l'autenticazione e la sicurezza dei messaggi, per questo motivo è stato successivamente sviluppato **ESMTP (Extended SMTP)** che contiene diverse estensioni per consentire nuove funzionalità.

Dopo aver inviato la mail al transfer agent apposito, il transfer agent non utilizza SMTP per consegnare la mail allo user agent, ma utilizza un altro protocollo detto **IMAP (Internet Message Access Protocol)** oppure in alternativa un **sistema webmail**.

## FTP

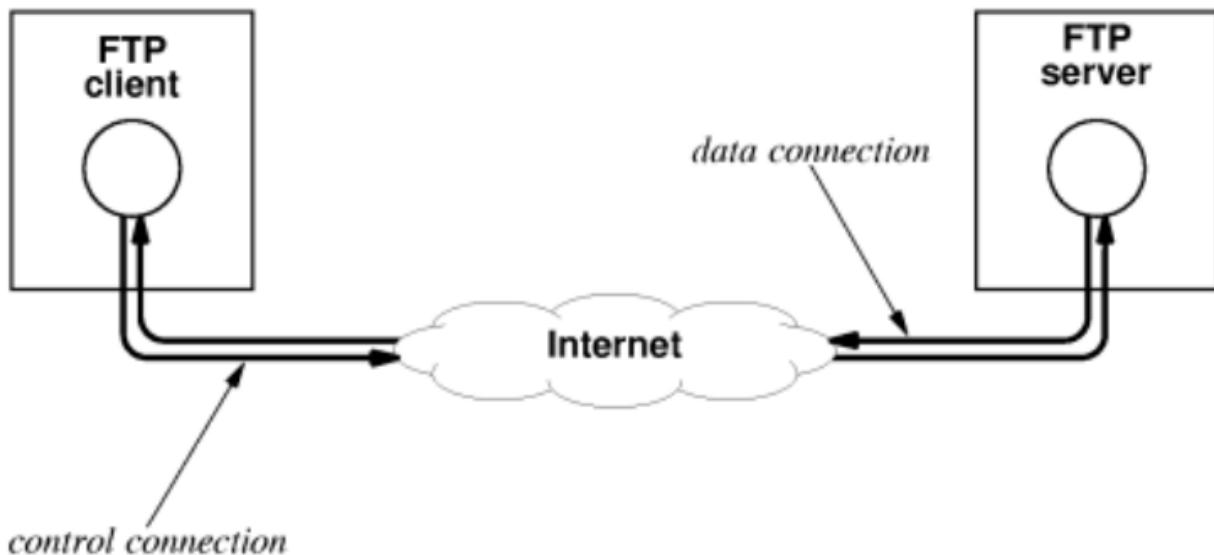
Protocollo di trasferimento dati tra macchine connesse attraverso una rete che supporta TCP/IP.

Si basa sul modello client/server:

- Il server FTP viene eseguito sulla macchina che accede fisicamente ai file e attende richieste.
- Il client FTP si collega al server e può richiedere al server operazioni inerenti il trasferimento dati

Il trasferimento dati e il controllo delle operazioni avviene via TCP mentre il server si trova in ascolto sulla porta TCP 21. Infatti, FTP utilizza la **porta 21 per connessione di controllo**, dove vengono inviati i comandi e le risposte del server, e la **porta 20 per connessione dati**, dove vengono inviati i dati veri e propri.

La differenza tra le due connessioni è, inoltre, che la connessione di controllo è sempre aperta all'interno di una sessione, mentre l'altra viene richiesta dal server e rimane aperta solo durante gli effettivi trasferimenti di dati. Utile come separazione per consentire il controllo fuori banda dei trasferimenti file.



I processi **FTP** sono distinti, quindi, in base alla loro funzione e sono **PI** (Protocol Interpreter) e **DTP** (Data Transfer Process), hanno le funzioni indicate sopra.

### Data Transfer Process

Può essere di due tipi:

- **Active mode:** il client si connette al server sulla porta 21 (**control connection**) e, prima del trasferimento dati, si mette in ascolto su una porta TCP (>1024). Il client notifica al server il numero di porta su cui si è messo in ascolto, il server quindi si connette alla porta aprendo la **data connection** e inizia il trasferimento dati. Questa modalità è **la più indicata dal punto di vista della sicurezza del server** siccome ha solo due porte utilizzabili (20-21).
- **Passive mode:** apre la **control connection** sulla **porta 21** del server. Il server, alla richiesta di trasmissione dati, apre una porta TCP (>1024) e trasmette attraverso la connessione di controllo il numero della porta. Il client si connette alla suddetta porta aprendo la **data connection**, iniziando il trasferimento dati. In questa modalità è **il client ad iniziare la connessione di trasferimento dati**.

### File Transfer Protocol

Ad ogni comando inserito il server risponde inviando un codice che identifica la riuscita o meno dell'operazione richiesta. I codici sono numerici e composti da tre caratteri **xyz**, dove **X** è il valore più significativo, seguito da **Y** e poi da **Z** che daranno maggiori dettagli in relazione a X.

- 1yz: **Risposta preliminare positiva.** Indica che il comando è stato accettato e che si avrà un'ulteriore risposta prima del comando successivo;
- 2yz: **Comando terminato con successo;**
- 3yz: **Risposta intermedia positiva.** Comando eseguito correttamente e in attesa di ulteriori informazioni per completare l'operazione;
- 4yz: **Il comando non è stato eseguito correttamente;**
- 5yz: **Comando che il server non ha potuto eseguire;**

Per utilizzare quindi FTP per trasferire file è necessario avere un **FTP-Client** e un **FTP-Server**, questo solitamente è **incluso nella maggior parte dei sistemi operativi**, come **Windows** e **Linux**.

Quando si prova a stabilire una connessione con un sito FTP, quest'ultimi richiederanno **login e password**, a meno che non si tratta di archivi di software aperti al pubblico e quindi si parla di **FTP Anonimo**.

Un FTP anonimo solitamente, **permette di attivare la connessione ad esso indicando :**

Login: “**anonymous**” (nome convenzionale)

Password: indirizzo della **propria e-mail**

Normalmente sono costruite avendo una sottodirectory, della directory radice FTP, **pub** (pubblica) che contiene i file a cui si ha accesso e possono essere trasferiti.

Di seguito alcuni **comandi FTP**:

#### **ftp> open sito**

**login:**

**Password:** Si collega con il sito indicato, fornendo login e password.

**cd pippo** Entra nella directory **pippo** sul computer remoto.

**pwd** Scrive il nome completo della directory remota in cui vi trovate.

**ls** Mostra i file contenuti nella directory corrente.

**lcd pippo** Entra nella directory **pippo** sul vostro computer.

**binary (o bin)** Setta la modalità di trasferimento binaria.

**ascii (o asc)** Setta la modalità di trasferimento ASCII.

**get nomefile** Preleva il file **nomefile** e lo salva nella directory corrente sul vostro computer.

**prompt** Esclude la modalità interattiva, utile per trasferimenti multipli

**mget nomefile** Come get, ma permette l'uso di asterischi nel **nomefile**.

**put nomefile** Copia il file **nomefile** dal vostro computer a quello remoto.

**mput nomefile** Come put, ma permette l'uso di asterischi nel **nomefile**

**Hash** Stampa un # per ogni blocco di 1 KB trasferito con successo

**help** Mostra l'elenco dei comandi supportati.

**quit** Si scollega dal sito a cui si è collegati.

**bye** Si scollega dal sito a cui si è collegati ed esce dal programma

**CTRL-C** Per interrompere un trasferimento

#### **Criticità di FTP**

- **FTP** non implementa un meccanismo di **cifratura** dei dati: vulnerabile allo sniffing.

- **Alta latenza** dovuta al numero e al formato dei comandi.
- Non possiede un meccanismo di controllo per l'**integrità dei dati**.
- Non possiede un meccanismo di **controllo dell'errore**
- Non trasferisce gli **attributi dei file**.

## LEZIONE 14 – TELEFONIA CELLULARE

Nella telefonia cellulare ogni area è suddivisa in più celle, ciascuna servita da un trasmettitore di debole potenza su frequenze destinate ad essere riutilizzate in celle non contigue. Quando ci si muove tra le celle, il sistema cellulare commuta agganciando automaticamente il segnale più intenso in quel punto, questo è il cosiddetto **handover** tra le celle.

Abbiamo tre aspetti importanti da considerare quando si parla di reti wireless:

- **Attenuazione del segnale:** le radiazioni elettromagnetiche si attenuano quando attraversano determinati ostacoli; nello spazio libero l'intensità del segnale si attenua al crescere della distanza percorsa (path loss).
- **Interferenze** da parte di altre sorgenti: frequenze wireless standard (es. 2,4 GHz) condivise da altri dispositivi (es. telefonini); anche rumori ambientali (es. motori) causano interferenza.
- **Propagazione** su più cammini: una parte delle onde elettromagnetiche si riflette su oggetti e sul terreno, compiendo cammini di diversa distanza tra trasmittente e ricevente.

Quindi, abbiamo diversi fenomeni che potrebbero causare problemi alla rete, che sono:

**Riflessione, diffrazione, assorbimento, rifrazione e diffusione**

### CDMA, FDMA e TDMA

Agli utenti di uno stesso servizio l'accesso simultaneo al mezzo trasmissivo è consentito mediante tecniche di divisione:

- di *frequenza (FDMA)*: utenti diversi trasmettono contemporaneamente su frequenze diverse.
- di *tempo (TDMA)*: utenti diversi trasmettono in tempi diversi sulla stessa frequenza.
- di *codice (CDMA)*: utenti diversi trasmettono contemporaneamente sulla stessa frequenza usando codici diversi.

### CDMA

È il protocollo di accesso al canale condiviso più diffuso nelle reti wireless e nelle tecnologie cellulari, utilizza una tecnica ad accesso multiplo basato su **codici ortogonali**. Consiste nell'assegnare un codice unico a ciascun utente (*code set partitioning*), poiché tutti gli utenti condividono la stessa frequenza, ma ciascuno ha la propria sequenza chipping per codificare i dati.

Consente, quindi, a più utenti di coesistere e trasmettere simultaneamente con un'interferenza minima.

Come abbiamo detto, le sequenze di chip sono scelte in modo da essere tra loro ortogonali. Queste vengono create dividendo ciascun tempo di bit in m sottointervalli.

$$\text{chip } S=[s_1, s_2, \dots, s_m] \quad s_i \in \{-1, +1\}$$

Ogni bit è rappresentato da una sequenza di chip e per ogni bit da trasmettere, si trasmette il chip o il suo opposto.

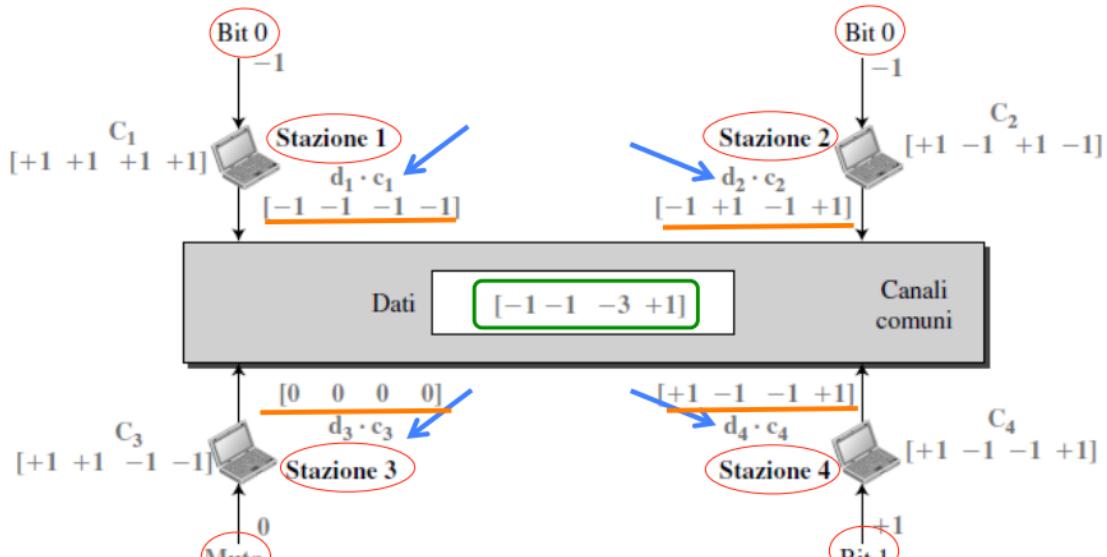
Esempio:

$$b=1 \rightarrow s = [-1, -1, -1, +1, +1, -1, +1, +1]$$

$$b=0 \rightarrow -s = [+1, +1, +1, -1, -1, +1, -1, -1]$$

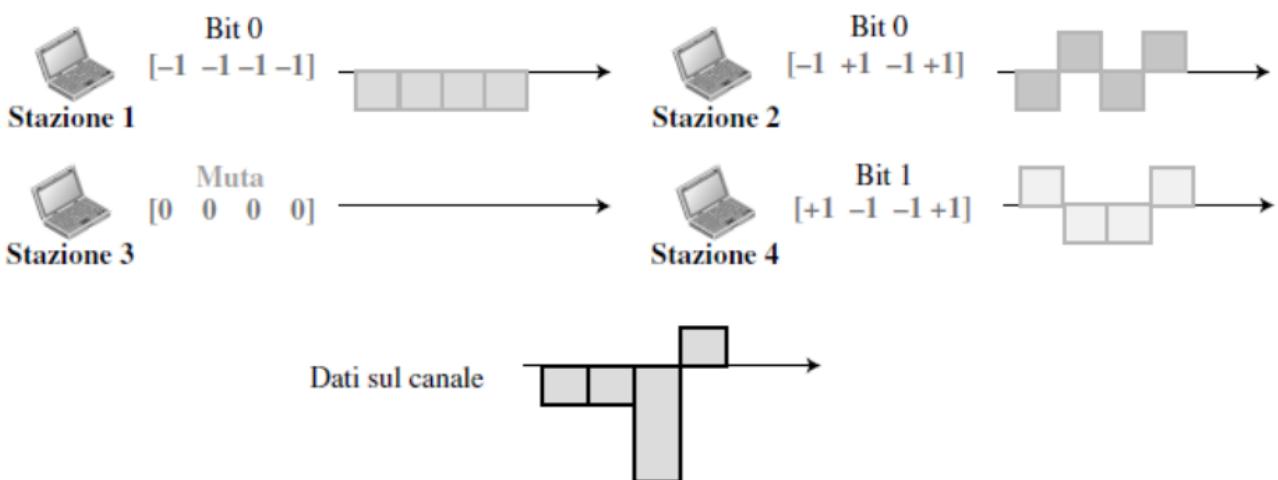
Ogni sequenza è composta da N elementi (N=stazioni), N deve essere una potenza di 2.

Esempio con 4 stazioni:



- ❖ La sequenza sul canale è la somma delle quattro sequenze inviate dalle stazioni
- ❖ La stazione 3 ascolta la 2 →  $[-1 -1 -3 +1] \cdot [+1 -1 +1 -1] = -4$   
 $-4/4 = -1 \rightarrow \text{bit } 0$

Segnale digitale creato dalle 4 stazioni:



Il CDMA permette, quindi, a **più utenti** di trasmettere **contemporaneamente** (al contrario di TDMA) senza **suddividere lo spettro** disponibile (al contrario di FDMA).

Trasmettendo  $m$  chip per ogni bit generato, si passa da una bit rate  $R$  a una chip rate  $R_c = mR$ . Avendo così lo spettro del segnale che risulta  $m$  volte più largo e un segnale più resistente ai disturbi.

## LEZIONE 15 – NAT (Network Address Translation)

Il **NAT (Network Address Translation)** nasce come risoluzione di un problema:

Si ha a disposizione un numero **N** di macchine e un numero **M** di indirizzi IP, con **M < N**.

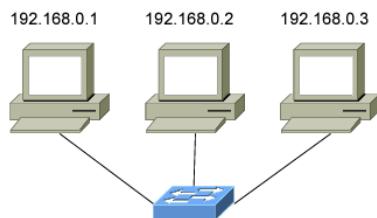
Gli indirizzi IP, infatti, hanno un costo che, generalmente, risulta tanto più elevato quanto minori sono le caratteristiche del contratto di connettività acquistato. Si rende pertanto necessario un sistema per consentire alle restanti macchine di collegarsi ad Internet. Per semplicità, nel seguito, supponiamo di disporre di un solo indirizzo IP.

Nell'assegnazione degli indirizzi IP, vi sono alcuni blocchi di indirizzi, descritti in RFC1918, che vengono allocati per reti ad uso privato:

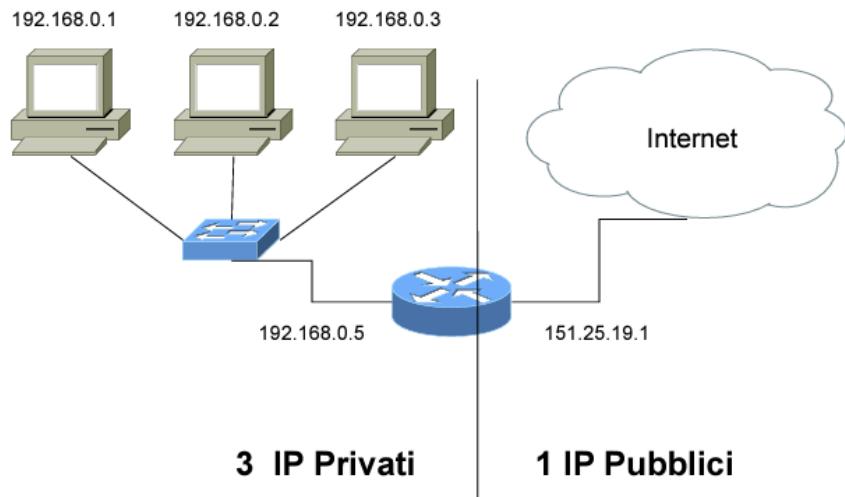
- La rete di classe A 10.0.0.0 – 10.255.255.255
- Le 16 reti di classe B 172.16.0.0 – 172.31.255.255
- Le 256 reti di classe C 192.168.0.0 – 192.168.255.255

Non esiste in Internet una rete con questi blocchi di indirizzi e i router della rete sono programmati (o, almeno, dovrebbero esserlo) per non instradare pacchetti provenienti da questi indirizzi.

Prendiamo l'esempio di una semplice Intranet senza uscita su Internet.



Assegnamo l'unico IP address pubblico disponibile ad un router con due interfacce di rete: una collegata all'intranet e una su internet (extranet).



**Problema:** un pacchetto proveniente da 192.168.0.1 **non può essere mandato su Internet**, in quanto verrebbe automaticamente cancellato dal primo router in rete e, quand'anche così non fosse, la risposta non arriverebbe a destinazione. L'unica via d'uscita è che un pacchetto esca con 151.25.19.1

Quindi, il router riceve una richiesta di connessione da 192.168.0.1 verso il nodo X. Sostituendosi alla macchina locale, il router effettua la richiesta con il SUO indirizzo IP pubblico. Ricevuta risposta, la inoltra sulla rete privata per il tramite del suo IP privato e procede così per tutta la durata della connessione.

Quindi il NAT è una tecnica che consente al **router** di agire come **intermediario** tra Internet (rete pubblica) e una rete privata. In questo modo, un **unico indirizzo IP** può rappresentare un **intero gruppo** di computer di **una rete privata**. Questo permette di ovviare al problema di **macchine>indirizzi IP**.

L'uso più comune del NAT è quello di **mappare** un insieme di indirizzi privati su di un unico indirizzo pubblico, utilizzando **differenti porte** (di livello **trasporto**) per mantenere traccia dell'indirizzo privato di provenienza.

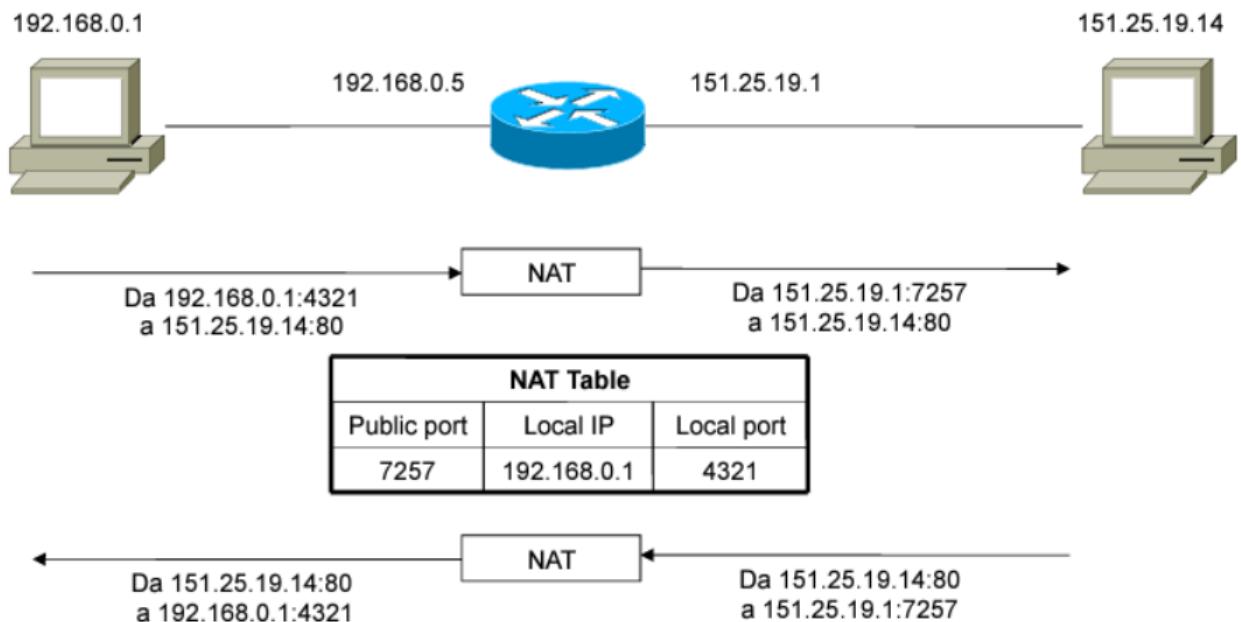
Infatti, l'intero meccanismo del **NAT** si regge interamente sull'utilizzo delle **porte**.

Quando il router riceve una richiesta di instradamento di connessione, effettua la connessione con il proprio IP pubblico e memorizza in una tabella:

- **l'indirizzo IP e la porta sorgente locale**
- la porta locale con cui effettua la connessione

Così facendo, è in grado, quando arriva un pacchetto di risposta sulla porta locale, di sapere a quale macchina rigirarlo.

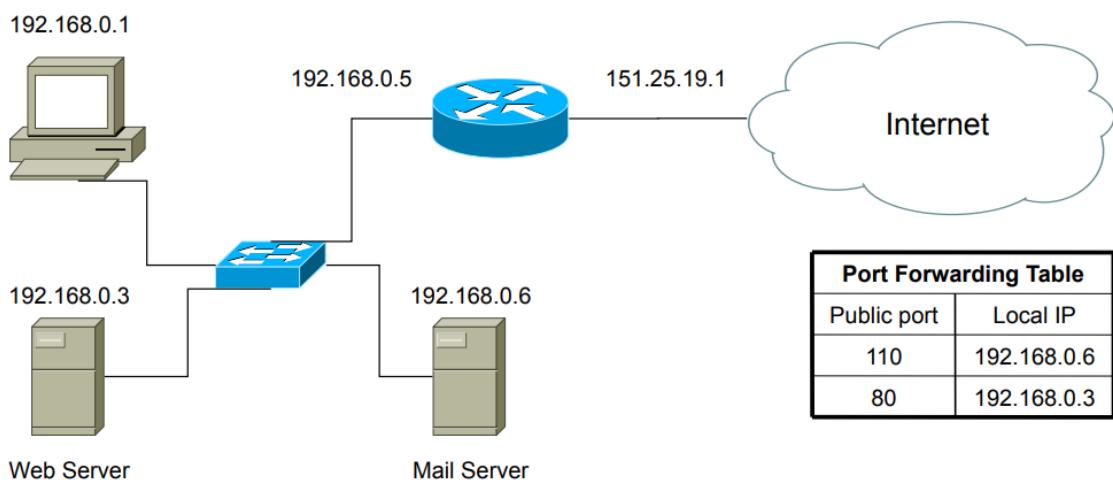
## Esempio



## Limitazioni del NAT

Se, tuttavia, dall'esterno arriva una richiesta di connessione ad una porta che non è stata precedentemente allocata da una macchina interna tramite una connessione, il NAT non sa a quale delle macchine interne rigirarla e scarterà il pacchetto. A meno che non si applichi una configurazione che prende il nome di **PORT FORWARDING**. Quest'ultima consiste nel configurare a priori il NAT per far sì che, alla richiesta (dall'esterno) di una connessione su una determinata porta di cui non si ha traccia, il NAT invii il pacchetto a una macchina predefinita all'interno della rete.

Esempio:





## LEZIONE 16 – Multimedia Network Application

Per **Multimedia Network Application** si intendono tutte quelle applicazioni di rete che prevedono l'utilizzo di audio e video.

Le varie categorie sono: **streaming audio video (stored), voice/video-over-IP (VOIP), straming live audio/video.**

Ognuna di queste categorie di applicazioni ha il suo **set di requisiti di servizio** e problematiche di progettazione.

### Streaming audio video

Si tratta di audio/video **pre-registrati** come film, show e serie televisive come quelle presenti su YouTube, Netflix ecc. La richiesta è **on-demand**

Caratteristiche chiave: **Streaming, Interattività, Riproduzione continua.**

Storing: **CDN, P2P**

### Voice-and Video-over-IP

Si tratta di **servizi voce in real-time su internet**, comunemente conosciuti come **VOIP ( Voice over IP).**

Esempi di applicazione sono Skype, Google Talk ecc.

### Caratteristiche:

1. Bisogna fare considerazioni sul **Timing** sono importanti perché questa tipologia di applicazioni sono altamente sensibili ai ritardi
2. le applicazioni multimediali sono **tolleranti alle perdite di informazioni** (in piccola parte)

### Streaming Audio Video Live

Simili al tradizionale broadcast radio e televisivo

E' basato su tecniche di IP multicasting.

## LEZIONE 17 – Streaming Stored Video

Nello **Streaming Stored Video**, i video, **pre-registrati**, sono salvati su **server**. Gli **utenti** inviano **richieste** a questi server per vedere contenuti **on-demand**

### Categorie di Streaming Stored Video:

1. UDP Streaming
2. HTTP Streaming
3. Adaptive HTTP Streaming

A prescindere dalla categoria, viene di solito applicato un **client-side application buffering**, per **mitigare** gli effetti della variazione dei ritardi end-to-end. Ciò è ottenuto **variando la quantità** della larghezza di banda a disposizione tra server e client.

### UDP Streaming

#### Nell'UPD Streaming:

1. Il server trasmette **video** ad un velocità che corrisponde alla **capacità di utilizzo** del video del client
2. Vengono inviati dei **chunks** (pezzi) su UDP ad una velocità **costante**: 3Mbps Video con trasmissione sulla socket ogni 4 msec
3. UDP **non implementa un meccanismo della congestione**

#### Gli **svantaggi** dell'UDP Streaming sono:

1. **constant-rate**: UDP streaming può **fallire** non riuscendo a fornire una riproduzione continua
2. È necessario un media control server, come un server **RTSP**, per processare richieste di interazione dal client al server e per tracciare lo stato del client
3. Molti **firewalls** sono configurati per **bloccare il traffico UDP**

### HTTP Streaming

Nell'**HTTP Streaming**, il video è semplicemente conservato in un **server HTTP** come un comune file con un URL specifico.

Il client stabilisce una connessione **TCP** con il server ed effettua una richiesta «**HTTP GET**» per quell'URL.

Il Server **serve** il file video, inserendolo in un messaggio di risposta http.

Lato client, i bytes sono collezionati in un buffer e, una volta che il numero di bytes nel buffer **superà** un predeterminato **threshold** (soglia), l'applicazione client inizia la riproduzione.

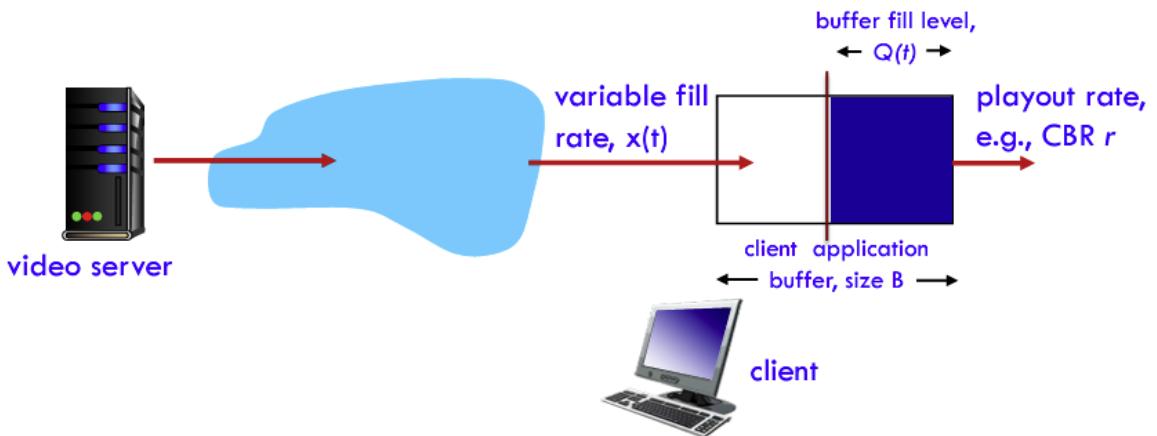
Periodicamente recupera i frames del video dall'applicazione del client buffer, decomprime i frames, e li visualizza (display) sullo schermo dell'utente.

#### Caratteristiche:

1. **Consente** al Video di transitare attraverso i **firewalls**
2. **Sopperisce** alla necessità di avere un media control server, come ad esempio per **RTSP** server.

Esempi di applicazioni HTTP Streaming sono YouTube e Netflix.

## Prefetching and Buffering



- Quando la velocità di trasferimento sulla rete è minore della velocità del video, la riproduzione alterna tra periodi di riproduzione continua e periodi di blocco (freezing)
- quando la velocità di trasferimento a disposizione alla rete è maggiore della velocità del video, dopo la prima fase di ritardo dovuta al buffering, l'utente può avere una riproduzione continua fino alla fine del video

### Adaptive Streaming e DASH

Nel **Dynamic Adaptive Streaming over HTTP (DASH)**, il video è codificato in molte **differenti versioni**.

Ogni versione ha a disposizione un **differente bit rate** e, di conseguenza, **una differente qualità**. Quando la larghezza di banda (bandwidth) è alta, vengono selezionati i chunks dalla versione a migliore qualità (**high-rate**), altrimenti, se è bassa, vengono selezionati i chunks dalla versione a bassa qualità (**low-rate**).

Ogni versione del **video** è conservata nel server HTTP, ognuna con un **differente URL**

Il server HTTP ha un file detto **MANIFEST** che **fornisce** un **URL** per ogni versione **insieme al suo bit rate**.

Mentre si scaricano i chunks, il client **misura** la larghezza di banda in ricezione ed esegue un algoritmo di **determinazione del tasso di trasferimento** per selezionare il **chunk** da richiedere successivamente. E quindi è possibile passare senza vincoli tra differenti livelli di qualità, mitigando il blocco della riproduzione.

## LEZIONE 18 – VOIP

Per **VoIP (Voice Over Ip)** si intendono quei **servizi voce in real-time su internet**, comunemente conosciuti come ad esempio Skype, Google Talk ecc.

Basandosi su **IP**, che si tratta di un protocollo che offre un servizio di tipo **best effort**, è soggetto a ritardi e possibili perdite di pacchetti.

La trasmissione avviene inviando **segmenti UDP** per ottenere una maggiore velocità di trasmissione (a discapito di possibili perdite di pacchetti ed errori), incapsulati in un **datagram IP**. I segmenti di voce viaggiano per la rete come se fossero normali pacchetti e quindi instradati dai router. E' possibile che **uno o più** router durante il tragitto abbiano un **buffers pieno** e in questi casi il pacchetto viene scartato.

L'eliminazione delle perdite potrebbe essere ottenuta utilizzando **TCP** invece di UDP. Ciò risulta **inaccettabile** per applicazioni convenzionali come l'**audio real-time**.

UDP è utilizzato da **Skype** anche se un utente è dietro una NAT o un firewall che **blocca i segmenti UDP** (nel qual caso viene utilizzato TCP)

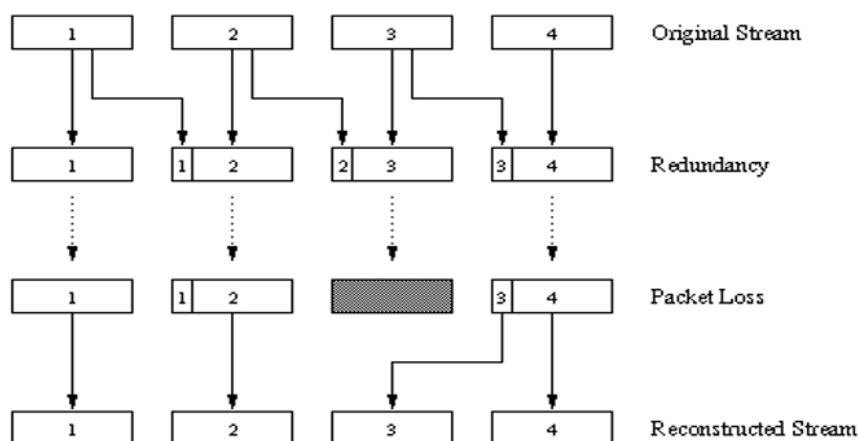
La perdita dei pacchetti in applicazioni real-time audio è sì un problema, ma se la **percentuale** si aggira tra l'1 e il 20 percento, allora **può essere tollerata**.

Se **il tempo** da quando un pacchetto è generato dalla sorgente a quando il destinatario lo riceve **varia da pacchetto a pacchetto**, a causa ad esempio di **code differenti nei router**, si parla di **packet jittering**.

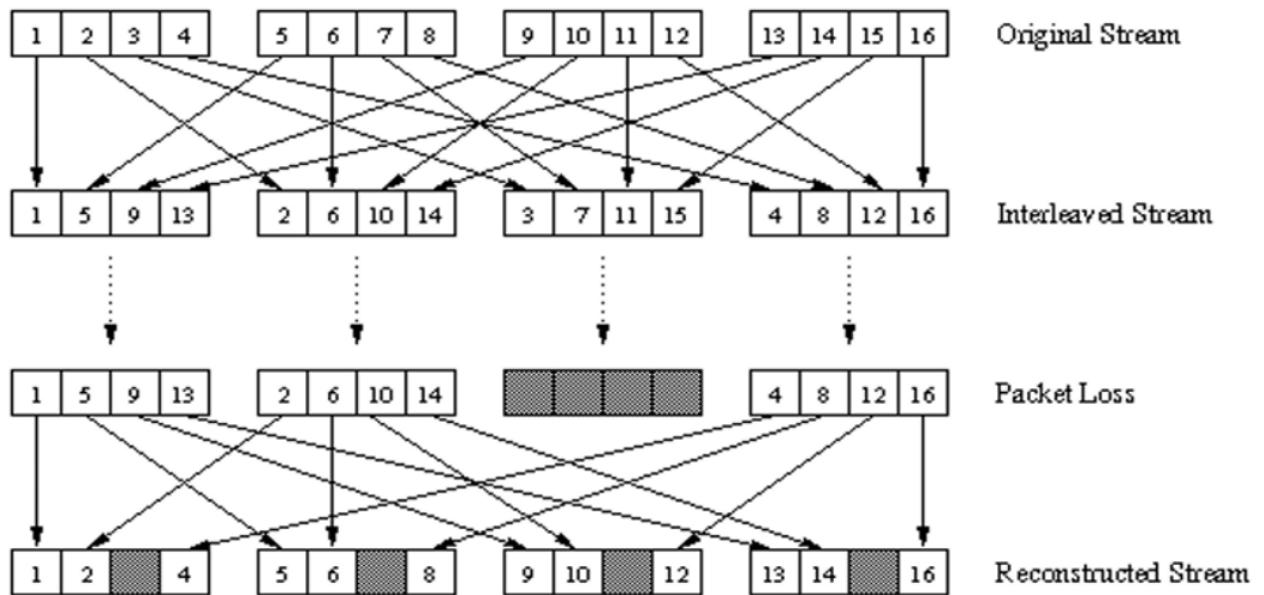
Per quanto riguarda gli errori come ad esempio possibile perdita di pacchetti, possono essere adottate due soluzioni:

1. Una soluzione può essere quella di applicare una **correzione anticipata dell'errore (FEC – Forward Error Correction)**.

Nel caso delle applicazioni audio real-time, viene inviato un flusso audio a bassa qualità come informazione ridondante nel pacchetto successivo.



2. Una seconda soluzione consiste nell'**interlacciamento (interleaved)**, ovvero interlacciare tra di loro i dati di pacchetti diversi, in modo che un **burst di errori** si presenti come un singolo errore in ogni pacchetto una volta che la destinazione ha ripristinato l'ordine.



Per occultare l'errore si tenta di produrre un rimpiazzo per un pacchetto che sia simile all'originale oppure effettuare l'interpolazione dei pacchetti (precedente e successivo)

## LEZIONE 21 – P2P

Il modello Peer-to-Peer si differenzia dal modello Client/Server in quanto ogni Peer ha funzionalità largamente simile.

Il modello P2P è nato a metà anni 2000 con l'intento di far ritornare il contenuto, la scelta e il controllo agli utenti. P2P permette di creare community e scambiare informazioni tra minuscoli endpoint nell'Internet senza nemmeno doversi conoscere.

Il modello P2P si basa su una rete di nodi con capabilità e responsabilità equivalenti, i nodi sono sia server che client **“Servents”**, e costituiscono una rete effimera che utilizzano per collaborare.

I peer connessi costituiscono una rete sovrapposta (overlay network) al di sopra dell'infrastruttura sottostante.

Le reti sovrapposte sono grandi gruppi di connessioni logiche tra gli end host, e possono essere **strutturate** oppure **non strutturate**.

### Cos'è un overlay network?

Una rete overlay è un insieme di connessioni logiche tra gli host finali.

Viene “definita” nel livello applicazione e la messaggistica avviene tramite l'utilizzo di TCP, UDP, ICMP.

### Caratteristiche:

- **Sistema distribuito e decentralizzato:** Nel modello P2P, non c'è un'entità centralizzata che controlla o gestisce l'intera rete. I partecipanti sono equivalenti tra loro e collaborano per fornire e accedere alle risorse.
- **Condivisione di risorse:** I partecipanti nel modello P2P condividono le proprie risorse, come potenza di calcolo, larghezza di banda, file e altro. Ogni partecipante può agire sia come cliente che come fornitore di risorse.
- **Autoorganizzazione:** Il modello P2P è progettato per essere resiliente e autoorganizzante. Quando nuovi partecipanti si uniscono alla rete o quando alcuni partecipanti vanno offline, la rete può continuare a funzionare senza dipendere da un singolo punto di fallimento. Nei sistemi P2P gli utenti accedono alle risorse in seguito ad una fase di **ricerca** di un **nodo già connesso**. E' comunque necessario uno spazio di indirizzamento ed un algoritmo di routing.
- **Architettura overlay:** Spesso, il modello P2P utilizza un'architettura overlay, che è una rete virtuale costruita in cima alla rete fisica. Questa rete overlay facilita le comunicazioni tra i partecipanti e può essere più efficiente nella gestione delle risorse rispetto a una rete puramente fisica.

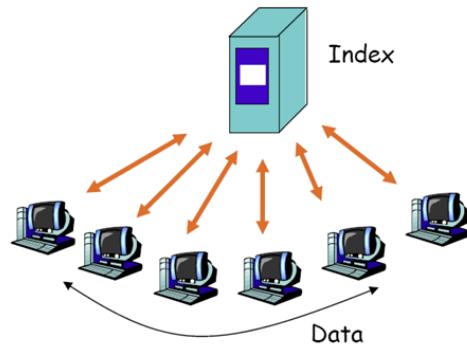
Gli **obiettivi** del modello P2P sono:

- Riduzione dei **costi** condividendo il costo stesso.
- Grande **scalabilità e affidabilità**
- **Interoperabilità** attraverso l'aggregazione di risorse diverse
- **Maggiore autonomia**.
- **Anonimità/Privacy**, che è difficile da ottenere quando il server è centrale
- **Dinamismo e comunicazioni ad hoc**.

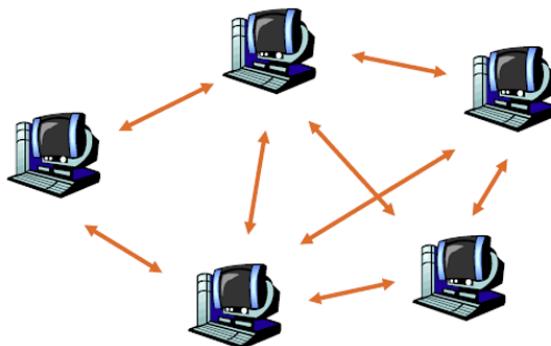
Nel P2P le risorse entrano e lasciano il sistema continuamente, quindi le reti P2P non possono affidarsi a infrastrutture esistenti.

I sistemi P2P possono essere classificati in base al grado di decentralizzazione:

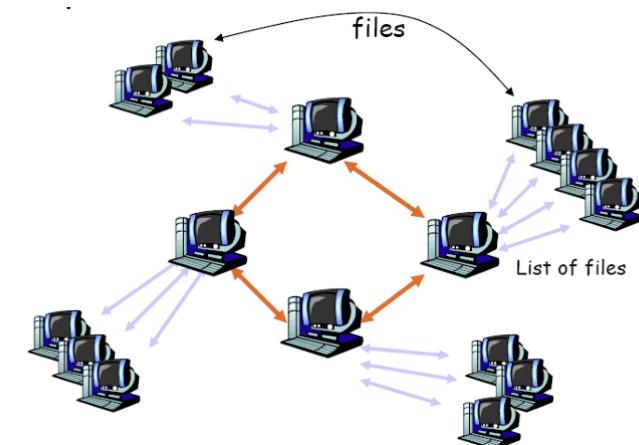
- L'**Hybrid decentralized P2P** sfrutta un server centrale per facilitare l'interazione tra i peer. Il server effettua delle ricerche e identifica i nodi nel network. Gli svantaggi principali sono la presenza di un singolo punto di fallimento e la scalabilità (Esempi: Napster)



- La **Purely decentralized P2P** si basa sui **servents** che effettuano le medesime mansioni senza una coordinazione centrale. Gli svantaggi principali sono la possibile mancanza di consistenza dei dati, l'overhead di comunicazione e la sicurezza. (Esempi: Gnutella inizialmente, Freenet)



- Il **Partially centralized P2P** utilizza alcuni nodi con ruoli più importanti, che agiscono come indici centrali, detti **supernodi**. (Esempi: Kazaa, Gnutella recentemente)



Inoltre i sistemi P2P possono essere classificati in base al grado di strutturazione:

- **Unstructured P2P**: i dati sono distribuiti casualmente tra i peer e meccanismi di broadcast sono utilizzati per la ricerca. Il piazzamento dei dati non è correlato alla topologie dell'overlay. (Esempi: Napster, Gnutella, Kazaa)
- Nello **Structured P2P** la topologia della rete è strettamente controllata e i file sono piazzati in posti specifiche, provvedendo un mapping tra gli identificatori dei file e le posizioni. (Esempi: Chord, CAN, Past, Tapestry)
- **Loosely Structured P2P**: si piazza a metà tra structured e unstructured, le posizioni dei file sono effettuate attraverso delle *routing hints*, ma non sono completamente specificate.

Il file sharing è la **killer application** del P2P, in quanto le aree di scambio sono potenzialmente illimitate e lo scambio è totalmente anonimo. P2P può essere utilizzato anche per l'instant messaging, conferenze audio/video, applicazioni condivise come per esempio giochi o powerpoint condivisi.

La computazione distribuita è un altro applicativo importante di P2P data la grande scalabilità ottenuta combinando un gran numero di PC.

P2P può anche essere utilizzato per i database distribuendo i nodi che contengono i dati.

La ricerca in una rete P2P deve risolvere diversi problemi, tra cui un possibile **bottleneck** della grandezza di banda, la necessità autonomia a livello di nodo, la robustezza, la scalabilità, l'espressività e la completezza.

### Tassonomie di ricerca

La ricerca può essere effettuata in maniera **blind (cieca)** oppure **informata**.

#### Blind Search

La blind search può essere effettuata con diversi metodi. Per esempio è possibile utilizzare la **BFS con flooding (Gnutella)**, ovvero inoltrando le query a tutti i vicini. Il flooding è ristretto dal TTL dei pacchetti.

E' possibile utilizzare una **BFS modificata**, che utilizza il flooding solo su una porzione dei vicini per ridurre i costi di traffico.

Altri metodi sono la **random walk**, che inoltra la query a un numero casuale di vicini e l'**iterative deepening** che utilizza BFS consecutive con profondità aumentata.

#### Informed Search

La ricerca informata può essere effettuata attraverso un **local index**, utilizzando metodi **proattivi o reattivi**.

Tra i metodi **proattivi** vi sono per esempio il **Neighborhood Signature Search**, dove ogni nodo mantiene un sommario delle keywords mantenute nei peer in un raggio vicino, oppure la **Associative Search** dove i nodi sono raggruppati in “gruppi di interesse” e le query utilizzano il flooding in uno o più gruppi di interesse.

Tra i metodi **reattivi** vi sono la **DRLP (Distributed Resource Location Protocol)** che usa una random walk quando non vi sono informazioni disponibili, oppure l'**intelligent BFS** dove l'instradamento delle query è basato sull'instradamento delle query passate e ogni nodo memorizza in una tabella query-idVicino. In caso di successo, le informazioni di instradamento su tutti i nodi sul percorso inverso vengono aggiornate.

### Approcci DHT (Distributed Index)

Una **Distributed Hash Table** (DHT) è un sistema di archiviazione decentralizzato che fornisce schemi di ricerca e archiviazione simili a una tabella hash, permettendo l'archiviazione di dati sotto forma di coppie chiave-valore. Gestisce i dati distribuendoli su un numero di nodi e implementando uno schema di routing che consente di cercare in modo efficiente il nodo su cui si trova l'elemento della ricerca. Ogni nodo in un DHT è responsabile delle chiavi assegnategli insieme ai valori mappati. Inoltre, ogni nodo memorizza una vista parziale dell'intero sistema distribuito, il quale distribuisce efficacemente le informazioni di instradamento. In base a queste informazioni, la procedura di instradamento attraversa tipicamente diversi nodi, avvicinandosi alla destinazione a ogni hop, fino a quando non viene raggiunto il nodo di destinazione.

Questo tipo di infrastruttura è molto utilizzata per servizi quali sistemi peer-to-peer di file sharing, file system distribuiti, multicast, web caching cooperativo. In un sistema DHT, ad ogni elemento che viene inserito viene assegnato un ID identificativo, un valore univoco dallo spazio degli indirizzi. Questo valore può essere scelto liberamente dall'applicazione, ma spesso è derivato dai dati stessi tramite una funzione hash resistente alle collisioni. Ad esempio, l'ID di un file potrebbe essere il risultato dell'hashing del nome del file o del file binario completo. Pertanto, la DHT memorizzerebbe il file nel nodo responsabile della parte dello spazio degli indirizzi che contiene l'identificatore.

Le DHT hanno le seguenti proprietà:

- **Decentralizzazione:** i nodi formano nell'insieme il sistema di rete senza alcuna autorità centrale.
- **Scalabilità:** il sistema è configurato per il funzionamento anche con migliaia o milioni di nodi connessi.
- **Tolleranza ai guasti:** il sistema è in grado di risultare sempre affidabile, continuamente adoperabile, nonostante quantità anche elevate di nodi che si uniscono, si disconnettono o escono volontariamente dalla rete in ogni momento.

Esistono diversi modi per realizzare la DHT:

- **Consistent hashing:** la rete sovrapposta è un cerchio, ogni nodo ha un id casuale, il successore è un nodo con il prossimo id più alto, e la chiave è mantenuta nel successore più vicino. Ogni nodo mantiene almeno due successori, quando il successore va via, chiede a quello successivo la lista dei suoi successori. Quando un nuovo nodo entra, trova il suo predecessore e il suo successore opportunamente.

- **Chord** utilizza un overlay circolare, con un ID monodimensionale nello spazio di hash. Il range coperto è [previous\_id, own\_id]. Un nodo è raggiungibile da un qualsiasi altro nodo in non più di  $\log_2(N)$  hops nell'overlay.

- **CAN** utilizza un hypercube come geometria di routing, un valore hash è un punto nel piano cartesiano a D dimensioni, e ogni nodo è responsabile di un cubo D-dimensionale. I vicini sono i nodi che toccano in più di un punto. L'aggiunta di un nuovo nodo utilizza un procedura ricorsiva per trovare i vicini, rappresentando il cubo come un albero.

- **Tapestry** utilizza una routing mesh dove i nodi adiacenti condividono un prefisso ID.

- **Pastry** è simile a Tapestry con alcune differenze sul processo di routing.

- **Kademlia** distribuisce gli indici tra i nodi che sono trattati come foglie di un albero binario. Il routing avviene esplorando l'albero in base ai prefissi degli ID.

Il file sharing nelle reti P2P utilizza un sistema basato su reputazione, ovvero in base al numero di file condivisi, ma che soprattutto siano genuini.

La sicurezza nelle reti P2P deve affrontare la presenza di Peer con intenti maliziosi, bisogna quindi provvedere a diversi livelli di sicurezza. La sicurezza dello storage viene affrontata utilizzando dati auto-certificanti, dispersione delle informazioni e SHA.

Il routing sicuro si basa su tre primitive: **assegnazione sicura degli ID ai nodi, manutenzione sicura delle tabelle di routing, e inoltro sicuro dei messaggi**.

### Casi di studio:

#### NAPSTER:

## Napster

- Hybrid decentralized, unstructured
  - Combination of client/server and P2P approaches
  - A network of registered users running a client software, and a central directory server
  - The server maintains 3 tables:
    - (File\_Index, File\_Metadata)
    - (User\_ID, User\_Info)
    - (User\_ID, File\_Index)
- 
- The diagram illustrates the Napster architecture. At the top is a purple circular icon representing the 'Napster Server'. Below it are two green square icons representing 'Client A' and 'Client B'. A green arrow points from Client A to the server, labeled 'register (user, files)'. From the server, a blue arrow points down to Client A, labeled 'A has X.mp3'. Another blue arrow points from the server to Client B, labeled 'Where is X.mp3?'. From Client A, a pink arrow points down to Client B, labeled 'Download X.mp3'.

## GNUTELLA:

### Gnutella

- ❑ Pure decentralized, unstructured
- ❑ Characteristic:
  - Few nodes with high connectivity.
  - Most nodes with sparse connectivity.
- ❑ Goal: distributed and anonymous file sharing
- ❑ Each application instance (node) :
  - stores/serves files
  - routes queries to its neighbors
  - responds to request queries

## BitTorrent

BitTorrent è un protocollo di condivisione di file peer-to-peer (P2P) che consente agli utenti di scaricare e condividere file in modo efficiente su Internet. Ecco come funziona il protocollo BitTorrent:

1. **Creazione del File Torrent:** Per condividere un file utilizzando BitTorrent, un utente deve creare un file di metadati chiamato "file torrent". Questo file contiene informazioni sul file da condividere, inclusi il nome, le dimensioni, il percorso e una lista di chunk (blocchi) che compongono il file.
2. **Tracker:** Il file torrent contiene anche l'indirizzo di almeno un tracker. Il tracker è un server che aiuta a coordinare le comunicazioni tra i client BitTorrent. Quando un utente avvia il download di un file torrent, il client BitTorrent contatta il tracker per ottenere informazioni su altri peer (utenti) che stanno condividendo o scaricando lo stesso file.
3. **Peers e Connessioni:** Una volta ottenute le informazioni dal tracker, il client BitTorrent stabilisce connessioni con altri peer che stanno condividendo lo stesso file. Ogni peer può essere sia un "seed" (seme), che ha già scaricato tutto il file e lo sta condividendo, sia un "leecher" (parassita), che sta ancora scaricando parti del file.
4. **Scambio di Chunk:** Piuttosto che scaricare il file intero da un singolo peer, BitTorrent divide il file in piccoli chunk. I peer condividono e scambiano questi chunk tra loro in parallelo. Questo approccio consente di utilizzare in modo efficiente la larghezza di banda disponibile e accelera il processo di download.
5. **Rarest First:** BitTorrent utilizza un algoritmo chiamato "rarest first" (il più raro prima) per ottimizzare il processo di download. Invece di scaricare i chunk più comuni, il client BitTorrent priorizza i chunk meno comuni, cioè quelli che sono disponibili in meno peer.

Questo aiuta a bilanciare il carico sui peer e garantire che tutti i chunk del file vengano scaricati il prima possibile.

6. **Seeding:** Dopo aver completato il download di un file, un utente può scegliere di rimanere connesso alla rete e condividere il file completato con altri peer. Questo è chiamato seeding e contribuisce alla salute complessiva della rete BitTorrent.

Tuttavia, è importante notare che BitTorrent è anche noto per la sua natura decentralizzata. Anche se il tracker facilita l'incontro iniziale tra i peer, una volta che sono stati stabiliti i collegamenti, i peer possono comunicare direttamente tra loro per lo scambio effettivo dei dati, senza la necessità di continuare a interagire con il tracker.

Quindi, BitTorrent può essere considerato un ibrido tra un sistema peer-to-peer decentralizzato e un sistema con un elemento centralizzato (il tracker) per facilitare l'avvio e la gestione delle connessioni tra i peer.