

Telecomunicazioni

Introduzione alle reti

Una rete è formata da:

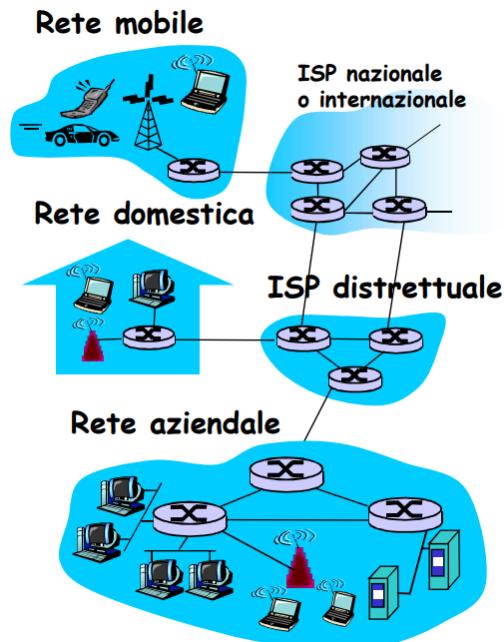
Host => sistema terminale (calcolatore)

Router => nodi che permettono l'instradamento e quindi la creazione di path

Communication link => interfaccia fisica di connessione che permette il trasferimento dei dati, la loro velocità (ampiezza di banda) si misura in bit/secondo (bps)

L'invio e la ricezione delle informazioni è regolata dai protocolli¹ di rete, i due più importanti sono:

- **TCP** (transmission control protocol)
- **IP** (internet protocol)



Internet non è altro che una un'infrastruttura di comunicazione per applicazioni distribuite, ovvero una rete di calcolatori, al centro della quale si trovano gli **ISP (internet service provider)**.

Esistono più livelli di ISP, quelli di livello 1 sono pochi e tutti connessi in peer to peer tra loro creando la rete dorsale (backbone) di internet, hanno una connessione molto veloce e si interfacciano con quelli di livello 2 (generalmente nazionali).

Internet offre 2 tipi di servizi:

- **Servizio connection-oriented** (con connessione) => garantisce la consegna dei dati a destinazione e la loro integrità
- **Servizio "best effort"** (senza connessione) => non affidabile, senza garanzie di consegna dei pacchetti, che possono essere perduti o consegnati fuori sequenza

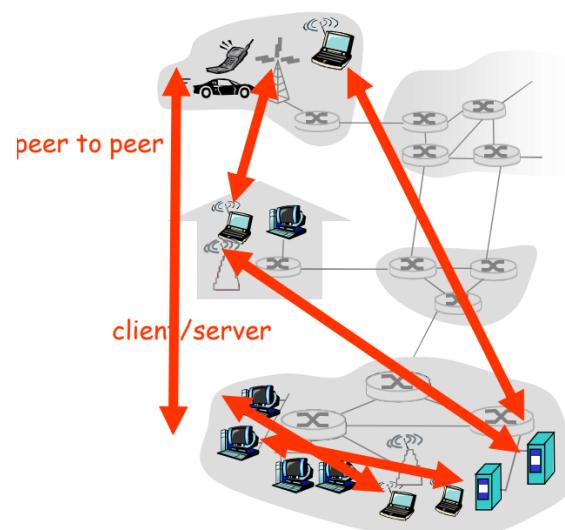
Struttura della rete

Sezione periferica

Costituita dagli End System ovvero gli host che eseguono i programmi, possono essere sia **client** che **server**.

Esistono 2 tipi di architetture host:

- **Client/Server** => il programma client chiede e riceve il servizio da un programma server su un altro terminale
- **Peer to Peer** => gli host comunicano direttamente senza passare per altri server



¹ protocollo = definisce il formato e l'ordine dei messaggi scambiati tra le entità di comunicazione, così come le azioni intraprese in fase di trasmissione e ricezione

Sezione interna

Rete magliata di router che interconnettono i terminali, permettono 2 tipi di trasmissione:

- **Circuit switching** (Commutazione a circuito) => crea un canale riservato che rimane aperto per tutta la durata della connessione, anche se non si stanno trasferendo dati.

Questo permette una velocità di trasmissione costante ma richiede l'impostazione della chiamata.

Le risorse di rete (banda) possono comunque essere utilizzate da più utenti contemporaneamente dividendole in più canali, esistono 2 tipologie di divisione:

- **Mutilazione di frequenza (FDM)**
- **Mutilazione di tempo (TDM)**

ATTENZIONE anche se è permesso il parallelismo tra più utenti questa modalità associa ad ogni utente i suoi slot, quindi anche se uno non sta trasmettendo in un determinato momento la sua fetta rimane occupata e non utilizzabile dagli altri

- **Packet switching** (Commutazione a pacchetto) => utilizzata dalla rete internet, le risorse non vengono riservate ma impegnate su richiesta, i dati vengono divisi in pacchetti ed immessi subito in rete per essere instradati tramite i router.

Permette un numero infinito di utenti in contemporanea ed un utilizzo continuo ed ottimale delle risorse, non avendo canali "prenotati".

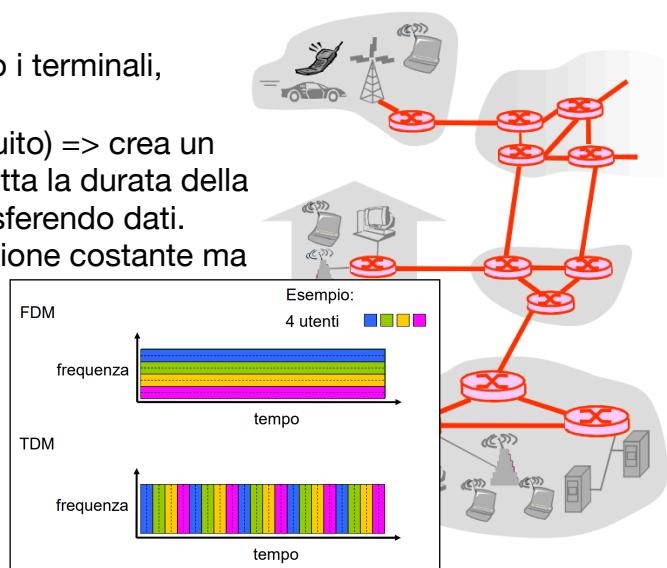
Se troppi pacchetti viaggiano in contemporanea nella rete si può incorrere ad una congestione che porta a rallentamenti rendendo imprevedibile il tempo di trasmissione.

Richiede quindi l'utilizzo di buffer che mantengano i pacchetti fino al loro invio, se la congestione è troppo alta si possono perdere dati.

Molti router inoltre utilizzano la modalità **store-and-forward**, finché non è stato ricevuto tutto il pacchetto questo non può essere rinvia, portando ad un ulteriore ritardo che aumenta all'aumentare dei router utilizzati.

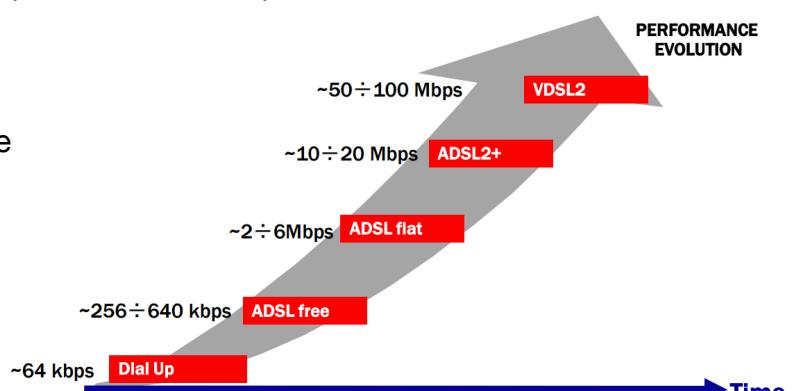
L'invio dei pacchetti non segue uno schema prefissato, si utilizza la

multiplazione statistica, ovvero i dati vengono inviati e ricevuti in maniera casuale per poi essere ricomposti alla fine del processo.



Accesso alla rete

Collegamento tra utente e centrale di commutazione (provider) avviene tramite il **modem**², nel tempo la tecnologia si è evoluta passando dai Dial Up, che non consentivano di chiamare e navigare, fino ai VDSL2 di adesso.



² modulatore-demodulatore => permette di trasformare segnale digitale in analogico e viceversa.

Mezzi trasmissivi

Indica ciò che si trova tra trasmittente e ricevente, principalmente di 2 tipi:

- Mezzi trasmissivi guidati => onde guidate tramite mezzo solido, ovvero cavi.
Possono essere: twisted pair, coassiali e fibre ottiche
- Mezzi trasmissivi non guidati => le onde si propagano attraverso l'atmosfera, ovvero canali radio.
Possono essere: satellitari, wifi, ...

Durante il loro percorso i dati possono passare per diverse tecnologie di mezzi

Ritardi nell'invio dei pacchetti

Se il tasso di arrivo dei pacchetti eccede la capacità di collegamento i pacchetti si accodano nel buffer del router in attesa del proprio turno, se il buffer non è abbastanza capiente si può avere anche una perdita di pacchetti.

Le cause che determinano il ritardo:

1. Ritardo di **elaborazione** nodo => il tempo che il nodo impiega per controllare gli errori e reindirizzare il pacchetto, dipende dal nodo
2. Ritardo di **accodamento** => dipende dall'intensità di traffico, calcolabile come La/R con a il tasso medio di arrivo dei pacchetti, se questo fattore raggiunge 1 si ha ritardo infinito poiché arrivano più pacchetti di quanti si riesce a servirne
3. Ritardo di **trasmissione** => tempo impiegato per trasferire il pacchetto, si calcola come L/R
4. Ritardo di **propagazione** => tempo impiegato per raggiungere la destinazione, si calcola come d/s

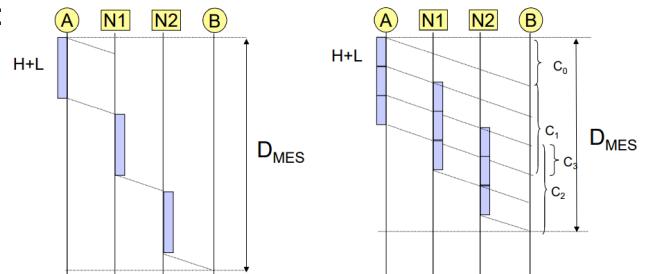
Quindi il ritardo di link teorico totale è dato da:

$$d_{link} = d_{elab} + d_{queue} + d_{trasm} + d_{prop}$$

OSS l'utilizzo dei pacchetti rende possibile la parallelizzazione dei processi di trasferimento passando per più router, poiché come arriva un pacchetto è possibile inviare il prossimo.

Legenda

- N** = numero nodi
- R** (o **C**) = bit rate (bit/s)
- H** = lunghezza intestazione
- L** = lunghezza testo pacchetto
- X** = numero bit messaggio completo
- d** = distanza
- s** = velocità



Nella realtà il calcolo diventa quindi più complesso, nel caso in cui il testo dei pacchetti sia sempre di dimensione costante, si usa la formula:

C_0 =ritardo di propagazione complessivo sulle N interfacce

C_1 =ritardo di trasmissione del primo pacchetto attraverso tutte le N interfacce

C_2 =ritardo di trasmissione su una interfaccia di tutti i pacchetti necessari a trasferire il messaggio

C_3 =ritardo di trasmissione del primo pacchetto attraverso una interfaccia

$$D_{MES} = d_{prop,AB} + \frac{H+L}{C} \cdot N + \frac{1}{C} \left\{ \left\lceil \frac{X}{L} \right\rceil \cdot (H+L) \right\} - \frac{H+L}{C}$$

ATTENZIONE la d_{prop} deve essere moltiplicata per il numero di nodi N
Nel caso in cui il testo dei pacchetti sia di dimensione variabile, si usa:

$$D_{MES} = d_{prop, AB} + \frac{H + L_{max}}{C} \cdot N + \frac{1}{C} \left\{ X + \left\lceil \frac{X}{L_{max}} \right\rceil \cdot H \right\} - \frac{H + L_{max}}{C}$$

La scelta della dimensione dei pacchetti è influenzata da 2 fattori:

- Al crescere di L_{max} diminuisce l'effetto pipeline (parallelismo)
- Al diminuire di L_{max} cresce il peso dell'intestazione H

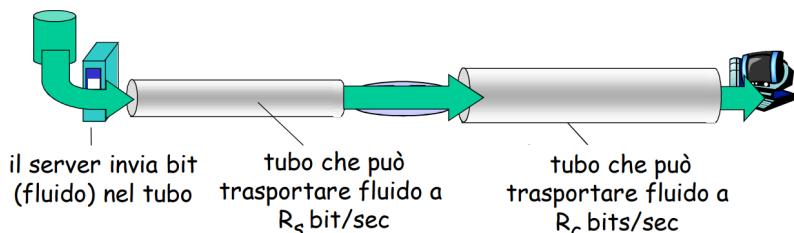
Quindi per decidere la grandezza dei pacchetti si usa la formula:

$$L_{max} = \sqrt{\frac{H \cdot X}{N - 1}}$$

Throughput

Frequenza (bit / unità di tempo) alla quale i bit sono trasmessi da mittente a ricevente, indica la capacità di trasmissione effettiva.

Spesso accade che si formi un **bottleneck**, ovvero ci sia una parte del collegamento molto più lenta che rallenta tutta la connessione

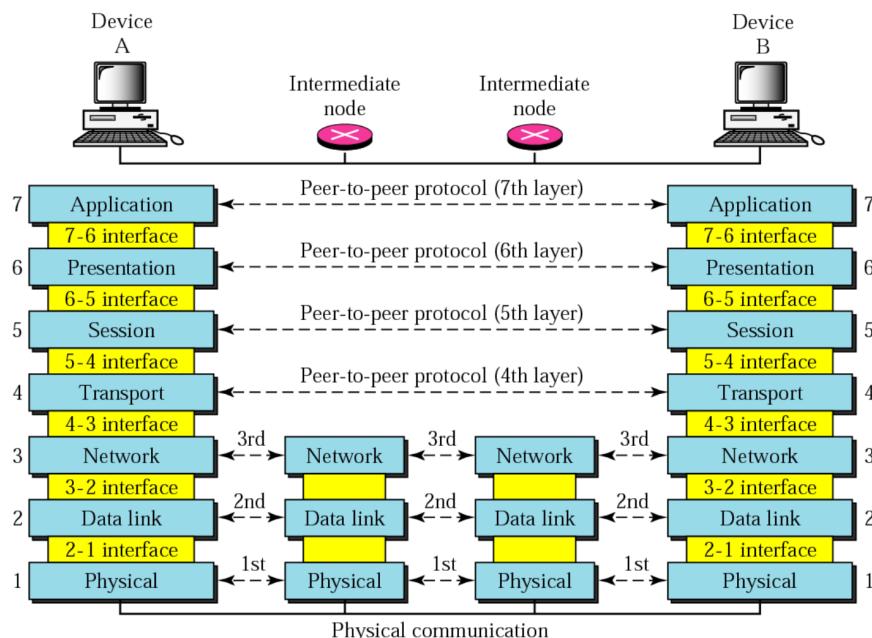


Architettura a strati (OSI) e TCP/IP

L'architettura a strati è l'organizzazione gerarchica di una rete, questa è infatti composta da **layers** (strati), ognuno dei quali con i suoi protocolli.

La struttura protocollare è preferita a quella monolitica poiché l'indipendenza degli strati la rende modulare favorendo la flessibilità di manutenzione ed aggiornamenti del sistema. La modularità può però portare a ridondanze nelle funzioni, ovvero più moduli eseguono la stessa operazione, inoltre si può avere che un modulo non possa portare a termine il lavoro poiché le informazioni di cui necessita sono disponibili solo in un altro.

Il modello **OSI (open System interconnection)** descrive un modello standard per la divisione degli strati di un architettura di rete, in realtà è stato ormai superato dal modello **TCP/IP**, che incorpora gli strati di presentazione e sessione nell'application layer.

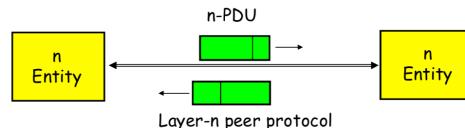


Applicativo	<ul style="list-style-type: none"> Forniscono supporto ad applicazioni di rete Consentono ad applicazioni di interpretare i dati ricevuti Sincronizzano e controllano dialogo
Presentazione	
Sessione	
Trasporto	<p>Trasferisce dati end-to-end tra gli host in modo affidabile</p> <ul style="list-style-type: none"> Controlla che tutti i segmenti siano arrivati Si occupa di frammentazione e deframmentazione dei dati Si occupa del port number per fornire i dati alla giusta applicazione
Rete	<p>Si occupa del trasferimento dei pacchetti tramite i link</p> <ul style="list-style-type: none"> Indirizzamento dei pacchetti Definisce procedure di routing (instradamento) e forwarding (rilancio dei pacchetti) Definisce funzioni di controllo della congestione
Collegamento	<p>Realizza trasferimento affidabile delle informazioni nel link</p> <ul style="list-style-type: none"> Forma i frame da inviare Rivela e corregge errori nei frame ricevuti Nel caso di reti condivise regola l'accesso Controllo del flusso
Fisico	Trasferisce i segnali che rappresentano i bit e definisce le caratteristiche fisiche del link di comunicazione

Comunicazione tra strati ed entità

Le entità che seguono le funzioni su uno stesso strato sono dette **peer processes**. Ogni strato ha le sue funzioni e comunica solo con gli strati adiacenti, la comunicazione è regolata dal **layer-n protocol** (protocollo di strato), ed avviene tramite le **PDU (protocol data unit)**.

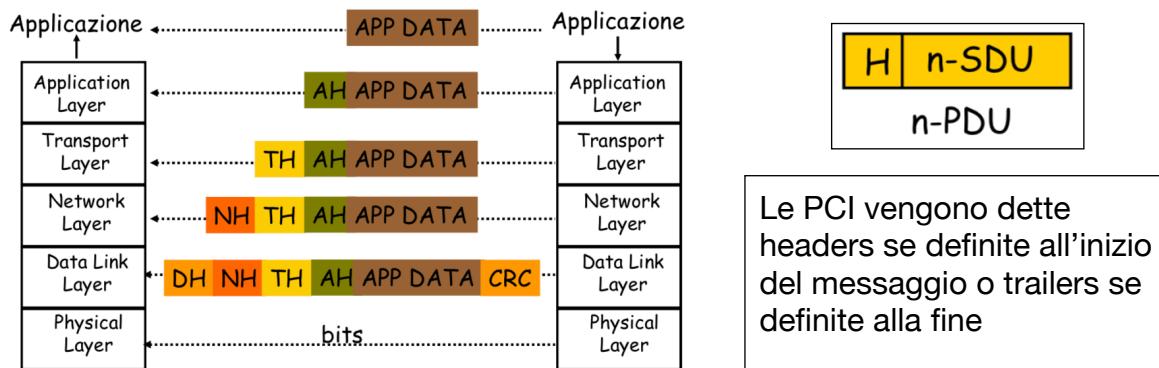
L'insieme dei protocolli è detto **protocol stack**.



Il passaggio dei dati è scandito da:

- Lo strato $n+1$ trasferisce le proprie informazioni allo strato n tramite il **SAP (service access point)**, le informazioni contenute nelle PDU ricevute dallo strato n vengono chiamate **SDU (service data unit)**
- Lo strato n incapsula le SDU in PDU aggiungendo le informazioni di controllo per l'esecuzione delle funzioni di strato **PCI³ (protocol control information)**
- Ogni strato comunica a quello inferiore le operazioni da seguire fino a raggiungere lo strato fisico che esegue il passaggio effettivo dei dati.

In caso di multiplexing e quindi condivisione del servizio tra più utenti bisogna utilizzare delle etichette in ogni PDU per indicare a quale utente appartiene la SDU.



Anche qui l'interazione può avvenire con:

- Servizio con connessione => usato da TCP e telefonia, è diviso in 3 tempi, inoltre richiede la negoziazione dei parametri di trasferimento e l'indirizzamento con identificatori di connessione.
Crea legame logico tra i segmenti informativi scambiati
- Servizio senza connessione => avviene in un unico tempo e non ha bisogno di negoziazione, utilizza indirizzi esplicativi per origine e destinazione.
Non si ha sicurezza su arrivo dati, utilizzato da IP e UDP

	OSI Layer	TCP/IP	Datagrams are called
Software	Layer 7 Application	HTTP, SMTP, IMAP, SNMP, POP3, FTP	Upper Layer Data
	Layer 6 Presentation	ASCII Characters, MPEG, SSL, TSL, Compression (Encryption & Decryption)	
	Layer 5 Session	NetBIOS, SAP, Handshaking connection	
	Layer 4 Transport	TCP, UDP	Segment
	Layer 3 Network	IPv4, IPv6, ICMP, <u>IPSec</u> , MPLS, ARP	Packet
Hardware	Layer 2 Data Link	Ethernet, 802.1x, PPP, ATM, <u>Fiber</u> Channel, MPLS, FDDI, MAC Addresses	Frame
	Layer 1 Physical	Cables, Connectors, Hubs (DLS, RS232, 10BaseT, 100BaseTX, ISDN, T1)	Bits

³ possono contenere indirizzi, codici di controllo errori, flag, ecc.

Lo strato fisico

Le informazioni possono essere divise in 2 tipologie:

- Informazioni a blocchi => strutturate in unità indipendenti la cui dimensione è data dal numero di bit per blocco, come messaggi di testo, immagini, ecc..
- Informazioni stream => prodotte e trasmesse in modo continuo, in questo caso si considera il **bit rate**, ovvero la quantità di bit prodotti dalla sorgente in un'unità di tempo

Il **delay di trasferimento** di un blocco è dato da:

$$\text{delay minimo} = t_{\text{prop}} + t_{\text{trasmissione}} = \frac{d}{c} + \frac{L}{R}$$

L si riduce mediante tecniche di compressione

R si aumenta mediante tecniche di trasmissione

Legenda

R (o **C**) = bit rate (bit/s)
L = numero bit messaggio
d = lunghezza collegamento
c = velocità trasmissione del mezzo

La compressione serve per ridurre il numero di bit necessari alla rappresentazione delle informazioni riducendo le ridondanze in esse contenute, di 2 tipi:

- **Lossless** (senza perdita)
- **Lossy** (con perdita)

Il **compression ratio** (rapporto di compressione) è dato da

$$R_c = B_{\text{orig}} / B_{\text{compr}}$$

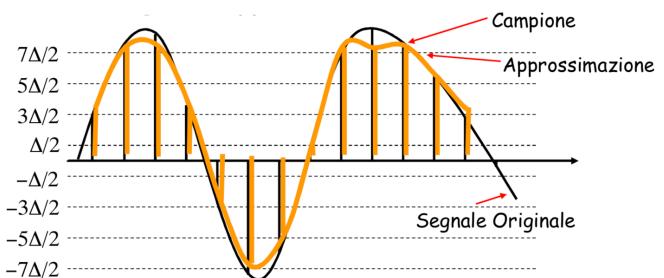
Tipo	Metodo	Formato	Originale	Compressed Ratio
Text	Zip	ASCII	Kbyte-Mbyte	$2 < R_c < 6$
Fax	CCITT Group 3	A4 page 200x100 pixel/in ²	256 kbyte	5-54 kbyte ($5 < R_c < 50$)
Immagine a Colori	JPEG	8x10 in ² photo 400 ² pixel/in ²	38.4 Mbyte	1-8 Mbyte ($5 < R_c < 30$)

Nel caso degli stream si introduce il **sampling** (campionamento) del segnale analogico che associato alla **quantizzazione** (approssimazione) ne permette la digitalizzazione.

Il numero di bit dipende dal numero di livelli in cui si decide di dividere l'asse verticale.

Il bit rate è dato da

$$\text{bit rate} = \frac{n^{\circ}\text{bit}}{\text{campioni}} * \frac{n^{\circ}\text{ campioni}}{\text{secondo}}$$



Il **bandwidth Ws** (larghezza di banda) è dato dall'intervallo delle frequenze di cui è composto il suo spettro ovvero quanto "velocemente" il segnale può variare nel tempo, maggiore è il valore più frequenti dovranno essere i campioni.

Secondo il teorema del campionamento, il tempo di campionamento deve essere

$$T_c = \frac{1}{F_c} \quad \text{con} \quad F_c = 2 Ws \quad (\text{frequenza di campionamento})$$

Se il valore è inferiore il segnale non viene riprodotto fedelmente, se è maggiore non si hanno cambiamenti nella percezione.

Per l'uomo la Ws va da 0 a 4KHz quindi la Fc = 1/8 KHz = 125 micro secondi

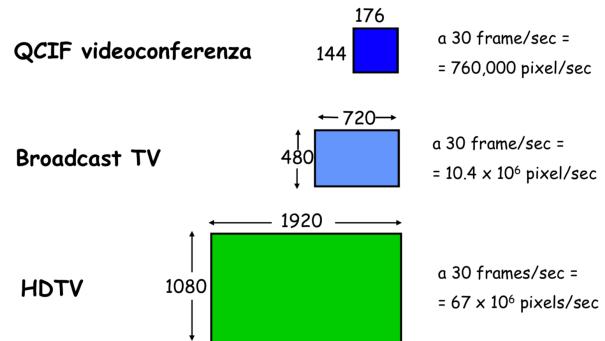
I segnali audio vengono generalmente compressi risparmiando sulla codifica delle pause oppure utilizzando un sintetizzatore in uscita ed inviando solo i bit essenziali per la corretta riproduzione.

Per i segnali video si usa il **picture frame**, ovvero una sequenza di immagini digitalizzate e compresse, la frequenza di ripetizione va tra 10 e 60 frame/secondo.

$$rate = M \frac{bits}{pixel} * (W * H) \frac{pixel}{frame} * F \frac{frame}{second}$$

H e W indicano le dimensioni dell'immagine

Tipo	Metodo	Formato	Originale	Compresso
Video Conferenza	H.261	176x144 or 352x288 pix a 10-30 fr/sec	2-36 Mbit/s	64-1544 kbit/s
Full Motion	MPEG2	720x480 pix a 30 fr/sec	249 Mbit/s	2-6 Mbit/s
HDTV	MPEG2	1920x1080 a 30 fr/sec	1.6 Gbit/s	19-38 Mbit/s



Una volta digitalizzata l'informazione stream si possono avere 2 tipologie di bit-rate:

- **Costant bit-rate** => si emette sempre la stessa quantità di bit ogni secondo, questo implica che la rete deve fornire un canale di comunicazione con banda almeno uguale al bit-rate della sorgente
- **Variabile bit-rate** => usato in alcuni video, il bit-rate varia durante la riproduzione OSS con **jitter** si indica la variabilità del ritardo

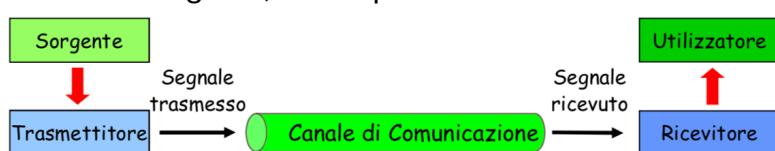
Trasmissione numerica

Nello schema di un sistema di trasmissione sono sempre presenti:

- **Trasmettitore** => converte il flusso informativo prodotto dalla sorgente in un segnale adatto alla trasmissione
- **Ricevitore** => converte un segnale ricevuto in una forma leggibile dal destinatario

Il trasferimento porta a delle alterazioni nel segnale, tra le quali:

- Attenuazione del segnale
- Distorsione del segnale
- Rumore additivo
- Interferenza con altri segnali



Per le trasmissioni si possono usare sia **segnale analogici** che **segnali digitali**.

I segnali digitali sono generalmente più vantaggiosi poiché risentono meno delle alterazioni, non è infatti necessario ricostruire tutto il segnale ma solo capire ad ogni istante se il segnale in ingresso è positivo o negativo.

I segnali analogici hanno bisogno di **ripetitori**, i quali si occupano di rigenerare il segnale in uscita in modo che sia il più possibile simile a quello in ingresso, la rigenerazione però non è ideale e la qualità del segnale diminuisce aumentando i ripetitori.

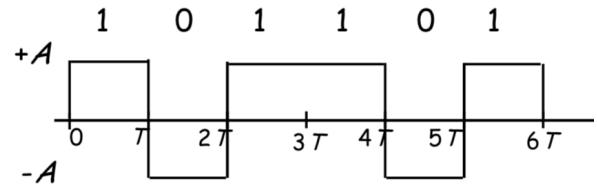
Le comunicazioni analogiche sono quindi distance-limited.

Nel caso dei segnali digitali si necessita dei **rigeneratori**, i quali ricostruiscono la sequenza iniziale di bit per inviarla nuovamente nella tratta successiva, in questo caso il segnale rigenerato è praticamente identico a quello originale.

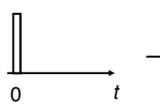
Quindi le trasmissioni numeriche (digitali) non risentono della distanza, inoltre permettono di implementare più funzioni ed hanno costi ridotti.

Le trasmissioni numeriche vengono codificate riportando le ampiezze sull'asse delle ordinate.

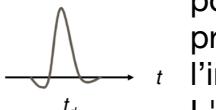
$$\text{bit rate} = \frac{1 \text{ bit}}{T \text{ secondi}}$$



Un modo per aumentare il bit rate consiste nel diminuire il tempo T , in canali ideali lo si



Canale



potrebbe far tendere a 0 ma in canali reali questo provocherebbe alterazioni nel segnale con l'impulso in uscita allungato e arrotondato. L'alterazione porta a 2 tipi di errori:

- **Interferenza intersimbolica** => le code finali di 2 simboli possono sommarsi creando una nuova curva diversa dall'originale
- **Interferenza di decisione** => non è possibile valutare il segnale perfettamente al centro di T e perciò l'arrotondamento dell'onda incrementa la probabilità d'errore, si rischia di prendere il campione quando la curva scende sotto asse.

Per evitare questi problemi bisogna mantenere la frequenza di trasmissione degli impulsi a $F = 2 / T$ detta frequenza di Nyquist.

Altrimenti per aumentare la fedeltà del segnale è possibile usare la **codifica multilivello**, ovvero aumentare il numero di bit che rappresentano ogni istante del segnale.

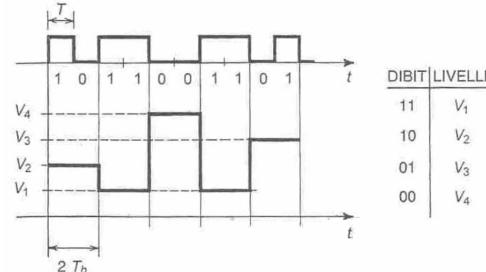
Questo si fa aumentando i livelli presenti nell'asse

verticale, con M livelli si ha 2^M e quindi:

$$\text{bit rate} = \frac{m \text{ bit}}{T \text{ secondi}}$$

Aumentando il numero di livelli diminuisce la loro distanza, il che può portare a problemi decisionali ed aumentare il **BER (bit-error-rate)**, soprattutto in caso di sovrapposizione con rumore additivo.

$$BER = \frac{\text{bit corretti}}{\text{bit totali}}$$



Per studiare la qualità della comunicazione si introduce il **SNR (signal-to-noise-ratio)**, più alto è questo parametro migliore è la qualità del segnale.

$$SNR = \frac{\text{potenza media del segnale}}{\text{potenza media del rumore}} \quad SNR(dB) = 10 \log_{10} SNR$$

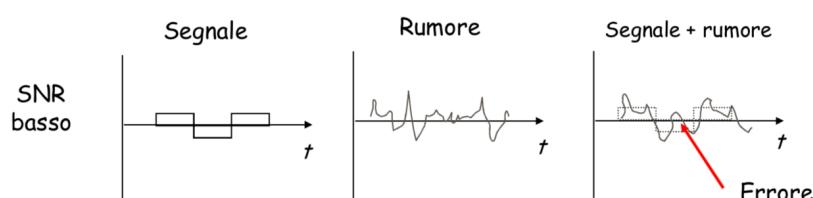
Il limite della capacità di un canale è dato dal

Limite di Shannon:

$$C = W_c \log_2 (1 + SNR) \text{ bit/s}$$



Quindi con $C > R$ il BER sarà basso, con $C < R$ alto.



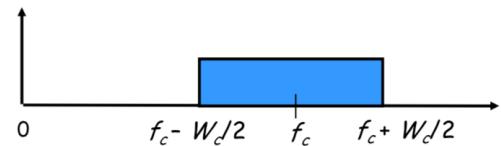
Banda del canale

Per canale di comunicazione si intende l'insieme dei mezzi trasmittivi e dei dispositivi in cui passa il segnale che parte dalla sorgente ed arriva a destinazione, si utilizza invece il termine **filtro** per indicare gli effetti del canale sul segnale.

Un **filtro passa-basso** fa passare inalterate tutte le frequenze $f < W_c$, bloccando le $f > W_c$.

Un **filtro passa-banda** permette il passaggio solo alle frequenze che rientrano nella larghezza di banda del segnale, bloccando frequenze troppo basse o alte.

La **frequenza portante** f_c indica il centro della larghezza di banda su cui lavora il filtro, molto utile nel caso di **frequency division multiplexing**, che associa una frequenza portante ad ogni utente.



Tutti i segnali hanno una rappresentazione in frequenza detta **rappresentazione di Fourier**, ovvero una raffigurazione del segnale come composizione di sinusoidi, tra cui la frequenza portante.

La sinusoide principale è la rappresentazione in frequenza della sua onda quadra alla quale si aggiungono altre componenti con frequenze diverse dalla sua.

Queste ultime hanno coefficienti decrescenti, che indicano un decrescere dell'ampiezza delle curve.

La potenza del segnale è la somma delle ampiezze, modulo al quadrato.

Lo spettro in frequenza del segnale è il set delle ampiezze delle componenti infinite della rappresentazione di Fourier.

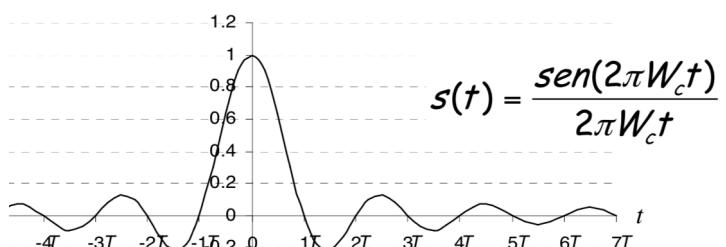
La banda del segnale è un sottoinsieme dello spettro, in cui vengono escluse quelle frequenze di cui le sinusoidi corrispondenti non alterano significativamente la curva.

Impulsi ad interferenza intersimbolica nulla

Gli impulsi sono le curve dipendenti da $t(f(t))$ su cui si trasportano i bit da inviare. Inviando un impulso questo potrebbe deformarsi ma soprattutto potrebbe confondersi con le code di quelli successivi e precedenti così da creare interferenza intersimbolica. Un modo per rendere questo fenomeno nullo è quello di spedire un **segnale modulato**⁴

chiamato sync, del tipo $\frac{\sin(x)}{x}$.

Con questo tipo di segnale le code interferiscono solo in corrispondenza degli zeri, annullandosi.



Questa tecnica non è comunque esente da errori, infatti il ricevente deve essere perfettamente sincronizzato con chi trasmette, poiché se la lettura non avviene negli istanti significativi potrebbe verificarsi ambiguità.

OSS Lo stesso discorso può essere applicato aumentando i bit di ogni impulso, in questo caso la probabilità che fraintenda il livello aumenta aumentando la potenza del segnale.

⁴ L'idea fondamentale della modulazione consiste nell'iniettare il segnale da trasmettere su un altro segnale (portante) che presenti le caratteristiche adatte per la trasmissione sul canale.

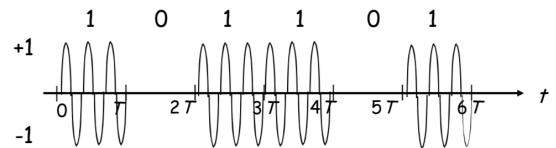
Modulazione numerica

I modulatori numerici (modem) utilizzano forme d'onda con frequenze che sono passanti per una canale passa-banda.

Esistono vari modi per effettuare la modulazione di un segnale, tra questi:

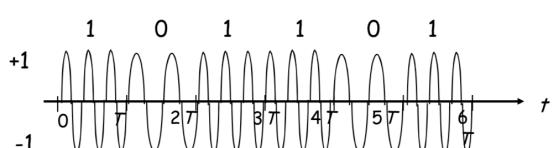
- **ASK (Amplitude Shift Keying)** => mappa ogni bit informativo nell'ampiezza di una sinusoide a frequenza f_c , si hanno le corrispondenze:

1 => trasmissione del segnale
0 => nessuna trasmissione



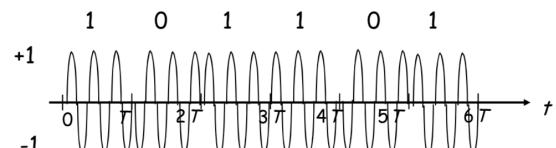
- **FSK (Frequency Shift Keying)** => mappa ogni bit informativo nella frequenza di un segnale sinusoide, si hanno le corrispondenze:

1 => segnale $f_c + \delta$
0 => segnale $f_c - \delta$

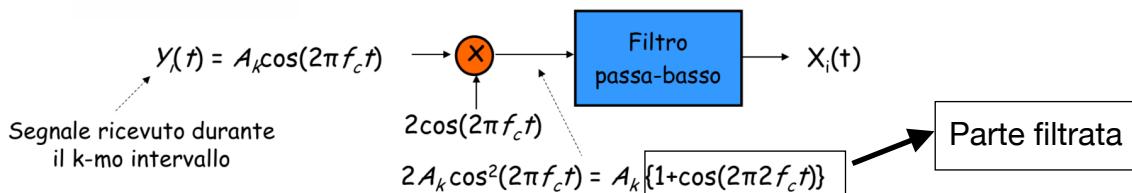


- **PSK (Phase Shift Keying)** => mappa ogni bit informativo nella fase di un segnale sinusoide, si hanno le corrispondenze:

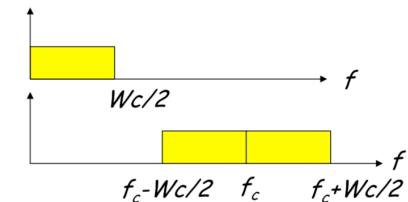
1 => segnale $A \cos(2\pi f_c t) = \text{fase } 0$
0 => segnale $A \cos(2\pi f_c t + \pi) = \text{fase } \pi$
(equivalente con $-A$ invece di $+\pi$)



Per demodulare il segnale basta rimoltiplicarlo per $2A \cos(2\pi f_c t)$ e poi filtrarlo con un filtro passa-basso che esclude le frequenze superiori a $2f_c$.



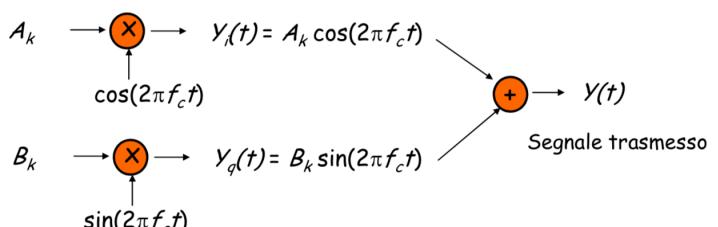
È da notare che usando la PSK non si usa tutta la banda traslata poiché se il segnale in banda base $x(t)$ ha banda $W_c/2$ Hz, il segnale modulato $x(t) \cos(2\pi f_c t)$ ha una banda uguale a W_c Hz.



Si utilizza quindi la **QAM (quadrature amplitude modulation)** che opera su 2 dimensioni.

Il segnale viene diviso in 2 componenti:

- A_k che opera in fase (cos)
- B_k che opera in quadratura (sin)



I quali vengono poi spediti contemporaneamente e riassemblati dal

demodulatore, il cui filtro questa volta deve tagliare tutte le frequenze superiori a $4f_c$. Questo è possibile poiché seno e coseno sono curve ortogonali ed in quadratura.

Ad esempio un segnale 110101 diventa:

$$A_k = 110 \Rightarrow A_k \cos(2\pi f_c t)$$

$$B_k = 101 \Rightarrow B_k \sin(2\pi f_c t)$$

La cui composizione è $y(t) = A_k \cos(2\pi f_c t) + B_k \sin(2\pi f_c t)$

Il concetto di quadratura può essere esteso in modo da avere costellazioni di segnali aumentando il numero di sfasamenti che si considerano.

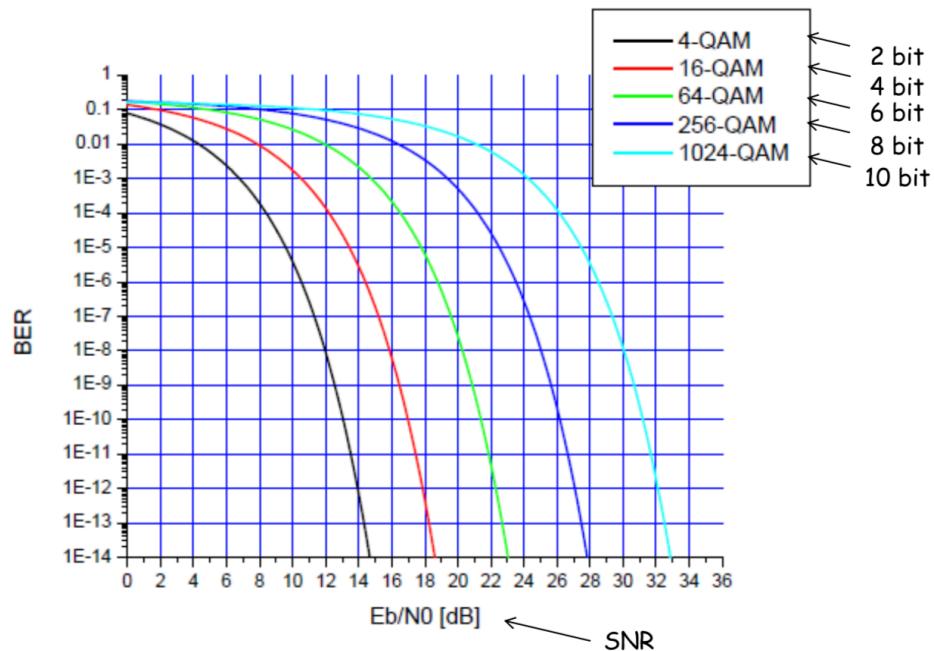
La costellazione di un segnale è l'insieme dei punti che può assumere un segnale avendo che ogni coppia (A_k, B_k) definisce un punto sul piano.

Ad ogni punto corrisponde una sequenza di bit, la cui lunghezza dipende dal tipo di QAM.

4-QAM => 4 possibili punti (bit) ogni T secondi $(0^\circ, 90^\circ, 180^\circ \text{ e } 270^\circ)$	16-QAM => 16 possibili punti (bit) ogni T secondi $(0^\circ, 45^\circ, 90^\circ, 135^\circ, 180^\circ, \dots)$

La modulazione QAM non può essere fatta in banda base.

Il numero di livelli che si possono creare non è infinito ma limitato dal rumore additivo, il quale può far ricadere il decisore del ricevitore su una regione diversa da quella corrispondente al segnale inviato in origine



Lo strato di collegamento

Per data link si intende tutto ciò che trasporta l'informazione a destinazione. I protocolli di questo strato si occupano del trasporto dei pacchetti lungo un singolo canale di comunicazione.

I servizi offerti da questo strato sono:

- **Framing** => si occupa di formare le PDU di strato (ovvero i **frames**), encapsulando i pacchetti ricevuti dallo strato superiore
 - **MAC (medium access control)** => protocollo necessario per trasmissioni **broadcast** (ovvero nel caso in cui siano presenti più terminali), identifica il destinatario corretto
 - **Controllo di flusso** => controlla l'invio dei dati in modo da non saturare il ricevente
 - **Consegna affidabile** => si occupa della ritrasmissione dei dati per evitare perdite (necessaria nelle connessioni senza filo)
 - **Rivelazione e correzione errori** => gli errori sono causati dal transito nel mezzo, possono essere di vari tipi
 - Ricevo “1” invece che “0”, o viceversa
 - Ricevo meno bit di quelli richiesti
 - Ricevo tutti i bit in disordine

Lo strato di collegamento è una combinazione di hardware, software e firmware. Il componente che se ne occupa è la scheda di rete **NIC (network interface card)**.

Framing

Si occupa di formare le PDU di strato (ovvero i **frames**), incapsulando i pacchetti ricevuti dallo strato superiore.

L'unità ricevente deve essere in grado di riconoscere senza ambiguità l'inizio e la fine di ogni frame, per far questo vi si aggiunge un **flag** all'inizio ed alla fine.

Essendo un flag una sequenza di bit, potrebbe essere confuso per una parte delle informazioni, si usa quindi una tecnica chiamata **bit stuffing**.

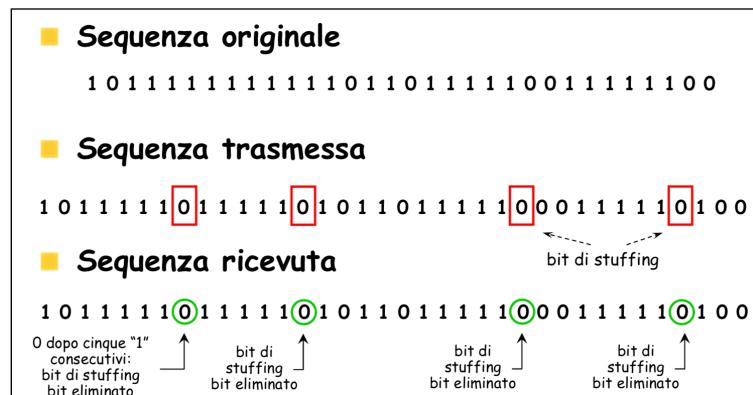
Il bit stuffing è un bit di controllo inserito dopo una “sequenza sospetta”, un esempio di utilizzo può essere uno “0” inserito dopo una sequenza di 5 “1” :

0111110

Si ha quindi una procedura del tipo:

1. In emissione, prima di trasmettere una sequenza di bit, inserisco uno "0" ogni 5 "1" consecutivi contenuti nella sequenza
 2. In ricezione, si effettua il bit destuffing controllando tutte le sequenze sospette:
 - Se alla fine della sequenza si trova uno "0", questo è un bit stuffing, quindi si elimina e si continua la lettura
 - Se alla fine della sequenza si trova un "1", la sequenza è un flag, frame terminato

OSS il PPP (point to point protocol) usa un intero byte come bit stuffing, se la stessa sequenza dovesse essere già contenuta nel messaggio se ne aggiunge un altro in modo da eliminarne solo una.



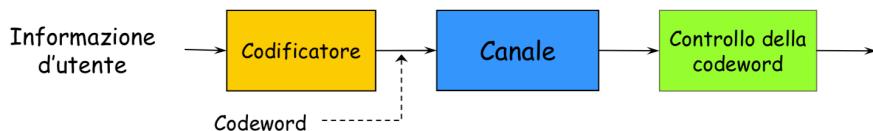
Rivelazione e correzione errori

La trasmissione dei dati introduce errori, esistono quindi 2 approcci:

- **ARQ (error detection & retransmission)** => rivelazione dell'errore e ritrasmissione
- **FEC (forward error correction)** => rivelazione e correzione errore

Alla base del controllo si ha l'organizzazione dei blocchi **codeword** (parole di codice), se il blocco ricevuto non è codeword allora è considerato errore.

Si necessita inoltre di un **overhead** di dati da aggiungere per il controllo.



Un primo metodo per effettuare il controllo è il **parity check** (controllo di parità).

Ne esistono di più tipi:

- Parità singola => data una sequenza di k bit si crea una sequenza di $k+1$ bit, dove il bit aggiuntivo è uguale ad "1" se gli "1" della sequenza sono dispari, "0" se sono pari.

Questo permette di rivelare solo gli errori in cui un numero dispari di bit sono stati scambiati, inoltre non permette di capire la posizione e quindi correggere l'errore

- | |
|--|
| ■ Bit informativi (7 bit): (0, 1, 0, 1, 1, 0, 0) |
| ■ Bit di parità: $b_8 = 0 + 1 + 0 + 1 + 1 + 0 + 0 = 1$ |
| ■ Codeword (8 bit): (0, 1, 0, 1, 1, 0, 0, 1) |

- Parità bi-dimensionale => stessa idea ma si aggiunge una nuova stringa di bit di parità alla fine, il tutto viene poi disposto in colonne in modo da rilevare le combinazioni che riportano errori.

Permette di rilevare fino a 3 errori ma l'overhead elevato non lo rende comunque un metodo ottimale, con 1 errore è possibile anche correzione

<table border="1"> <tr><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td></tr> </table>	1	0	0	1	0	0	0	1	0	0	0	1	1	0	0	1	0	0	1	1	0	1	1	1	1	0	0	1	1	1	1	0	0	1	1	1	<table border="1"> <tr><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td></tr> </table>	1	0	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	0	1	1	0	1	1	0	1	0	0	1	1	1	1	0	0	1	1	1	<table border="1"> <tr><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td></tr> </table>	1	0	0	1	0	0	0	0	0	1	0	1	1	0	0	1	0	0	1	0	0	0	1	0	1	0	0	1	1	0	1	0	0	1	1	1	<table border="1"> <tr><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td></tr> </table>	1	0	0	1	0	0	0	0	0	1	0	1	1	0	0	1	0	0	1	0	0	0	1	0	1	0	0	1	1	1	1	0	0	1	1	1
1	0	0	1	0	0																																																																																																																																														
0	1	0	0	0	1																																																																																																																																														
1	0	0	1	0	0																																																																																																																																														
1	1	0	1	1	1																																																																																																																																														
1	0	0	1	1	1																																																																																																																																														
1	0	0	1	1	1																																																																																																																																														
1	0	0	1	0	0																																																																																																																																														
0	0	0	0	0	1																																																																																																																																														
1	0	0	1	0	0																																																																																																																																														
1	1	0	1	1	0																																																																																																																																														
1	0	0	1	1	1																																																																																																																																														
1	0	0	1	1	1																																																																																																																																														
1	0	0	1	0	0																																																																																																																																														
0	0	0	1	0	1																																																																																																																																														
1	0	0	1	0	0																																																																																																																																														
1	0	0	0	1	0																																																																																																																																														
1	0	0	1	1	0																																																																																																																																														
1	0	0	1	1	1																																																																																																																																														
1	0	0	1	0	0																																																																																																																																														
0	0	0	1	0	1																																																																																																																																														
1	0	0	1	0	0																																																																																																																																														
1	0	0	0	1	0																																																																																																																																														
1	0	0	1	1	1																																																																																																																																														
1	0	0	1	1	1																																																																																																																																														
<table border="1"> <tr><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td></tr> </table>	1	0	0	1	1	1	1	0	0	1	1	1	<table border="1"> <tr><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td></tr> </table>	1	0	0	1	0	0	0	0	0	0	1	1	<table border="1"> <tr><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td></tr> </table>	1	0	0	1	0	0	0	0	0	1	0	1	1	0	0	1	0	0	1	0	0	0	1	0	1	0	0	1	1	1	1	0	0	1	1	1	<table border="1"> <tr><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td></tr> </table>	1	0	0	1	0	0	0	0	0	1	0	1	1	0	0	1	0	0	1	0	0	0	1	0	1	0	0	1	1	1	1	0	0	1	1	1																																																
1	0	0	1	1	1																																																																																																																																														
1	0	0	1	1	1																																																																																																																																														
1	0	0	1	0	0																																																																																																																																														
0	0	0	0	1	1																																																																																																																																														
1	0	0	1	0	0																																																																																																																																														
0	0	0	1	0	1																																																																																																																																														
1	0	0	1	0	0																																																																																																																																														
1	0	0	0	1	0																																																																																																																																														
1	0	0	1	1	1																																																																																																																																														
1	0	0	1	1	1																																																																																																																																														
1	0	0	1	0	0																																																																																																																																														
0	0	0	1	0	1																																																																																																																																														
1	0	0	1	0	0																																																																																																																																														
1	0	0	0	1	0																																																																																																																																														
1	0	0	1	1	1																																																																																																																																														
1	0	0	1	1	1																																																																																																																																														

Un secondo metodo è l'**internet checksum**⁵, in questo caso vengono sommati tutti i bit del messaggio per poi aggiungere il risultato (in complemento a 2) in un campo specifico dell'header delle PDU (RFC 1071).

Per creare una checksum si procede (esempio IP):

1. Si divide la stringa da proteggere in L parole di 16 bit (poiché intestazione IP da 32 bit)
2. Si considera ogni parola come un intero, quindi si sommano per avere $x = b_0 + b_1 + b_{L-1}$ a cui applicare *modulo* $2^{16} - 1$
3. Il checksum sarà dato da $b_L = -x \bmod 2^{16} - 1$

Il blocco trasmesso deve quindi rispettare la proprietà:

$$b_0 + b_1 + b_{L-1} + b_L \bmod 2^{16} - 1 = 0$$

⁵ implementato da protocolli come IP, TCP, UDP

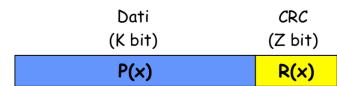
- | |
|--------------------------------------|
| ■ $b_0 = 1100 = 12$ |
| ■ $b_1 = 1010 = 10$ |
| ■ $b_0 + b_1 = 12 + 10 = 7 \bmod 15$ |
| ■ $b_2 = -7 = 8 \bmod 15$ |
| ■ $b_2 = 1000$ |

Il terzo metodo è il **CRC (codici polinomiali a ridondanza ciclica)**, dove la sequenza di k bit da inviare viene rappresentata come un polinomio $P(x)$ di grado $k-1$. Viene inoltre garantito che nodo emittente e ricevente conoscono ed utilizzano un polinomio comune $G(x)$, detto polinomio generatore, di grado z .

La trasmissione avviene nei seguenti passi:

$$1. \text{ L'emittente opera l'espressione } \frac{x^z P(x)}{G(x)} = Q(x) + \frac{R(x)}{G(x)}$$

Con $Q(x)$ quoziente e $R(x)$ resto.



2. L'emittente inserisce la sequenza di bit corrispondente a

$$T(x) = x^z P(x) + R(x), \text{ dove } T(x) \text{ rappresenta una codeword con grado } k+z-1.$$

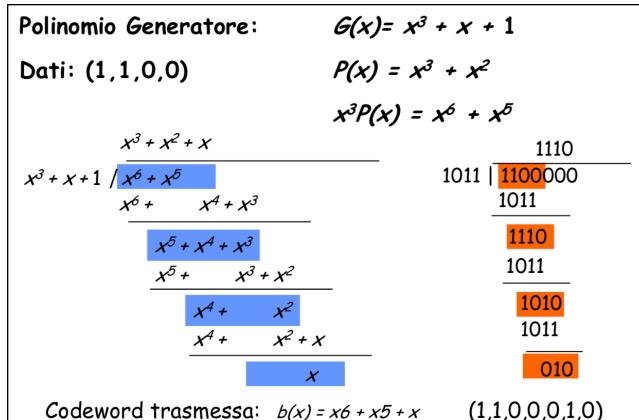
3. Il ricevente all'arrivo di $T(X)$ effettua

l'operazione $\frac{T(x)}{G(x)}$ ma siccome

$$\frac{T(x)}{G(x)} = \frac{x^z P(x)}{G(x)} + \frac{R(x)}{G(x)} = Q(x)$$

se l'operazione genera resto è stato commesso un errore durante la trasmissione, se il resto è nullo la trasmissione è avvenuta correttamente.

Questo metodo permette di rivelare tutti gli errori a burst inferiori a al grado del resto+1, inoltre in alcuni casi permette anche correzioni.



MAC (medium access control)

Esistono 2 tipi di collegamenti di rete:

- PPP (point to point protocol) => punto-punto come ad esempio le reti telefoniche
- Broadcast => condivise tra più utenti, ad esempio ethernet e wireless LAN

Nel secondo caso si necessita quindi di protocolli che gestiscano l'accesso multiplo in modo da evitare collisioni tra frame inviati contemporaneamente da più utenti.

La comunicazione può avvenire attraverso:

- Multiplazione centralizzata => un entità centralizzata gestisce l'uso delle risorse
- Multiplazione decentralizzata => nodi accedono autonomamente al canale

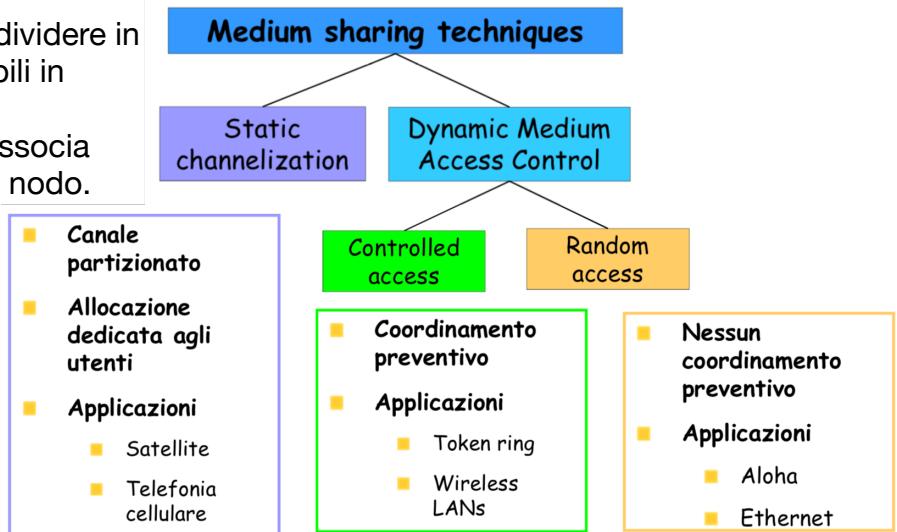
I protocolli si possono suddividere in diverse categorie riassumibili in questo albero.

La canalizzazione statica associa degli slot prefissati ad ogni nodo.

Nell'accesso dinamico le risorse vengono frazionate a seconda dell'esigenza.

È importante notare che nei protocolli randomici possono verificarsi collisioni, cosa che non può accadere in quelli ad accesso controllato.

Esempi di protocolli a

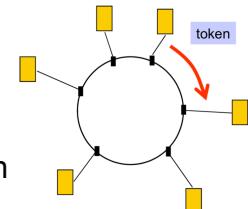


suddivisione del canale (statici) sono:

- **TDMA (time division multiple access)** => funziona come TDM, ovvero il canale viene suddiviso in slot di tempo e ad ognuno viene associato un utente.
In questo caso però non si ha una gestione centralizzata degli accessi ma sono i terminali stessi a controllare il loro accesso.
- **FDMA (frequency division multiple access)** => come TDMA ma basato sulle frequenze, usato ad esempio nelle radio
- **CDMA (code division multiple access)** => gli utenti trasmettono allo stesso tempo sulla stessa frequenza ed ogni comunicazione viene moltiplicata per un codice chiamato CHIP che individua la trasmissione.

Protocolli dinamici ad accesso controllato:

- **Polling protocol** => un nodo master (poll) controlla continuamente lo stato della rete e coordina l'invio dei dati, se il master non funziona smette di funzionare tutta la rete
- **Token passing** => crea configurazione di rete ad anello (**token-ring**) in cui solo chi possiede il token in quel momento può trasmettere.



Quando un terminale finisce la trasmissione passa il token al prossimo, stesso se non si ha niente da trasmettere.

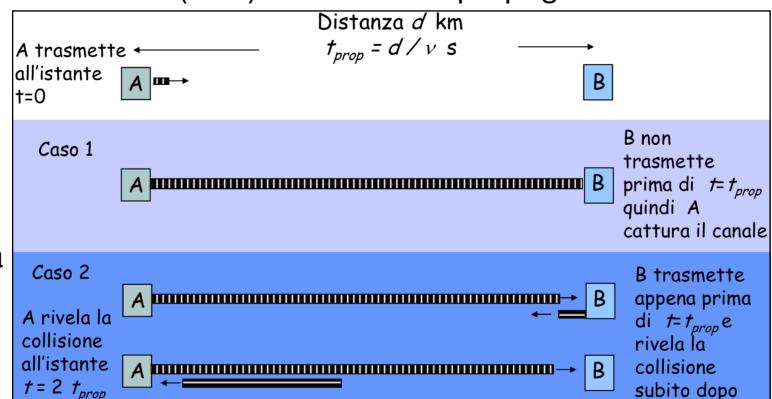
Protocollo decentralizzato ed altamente efficiente ma se uno dei terminali non funziona smette di funzionare tutta la rete.

Protocolli dinamici ad accesso casuale:

- **ALOHA**
- **SLOTTED ALOHA**
- **CSMA, CSMA/CD, CSMA/CA**

Un protocollo è condizionato da un parametro chiamato **PBR (prodotto banda-ritardo)**, dato da $PBR = R * d$, con R banda del canale (bit/s) e d ritardo di propagazione.

Ovvero il numero di bit che si trovano contemporaneamente sul canale, è quindi legato alla difficoltà di coordinamento delle collisioni. Come si vede dall'esempio una trasmissione ha un intervallo di vulnerabilità di $2t_{prop}$, poiché oltre a considerare il tempo necessario per l'invio bisogna considerare anche quello necessario per la ricezione.



$$\text{Efficienza} = \rho_{\max} = \frac{L}{L + 2t_{prop}R} = \frac{1}{1 + 2t_{prop}R/L} = \frac{1}{1 + 2a}$$

$$\text{Throughput Massimo} = R_{eff} = \frac{L}{L/R + 2t_{prop}} = \frac{1}{1 + 2a} R \text{ bit/s}$$

$$\text{Prodotto banda ritardo normalizzato} \quad a = \frac{t_{prop}}{L/R} \quad \begin{array}{l} \xrightarrow{\text{Ritardo di Propagazione}} \\ \xleftarrow{\text{Tempo di trasmissione di una frame}} \end{array}$$

Legenda

R (o **C**) = bit rate (bit/s)
L = numero bit messaggio
 Efficienza = tempo di trasmissione

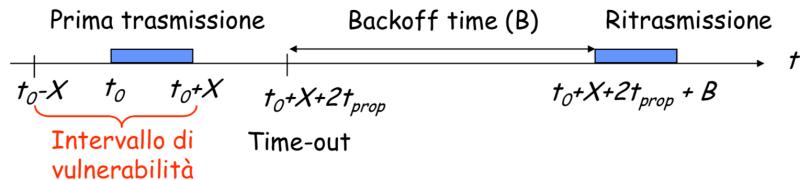
Quindi per avere la certezza che il messaggio sia arrivato a destinazione si necessita di una lunghezza dei frame $L_{min} \geq 2t_{prop} R$.

Protocollo ALOHA

Non prevede vincoli all'invio dei dati, quando il frame è pronto il terminale trasmette, comportando un alto rischio di collisione (con conseguente perdita del frame).

Una volta trasmesso il segnale, la stazione si mette in attesa di un **ACKNOWLEDGE** dal destinatario per un certo periodo, chiamato timeout.

L'ack è un avviso di correttezza, se la stazione allo scadere del timeout non riceve niente, calcola il tempo di ritrasmissione (**backoff time**), quando scade rinvia le informazioni.



Il throughput, ovvero il numero medio di trame trasmesse con successo in un certo intervallo di tempo X è dato da $S = G P_{success}$ con $0 < S < 1$

$G \Rightarrow$ load, ovvero numero di tentativi nell'intervallo X

$P_{success} \Rightarrow$ probabilità che la trama trasmessa non subisca collisioni

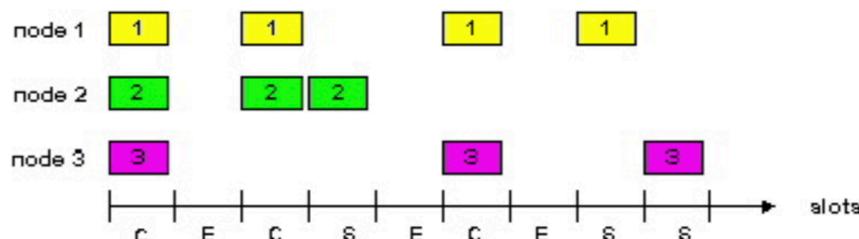
Si può dimostrare che per un numero di tentativi che tendono a ∞ $P_{success} = e^{-2G}$,

ovvero l'efficienza massima $S = G e^{-2G}$ di ALOHA è $\frac{1}{2e}$ (18%).

Protocollo slotted ALOHA

Basato sullo stesso principio di ALOHA ma con pacchetti che hanno tutti la stessa dimensione ed il tempo suddiviso in slot, i terminali sono quindi sincronizzati.

Appena il frame è pronto viene spedito all'inizio del prossimo slot, se non si verificano collisioni il terminale si prepara per inviare il frame successivo, altrimenti la collisione viene rivelata prima della fine dello slot ed il frame verrà ritrasmesso con probabilità p in uno degli slot successivi.



Questo protocollo è fortemente decentralizzato lasciando ad ogni nodo la libertà di decidere quando ritrasmettere e ricavare le collisioni in modo autonomo.

Inoltre consente di trasmettere continuamente pacchetti alla massima velocità sul canale. Si ha però che una certa percentuale degli slot presenta collisioni e quindi viene sprecata, inoltre non avendo un controllo centralizzato parte degli slot rimangono vuoti.

Riutilizzando la formula utilizzata per ALOHA e sapendo che questa la volta la

$P_{success} = e^{-G}$ si trova che le prestazioni di slotted ALOHA sono $\frac{1}{e}$ (36%).

Utilizzato generalmente con pochi utenti e poco traffico oppure inserito come metodo di prenotazione in altri protocolli, l'asse dei tempi è suddiviso in cicli all'inizio dei quali si trovano dei mini slot per effettuare le prenotazioni tramite lo slotted ALOHA.

Protocollo CSMA (carrier sensing multiple access)

Il nodo ascolta prima di trasmettere, se il canale è libero trasmette il frame, altrimenti utilizza uno degli algoritmi per il backoff.

Questo protocollo non è comunque esente da collisioni, poiché l'occupazione del canale necessita di un tempo t_{prop} (finché un nodo non riceve non sa che il canale è occupato), quindi se un altro nodo trasmette prima di questo periodo i frame verranno sovrapposti. Gli algoritmi di backoff da usare in caso di canale occupato sono:

- **1-persistent CSMA** => il nodo inizia a trasmettere come il canale si libera, basso ritardo ma bassa efficienza, utilizzabile solo con pochi utenti
- **Non-persistent CSMA** => il nodo applica backoff, quindi effettua un nuovo carrier sensing⁶ per poi ritrasmettere, alto ritardo ma alta efficienza, utile con molti utenti
- **P-persistent CSMA** => il nodo aspetta che il canale si libera, poi:
 - Con probabilità p trasmette
 - Con probabilità $(1-p)$ effettua carrier sensing
 Sia ritardo che efficienza possono essere modulari.

$$\text{L'efficienza di questo protocollo è data da } E = \frac{1}{2t_{prop} R e}$$

Protocollo CSMA/CD (with collision detection)

Il nodo ascolta prima di trasmettere e mentre trasmette, in questo modo può rivelare le collisioni ed interrompere la trasmissione immediatamente.

Questo permette di applicare immediatamente uno degli algoritmi per il backoff invece di dover aspettare la conclusione della trasmissione.

Affinché ci si accorga di una collisione devo trasmettere a

$$\frac{L_{min}}{R} \geq 2t_{prop} + t_R \quad \text{con} \quad L_{min} = \text{lunghezza minima frame} = 2\frac{d}{s}R$$

Si noti quindi che più è ampia e veloce la rete più il frame deve essere lungo, ciò è dovuto al fatto che serve più tempo per rivelare le collisioni.

Un esempio di standard che usa questo protocollo è LAN ethernet, sia nella sua forma cablata che quella wireless (WiFi).

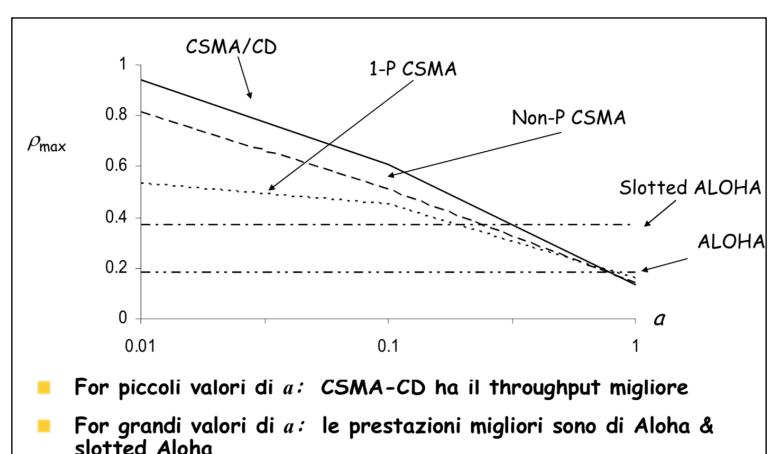
È 1-persistent, ha un bit rate di 10Mbits/sec ed un t_{prop} di 51,2 microsecondi.

Si ha quindi che la lunghezza minima di un frame deve essere di 512bits ovvero 64byte slot, la distanza massima 2.5 km (supporta massimo 4 ripetitori).

L'algoritmo per calcolare il tempo di backoff è il truncated binary exponential backoff:

Dopo la n -esima collisione si scegli il backoff in un intervallo di valori

$$\{0, 1, \dots, 2^k - 1\} \text{ con } k = \min\{n, 10\}$$



⁶ rilevamento della portante, ovvero il nodo si mette in ascolto fino a quando il canale si libera

Indirizzamento dello strato di collegamento: ARP ed indirizzi MAC

Quando un terminale opera su un canale broadcast **LAN** (**local area network**) i suoi pacchetti vengono in realtà fisicamente ricevuti da tutti i dispositivi connessi ad essa. Per capire a chi sono indirizzati i frame, nelle intestazioni di controllo dello strato 2 viene inserito un **MAC-ADDRESS** che permette di identificare univocamente⁷ la scheda di rete del dispositivo.

Nello standard Ethernet questo è un codice esadecimale da 6 byte (FF-FF-FF-FF-FF-FF).

Il MAC-address è equivalente al codice fiscale di una persona ma ha solo valenza locale, per le trasmissioni esterne è richiesto lo strato di rete che introduce un nuovo tipo di indirizzo, associabile al codice postale, chiamato **IP-ADDRESS**.

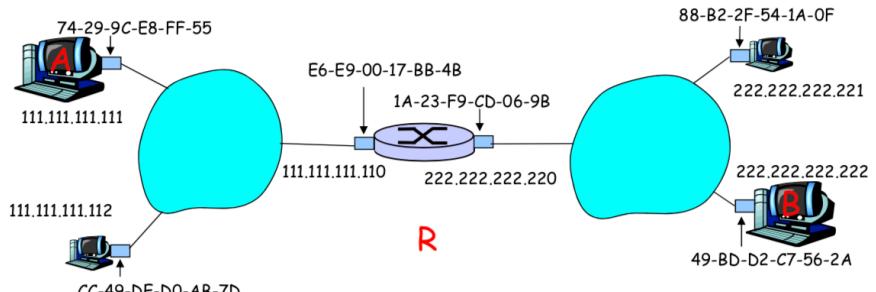
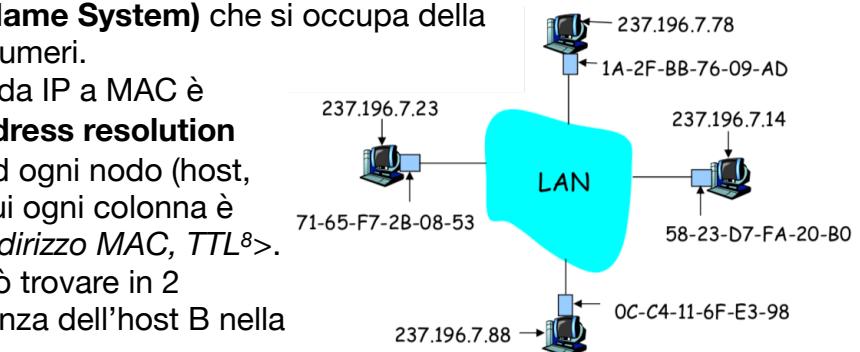
L'IP corrisponde agli indirizzi mnemonici utilizzanti normalmente (ex. www.google.com), è il protocollo **DNS (Domain Name System)** che si occupa della trasformazione da lettere a numeri.

Ad occuparsi del passaggio da IP a MAC è invece il protocollo **ARP (address resolution protocol)**, il quale associa ad ogni nodo (host, router) una tabella ARP in cui ogni colonna è composta da <indirizzo IP, indirizzo MAC, TTL⁸>.

Per il funzionamento ci si può trovare in 2 situazioni dettate dalla presenza dell'host B nella tabella di routing dell'host A:

- A e B appartengono alla stessa sottorete (B è presente):
 1. A conosce l'IP di B, quindi verifica se è già presente nella sua tabella ARP
 - Se è presente lo estrae ed invia normalmente => termina
 - Se non è presente prepara una **ARP REQUEST**, ovvero un messaggio encapsulato in un frame Ethernet contenente l'IP di B
 2. Tutta la rete LAN riceve il messaggio ma solo B risponde con un **ARP REPLY** contenente il suo indirizzo MAC
 3. A quindi prepara il pacchetto inserendo come destinatario l'indirizzo MAC ricevuto ed invia il pacchetto per poi aggiornare in plug-and-play⁹ la tabella
- A e B si trovano in 2 reti LAN differenti (B non è presente):
 1. La trasmissione avviene passando per un router R che contiene 2 tabelle ARP, una per ciascuna rete LAN
 2. Si ha quindi un primo passaggio equivalente al primo caso, qui però invece di avere direttamente la destinazione B la destinazione è il router R
 3. R, ricevuto il frame, lo spacchetta ed estrae l'IP di B, dopodiché crea un nuovo frame e ripete tutti i passaggi del primo caso comportandosi come se fosse A

Il protocollo ARP è quindi efficiente ma facilmente ingannabile, poiché un host potrebbe dichiararsi con un IP differente per ricevere informazioni non dirette a lui.



⁷ Per far questo le industrie che producono schede di rete devono comprare blocchi di codici univoci da associare ai loro dispositivi

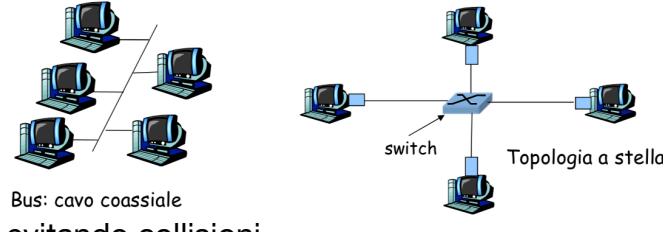
⁸ TTL (time to live) = indica la durata di vita di una riga della tabella, generalmente 20 minuti

⁹ plug-and-play = ovvero non ha bisogno di configurazioni da parte di un amministratore

Standard specifico per reti LAN: Ethernet

Standardizzato nel 1985 da IEEE (Institute of electrical and electronic engineers), appartiene alla famiglia 802 e .3 è la sua specifica, inizialmente prevedeva una velocità di 10Mbps adesso diffuso in reti da 10 Gbit.

La topologia iniziale prevedeva una rete a bus (tutti i terminali collegati ad un mezzo comune), sostituita in seguito dalla topologia a stella che prevede un **hub** (o **switch**) centrale che si occupa delle funzioni di commutazione dei frame, evitando collisioni.



Ethernet utilizza il protocollo CSMA/CD ha un bit rate di 10Mbits/sec ed un t_{prop} di 51,2 microsecondi, si ha quindi che la lunghezza minima di un frame deve essere di 512bits (ovvero 64byte slot), la distanza massima 2.5 km¹⁰ (supporta massimo 4 ripetitori).

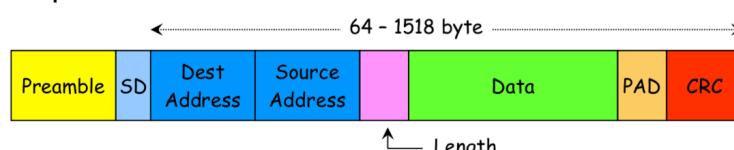
Apporta una leggera modifica al protocollo CSMA/CD , infatti quando un terminale in trasmissione si accorge di una collisione continua ad inviare bit in modo da rivelare il problema anche agli altri utenti, questo segnale continuo è detto **JAM** ed il tempo necessario per la rivelazione è detto **MAC JAM TIME**.

Inoltre ci si può accorgere della collisione osservando che l'onda ricevuta ha una potenza più alta di quella definita dallo standard Ethernet.

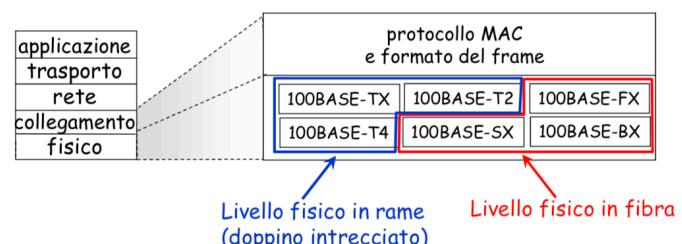
Implementa un servizio senza connessione, ovvero il mittente invia i propri dati senza aspettare nessuna risposta del ricevitore.

Un frame Ethernet è composto da:

- **Preamble** (7 byte) => serve per sincronizzare il Clock del ricevente con quello del mittente, ogni byte è composto dalla sequenza "10101010"
- **SD** (1 byte) => indica l'inizio del frame, sequenza "10101011"
- **Dest address** (6 byte) => MAC-ADDRESS del destinatario, il terminale continua a leggere solo se questo corrisponde al suo
- **Source address** (6 byte) => MAC-ADDRESS del mittente
- **Length** (2 byte) => indica lunghezza frame
- **Data** (variabile) => dati
- **PAD** (variabile) => byte inutili inseriti per raggiungere 64 byte in caso di lunghezza minore del campo dati
- **CRC** (4 byte) => polinomio di controllo errori



Oltre a definire i protocolli Ethernet si occupa anche della definire dei mezzi trasmittivi necessari per lo strato fisico, ogni sigla infatti corrisponde a velocità di trasmissione, frequenza e mezzo diversi.



¹⁰ ogni incremento di 10 volte del bit rate, determina la diminuzione di 10 volte della lunghezza massima della rete

Topologia a stella: HUB e Switch

Un dominio di collisione è una sezione di rete in cui qualsiasi coppia di stazioni che trasmettono contemporaneamente generano collisioni.

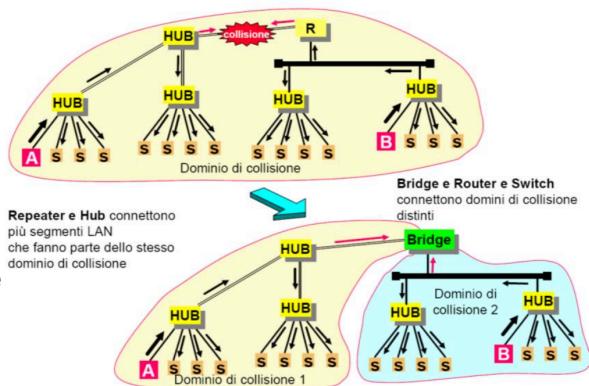
Un **hub** (repeater) opera solo sullo strato fisico, rigenera il segnale e lo ritrasmette a tutte le interfacce permettendo di estendere la rete, si accorge delle collisioni ma le ignora. La sezione di rete collegata ad esso è quindi un unico dominio di collisione.

Uno **Switch** è un'evoluzione dell'hub, incorpora anche lo strato di collegamento, ha quindi la capacità di filtrare ed inoltrare i frame Ethernet.

Potendo leggere gli indirizzi MAC può implementare il protocollo CSMA/CD che permette di evitare le collisioni e quindi creare diversi domini di collisione, uno per ogni porta. Per inviare i dati all'indirizzo giusto necessita di una tabella di inoltro, chiamata **switch table** (tabella di commutazione), in cui memorizza gli indirizzi MAC associati alle porte, ogni colonna è composta da <indirizzo MAC, interfaccia, TTL>.

La tabella è compilata in **plug-and-play** direttamente dallo switch, all'inizio la tabella è vuota, quando arriva il primo frame memorizza il MAC-ADDRESS della porta mittente, lo stesso per ogni nuovo pacchetto.

La tabella si dice a regime quando ha memorizzato tutti gli indirizzi. Gli switch quindi permettono di estendere la rete LAN senza grandi limiti creando sempre nuovi domini collisione, ma per l'esterno si necessita di router con strato di rete.

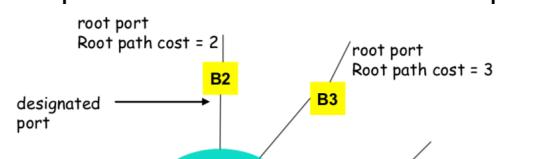
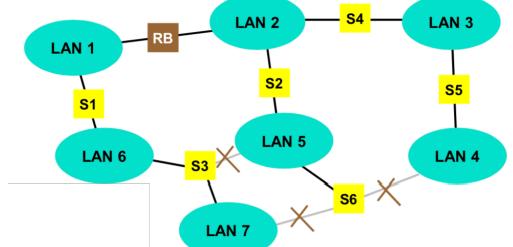


STP (spanning tree protocol)

Protocollo introdotto per evitare cicli di instradamento infiniti, appartiene allo standard IEEE 802.1D, ha il compito di tagliare fuori dalla rete i nodi che creano loop, generando un albero ricoprente dell'intera rete.

Opera in 3 fasi:

1. Determinazione **root bridge** (switch radice), vince lo switch con identificativo (root id) più basso
2. Selezione della **root port** per ogni switch, ovvero la porta con il percorso a costo minimo per raggiungere il root bridge dell'albero.
3. Selezione della **designated port** per ogni switch, ovvero, se esistono più switch che raggiungono la stessa LAN si sceglie quella con root path cost minore da utilizzare per ricevere/inoltrare le trame della LAN
4. Tutte le porte che non appartengono a una di queste categorie vengono bloccate



Per l'implementazione del protocollo i bridge si scambiano periodicamente trame di controllo

specifiche chiamate **BPDU (bridge protocol data unit)**, contenenti:

- *Root id* => identificativo del bridge candidato a diventare root
- *Switch id* => identificativo del bridge che trasmette la BPDU
- *Root path cost* => costo totale per raggiungere il root bridge
- *Flag* => TC (topology change) e TCA (TC acknowledgment)

Controllo di flusso ed errore: ARQ (automatic repeat request)

Metodo opposto al FEC, si occupa di richiedere ed inviare frame arrivati con errori, inoltre si assicura che la sequenza delle PDU sia in ordine e senza ripetizioni.

Questo metodo è utilizzato sia come **hop-by-hop** (tratta per tratta) nello strato di collegamento, sia come **end-to-end** nello strato di traporto.

Stop-and-wait ARQ

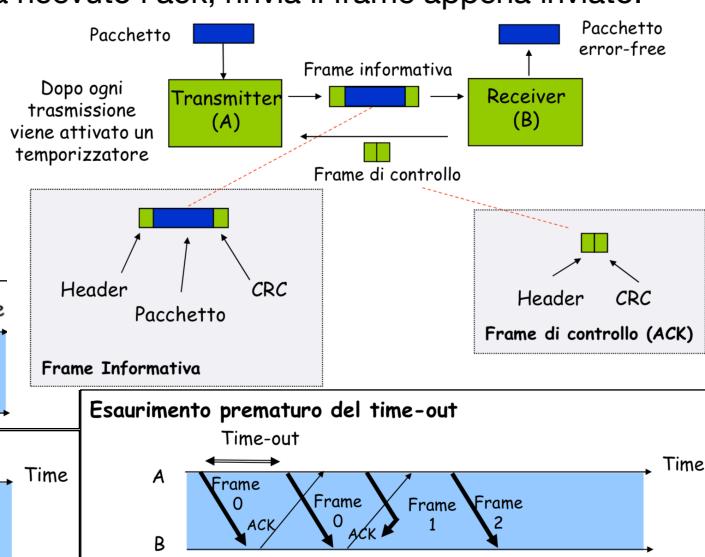
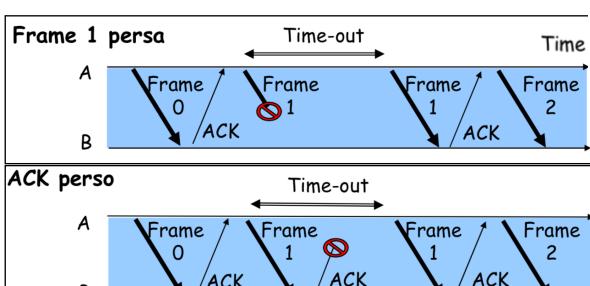
A invia un frame a B ed aspetta un ack come riscontro, funzionamento:

1. Quando A invia il frame si attiva un temporizzatore
 2. Allo scadere del timeout, se A non ha ricevuto l'ack, rinvia il frame appena inviato.

Questo può avvenire in 3 casi:

- A. Il frame non è stato ricevuto
 - B. l'ack non è arrivato
 - C. Esaurimento prematuro del timeout

OSS per evitare duplicati si necessita di una numerazione dei frame

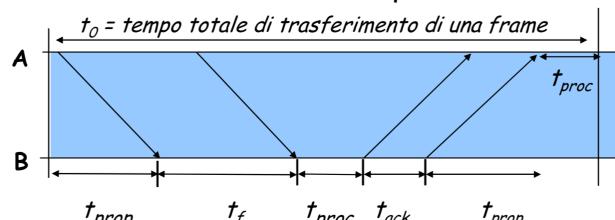


La numerazione necessaria per riconoscere i frame viene generalmente semplificata da un sistema di alternanza di “0” / “1”

Facendo riferimento al caso C si nota come l'utilizzo di un time-out errato può comportare un secondo invio non necessario

con conseguente spreco di risorse.
Per calcolare la lunghezza del treno, ci vu-

$$t_0 = 2t_{prop} + 2t_{proc} = \frac{L_f}{P} + \frac{L_{ack}}{P}$$



Efficienza su un canale senza-erri

Rete di trasmissione efficace

$$R_{\text{eff}}^0 = \frac{\text{numero di bit informativi consegnati a destinazione}}{\text{tempo totale necessario per la consegna dei bit informativi}} = \frac{n_f - n_o}{t_s},$$

Efficienza di trasmissione

$$\eta_0 = \frac{R_{eff}}{R} = \frac{\frac{n_f - n_o}{t_0}}{R} = \frac{1 + \frac{n_o}{n_f}}{1 + \frac{n_a}{n_f} + \frac{2(t_{prop} + t_{proc})R}{n_f}}$$

Effetto dell'overhead di una frame

Effetto di un ACK

Effetto del prodotto Banda-Ritardo

Efficienza su un canale con errori

- Sia $1-P_f$ = probabilità che una frame arrivi senza errori
 - $1/(1-P_f)$ = numero medio di trasmissioni necessarie per avere una trasmissione corretta di una frame
 - $t_o/(1 - P_f)$ = tempo medio di trasferimento di una frame

$$\eta_{SW} = \frac{R_{eff}}{R} = \frac{\frac{n_f - n_o}{t_0}}{\frac{1 - P_f}{R}} = \frac{1 - \frac{n_o}{n_f}}{1 + \frac{n_o}{n_f} + \frac{2(t_{prop} + t_{proc})R}{n}} (1 - P_f)$$

n corrisponde
alla lunghezza /

Effetto della probabilità di perdita delle frane

Questa tecnica non è efficace su link ad alta velocità o con elevati ritardi di propagazione, inoltre è altamente influenzata dalla presenza di errori.

Go-back N ARQ

Migliora lo stop-and-wait eliminando l'attesa dei riscontri, il canale è sempre mantenuto occupato continuando ad inviare frame.

Il riscontro è gestito da una **finestra di trasmissione** di ampiezza W_s frames.

Funzionamento (esempio con $W_s = 4$):

1. A invia frame F0, F1, F2, F3
2. Quando B riceve F0 invia l'ack

$$R_{next} = 1$$

3. A capisce che B ha ricevuto F0 quindi aggiorna la finestra facendo

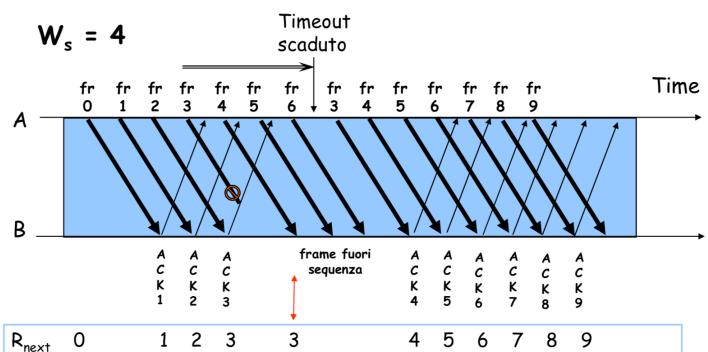
in modo che S_{last} abbia lo stesso numero dell'ack appena ricevuto, questo fa comparire un nuovo numero nella sequenza (4) detto S_{recent}

4. Arrivato ad F4 A riceve l'ack 2, quindi aggiorna la finestra (2,3,4,5) ed invia F5, stesso per F6 con l'arrivo dell'ack 3 (3,4,5,6)

Quindi se non si presentassero errori il protocollo continuerebbe ad inviare nuovi frame ininterrottamente, nell'esempio però F3 non arriva a destinazione, quindi B rivela un errore e non può inviare l'ack di risposta

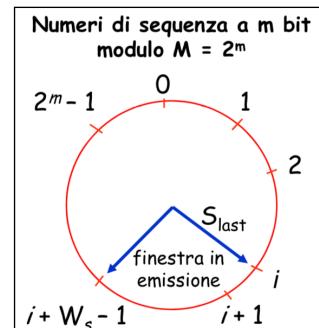
5. A rimane in attesa dell'ack 3 che non arriverà mai, quindi allo scadere del time-out di F3 rivela l'errore e comincia ad inviare nuovamente tutti i frame corrispondenti all'ultimo aggiornamento della finestra di trasmissione (3,4,5,6).

Infatti B essendo in attesa di F3 scarta tutti gli altri frame fuori sequenza



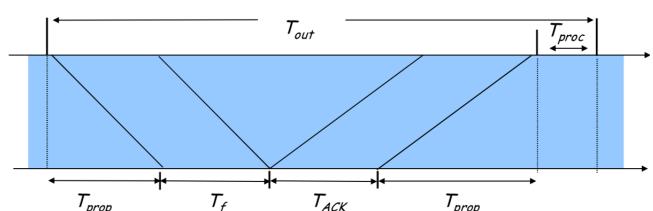
La finestra di trasmissione è chiamata **sliding window** (finestra scorrevole) e può anche essere rappresentata come un cerchio. Nell'esempio gli ack vengono inviati dopo ogni frame, in alcuni casi però si possono avere ack cumulativi, ovvero l'ack viene rilasciato solo dopo l'invio di un tot di frame.

La scomodità di questo protocollo sta nel fatto che, quando scade il time-out, A è costretto a rinviare tutti i frame contenuti nella finestra in quel momento (da notare che questo accade anche se B riceve i frame ma non riesce ad inviare gli ack).



Il mancato arrivo degli ack di riscontro può creare anche un altro problema, a livello teorico si possono associare ai frame numeri progressivi che tendono all'infinito, ma nella realtà la macchina ha a disposizione solo un certo numero m di bit, quindi in caso di rinvio di tutta la sequenza di possano creare delle ambiguità sul numero del frame, per ovviare a questo problema si utilizza finestra massima di $2^m - 1$ che permette di rilevare frame già ricevuti e scartarli.

Nel caso in cui la trasmissione sia bidirezionale si utilizza la tecnica **PiggyBacking** che incorpora l'ack nei frame, quindi ogni frame ricevuto da A o B corrisponde anche ad un ack di ritorno per i dati inviati in precedenza.



L'efficienza del protocollo dipende da W_s che deve essere abbastanza grande da poter mantenere il canale occupato per tutto il periodo t_{out} .

Ex. Il protocollo TCP ogni volta che inizia una nuova sessione valuta la distanza ed il bit rate in modo da calcolare di conseguenza il W_s adeguato.

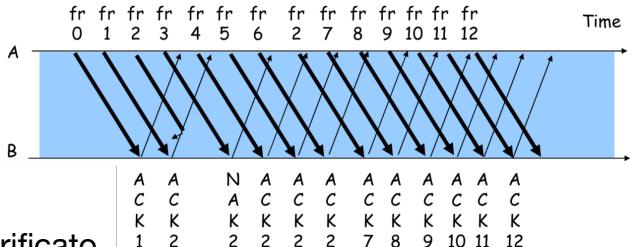
Selective repeat ARQ

Migliora il go-back N, quando deve ritrasmettere invece di rinviare tutti i frame della finestra invia solo quello sbagliato, questo poiché il ricevente memorizza tutti quelli successivi in un buffer ampio W_R .

Funzionamento (esempio):

1. A invia F1, F2, F3
 2. B riconosce F0, F1 ed invia ack 1 ed ack 2, ma F3 viene perso quindi B non invia l'ack corrispondente
 3. A continua ad inviare, ma siccome si è verificato un errore in precedenza, B, all'arrivo di F4, lo inserisce nel buffer ed invece di inviare l'ack 4 invia un nack corrispondente al frame perso, ovvero nack 2
 4. Quando A riceve il nack si accorge dell'errore ed invia nuovamente F2 a B, che una volta ricevuto il frame lo spedisce allo strato superiore insieme a tutti gli altri frame bufferizzati ricominciando quindi a mandare gli ack in modo corretto, partendo dal prossimo non bufferizzato

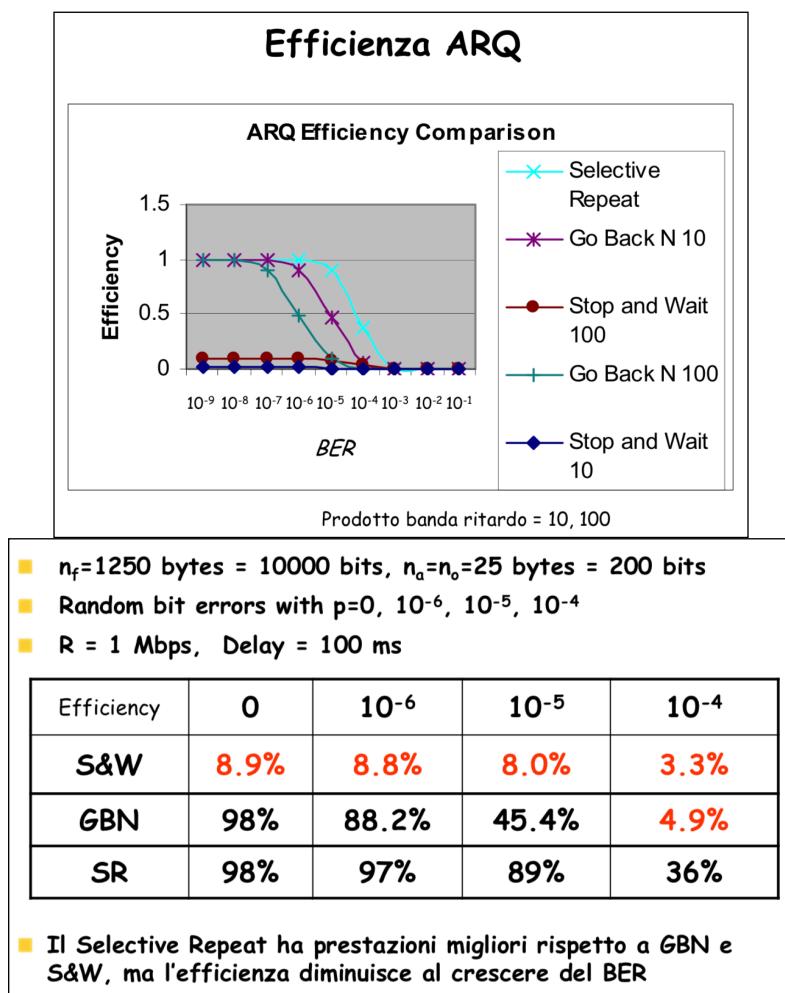
A	A	N	A	A	A	A	A	A	A	A	A
C	C	A	C	C	C	C	C	C	C	C	C
K	K	K	K	K	K	K	K	K	K	K	K
1	2	2	2	2	2	7	8	9	10	11	12



Il ricevente invia il **nack** (avviso d'errore) in 2 casi:

- Il frame arriva con un errore
 - Il frame o l'ack non arrivano entro il tempo di timeout

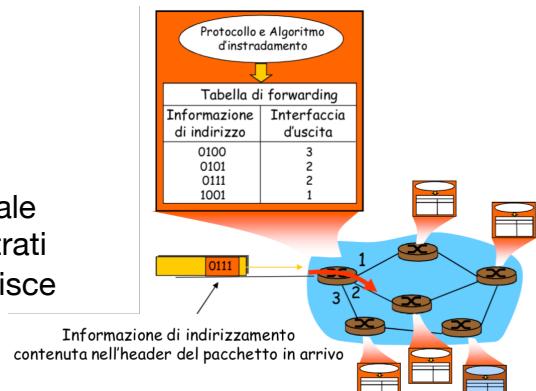
Il valore di W_R deve essere tale che $W_R + W_s = 2^m$, è più efficiente degli altri 2 ma anche più complicato da implementare, inoltre deve gestire una quantità maggiore di dati. Anche in questo caso è altamente influenzata dalla probabilità d'errore.



Lo strato di rete

Le funzioni dello strato di rete sono:

- **Routing** (Instradamento) => funzione decisionale, determina il percorso che seguiranno i pacchetti dall'origine alla destinazione, inoltre decide su quale interfaccia in uscita del router devono essere inoltrati
- **Forwarding** (inoltro) => funzione attuativa, trasferisce i pacchetti da un interfaccia di ingresso ad un opportuna interfaccia d'uscita

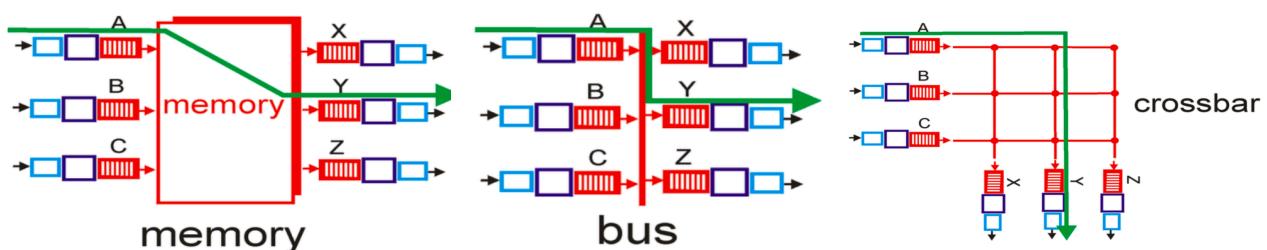
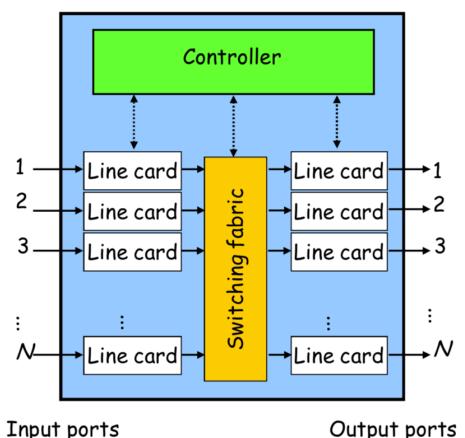


Le 2 funzioni sono implementate diversamente a seconda della modalità di rete:

- Rete a **circuito-virtuale** (servizio con connessione):
 - Comutazione a circuito, prima dell'invio dei pacchetti è necessaria una fase di set-up che instaura la connessione ed inizializza le tabelle di forwarding dei nodi, è qui che si decide il path che verrà seguito da tutti i pacchetti (non sarà quindi necessario inserire il destinatario in ogni pacchetto)
 - I nodi hanno funzionamento **statefull**, ovvero mantengono informazioni sullo stato delle connessioni
 - Alla fine della conversazione bisogna liberare le risorse e cancellare le info sugli stati
- Rete a **datagramma** (servizio senza connessione):
 - Comutazione a pacchetto, non si instaura la connessione quindi i pacchetti vengono inviati senza un accordo preventivo con il destinatario
 - l'instradamento è deciso pacchetto per pacchetto, quindi possono seguire path diversi (implica necessità di inserire destinatario in ogni pacchetto)
 - I nodi hanno funzionamento **stateless**, non memorizzano nulla

Architettura di un router

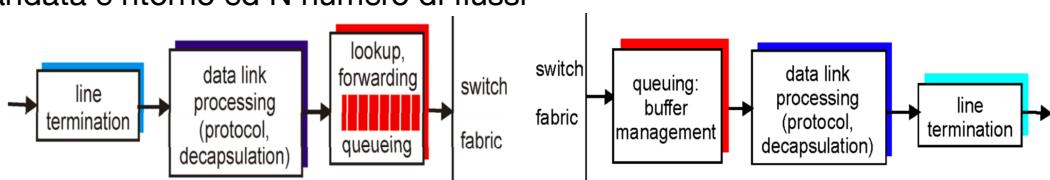
- **Controller** => esegue i protocolli di instradamento, gestisce le tabelle di inoltro e le informazioni sui collegamenti attivi
- **Switching fabric** (struttura di commutazione) => struttura di commutazione che connette fisicamente porte di uscita con porte d'entrata, ne esistono 3 tipi:
 - **Shared memory** => prima generazione di router, il pacchetto viene copiato nella memoria (da cui dipende la velocità) e la commutazione è gestita sotto diretto controllo del controller
 - **Bus interconnection** => le porte d'ingresso trasferiscono il pacchetto direttamente alle porte d'uscita tramite un bus condiviso, che quindi ne determina la velocità
 - **Crossbar** => rete di interconnessione formata da $2n$ bus che collegano n porte d'ingresso ad n porte d'uscita, superano limite di banda di un bus unico



- **Line card** => si occupano delle funzioni di strato 1 e 2, inoltre, la porta in ingresso si occupa del processo di forwarding passando per la tabella di routing, quella in uscita decide in quale ordine trasmettere i pacchetti in coda.

Il buffer per memorizzare i pacchetti accodati si può trovare o in entrata o in uscita, la versione in uscita ha prestazioni migliori, poiché nella versione in entrata un pacchetto potrebbe trovarsi accodato anche se la sua porta d'uscita è libera.

La dimensione dei buffer è data da $B = \frac{RTT * R}{\sqrt{N}}$ con RTT media del tempo di andata e ritorno ed N numero di flussi

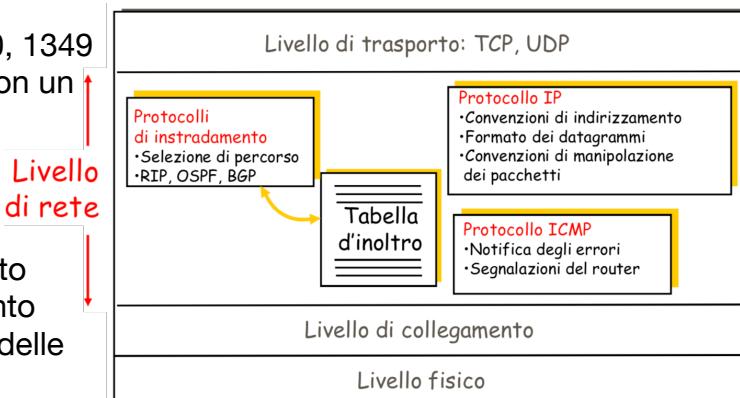


Il protocollo IP

Definito dalle RFC 791, 919, 922, 950, 1349 è il cuore dello strato di rete, opera con un servizio "best effort", ovvero senza connessione.

Le sue funzioni sono:

- Definizione formato pacchetti
- Definizione schema di indirizzamento
- Definizione modalità di instradamento
- Frammentazione e riassemblaggio delle unità dati



Per il suo corretto funzionamento IP necessita di altri protocolli di supporto, che sono:

- **Protocolli di instradamento (RIP, OSPF, BGP)** => compilano le tabelle di routing
- **Protocollo ICMP (internet control message protocol)** => ha una funzione di controllo sul trasferimento, comunica eventuali problemi alle sorgenti e cerca di risolvere le situazioni anomale

Pacchetto IP

- **Vers** => versione del protocollo a cui si riferisce il pacchetto
- **HLEN** => lunghezza dell'header, compresa tra 20 e 60 byte
- **Service type** => specifica i parametri di qualità del servizio richiesti dall'utente, diviso a sua volta in parti:

Bytes								
0	4	8	12	16	20	24	28	31
Vers	HLEN	Service Type			Total Length			
			Identification		Flag + Fragment Offset			
			Time To Live	Protocol	Header Checksum			
					Source IP Address			
					Destination IP Address			
					Options		Padding	
					Data			
							
					Data			

- Precedence: livello di priorità del pacchetto
- Type of service (TOS): indicano servizio richiesto, solo uno può essere impostato ad "1"

Precedence	Delay	Thput	Reliab.	Cost	0
1	2	3	4	5	6

1 0 0 0	Minimize delay
0 1 0 0	Maximize Throughput
0 0 1 0	Maximize Reliability
0 0 0 1	Minimize Monetary Cost
0 0 0 0	Normal Service

- **Total length** => lunghezza complessiva del pacchetto
- **Identification** => numero identificativo del pacchetto, assegnato dalla sorgente
- **Flags** => composto da 3 bit:
 - X: non usato e posto a 0
 - DF (don't fragment): frammentazione permessa se "1", altrimenti no
 - MF (more fragment): se "0" questo è ultimo frammento, altrimenti no
La frammentazione è un'operazione standardizzata dagli RFC, la suddivisione avviene quando l'unità massima di trasmissione (MTU) dello strato di collegamento è minore della grandezza del pacchetto (tutti i pacchetti mantengono header).
- **Fragment offset** => posizione del frammento all'interno del pacchetto
- **TTL (time to live)** => inizializzato dalla sorgente e decremento ad ogni passaggio su un nodo, indica il numero massimo di nodi che può attraversare per evitare che viaggi all'infinito, quando diventa nullo viene scartato (conseguente ICMP message)
- **Protocol** => indica a quale protocollo di strato superiore deve essere trasferito il contenuto informativo (TCP = 6, UDP = 17, ICMP = 1)
- **Header checksum** => bit di controllo header, se rivela errore il pacchetto viene scartato
- **Source address / destination address** => indirizzi di sorgente e destinazione (32 bit)
- **Options** => campo opzionale può contenere:
 - RRO (record route option): lista degli IP che attraversa
 - Timestamp option: aggiunge ad RRO l'istante in cui passa per il nodo
 - LSRO (loose source routing option): lista dei router che devono essere attraversati
 - SSRO (strict source route option): router attraverso i quali deve transitare
- **Padding** => rende intestazione multipla di 32 bit aggiungendo zeri

Protocollo ICMP (internet control message protocol)

Definito da RFC 792, 950 e parte integrante del protocollo IP, consente ai router di inviare all'host sorgente informazioni riguardo anomalie nel processamento di un pacchetto.

Possono essere:

- Errori di instradamento
- TTL scaduto
- Congestione eccessiva

OSS Si occupa solo delle notifiche, non prende decisioni, è l'host a decidere come comportarsi all'arrivo della notifica.

I messaggi ICMP sono incapsulati nella parte dati di pacchetto IP, un malfunzionamento su uno di questi pacchetti non genera un nuovo messaggio ICMP.

Composto da:

- **Type** => identifica tipo messaggio
- **Code** => codice errore
- **Checksum**
- **ICMP data** => permette di identificare il pacchetto e la causa dell'errore

0	8	16	24	31
Type	Code	Checksum		
ICMP data				
0	Echo reply	11	Time exceeded	
3	Dest. Unreachable	13	Time stamp request	
4	Source Quench	14	Time stamp replay	
5	Redirect	17	Address mask req.	
8	Echo	18	Address mask rep.	

Esempi di messaggi sono:

- Echo / echo reply => servono per stabilire attività di un host
- Destination unreachable => indica che non è possibile trovare il destinatario
- Redirect message => se emesso da un router indica che i successivi pacchetti dovranno essere inviati attraverso un altro router a causa di una modifica nelle tabelle
- Time exceeded => indica che il TTL è esaurito

Due protocolli che usano ICMP sono:

- **Ping** => individua l'attività di un host ed il tempo di transito tra sorgente e destinazione, utilizza i messaggi echo e echo reply

Per utilizzo da terminale: *ping "indirizzo mnemonico o IP"*

- **Traceroute** => traccia il pacchetto, ovvero mostra tutti i router che attraversa.

Per far questo forza il TTL:

1. invia un primo pacchetto con TTL=1 che raggiunge il primo router e ritorna un messaggio Time exceeded con i suoi dati

OSS alcuni router non hanno il permesso di inviare i propri dati, quindi ritornano come identificativo * * *

2. Invia un secondo messaggio TTL=2 per identificare il secondo

3. Continua così fino a destinazione

il procedimento viene ripetuto fino a 3 volte, il tempo necessario per raggiungere la destinazione è chiamato (RTT) round trip time.

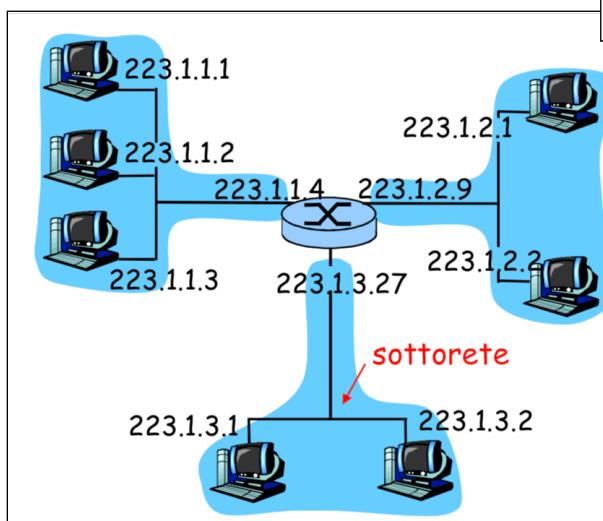
Per utilizzo da terminale: *tracert "indirizzo mnemonico o IP"*

Indirizzamento IP

l'IP-ADDRESS è una sequenza che identifica univocamente una singola interfaccia di rete, se un host è connesso a più reti ha un IP per ogni interfaccia.

Un router ha tanti indirizzi IP quante sono le interfacce di rete che gestisce.

Notazione Numerica	10010111	01100100	00001000	00010010
Notazione Dotted	151. 100. 8. 18			



L'indirizzo IP è una stringa binaria di 32 bit ma viene spesso scritto nella notazione dotted dove ogni gruppo da 8 bit è separato da “.” e tradotto in decimale.

Una sottorete è una rete isolata in cui i suoi punti terminali sono collegati all'interfaccia di un host o di un router.

Una sottorete è detta anche rete IP.

ATTENZIONE un link diretto tra 2 router è comunque una sottorete con 2 interfacce

Un indirizzo IP è formato da 2 parti:

- **Net_id** => identificativo sottorete, uguale per ogni host della sottorete
- **Host_id** => identificativo del singolo host nella sottorete

In origine fu adottato il metodo **classfull** dove le tipologie di sottorete erano divise in classi, identificate dai bit iniziali.

Da notare che bisogna escludere dagli host disponibili gli Host_id “speciali”:

- Tutti “0” identifica sottorete
- Tutti “1” indirizzo broadcast

Classe	Bit iniziali	Net_Id	Host_Id	“Reti” disponibili	“Host” disponibili
A	0	7 bit	24 bit	128	16.777.216
B	10	14 bit	16 bit	16384	65.536
C	110	21 bit	8 bit	2.097.152	256
D	1110			Indirizzo multicast: 28 bit Indirizzi possibili: 268.435.456	
E	11110			Riservata per usi futuri: 27 bit Indirizzi possibili: 134.217.728	

	0	8	16	24	31
Classe A	0	Net_Id		Host_Id	
Classe B	10		Net_Id		Host_Id
Classe C	110			Net_Id	Host_Id
Classe D	1110				Multicast Address
Classe E	11110				Reserved

Per migliorare questa tecnologia, nel 1984, fu introdotto il **subnetting**, che permette di introdurre un terzo livello gerarchico, il **Subnet_id**, separando alcuni bit dell'Host_id. Più bit si utilizzano per la subnet più sottoreti si possono indirizzare.

Original address	1 0	Net ID	Host ID	
Subnetted address	1 0	Net ID	Subnet ID	Host ID

Ad esempio alla sapienza è assegnato un blocco di indirizzi di classe B, il che la rende parte delle 16384 reti mondiali.

Non avendo tutti questi utenti li divide con Tor Vergata, ciò è permesso utilizzando il primo bit dell'Host_id come Subnet_it, da notare che adesso entrambe non hanno più a disposizione 2^{16} host ma 2^{16-1} .

Per utilizzare questo metodo, accanto all'IP address, viene inviata una **Subnet Mask**, ovvero una parola di 32 bit con tanti "1" quanto sono lunghi Net_it + Subnet_id e tanti "0" quanto è lungo l'Host_id.

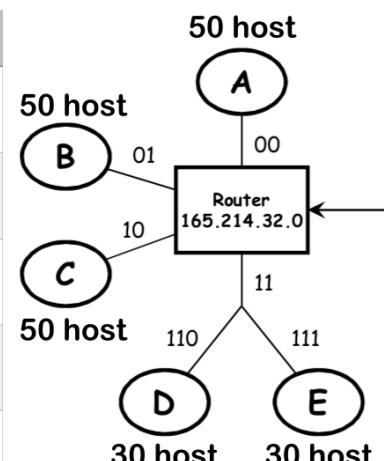
IP address	1 0	Net ID	Subnet ID	Host ID
Subnet Mask	1 1 1 1 1 1 1 1	...	1 1 1 1	0 0 0 0 0 0 0 0

Il subnetting può essere di 2 tipi:

- **Statico** => le sottoreti che vengono ricavate dalla stessa rete hanno tutte la subnet mask uguale, semplice da implementare ma comporta un alto spreco di indirizzi in sottoreti di piccole dimensioni
 - **Variabile** => ogni sottorete ha una propria maschera a seconda del numero di indirizzi di cui necessita, consente di gestire reti con dimensioni diverse

Ex. dato l'indirizzo di classe C: 193.214.32.0, assegnare ad ognuna delle 5 sottoreti un gruppo di indirizzi IP, sapendo che A,B,C posseggono 50 host e D,E invece 30

Indirizzo IP		Subnet mask
A	193.214.32.0 a 193.214.32.63 ...00000000 a ... 00111111	255.255.255.192 /26 1.1.1.11000000
B	193.214.32.64 a 193.214.32.127 ...01000000 a ... 01111111	255.255.255.192 /26 1.1.1.11000000
C	193.214.32.128 a 193.214.32.191 ...10000000 a ... 10111111	255.255.255.192 /26 1.1.1.11000000
D	193.214.32.192 a 193.214.32.223 ...11000000 a ... 11011111	255.255.255.224 /27 1.1.1.11100000
E	193.214.32.224 a 193.214.32.255 ...11100000 a ... 11111111	255.255.255.224 /27 1.1.1.11100000



1. Parto dalla sottorete con più host (A, B, C), per avere 50 host necessito di 6 bit per l'host, ho quindi $8-6 = 2$ di Subnet_id
 2. Con 2 bit di Subnet_id è possibile suddividere la rete in 2^2 sottoreti e siccome D ed E necessitano solo di 30 host possono unirsi e formare una sottorete D + E
 3. L'assegnazione delle subnet è quindi: A = 00, B = 01, C = 10, D + E = 11
 4. A questo punto bisogna suddividere nuovamente D ed E, si prende quindi un altro bit di subnet D = 110 e E = 111

OSS il numero dopo “/” è dato dal numero di “1” nella subnet mask

Routing in reti IP

Il routing sfrutta l'utilizzo delle **routing table** (tabelle di routing), composte da:

- Indirizzo IP destinatario
- Indirizzo IP del next-hop router, ovvero il prossimo nodo
- Identificatore d'uscita
- Informazioni statistiche

Nella routing table non è possibile mantenere tutti gli indirizzi mondiali, questi vengono quindi raggruppati a seconda del Net_id.

Ex. l'host A deve spedire un pacchetto all'host B

1. A compila l'header del pacchetto IP (per trovare l'IP destinazione utilizza il DNS) e controlla se B appartiene alla sua sottorete tramite il protocollo ARP e le normali procedure dello strato di collegamento
2. La risposta è negativa quindi il pacchetto deve essere inviato al **default gateway** (default router) che si occuperà di inoltrarlo
3. Una volta ricevuta l'unità, il router spacchetta il frame e ricava l'IP destinazione, dopodiché lo confronta con la sua routing table secondo i seguenti criteri di ricerca:
 - Destination address completo
 - Destination Net_id (prefisso)
 - Default router
 - "declare packet undeliverable", con conseguente messaggio ICMP
4. Il default router capisce che il pacchetto deve essere inviato ad un nuovo router appartenente alla sua sottorete, quindi tramite ARP effettua il passaggio
5. Il procedimento si ripete fino a trovare la sottorete che contiene il terminale con l'indirizzo MAC associato al destinatario

OSS IP di destinazione e sorgente non vengono mai modificati, cambiano solo i MAC address a cui vengono associati per il passaggio dei frame.

Nell'esempio vengono utilizzati 2 tipi di indirizzamento:

- **Indirizzamento diretto** => usa l'interfaccia direttamente collegata alla sua sottorete (da router finale a LAN B)
- **Indirizzamento indiretto** => sfrutta il next-hop (passaggio tra i vari router)

Ogni oggetto collegato ad internet ha un proprio IP, questo crea 2 problemi:

- Esaurimento degli indirizzi disponibili
- Crescita delle entry delle routing table, tra il 1991 ed il 1995 raddoppiavano le proprie dimensioni ogni 10 mesi

Una tecnica per ovviare a questo problema sarebbe quella di riciclare IP non più utilizzati, ma questi comunque non basterebbero al crescere dei dispositivi.

Come soluzione a lungo termine si è pensato di creare nuove versioni (IPv4 è a 32bit, la nuova IPv6 a 128), ma questo comporta una modifica di tutte le tabelle di routing.

Per quanto riguarda le soluzioni a breve termine invece si ha:

- Classless Inter Domain Routing (CIDR) [RFC 1518]
- Una nuova politica di allocazione [RFC 2050]
- Introduzione di IP privati nelle reti intranet

Classless Inter Domain Routing (CIDR)

Permette di assegnare ad una rete un certo numero di blocchi contigui di indirizzi, questa modalità viene chiamata **supernetting**, funziona come il subnetting ma invece di lavorare sulla parte host_id, lavora sul net_id.

Oltre ad un utilizzo con meno sprechi degli indirizzi IP, CIDR migliora le tabelle di routing utilizzando un unico prefisso per caratterizzare sottoreti raggruppate (supersottoreti).

In questo modo si attua anche il cambio della politica di allocazione, potendo assegnare blocchi consecutivi di classe C (fino a 64 blocchi) con stesso prefisso, indirizzi di classe A e B vengono assegnati solo se strettamente necessario.

Si ha inoltre una pianificazione geografica degli indirizzi di classe C, ogni regione viene identificata con gli stessi 7 bit del prefisso.

Ex.1 Assegnazione degli indirizzi in Nord America => CIDR mask 198.0.0.0 /7.

Multiregional	192.0.0	193.255.255
Europe	194.0.0	195.255.255
Others	196.0.0	197.255.255
North America	198.0.0	199.255.255
Central/South America	200.0.0	201.255.255
Pacific Rim	202.0.0	203.255.255
Others	204.0.0	205.255.255
Others	206.0.0	207.255.255

Un ISP richiede 2048 blocchi di indirizzi di classe C, ovvero:

dato 198.0.0.0 = 11000110.00000000.00000000

per avere 2048 si necessita di 11 bit ($2048 = 2^{11}$)

Quindi ad esempio i blocchi potranno essere:

da 198.24.0.0 (11000110.00011000.00000000.0)

a 198.31.255.0 (11000110.00011111.11111111.0)

con CIDR mask 198.24.00/13

Ex.2 Un altro ISP richiede 16 blocchi di indirizzi di classe C, partendo dalle stesse.

assunzioni si ha che per 16 blocchi questa volta si necessita di 4 bit ($16 = 2^4$).

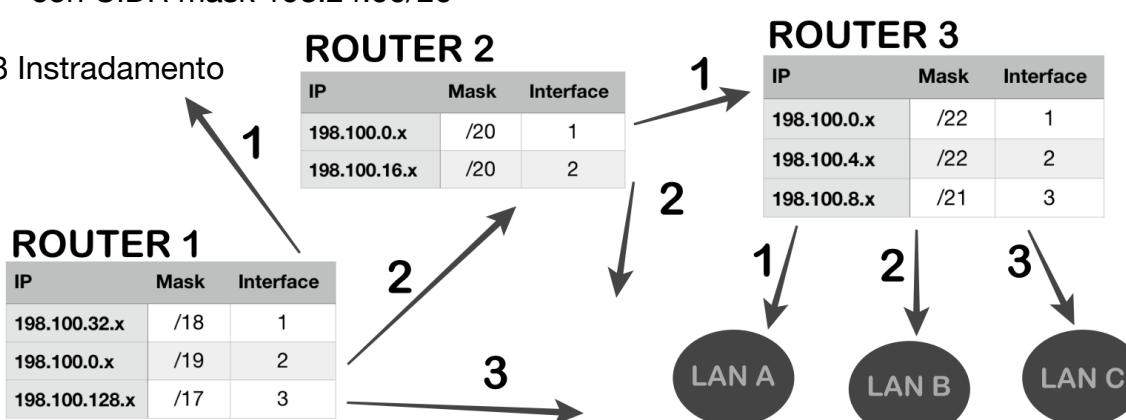
Quindi ad esempio i blocchi potranno essere:

da 198.24.0.0 (11000110.00011000.00010000.0)

a 198.31.255.0 (11000110.00011111.00011111.0)

con CIDR mask 198.24.00/20

Ex.3 Instradamento



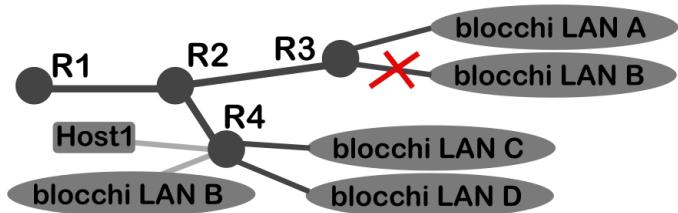
- Il router 1 legge l'ip 198.100.6.22 e gli applica ogni maschera delle proprie entry
In decimale risulta 11000110.01100100.00000110.00001010, quindi applicando la maschera /18 rimane 11000110.01100100.00000000.00000000 che equivale a 198.0.0.0 => non combacia
Si applica /17 che ritorna 198.100.0.0 => non coincide
Quindi il pacchetto viene trasferito su interfaccia 2 (longest prefix matching)
- Il router 1 legge l'ip 198.100.6.22 ed eseguendo gli stessi passaggi invia a router 3
- Stesso lavoro viene effettuato dal router 3 per arrivare all'host finale
OSS Nell'esempio l'ip coincide ad una sola entry, ma questo non è sempre vero.

Nel caso in cui ci si trovi con un ip che corrisponde a più entry del router si applica la regola del **longest prefix matching**, cioè si sceglie sempre l'entry con il prefisso maggiore, (ovvero quello con la maschera più lunga).

Ex.4 si vuole spostare l'intera sottorete LAN B dal nodo R3 al nodo R4.

Per far questo bisogna modificare la tabella di route:

- La tabella di R2 deve essere modificata in modo da reindirizzare i pacchetti destinati alla LAN B a R4 (prima andavano su R3)
- La tabella di R3 deve rimuovere l'interfaccia delle LAN B
- La tabella di R4 deve aggiungere una riga contenente il prefisso e la maschera verso LAN B



Se invece si volesse spostare un solo host di LAN B su R4 bisognerebbe il suo ip sia R2 (con next-hop R4), sia su R4.

Il passaggio effettivo di un blocco si può effettuare in 2 modi:

- Si passano uno ad uno i 254 utenti inserendo in loro ip in R2 e R4 con maschera /32
- I 4 blocchi, ad esempio, iniziano con 193.64.4.0 e finiscono con 193.64.7.255.

Si ipotizza di attaccare il blocco 2 a R4, ovvero da 193.64.5.0 a 193.64.5.255.

Affinché R2 capisca che un pacchetto indirizzato al blocco 2 non deve andare in R3, bisogna aggiungere una riga nella sua tabella.

Avendo ad esempio che 193.64.0.x/20 => R3 e 193.64.16.x/20 => R4, si dovrà aggiungere una terza riga con maschera maggiore alla prima, un esempio può essere: 193.64.5.x => R4

Protocollo DHCP (Dynamic Host Configuration Protocol)

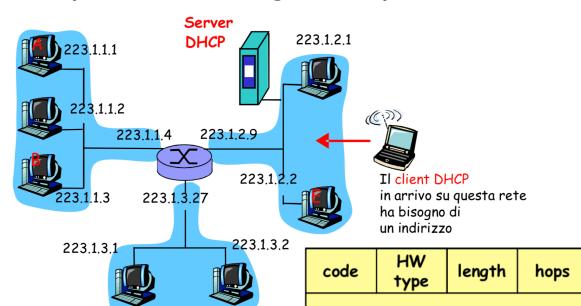
Protocollo introdotto per evitare sprechi nell'attribuzione di indirizzi IP sfruttandone il riuso, per fare questo i server DHCP hanno a disposizione un certo numero di ip da assegnare dinamicamente ai terminali che vogliono accedere alla rete.

Si occupa quindi di configurare dinamicamente i campi necessari agli host per accedere a internet, che sono:

- Indirizzo IP
- Subnet mask
- Gateway predefinito
- Server DNS

Supporta 3 meccanismi per la gestione:

- Allocazione automatica => permanente
- Allocazione dinamica => periodo limitato
- Allocazione manuale => tramite amministratore



code	HW type	length	hops
Transaction ID			
Seconds	Flags field		
Client IP address			
Your IP address			
Server IP address			
Router IP address			
Client HW address (16 bytes)			
Server host name (64 bytes)			
Boot file name (128 bytes)			
Options 312 bytes			

Funzionamento della procedura:

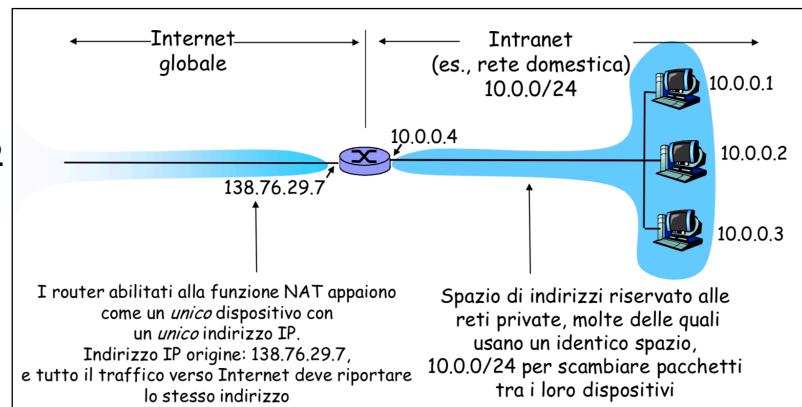
1. Il nuovo host invia un **DHCP_DISCOVER** in broadcast per identificare il server DHCP
2. Il server risponde con **DHCP_OFFER** con cui assegna l'ip
3. Host invia al server un **DHCP_REQUEST** per chiedere ulteriori parametri di configurazione ed estensione temporale per l'utilizzo
4. Server risponde con **DHCP_ACK** con le configurazioni richieste

OSS non molto sicuro, permette accesso a utenti non autorizzati

Network address translator (NAT)

Altro metodo per ridurre l'occupazione degli indirizzi, utilizzato per **Intranet**, associa ad ogni terminale della sottorete un IP privato non visibile all'esterno della rete LAN.

L'unico ad avere l'ip pubblico è il router che possiede anche la funzione NAT, ogni pacchetto indirizzato ad un host della sottorete deve riportare questo IP. Un router NAT quindi permette di rappresentare un intera rete LAN con un unico IP, nascondendo tutti i dettagli della rete intranet. Inoltre permette di scambiare gli indirizzi dei terminali senza che tutta la rete globale lo sappia.

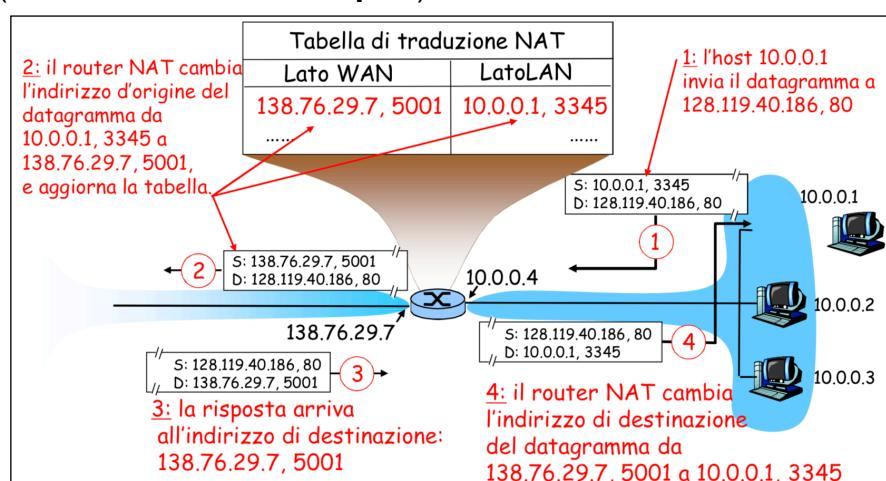


Per il suo funzionamento la NAT sfrutta l'indirizzo specifico dello strato di trasporto, la porta (sfocia quindi nell'utilizzo del 4 strato), in questo modo:

- Il router riceve un pacchetto di un host che vuole comunicare col mondo esterno
- Il router modifica l'indirizzo privato d'origine con quello pubblico, ed il numero di porta con uno nuovo casuale (evitando le **well-known port**).

Quindi aggiorna la tabella di traduzione

- Arriva la risposta dalla destinazione, la quale possiede come destination address l'ip del router e come porta quella scelta prima.
- Il router quindi legge il datagramma, modifica ip e porta con quelli privati, poi lo spedisce al destinatario corretto



OSS il campo numero di porta è a 16 bit, quindi NAT può supportare più di 60.000 connessioni simultanee con un solo IP sul lato WAN

Tra i metodi per la corretta associazione degli IP privati si hanno:

- Associare sempre alla stessa porta lo stesso terminale
- Universal Plug and Play (UPnP)
- Internal Gateway Device Protocol (IGD)
- Connessione tramite server esterno in relay (usato da Skype)

Ci sono alcuni punti che vengono contestati di questa tecnica:

- Operando su 4 strati è contrario ai principi dell'architettura a strati
- Un host non è visibile all'esterno (rende eventuali operazioni illegali irrintracciabili)
- Interferisce con applicazioni P2P
- Incompatibile con protocollo ICMP
- Al momento del cambio si IP deve essere ricalcolato checksum di pacchetti UDP e TCP

Forwarding (inoltro)

Consiste nella compilazione delle tabelle di routing, per far ciò i router si scambiano appositi pacchetti di controllo, che tramite appositi protocolli forniscono ai nodi un'idea della topologia della rete.

Questi sono implementati tramite gli algoritmi di routing che permettono di determinare i percorsi migliori, ovvero cammino minimo, costo minore e massima affidabilità.

I path non rimangono sempre gli stessi ma vengono aggiornati ogni 30 secondi.

I protocolli di routing necessitano dei seguenti requisiti:

- **Risposta alla variazione di stato** => cambiamenti topologici, congestione, guasti
- **Ottimalità** => utilizzo ottimale delle risorse di rete, minimizzazione dei cammini
- **Robustezza** => continuità del servizio anche in condizioni anomale
- **Semplicità** => basso carico di elaborazione

l'instradamento può essere di 2 tipi:

- **Statico** => cammini configurati manualmente e non subiscono cambiamenti, usati solo con reti piccole o per impostare determinati percorsi
- **Dinamico** => i cammini vengono calcolati autonomamente sulla base delle informazioni ricevute per mezzo dei protocolli di routing

Inoltre si può avere un instradamento centralizzato gestito da un solo nodo centrale oppure instradamento distribuito in cui ogni nodo determina i suoi cammini.

Le tabelle di routing sono costituite da

<<IP destinazione, subnet_mask, IP next-hop, interfaccia, metrica¹¹>>

e sono formate da tante righe quanti sono i nodi e le sottoreti a cui inviano pacchetti, motivo per cui le reti IP sono state gerarchizzate, quindi non conoscono tutti i next-hop.

Inoltre i router posseggono un **topological database** che memorizza le informazioni topologiche, queste favoriscono la determinazione periodica dei percorsi minimi.

La rete è quindi controllata da un insieme di **Autonomous System (AS)**, ovvero router ed host controllati da un'autorità amministrativa (ex. Un ISP).

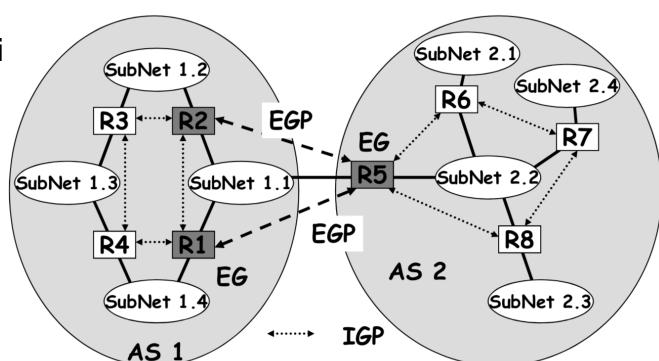
Ogni AS ha il proprio protocollo di instradamento, questi sono poi collegati tra loro tramite i router di confine.

OSS Un particolare AS è il “core AS” che costituisce il back bone di internet.

I protocolli di routing che girano all'interno di un sistema autonomo sono chiamati

interior gateway protocols (IGP), quelli che girano nei sistemi esterni vengono detti **exterior gateway protocols (EGP)**.

È importante notare che molti pacchetti non prendono determinate strade, anche se più efficienti, per questioni politiche o amministrative.



¹¹ metrica = costo per arrivare al nodo

Algoritmi di instradamento

La rete può essere rappresentata come un **grafo pesato** $G = (N, E, c)$, dove:

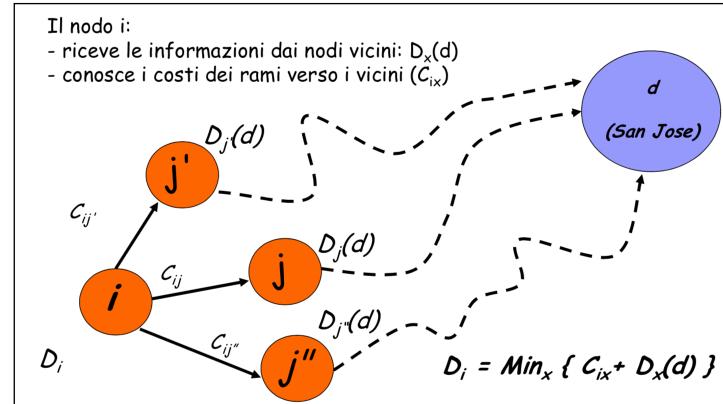
- $N \Rightarrow$ insieme dei nodi
- $E \Rightarrow$ insieme degli archi
- $c =$ costi associati ai nodi

Il costo di un cammino è definito dalla somma dei costi di tutti gli archi lungo il percorso, per il calcolo del cammino si possono usare 2 algoritmi:

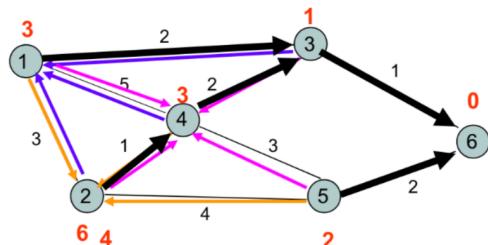
Bellam-Ford

Implementato dalla famiglia dei **distance vector protocols**, ogni nodo comunica solo con i suoi rami, questi a loro volta comunicano solo con i loro rami senza conoscere la topologia. I router vicini quindi si scambiano i **distance vector**

[DV = (destinazione, distanza)] con i quali determinano il next-hop migliore.



Ex. Dato il grafo in figura ricavare albero costi minimi per arrivare alla destinazione (6)



Iteration	Node 1	Node 2	Node 3	Node 4	Node 5
Initial	(-1, ∞)				
1	(-1, ∞)	(-1, ∞)	(6, 1)	(-1, ∞)	(6, 2)
2	(3, 3)	(5, 6)	(6, 1)	(3, 3)	(6, 2)
3	(3, 3)	(4, 4)	(6, 1)	(3, 3)	(6, 2)

Funzionamento:

5. Per prima cosa si inizializzano tutti i nodi con i valori di default $(-1, \infty)$
6. Partendo dalla destinazione si chiede il valore dei distance vector ai rami vicini che vengono quindi salvati in una nuova riga della tabella
7. Si ripete quindi la stessa operazione per i nodi appena trovati
ATTENZIONE il valore della distanza che viene salvato sulla tabella non è quello per raggiungere il nodo che effettua la richiesta ma quello per arrivare a destinazione.
8. Si continua quindi ad iterare il passaggio 3, da notare che, anche se si arriva a raggiungere tutti i nodi, potrebbero esserci archi ancora non visitati con costi minori.
È quindi importante arrivare a visitare tutti i nodi del grafo

La prima colonna della tabella indica il numero massimo di hop da attraversare. Ogni cella dell'ultima riga rappresenta una riga della routing table del rispettivo nodo, ad esempio la tabella del nodo 2 avrà :

`<<destination IP: ip di 6, subnet_mask, IP next-hop: ip di 4, metrica: 4>>`

Da notare che la subnet_mask di un link è sempre /30, poiché si necessita solo di 4 indirizzi (2 speciali + 2 per i router), ovvero 2 bit per l'host

Tornando all'esempio, se si guastasse il link (3,6) ed il nodo 3 modificasse la sua tabella senza notificare il guasto, si avrebbe che 4 continuerebbe ad inviargli i pacchetti per vederseli tornare indietro, creando loop infiniti.

Una soluzione per ovviare al problema dei loop infiniti è la **split horizon con poison reverse** nella quale il router imposta la propria distanza ad ∞ .

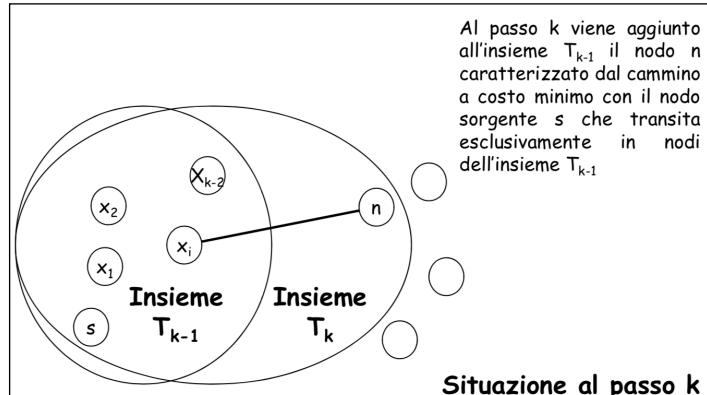
Dijkstra

Implementato dalla famiglia dei **link state protocols**, i router conoscono l'intera topologia della rete ed ognuno calcola lo shortest path verso destinazione.

Individua un cammino di lunghezza minima tra un nodo s e tutti gli altri nodi del grafo procedendo in modo da aumentare progressivamente la distanza.

Procede a passi successivi:

1. Al k -esimo passo sono individuati i k nodi raggiungibili dal nodo sorgente tramite cammini minimi (i k nodi formano l'insieme T_k)
2. Al passi $k+1$ si individua il nodo n che è caratterizzato dal cammino minimo verso il nodo s e che transita esclusivamente nei nodi dell'insieme T_k , in questo modo si forma un nuovo insieme T_{k+1}
3. Termina quando sono stati esplorati tutti i nodi



Situazione al passo k

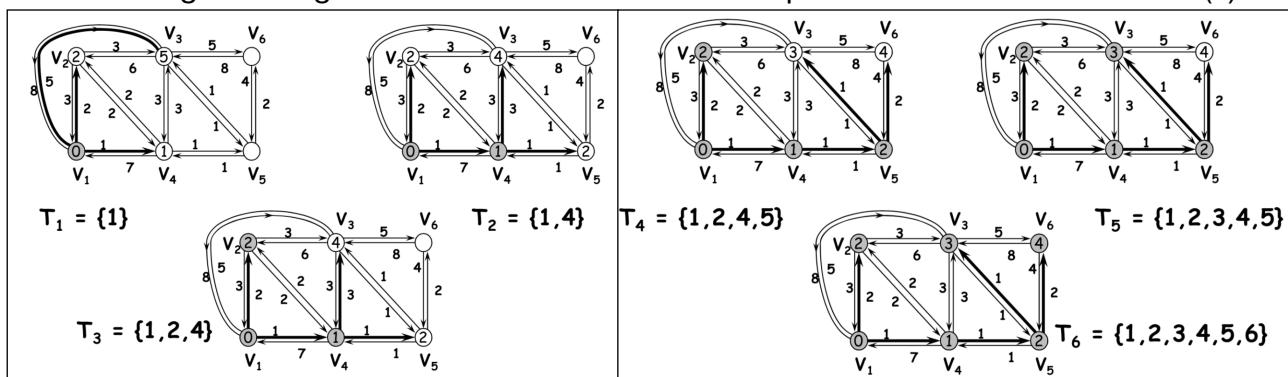
Costo $O(n^2)$

Ex.1 Dato il grafo in figura ricavare albero costi minimi per arrivare alla destinazione (u)

passo	N'	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	u	2,u		5,u	1,u	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					

Ogni elemento cerchiato rappresenta il costo minimo per il nodo della colonna corrispondente, inoltre indica quale sarà il nodo da visitare al prossimo passo. Rappresenta anche una riga della routing table del rispettivo nodo.

Ex.2 Dato il grafo in figura ricavare albero costi minimi per arrivare alla destinazione (0)



Protocolli di routing

I protocolli di instradamento che implementano algoritmi sono IGP, quindi girano su AS. I 2 protocolli più famosi sono:

RIP (routing information protocol)

Implementato tramite Bellam-Ford, ogni router invia al vicino la propria tabella di routing contenente tutte le sue sottoreti.

Questo comporta un alto costo quindi viene usato solo per piccole-medie reti.

Utilizza protocollo UDP (port Number 520), i router si scambiano 2 tipi di messaggi:

- **Request** => per chiedere ai vicini il distance vector
- **Response** => annuncia il distance vector

Gli aggiornamenti vengono scambiati ogni 30 secondi. Se non ricevo notizie da un nodo adiacente entro 180 secondi allora questo viene considerato spento o guasto, quindi ogni router interessato effettua una modifica alla routing table ed avvisa i vicini.

Command	Version	0
Address Identifier	0	
IP Address 1		
0		
0		
Metric for address 1		
Address Identifier	0	
IP Address 2		
0		
0		
Metric for address 2		
⋮		
Address Identifier	0	
IP Address N		
0		
0		
Metric for address N		

OSPF (open shortest path first)

Implementato tramite Dijkstra, i router si scambiano le informazioni per avere un immagine topologica della rete, utilizzato per reti geografiche.

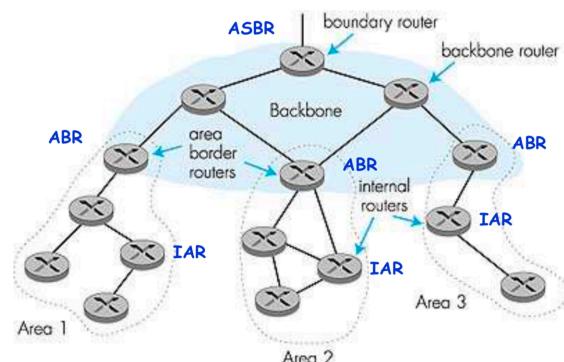
I messaggi che vengono scambiati sono i **link state advertisement (LSA)** e vengono trasmessi tramite la tecnica di **flooding**, con le seguenti proprietà:

- Vengono emessi ogni volta che si ha un cambiamento di stato di un link o dopo determinati periodi di tempo
- Hanno dei numeri di sequenza o riferimenti temporali
- Vengono rilanciati da ogni router su tutte le interfacce tranne quella d'arrivo, inondando la rete.

L'ultima proprietà crea però dei problemi, infatti con una rete molto ampia il traffico può essere elevato, si è pensato quindi all'istradamento gerarchico.

Una rete IP viene suddivisa in aree in aree interconnesse dalla backbone, i router sono suddivisi in:

- **Intra-area router (IAR)** => router interni all'area che inviano LSA agli altri router interni
- **Area border router (ABR)** => router collegati a 2 o più aree, gerarchicamente più alti
- **As boundary router (ASBR)** => router di confine delle AS, scambiano LSA all'interno con informazioni sui percorsi esterni



Dettagli sui messaggi nelle slides del corso

BGP (border gateway protocol)

Standard attuale per protocolli EGP, utilizzato per far comunicare AS diversi.

Protocollo sofisticato che deve considerare la coesistenza di sottoprotocolli differenti ed eventuali problemi topologici o politici-amministrativi.

Lo strato di trasporto

Strato che opera da estremo a estremo, ha il compito di connettere le applicazioni. Ogni applicazione è identificata da un indirizzo chiamato **port number**, l'assegnazione viene effettuata da **IANA (internet assigned numbers authority)**, stessa di IP.

Le prime 1023 porte sono utilizzate per servizi specifici e sono dette **well known port** (ex. 80 HTTP), le altre possono essere utilizzate liberamente dalle applicazioni sconosciute o per stabilire comunicazioni.

Ogni connessione è identificata da un socket che dipende dal protocollo:

- UDP => *IP destinazione + n° di porta destinazione*
- TCP => *IP destinazione + n° di porta destinazione + IP sorgente + n° di porta sorgente*

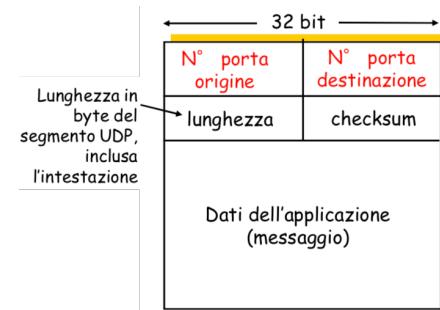
Il numero di porta sorgente viene scelto >1023 in modo da essere unico e riconoscibile, questa tecnica rappresenta la multiplazione.

Quindi i protocolli principali di questo strato sono 2:

- **UDP (user datagram protocol)** [RFC 768] => protocollo semplice senza connessione, molto usato (ex. DNS).

Il segmento è composto da:

- n° di porta origine
- n° di porta destinazione
- Lunghezza totale
- Checksum
- Dati

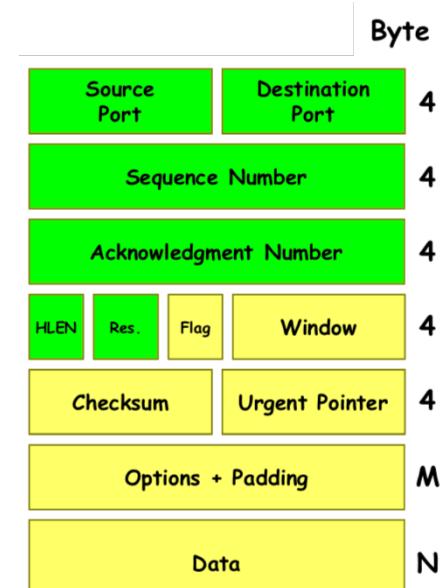


- **TCP (transport control protocol)** [RFC 793, 1122, 1323, 2018, 2581] => molto più complesso, servizio con connessione permette le seguenti funzioni:

- A. Indirizzamento applicazioni
- B. Controllo sequenza delle unità informative
- C. Controllo e recupero d'errore
- D. Controllo di flusso
- E. Controllo di congestione

Un segmento TCP è formato da:

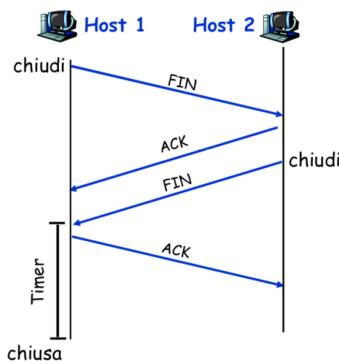
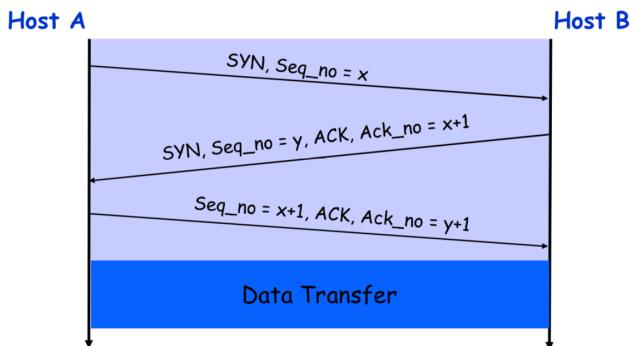
- Porte sorgente e destinazione
- Numero di sequenza del pacchetto
- Numero di acknowledgment
valido solo se bit ack del campo flag è “1”
- Lunghezza (HLEN) e bit per usi futuri
- Flag (vedere slides)
- Window, campo usato per controllo di flusso
- Checksum
- Urgent pointer che contiene numero di sequenza dell'ultimo byte dei dati che devono essere consegnati urgentemente al processo ricevente
- Options non sempre presente con padding per renderlo multiplo di 32



Approfondimento TCP

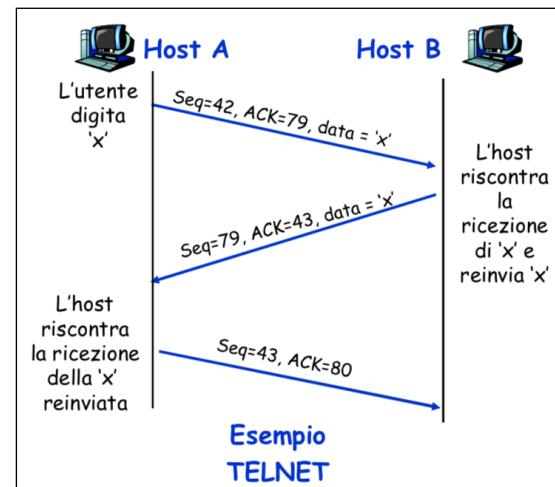
TCP è un protocollo connection-oriented, per stabilire la connessione utilizza una tecnica chiamata **three-way handshake**, il cui funzionamento:

1. Host A invia un segmento SYN a B in cui specifica solo il numero di sequenza iniziale senza inviare dati
2. Host B risponde con un SYN ack in cui specifica il numero di sequenza iniziale del server
3. A risponde con un ack, questo può già contenere dati



Una connessione aperta occupa un socket, quindi per chiuderla:

1. Host A invia segmento FIN a B
2. B riceve FIN e risponde con un ack
3. B chiude la connessione ed invia FIN
4. A riceve FIN e risponde con ack



Funzioni del TCP:

- **Controllo di sequenza** => il numero di sequenza corrisponde al primo byte del segmento nel flusso di byte.
L'ack è cumulative e riporta il numero del prossimo byte atteso dall'altro lato.
- **Controllo d'errore** => uguale a strato 2, il tempo di rinvio però è stimato dinamicamente a seconda della situazione del traffico.
Sfrutta il round trip time (RTT) e le sue oscillazioni nel tempo.

Retransmission TimeOut (RTO) = RTT stimato + 4devRTT
con devRTT deviazione standard del RTT stimato

- **Controllo di flusso** => utilizza il campo window, chi riceve scrive quanti segmenti è disposto a ricevere così che il mittente adatti la propria finestra di trasmissione
- **Controllo di congestione** => si compone di 2 fasi:
 1. Slowstart => un host invia i segmenti con velocità sempre più alta per testare la rete, fino a quando riceve riscontri che aumentano, altrimenti passa a fase 2
 2. Congestion avoidance => si è raggiunto il limite quindi protocollo ha superato velocità consentita