



Reti di Calcolatori

Prof. Riccobene Salvatore
AA 2022/23



REALIZZATO DA
Giulio Pedicone

REALIZZATO DA
Vincenzo Villanova



Indice

- **LIVELLO APPLICATIVO**

- [Architettura delle applicazioni di rete](#)
- [Processi comunicanti](#)
- [Well Known Ports, Registered Ports e User Ports](#)
- [Servizi di trasporto disponibili per le applicazioni](#)
- [Servizi di trasporto offerti da Internet](#)
- [Panoramiche di HTTP](#)
- [Connessioni persistenti e non persistenti](#)
- [Formato dei messaggi HTTP](#)
- [Interazione utente-server: i cookie](#)
- [FTP e differenze con HTTP](#)
- [Protocollo SMTP](#)
- [POP e IMAP](#)
- [DNS](#)
- [SNMP](#)

- **LIVELLO DI TRASPORTO**

- [Introduzione e servizi a livello di trasporto](#)
- [Multiplexing e Demultiplexing](#)
- [Server Multipli](#)
- [Trasporto non orientato alla connessione: UDP](#)
- [Struttura dei segmenti UDP](#)
- [Costruzione di un protocollo di trasferimento dati affidabile](#)
- [Definizioni utili per il livello di trasporto](#)
- [Stop And Wait, Go Back-N, Ripetizione Selettiva](#)
- [Trasporto orientato alla connessione: TCP](#)
- [Struttura dei segmenti TCP](#)
- [Timeout e stima del tempo di andata e ritorno](#)
- [Ritrasmissione Rapida](#)
- [Controllo di Flusso](#)
- [Gestione della connessione TCP](#)
- [Cause e costi della congestione](#)
- [Approcci al controllo di congestione](#)
- [Controllo di congestione TCP](#)
- [Slow Start](#)
- [Congestion Avoidance](#)
- [Fast Recovery](#)
- [Retrospettiva sul controllo di congestione di TCP](#)
- [TCP Tahoe VS TCP Reno](#)

- **LIVELLO DI RETE**
 - [Inoltro ed Instradamento](#)
 - [Piano di controllo: Tradizionale VS SDN](#)
 - [Modelli di servizio](#)
 - [Che cosa si trova all'interno di un router?](#)
 - [Inoltro basato sull'indirizzo di destinazione](#)
 - [Struttura di commutazione](#)
 - [Dove si verifica l'accodamento?](#)
 - [Schedulazione dei pacchetti](#)
 - [Formato dei datagrammi IPv4](#)
 - [Frammentazione dei datagrammi IPv4](#)
 - [Indirizzamento IPv4](#)
 - [LAN Interconnesse](#)
 - [Come ottenere l'indirizzo di un host: DHCP](#)
 - [NAT \(Network Address Translation\)](#)
 - [ICMP](#)
 - [ARP, RARP, BOOTP](#)
 - [IPv6](#)
 - [Formato dei datagrammi IPv6](#)
 - [Firewall e DMZ](#)
 - [Algoritmi di Routing](#)
 - [Flooding](#)
 - [Distance Vector \(DV\)](#)
 - [Link State Routing \(LSR\)](#)
 - [Link State Routing VS Distance Vector](#)
 - [Internet Routing](#)
 - [Routing Internet Protocol \(RIP\)](#)
 - [Open Shortest Path First \(OSPF\)](#)
- **LIVELLO DLL**
 - [Framing dei dati](#)
 - [Tecniche di rilevazione e correzione degli errori](#)
 - [Distanza di Hamming](#)
 - [Codice di Hamming](#)
 - [Protocolli del Data Link per il MAC](#)
 - [Protocolli a suddivisione del canale](#)
 - [Protocolli ad accesso casuale](#)
 - [Protocolli senza collisioni](#)
 - [Protocolli a turno](#)
 - [IEEE 802.1 / 802.2 / 802.3](#)
 - [Cablaggi](#)
 - [Ethernet](#)
 - [Codifica Manchester](#)
 - [Fast Ethernet e Gigabit Ethernet](#)
 - [Schemi di Trellis e decodifica di Viterbi](#)
 - [Schemi di interconnessione in Ethernet](#)
 - [Repeater, Hub, Bridge, Switch](#)
 - [Schemi di indirizzamento: Piatto VS Gerarchico](#)
 - [LAN Virtuali \(VLAN\)](#)

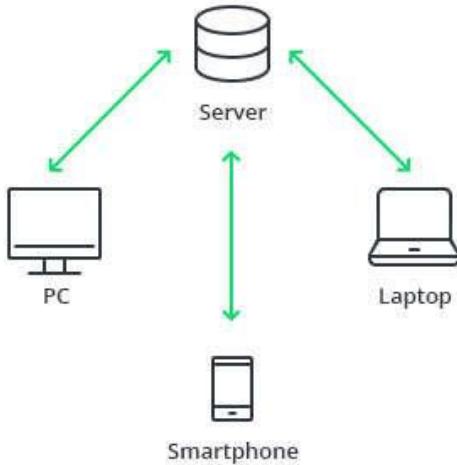
Livello Applicativo

Architettura delle applicazioni di rete

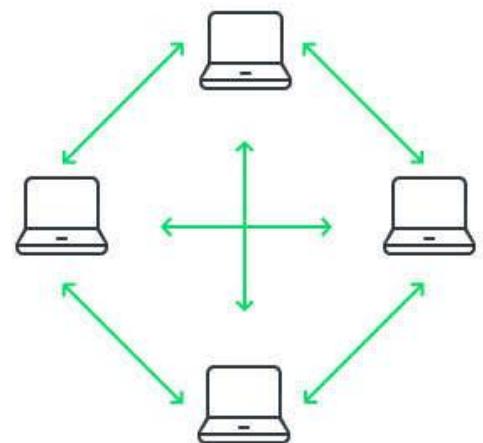
Il compito del livello applicativo è quello di pianificare come l'applicazione funzionerà su vari dispositivi, scegliendo tra due tipi principali di reti: **Client-Server** o **peer-to-peer**.

Nell'architettura **Client-Server** vi è un host sempre attivo, chiamato **server**, che risponde alle richieste di servizio di molti altri host, detti **client**. I client non possono interagire direttamente tra loro. Inoltre, il server dispone di un indirizzo fisso, e noto, detto indirizzo IP. Il client può quindi contattare il server in qualsiasi momento, inviandogli un pacchetto. Spesso in un'applicazione **Client-Server** un solo host server non è in grado di rispondere a tutte le richieste dei client. Per questo motivo, nelle architetture **Client-Server**, è comune utilizzare i **data center**.

L'architettura **P2P**, si basa sulla comunicazione diretta tra coppie di host chiamati "**peer**", che sono collegati in modo intermittente. Ogni **peer** è autonomo e può comunicare direttamente con altri peer all'interno della rete. Il modello P2P è altamente **scalabile**. Poiché i peer possono essere aggiunti o rimossi facilmente, adattandosi facilmente alle dimensioni della rete.



Client-server

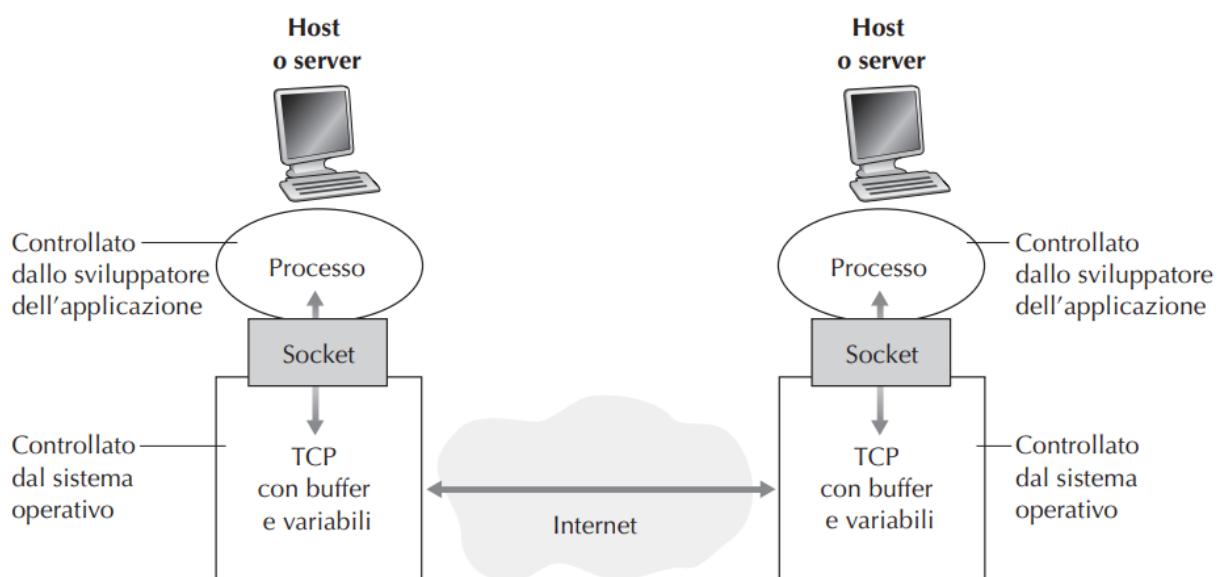


P2P network

Processi comunicanti

Si può pensare a un **processo** come a un **programma in esecuzione** su un sistema, i processi su due dispositivi si **comunicano** scambiando **messaggi** tramite una **rete**. Nel contesto di una sessione di comunicazione tra una coppia di processi quello che **avvia la comunicazione** (cioè, contatta l'altro processo all'inizio della sessione) è indicato come **client** mentre quello che **attende** di essere contattato per iniziare la sessione è detto **server**. Un **processo** invia **messaggi** nella rete e riceve messaggi dalla rete **attraverso** un'interfaccia software detta **socket**. Un processo è assimilabile a una **casa** e le socket sono il corrispettivo delle **porte**. Un processo che vuole inviare un messaggio a un altro processo o a un altro host, fa uscire il messaggio dalla propria porta (socket).

In Internet, gli host vengono identificati attraverso i loro **indirizzi IP**. Oltre a conoscere l'indirizzo dell'host cui è destinato il messaggio, il mittente deve anche identificare il processo destinatario, più specificatamente **la socket** che deve ricevere il dato. Un **numero di porta** di destinazione assolve questo compito.



Well Known Ports, Registered Ports e User Ports

Le porte in una rete possono essere suddivise in tre categorie principali: **porte ben note** (well-known), **porte registrate** (registered), e **porte utente** (user). Ecco le differenze tra queste categorie:

- **Porte Ben Note** (Well-Known Ports):
 - **Numeri di Porta:** Le porte ben note variano da 0 a 1023.
 - Queste porte sono riservate per servizi noti e ampiamente riconosciuti a livello globale.
 - **Esempi:** Porta 80 (HTTP), Porta 443 (HTTPS), Porta 22 (SSH).
- **Porte Registrate** (Registered Ports):
 - **Numeri di Porta:** Le porte registrate variano da 1024 a 49151.
 - Sono utilizzate da applicazioni personalizzate e specifiche dell'utente.
 - **Esempi:** Porta 3306 (MySQL), Porta 5432 (PostgreSQL).
- **Porte Utente** (Dynamic Ports - User Ports):
 - **Numeri di Porta:** Le porte utente variano da 49152 a 65535.
 - Liberamente utilizzabili da tutte le applicazioni utente

Ports	
20/tcp	FTP - data
21/tcp	FTP - control
22/tcp	SSH - Secure login
23/tcp	Telnet
25/tcp	SMTP
53/tcp	DNS
53/udp	DNS
67/udp	BOOTP (Server) and DHCP (Server)
68/udp	BOOTP (Client) and DHCP (Client)
69/udp	TFTP
70/tcp	Gopher
80/tcp	HTTP
88/tcp	Kerberos Authenticating agent
110/tcp	POP3
123/udp	NTP
143/tcp	IMAP4
161/udp	SNMP (Agent)
162/udp	SNMP (Manager)
443/tcp	HTTPS
465/tcp	SMTP over SSL
993/tcp	IMAP4 over SSL
995/tcp	POP3 over SSL

Servizi di trasporto disponibili per le applicazioni

Li possiamo classificare a grandi linee secondo **quattro dimensioni**:

- **Trasferimento dati affidabile**
- **Throughput** (Tasso al quale il processo mittente può inviare i bit al processo ricevente)
- **Temporizzazione** (Ritardo di comunicazione minimo)
- **Sicurezza** (Integrità, Confidenzialità, Autenticazione, Non ripudio)

Servizi di trasporto offerti da Internet

Internet (come ogni rete TCP/IP) mette a disposizione delle applicazioni due protocolli di trasporto: **UDP** e **TCP**.

- **UDP** è un protocollo di trasporto che offre un servizio di trasferimento dati **non affidabile**, senza connessione preliminare, consegna garantita o controllo della congestione.
- **TCP** offre un servizio orientato alla connessione, che include una procedura di **handshaking** per stabilire una connessione. Inoltre, TCP fornisce un servizio di **trasferimento affidabile dei dati**, garantendo che i dati siano consegnati senza errori e nell'**ordine corretto** tra le **socket** di mittente e destinatario. TCP include anche un meccanismo di **controllo della congestione** per gestire il traffico in rete in modo da evitare eccessive sovraccariche.

Panoramiche di HTTP

HTTP (Hypertext Transfer Protocol) è il protocollo fondamentale del World Wide Web. Questo protocollo funziona attraverso **client** e **server** che si scambiano messaggi HTTP. Le pagine web sono composte da **oggetti**, come file HTML, immagini, video, ecc., indirizzabili tramite URL.

Ogni **URL** ha due componenti:

- il **nome dell'host** del server che ospita l'oggetto
- il **percorso** dell'oggetto.

Per esempio, l'URL `http://www.someSchool.edu/someDepartment/picture.gif` ha:

`www.someSchool.edu` come **nome dell'host**

`/someDepartment/picture.gif` come **percorso**.

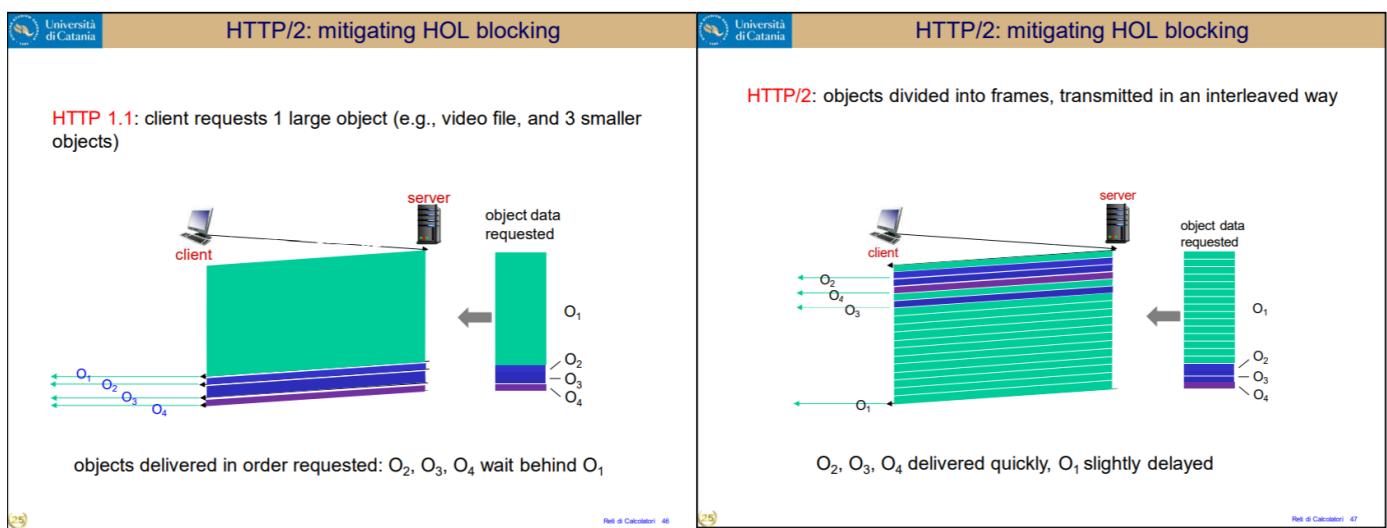
HTTP utilizza **TCP** come protocollo di trasporto per garantire la consegna affidabile dei dati. Quando un utente richiede una pagina web, il browser invia **richieste** HTTP al server, che risponde con messaggi di **risposta** HTTP contenenti gli oggetti richiesti. HTTP è uno dei protocolli "senza memoria di stato" (**stateless**) in quanto i server non mantengono informazioni sullo stato delle richieste dei client. Questo protocollo è essenziale per la **navigazione web** e la comunicazione tra client e server nel contesto del World Wide Web.

Connessioni persistenti e non persistenti

In molte applicazioni Internet, **client** e **server** comunicano a lungo termine, con il client che **invia** richieste e il server che **risponde**. La scelta chiave è se utilizzare **connessioni non persistenti**, dove ogni richiesta/risposta avviene su una connessione TCP separata, o **connessioni persistenti**, dove molte richieste/risposte possono essere gestite sulla stessa connessione. Connessioni non persistenti comportano **ritardi** e ulteriore carico sul server, mentre connessioni persistenti **riducono questi problemi**. HTTP 1.1 e HTTP/2 supportano connessioni persistenti con opzioni come il pipelining e la priorità.

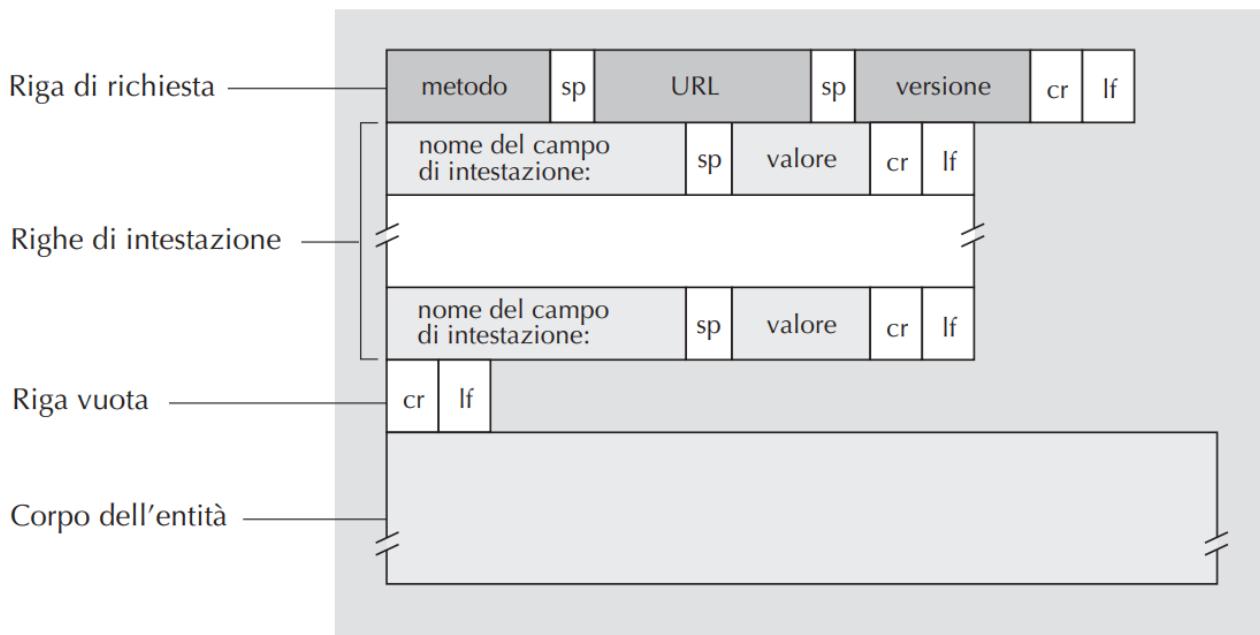
Versioni **HTTP**:

- **HTTP/1:** chiude ogni connessione TCP dopo che l'oggetto è stato trasferito.
- **HTTP/1.1:** utilizza connessioni TCP persistenti. Diversi oggetti possono essere inviati in pipeline utilizzando la stessa connessione TCP, che verrà chiusa solo alla fine.
- **HTTP/2:** Permette la compressione dell'intestazione (header), l'operazione di push ed il download parallelo.
- **HTTP/3:** È basata sul protocollo di trasporto QUIC (Quick UDP Internet Connections) e utilizza UDP al posto di TCP per ridurre la latenza. È una versione avanzata di HTTP che mira a fornire una navigazione web più veloce e sicura.



Formato dei messaggi HTTP

Messaggio di richiesta HTTP



Il **messaggio di richiesta** è composto da **cinque righe** separate da caratteri di ritorno a capo e nuova linea. La prima riga è la "riga di richiesta", contenente il **metodo** (**GET, POST, HEAD, PUT, DELETE**), l'**URL** e la **versione** di HTTP.

Le righe di intestazione forniscono ulteriori informazioni (alcuni esempi):

- **Host** specifica l'host dell'oggetto richiesto.
- **Connection**: "close/keep-alive" indica al server se chiudere la connessione dopo l'invio dell'oggetto o meno.
- **User-agent** specifica il browser che effettua la richiesta.
- **Accept-language** indica la preferenza dell'utente per la lingua dell'oggetto.

Il messaggio di richiesta può contenere un **corpo** utilizzato principalmente nel metodo **POST** quando l'utente invia dati tramite un form. Il metodo **POST** consente di **inviare** dati specifici nel **corpo** del messaggio.

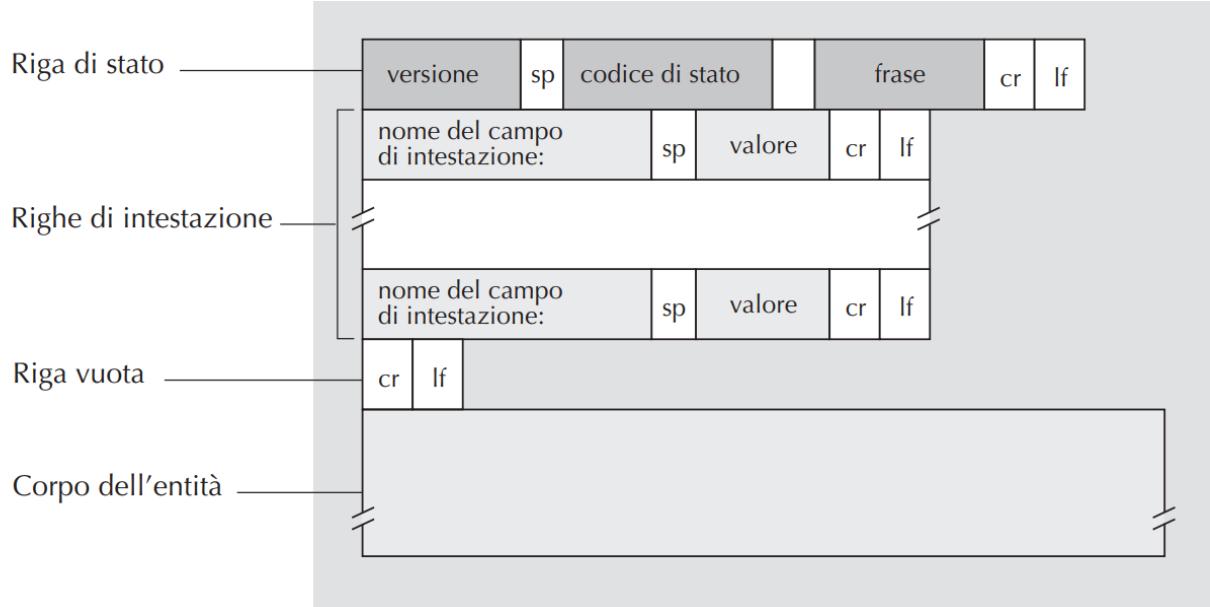
È importante notare che i form HTML spesso utilizzano il metodo **GET**, includendo i dati immessi nell'URL richiesto.

Altri **metodi** come **HEAD** consentono di verificare la correttezza del codice senza ottenere l'oggetto completo, mentre **PUT** e **DELETE** permettono di **inviare** e **cancellare** oggetti su un server web, rispettivamente.



PUT e **DELETE** sono stati rimossi per motivi di sicurezza!!

Messaggio di risposta HTTP



Il **messaggio di richiesta** è composto da **tre sezioni**:

- La **riga di stato** indica la **versione** del protocollo, il **codice di stato** e il **messaggio di stato**, che rappresentano il risultato della richiesta.
- Le **righe di intestazione** forniscono informazioni aggiuntive come la **chiusura della connessione**, la **data** di creazione e invio, il tipo di **server**, la **data** di **modifica** dell'oggetto, la **lunghezza** dell'oggetto e il **tipo** di contenuto.
- La **riga del corpo** del messaggio contiene **l'oggetto richiesto** o l'entità associata alla richiesta o alla risposta.

I **codici di stato** comuni includono:

- **200 OK**: Richiesta avvenuta con **successo** e informazioni inviate.
- **301 Moved Permanently**: L'oggetto richiesto è stato **trasferito** permanentemente con un nuovo URL specificato nell'intestazione "Location".
- **400 Bad Request**: **Errore generico** indicante che la richiesta non è stata compresa dal server.
- **404 Not Found**: Il documento richiesto **non è disponibile** sul server.
- **505 HTTP Version Not Supported**: Il server **non supporta** la versione del protocollo HTTP richiesta.

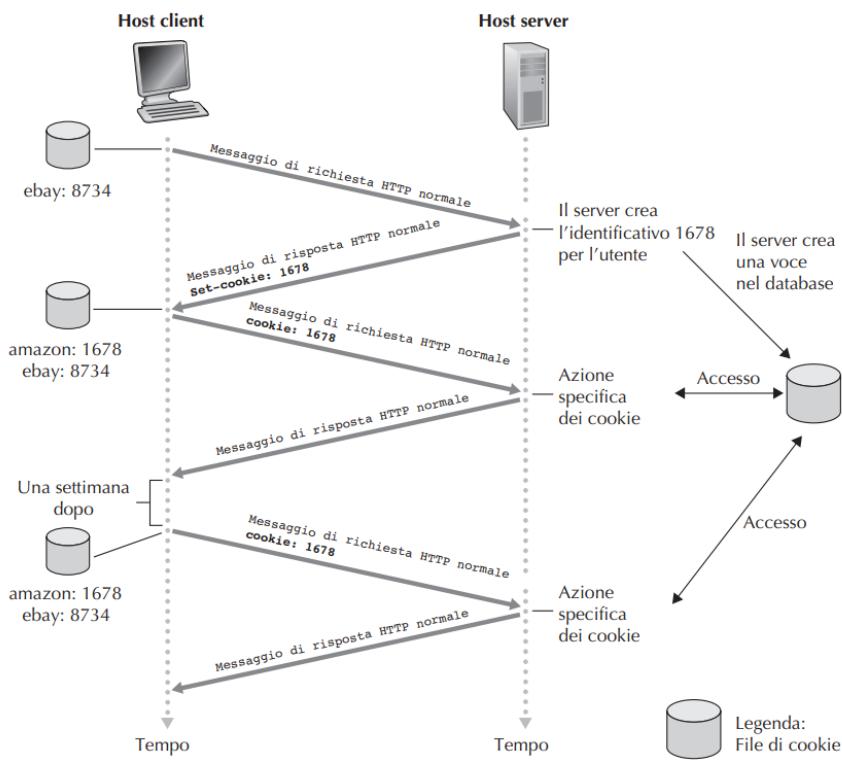
Interazione utente-server: i cookie

I **cookie** sono composti da **quattro componenti**: **una riga di intestazione** nel messaggio di **risposta HTTP**, **una riga di intestazione** nel messaggio di **richiesta HTTP**, un **file** memorizzato sul sistema dell'utente **gestito dal browser** e un **database** sul sito web.

Ecco come funzionano i **cookie**:

1. Quando un **utente** accede a un sito web per la prima volta, il **server** crea un identificativo unico e lo invia al **browser** dell'utente tramite l'intestazione "**Set-cookie**" nel messaggio di risposta **HTTP**.
2. Il **browser** dell'utente **memorizza** questo identificativo nel proprio file dei **cookie**, associandolo all'host del server.
3. Durante le visite successive al sito web, il browser **include** l'intestazione "Cookie" nei messaggi di richiesta **HTTP**, contenente **l'identificativo** unico.
4. Il server utilizza l'identificativo per **tenere traccia** delle **attività dell'utente** nel sito, come le pagine visitate e gli acquisti effettuati.
5. I **cookie** consentono al server di **offrire servizi** personalizzati, come il carrello della spesa virtuale o la memorizzazione delle preferenze dell'utente.
6. Se l'utente si registra nel sito web, il server può **associare ulteriori informazioni** personali all'identificativo nei suoi **database**.

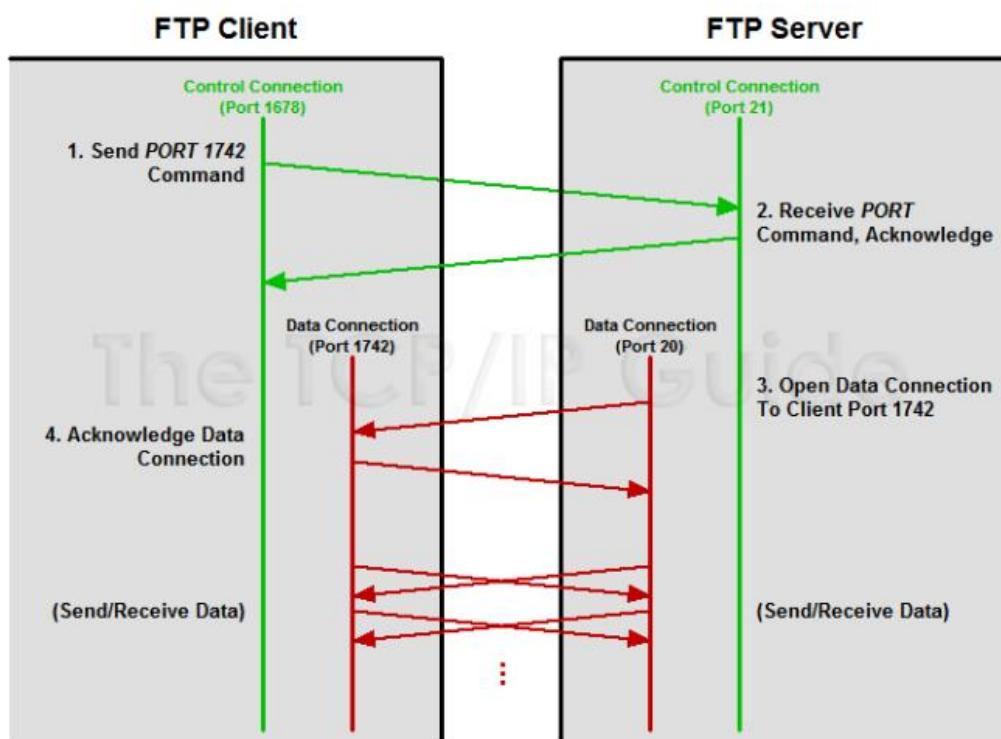
L'uso dei **cookie** può sollevare questioni sulla **privacy** degli utenti, poiché i siti web possono **raccogliere informazioni** sull'utente e venderle a terze parti. Tuttavia, i cookie sono una parte comune delle attività online e vengono utilizzati per **migliorare l'esperienza dell'utente** e fornire servizi personalizzati.



FTP e differenze con HTTP

Il **File Transfer Protocol** (FTP) è uno dei protocolli più antichi, utilizzato principalmente per il trasferimento di file su una rete. A differenza di HTTP, che è più moderno, FTP **utilizza due connessioni TCP separate su porte diverse**, ovvero la porta **20** per il **trasferimento dei dati** e la porta **21** per i **comandi**. Questo approccio è noto come "Sistema di connessione con dati fuori banda" e consente di avere canali distinti per l'invio di comandi e dati.

Per utilizzare **FTP**, è richiesta un'**autenticazione** e viene stabilita una **connessione persistente**, che rimane **aperta** finché l'utente non la **chiude** o finché non scatta un **timeout**. Una **differenza** rilevante con **HTTP** è che **FTP** supporta il **trasferimento bidirezionale** di dati.



Protocollo SMTP

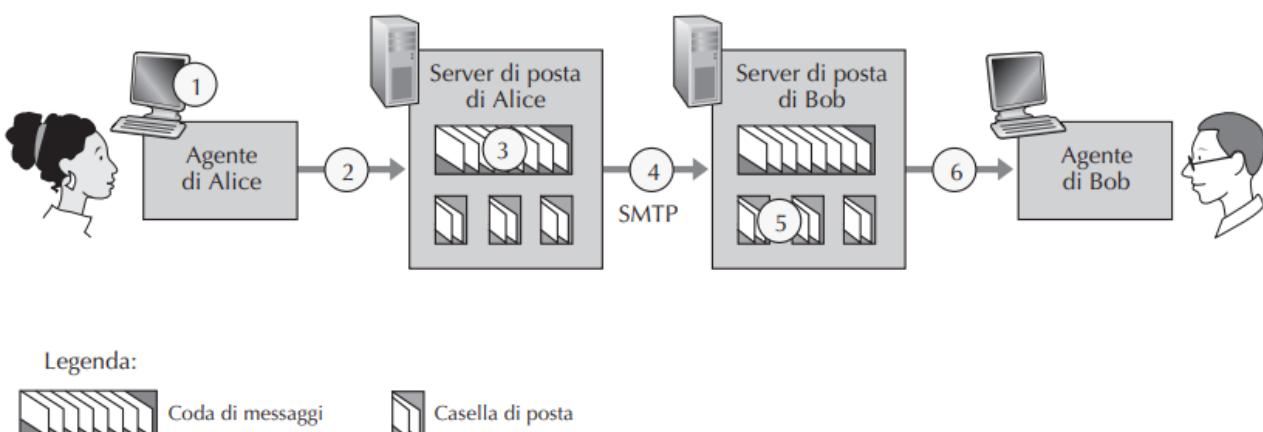
Il protocollo **SMTP** (Simple Mail Transfer Protocol) è utilizzato per **inviare** messaggi di posta elettronica su Internet. È il cuore della comunicazione **e-mail**. Tuttavia, ha alcune **caratteristiche "arcaiche"** **che possono limitare** la gestione dei dati multimediali, come immagini, audio o video, a causa della sua restrizione **all'ASCII a 7 bit**.

Ecco un riassunto delle **operazioni di base** di SMTP:

1. Il **mittente** (Alice) **compone un messaggio** e lo **invia** al suo **mail server** tramite un user agent per la posta elettronica.
2. Il **mail server** di Alice mette il messaggio in una **coda di messaggi**.
3. Il **client** SMTP sul mail server di Alice **stabilisce una connessione TCP** con il mail server del **destinatario** (Bob).
4. Dopo un **handshaking** SMTP, il client SMTP invia il messaggio.
5. Il mail server di Bob **riceve** il messaggio e lo **consegna nella casella di posta** di Bob.
6. Bob **legge il messaggio** quando lo desidera, utilizzando il suo **user agent** per la posta elettronica.

SMTP si basa su **TCP** per garantire la consegna **affidabile** dei messaggi. Una volta stabilita la connessione, il client SMTP invia il **messaggio**, e può contare su TCP per la consegna **senza errori**.

I **comandi** utilizzati tra i due mail server includono **HELO** (saluto), **MAIL FROM** (mittente), **RCPT TO** (destinatario), **DATA** (inizio del corpo del messaggio), **"."** (fine del messaggio), e **QUIT** (uscita).



POP e IMAP

POP (Post Office Protocol) e **IMAP** (Internet Message Access Protocol) sono due protocolli utilizzati per accedere alle caselle di posta elettronica su un server.

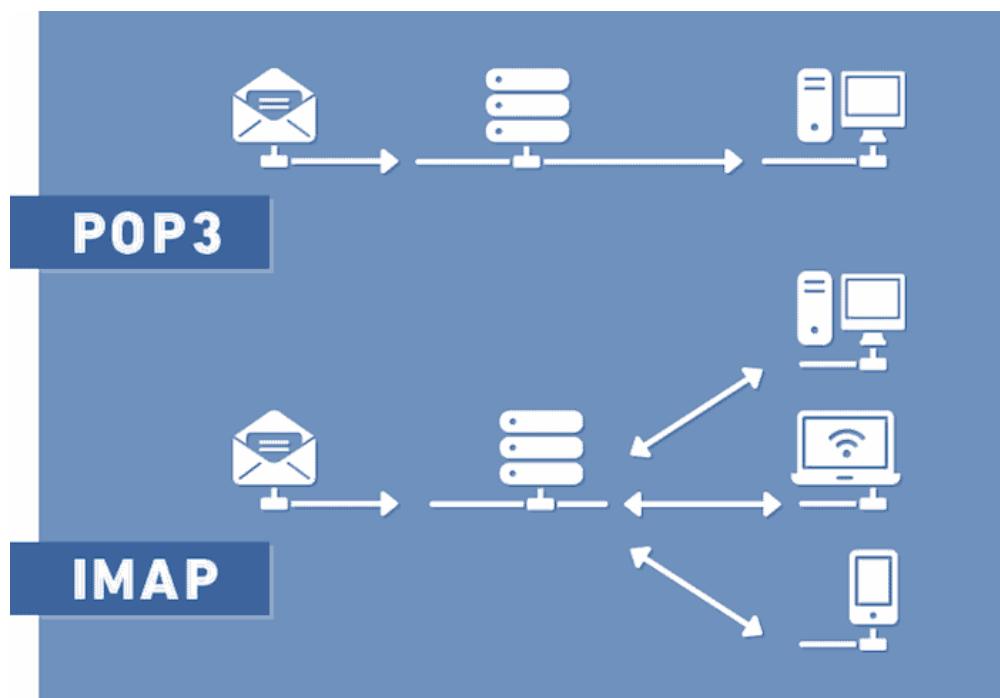
- **Download vs. Sincronizzazione:**

- **POP:** Scarica i messaggi dal server alla macchina del cliente. Una volta **scaricati**, i messaggi sono **rimossi** dal server. I messaggi sono quindi gestiti **localmente** dal cliente.
- **IMAP:** Sincronizza i messaggi tra il server e il client. I messaggi **rimangono sul server**.

- **Gestione dei Messaggi:**

- **POP:** Se il cliente accede da più dispositivi, i messaggi **non** sono **sincronizzati** tra di essi.
- **IMAP:** I messaggi sono gestiti in modo **centralizzato** sul **server**. Questo permette agli utenti di **accedere** alla stessa casella di posta da **più dispositivi**, mantenendo la coerenza tra di essi.

In sintesi, **POP** è più adatto per chi desidera **scaricare e archiviare** i messaggi **localmente**, mentre **IMAP** è ideale per coloro che desiderano **accedere** alle loro caselle di posta da **più dispositivi** e mantenerle **sincronizzate**. La **scelta** tra POP e IMAP dipende dalle **esigenze** dell'utente e dal modo in cui intende gestire la posta elettronica.



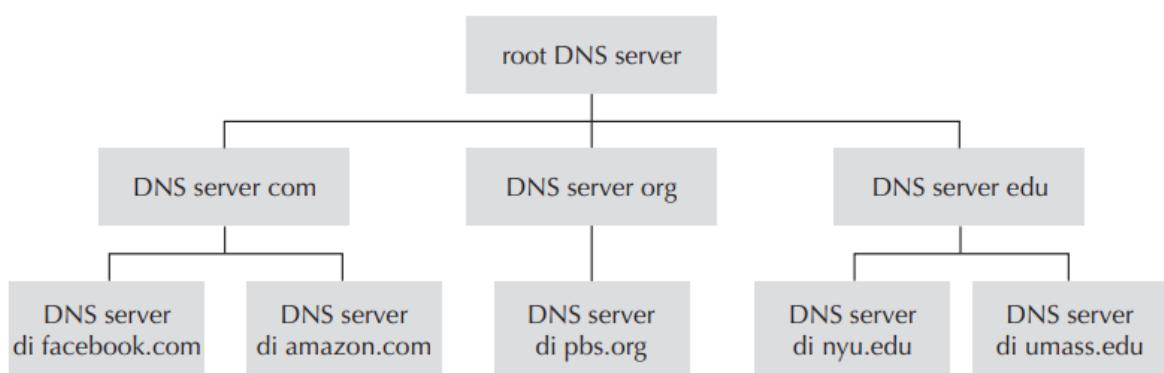
DNS: il servizio di directory di Internet

DNS (Domain Name System) ha lo scopo di **tradurre i nomi** di dominio (hostname) in **indirizzi IP** per identificare i dispositivi in rete. Il DNS funge sia da **database** distribuito contenente gli indirizzi IP, sia da **protocollo** che consente agli host di **interrogare** questi server per ottenere traduzioni. Tuttavia, questa traduzione aggiunge un **ritardo** nell'accesso a Internet.

Se il DNS **fosse basato** su un unico **server centralizzato**, ci sarebbe un **sovraffollamento** del database, più **vulnerabilità**, maggiore **lentezza** nell'accesso dalle regioni lontane e **difficoltà** di manutenzione. Di conseguenza, il **DNS** è progettato in modo **distribuito e gerarchico**, suddiviso in tre tipi principali di server:

- **Root server:** Questi **server** rappresentano **il livello più alto della gerarchia** e **forniscono** gli **indirizzi IP** dei server **top-level domain**.
- **Server top-level domain (TLD):** Questi server **gestiscono** i **domini** di primo livello come **com**, **org**, **edu**, ecc., e **forniscono** gli **indirizzi IP** dei server **autoritativi**.
- **DNS server autoritativi:** Ogni organizzazione con host pubblicamente accessibili su Internet deve fornire i propri indirizzi IP, ospitati sui server DNS autoritativi dell'organizzazione stessa.

I **server DNS locali** associati agli ISP fungono da **intermediari** tra gli host **utenti** e la gerarchia dei **server DNS**. Un aspetto cruciale del DNS è il servizio di **caching**, in cui ogni server DNS **conserva** in cache gli **indirizzi IP** delle ricerche **recenti** per velocizzare le future richieste e ridurre il carico sui server. Questo servizio permette al DNS di essere visto come un **grafo** invece di un albero, poiché ogni server può avere già la risposta in **cache** o indirizzare la richiesta al **server** giusto.



Cenni sul protocollo SNMP

SNMP (Simple Network Management Protocol) è un **protocollo** per la **gestione** e il **monitoraggio** di **dispositivi** di rete come **router** e **switch**. Consente agli amministratori di **raccogliere informazioni** sullo stato dei **dispositivi** e di eseguire **operazioni di gestione**. SNMP **organizza** le informazioni in una **struttura gerarchica** chiamata **MIB** (Management Information Base) e supporta **diverse versioni** con livelli di sicurezza variabili. In sostanza, è uno strumento cruciale per la gestione delle reti.

Livello Di Trasporto

Introduzione e servizi a livello di trasporto

Questo livello fornisce una **comunicazione logica**, consentendo ai processi di comunicare come se fossero **direttamente connessi**, anche se sono fisicamente separati da router e collegamenti di rete. I protocolli a livello di trasporto **convertono** i messaggi dei processi in **segmenti** e li **incapsulano** nei pacchetti di rete, che vengono inviati alla destinazione. Il livello di trasporto **estrae i segmenti** alla destinazione e li rende disponibili all'applicazione **destinataria**. Inoltre, diverse applicazioni di rete possono utilizzare protocolli a livello di trasporto diversi, come **TCP** e **UDP** in Internet, che forniscono servizi diversi alle applicazioni.

Multiplexing e Demultiplexing

Il **multiplexing** coinvolge la raccolta dei dati da diverse **socket**, **l'incapsulamento** in segmenti e l'inoltro ai livelli superiori. D'altra parte, il **demultiplexing** consiste nell'indirizzare i segmenti al processo **applicativo** corretto in base alle **socket** e agli identificatori univoci.

In sintesi, il **multiplexing** raggruppa i dati da diverse fonti in segmenti, mentre il **demultiplexing** instrada i segmenti ai processi appropriati sulla base delle socket e degli identificatori univoci.

Server Multipli

I **server multipli** sono una configurazione di rete in cui più server lavorano insieme per fornire servizi o risorse agli utenti. Servono a **distribuire il carico** di lavoro in modo **equo** tra i server, migliorando **l'affidabilità**, **la disponibilità** e **le prestazioni** dei servizi. Questa configurazione è spesso utilizzata per gestire siti web ad **alto traffico**, applicazioni online, servizi **cloud** e applicazioni aziendali, garantendo che un server possa coprire il lavoro degli altri in caso di **guasto** o aumento del carico di **lavoro**.

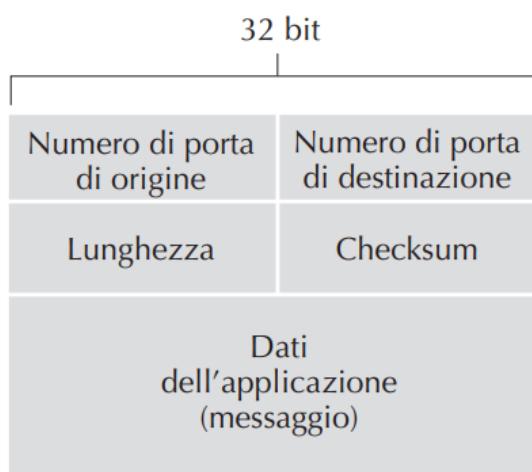
Trasporto non orientato alla connessione: UDP

Il protocollo **UDP** (User Datagram Protocol) è un protocollo di trasporto **minimale** che svolge il ruolo **essenziale** di multiplexing/demultiplexing, aggiungendo una minima gestione degli **errori**. UDP è noto per la sua mancanza di controllo di **congestione** e la mancanza di **handshaking** tra le entità mittente e destinataria a livello di trasporto, motivo per cui è definito "**non orientato alla connessione**".

UDP è spesso preferito a **TCP** in alcune applicazioni, come quelle in tempo reale (ad esempio, telefonia su Internet e **streaming** multimediale). Inoltre, UDP è utilizzato in applicazioni di gestione di rete (**SNMP**) e nel **DNS**.

In sintesi, UDP è un protocollo di trasporto **leggero** che offre un servizio di base senza alcune delle complessità di TCP, ma le applicazioni devono assumersi la **responsabilità dell'affidabilità** se la richiedono.

Struttura dei segmenti UDP



L'**intestazione** UDP è **semplice** e comprende **quattro** campi di due byte ciascuno:

- numeri di porta di **origine**
- numeri di porta di **destinazione**
- **lunghezza** che indica il numero di byte nel segmento UDP
- **checksum**

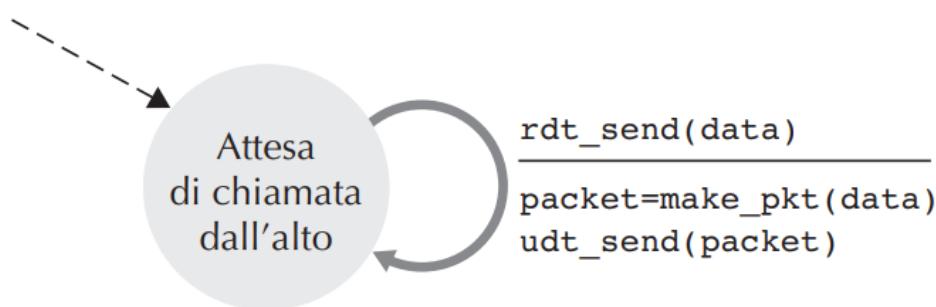
Il **checksum** UDP viene utilizzato per rilevare **errori** durante il trasferimento dei dati. Il checksum si basa su una somma di **complemento a 1** delle parole a 16 bit nel segmento, con l'eventuale riporto sommato al primo bit. Se durante il calcolo del checksum si **rilevano errori**, il **pacchetto** è considerato **danneggiato**.

Costruzione di un protocollo di trasferimento dati affidabile

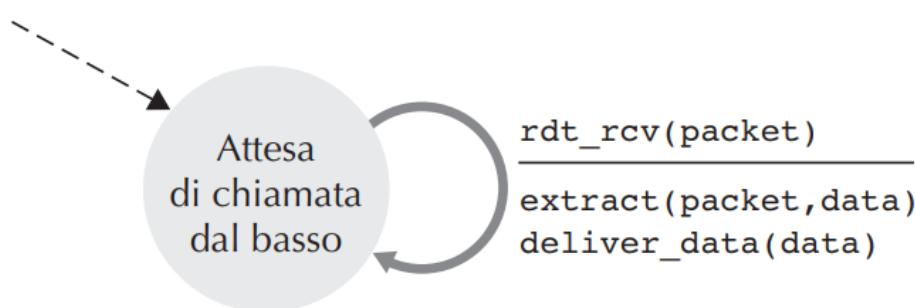
RDT 1.0: Trasferimento dati affidabile su un canale perfettamente affidabile

Il protocollo **rdt1.0** affronta il caso in cui il canale sottostante è **completamente affidabile**, cioè non si verificano **errori** di trasmissione. Questo protocollo è estremamente **semplice** e prevede **due macchine** a stati finiti separate: una per il **mittente** e una per il **destinatario**.

Non è necessario che il destinatario fornisca **feedback** o informazioni al mittente, dato che non ci sono **errori** di trasmissione e nulla può andare storto in un canale perfettamente affidabile. Inoltre, si assume che il destinatario possa ricevere dati al ritmo di invio del mittente, quindi non è richiesto alcun **controllo** sul tasso di **trasmissione** tra **mittente** e **destinatario**.



a. rdt1.0: lato mittente



b. rdt1.0: lato ricevente

RDT 2.0: Trasferimento dati su un canale con errore sui bit

I protocolli RDT (Reliable Data Transfer) 2.0, 2.1 e 2.2 sono progettati per garantire la trasmissione affidabile dei dati su un canale con errori sui bit. Ogni versione migliora le prestazioni e la robustezza rispetto alla versione precedente. Di seguito, verranno illustrate le principali differenze tra RDT 2.0, 2.1 e 2.2:

- **RDT 2.0:**

- Utilizza **acknowledgment** positivi (ACK) e notifiche negative (NAK) per gestire la ritrasmissione dei pacchetti in caso di errori.
- Non tiene conto della possibilità che gli ACK o NAK stessi possano essere **corrotti**, il che può causare problemi di affidabilità.
- Non utilizza **numeri di sequenza** nei pacchetti.

- **RDT 2.1:**

- Il mittente utilizza **numeri di sequenza** nei **pacchetti** per gestire la **ritrasmissione** e l'ordinamento degli stessi
- Il mittente invia pacchetti con numeri di sequenza e il destinatario risponde con **ACK** per confermare la ricezione di un pacchetto o **NAK** per richiedere la ritrasmissione di un pacchetto mancante.
- Risolve il problema dell'**ambiguità** causato da ACK o NAK corrotti nel protocollo RDT 2.0.

- **RDT 2.2:**

- Il protocollo viene ulteriormente migliorato introducendo **numeri di sequenza** sia nei **pacchetti** dati che nei pacchetti di conferma (ACK).
- **Elimina** la necessità di utilizzare **NAK**, poiché il mittente può dedurre se c'è la necessità di ritrasmettere in base ai numeri di sequenza dei pacchetti di conferma.

RDT 3.0: Trasferimento dati affidabile su un canale con perdite ed errori sui bit

RDT 3.0 estende RDT 2.2 per gestire anche la **perdita** di pacchetti, utilizzando **timeout** e numeri di sequenza per assicurare il **trasferimento affidabile** dei dati, anche in presenza di **pacchetti persi** o **ritardati**, sebbene ciò possa causare pacchetti **duplicati** da gestire.

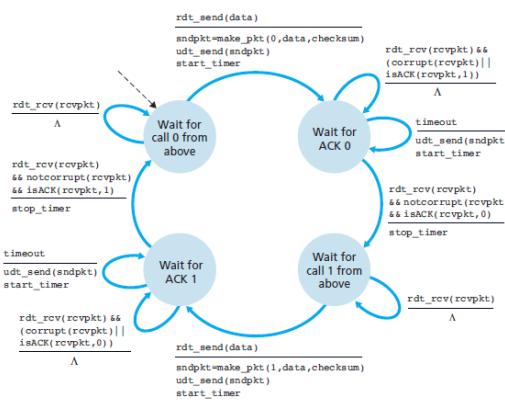


Figure 3.15 • rdt3.0 sender

Riepilogo di RDT

- **RDT 1.0:**
 - In **RDT 1.0**, il mittente invia i dati e il destinatario risponde con un messaggio di conferma (**ACK**) per indicare che i dati sono stati ricevuti **correttamente**. Tuttavia, questa versione **non** è ancora molto **affidabile** in quanto non gestisce situazioni in cui i pacchetti si **perdonano** o arrivano **fuori ordine**.
- **RDT 2.0:**
 - **RDT 2.0** migliora RDT 1.0 aggiungendo una logica più sofisticata per garantire **l'affidabilità**. Introduce l'uso di **numeri di sequenza** per i pacchetti e richiede al destinatario di inviare **ACK** o **NACK** (acknowledgment o non-acknowledgment) per indicare se il pacchetto è stato ricevuto correttamente o meno. In questo modo, il mittente può ritrasmettere i pacchetti mancanti.
- **RDT 2.1:**
 - Questa versione introduce la **ritrasmissione selettiva**. Oltre a inviare **NACK**, il destinatario può specificare quale pacchetto è stato perso e richiederne la **ritrasmissione**. Ciò evita la ritrasmissione di pacchetti che il destinatario ha già ricevuto correttamente.
- **RDT 2.2:**
 - In questa versione, vengono introdotti i **timeout**. Se il mittente non riceve un **ACK** entro un certo periodo di tempo, suppone che il pacchetto sia stato **perso** e lo **ritrasmette**. Questo migliora ulteriormente l'affidabilità del protocollo.
- **RDT 3.0:**
 - **RDT 3.0** introduce il concetto di finestra scorrevole (**sliding window**). In questa versione, il mittente può inviare **più pacchetti** prima di attendere **l'ACK**. Il destinatario può confermare i pacchetti ricevuti in sequenza e il mittente può **rimuovere** dalla **finestra** i pacchetti **confermati**. Ciò aumenta **l'efficienza della trasmissione**.

Velocità di trasferimento dati di un canale

La **velocità di trasferimento dati** di un canale, spesso chiamata semplicemente "**larghezza di banda**", rappresenta la **quantità di dati** che può essere **trasmessa** attraverso il **canale** di comunicazione in un dato **intervallo di tempo**. È misurata in bit al secondo (**bps**) o in unità più grandi come kilobit al secondo (**Kbps**), megabit al secondo (**Mbps**) o gigabit al secondo (**Gbps**), a seconda della capacità del canale.

Tempo di latenza

Il **tempo di latenza** si riferisce al **ritardo** che un segnale o un **pacchetto** di dati **subisce** durante il suo percorso da una sorgente a una destinazione. Esistono diversi tipi di latenza, tra cui:

- **Latenza di trasmissione:** Il tempo impiegato per **inviare** dati da una sorgente a una **destinazione** sulla rete.
- **Latenza di propagazione:** Il tempo necessario affinché un segnale o un pacchetto **attraversi fisicamente la rete** o il mezzo di trasmissione, in genere dipendente dalla distanza fisica tra i due punti.
- **Latenza di elaborazione:** Il tempo che un dispositivo o un nodo di rete impiega per **processare un pacchetto** o un segnale.
- **Latenza di accodamento:** Il tempo che i pacchetti devono **attendere in una coda** prima di essere elaborati o trasmessi.

Velocità di trasferimento end-to-end

La **velocità di trasferimento** end-to-end rappresenta la **quantità** di dati che può essere **trasmessa** dall'inizio alla fine di una **comunicazione**. È la **velocità effettiva** percepita dall'utente o dall'applicazione alla fine del processo di comunicazione ed è solitamente **inferiore** alla **larghezza di banda** nominale del canale a causa dei **ritardi** e delle **perdite**.

Banda disponibile ai livelli superiori

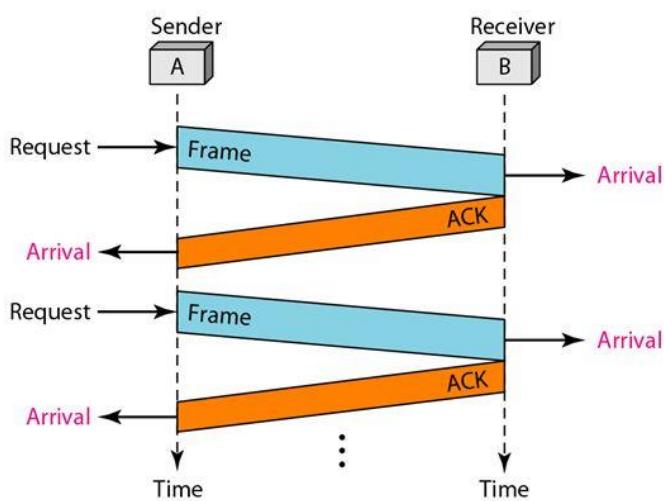
Questa è la quantità di larghezza di banda **effettivamente utilizzata** dagli strati superiori del protocollo di comunicazione. In altre parole, è la **quantità di dati** che può essere **effettivamente trasmessa e ricevuta** dalle applicazioni senza considerare le **informazioni aggiuntive** ed i protocolli di comunicazione sottostanti.

Protocolli Stop and Wait, Go back-N, Ripetizione selettiva

Le tecniche di trasmissione dati affidabili "**Stop-and-Wait**," "**Go-Back-N**," e "**Ripetizione Selettiva**" sono tre approcci per gestire la **trasmissione di dati affidabili** su un canale soggetto a **errori e perdite**. Ecco le principali differenze tra queste tre tecniche:

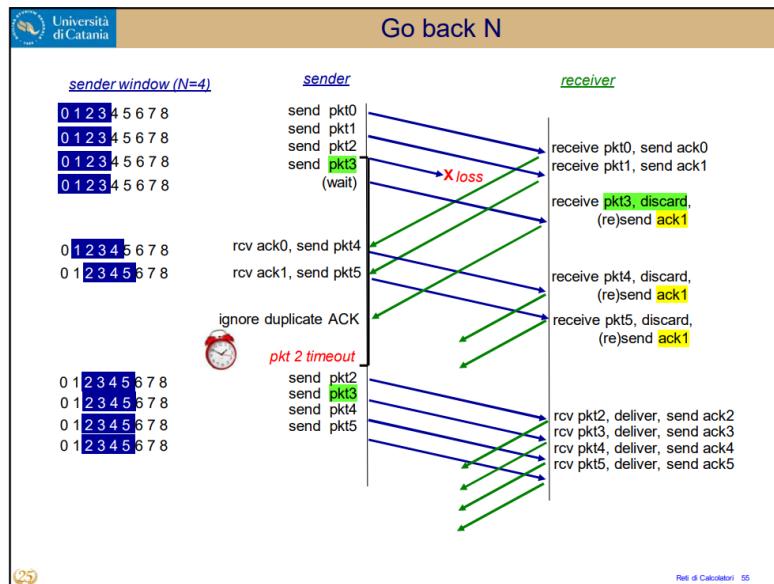
Stop-and-Wait (Stop e attesa)

Il mittente invia un **singolo pacchetto** alla volta e **attende** il suo **ACK** (acknowledgment) prima di **inviare il successivo**. Mantiene un **buffer** con un **solo pacchetto** in attesa di **conferma**. Se il mittente non riceve l'ACK entro un **timeout** specificato, **rtrasmette il pacchetto**.



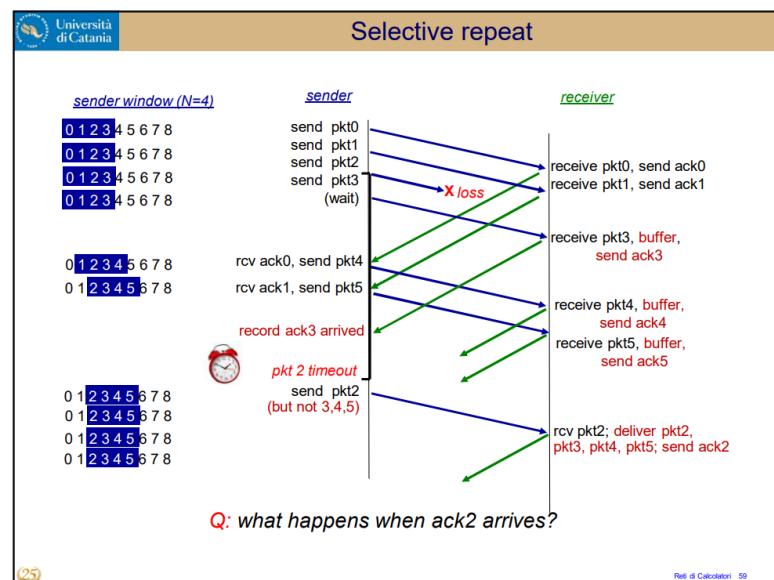
Go Back-N (GBN)

Il mittente invia una **sequenza** di pacchetti **senza attendere l'ACK** per ciascuno di essi. Mantiene un buffer per **più pacchetti trasmessi** ma **non ancora confermati**. Se un pacchetto viene **perso** o **corrotto**, il mittente **rtrasmette** tutti i **pacchetti** a partire da quello perso. Il destinatario può accettare pacchetti **fuori sequenza**, ma **scarta i duplicati**.



Ripetizione Selettiva (Selective Repeat)

Il **mittente** invia una **sequenza** di pacchetti senza attendere l'**ACK** per ciascuno di essi. Mantiene un buffer per i pacchetti trasmessi ma non ancora confermati. **Solo i pacchetti persi** o corrotti vengono **rtrasmessi** individualmente. Il **destinatario** può accettare pacchetti **fuori sequenza** e gestisce i **duplicati**.

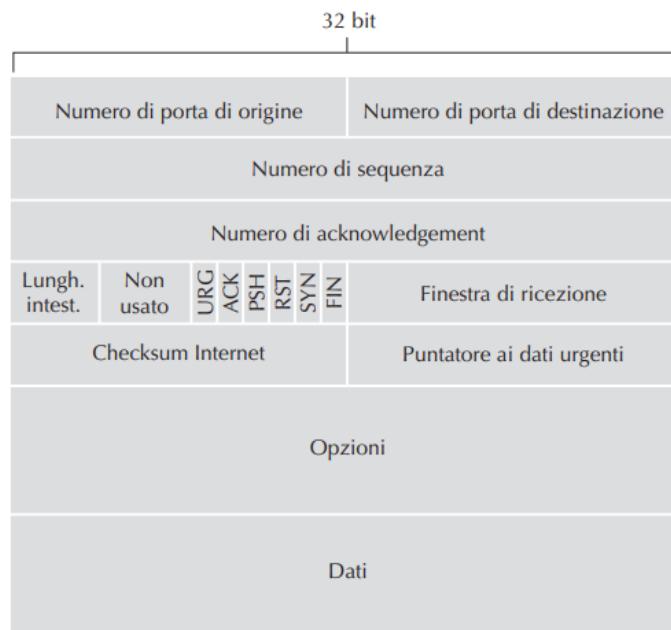


Trasporto orientato alla connessione: TCP

Il **TCP** è un protocollo **orientato alla connessione** che richiede un **handshake** preliminare prima di scambiare dati. Le connessioni TCP sono **full-duplex** e **punto a punto**, ma non supportano il Multicast.

Per stabilire una connessione, il client invia un segmento speciale al server, che risponde con un altro segmento e quindi il client risponde con un terzo. Questo è noto come "**handshake a tre vie**". Una volta stabilita la connessione, i dati vengono inviati e ricevuti attraverso buffer di invio e ricezione.

Struttura dei segmenti TCP



La struttura dei segmenti **TCP** è fondamentale per comprendere come avviene la comunicazione **affidabile** e efficiente tramite questo protocollo. Un **segmento** TCP è **composto** da campi di **intestazione** e un campo **dati** proveniente dall'applicazione. La **dimensione massima** del campo dati è vincolata dalla **MSS** (Maximum Segment Size), che è una parte essenziale della comunicazione TCP.

Per comprendere meglio questa **struttura**, consideriamo alcuni dei principali **campi** presenti nei segmenti TCP:

- **Numeri di Porta**: Ogni segmento TCP include numeri di porta di **origine** e **destinazione** (16+16 bit).
- **Numero di Sequenza e Numero di Acknowledgment**: Questi campi, ciascuno di 32 bit, svolgono un ruolo cruciale nell'implementazione del trasferimento dati affidabile. Il mittente utilizza il numero di **sequenza** per numerare i **segmenti** in modo univoco, mentre il destinatario utilizza il numero di **acknowledgment** per indicare il prossimo **segmento atteso**.
- **Lunghezza dell'Intestazione**: Il campo "header length" di 4 bit, specifica la **dimensione dell'intestazione TCP** in multipli di 32 bit. In genere, l'intestazione TCP ha una lunghezza di **20 byte**, ma può aumentare se vengono utilizzate opzioni aggiuntive.
- **Flag**: Il campo dei flag è costituito da 6 bit ed è di vitale importanza per il controllo della connessione TCP. Alcuni flag notevoli includono:
 - **ACK**: Indica che il valore nel campo di **acknowledgment** è **valido** e che il segmento contiene un acknowledgment per un segmento ricevuto con successo.
 - **RST, SYN, e FIN**: Vengono **utilizzati** per **stabilire** e **chiudere** la **connessione**, come parte del processo di handshake e chiusura.
 - **CWR ed ECE**: Sono utilizzati nel **controllo di congestione** esplicito, che regola il flusso di dati per prevenire la congestione di rete.
 - **PSH**: Indica al destinatario di **inviare** immediatamente i **dati** al **livello superiore**.
 - **URG**: Indica la **presenza** di **dati urgenti** nel segmento, con il puntatore ai dati urgenti che indica la posizione dell'ultimo byte urgente.
- **Finestra di Ricezione**: La finestra di ricezione, un campo di 16 bit, è **utilizzata** per il **controllo di flusso**. Indica al mittente quanti byte il destinatario è disposto ad accettare senza sovraccaricare il proprio buffer.
- **Checksum**: Il campo checksum è utilizzato per **rilevare** eventuali **errori** nei dati del segmento durante la trasmissione.

- **Puntatore ai dati urgenti** (Urgent Pointer) è un componente importante nell'intestazione di un segmento TCP. Esso serve a **indicare** la posizione **dell'ultimo byte** di **dati urgenti** nel segmento. Questo campo è utilizzato quando i dati all'interno del segmento sono marcati come "urgenti" dall'entità mittente a livello superiore.
- **Opzioni:** Le opzioni TCP, di lunghezza variabile e facoltative, sono **utilizzate** per **negoziare parametri** come la dimensione massima del segmento (**MSS**) o per stabilire fattori di scala per la finestra di ricezione in reti ad alta velocità.

Timeout e stima del tempo di andata e ritorno

TCP utilizza un meccanismo di **timeout** e ritrasmissione per **recuperare i segmenti** di dati **persi** durante la trasmissione. Tuttavia, determinare il momento esatto in cui ritrasmettere un segmento richiede una gestione accurata.

Stima del Tempo di Andata e Ritorno (RTT)

TCP misura il tempo di andata e ritorno (Round-Trip Time, RTT) tra il mittente e il destinatario. L'RTT di un segmento, chiamato "**SampleRTT**," è il **tempo** trascorso tra **l'invio del segmento** e la **ricezione dell'acknowledgment** corrispondente. In ogni istante, si misura il SampleRTT per un solo segmento non ancora confermato.

Stima di RTT Mediante Media Mobile Esponenziale Ponderata (EWMA)

Per ottenere una stima più stabile del RTT, TCP calcola una **media ponderata** dei valori di **SampleRTT**. Questa media, chiamata "**EstimatedRTT**," è aggiornata utilizzando la seguente formula:

$$\text{EstimatedRTT} = (1 - \alpha) \times \text{EstimatedRTT} + \alpha \times \text{SampleRTT}$$

Qui, "**a**" è un valore di pesatura (di solito **1/8** o **0,125**), che assegna più importanza ai campioni recenti rispetto a quelli più vecchi.

Stima della Variabilità RTT (DevRTT)

È importante conoscere la variabilità del RTT. **DevRTT** indica quanto **SampleRTT** si **discosta** da **EstimatedRTT**. Un valore raccomandato per la pesatura in DevRTT è **$\beta=0,25$** .

$$\text{DevRTT} = (1 - \beta) \times \text{DevRTT} + \beta \times | \text{SampleRTT} - \text{EstimatedRTT} |$$

Impostazione e Gestione del Timeout di Ritrasmissione

Ora che TCP ha stime di **EstimatedRTT** e **DevRTT**, deve determinare quando ritrasmettere un segmento non confermato. L'intervallo di **timeout**, chiamato "RTO," viene calcolato come segue:

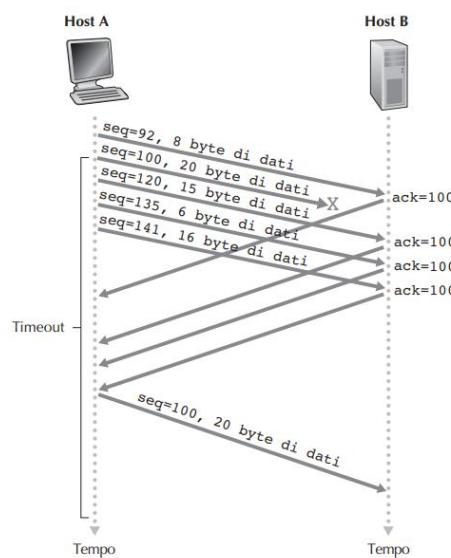
$$RTO = \text{EstimatedRTT} + 4 \times \text{DevRTT}$$

Un valore iniziale comune per **RTO** è di **1 secondo**.

In sintesi, TCP utilizza un meccanismo di **timeout** e ritrasmissione basato su stime del tempo di andata e ritorno (RTT) e della sua variabilità per garantire **l'affidabilità** della comunicazione e per adattarsi alle condizioni di rete. Questo meccanismo è **fondamentale** per gestire le ritrasmissioni in modo efficiente senza causare eccessivi **ritardi** nella consegna dei dati.

Ritrasmissione rapida

TCP affronta il **problema** delle **ritrasmissioni** quando i dati vengono persi durante la trasmissione. Un problema è la **lunghezza** dei periodi di **timeout**, poiché un timeout prolungato **ritarda** la ritrasmissione dei dati persi. TCP può **rilevare** la perdita di dati **prima** che scatti il **timeout grazie** agli ACK duplicati. La politica di generazione degli ACK in TCP prevede l'invio di ACK duplicati quando il destinatario rileva un buco nel flusso di dati, indicando segmenti mancanti. Il mittente, se riceve **tre ACK duplicati** consecutivi relativi allo stesso dato, **interpreta** che il **segmento** successivo è stato **perso**. TCP esegue una "**ritrasmissione rapida**," **rispedendo il segmento mancante** prima che scada il timer associato ad esso. La ritrasmissione rapida permette di recuperare i dati persi in modo più veloce rispetto all'attesa del timeout, riducendo al minimo il ritardo nella consegna dei dati persi durante la comunicazione.



Controllo di flusso

TCP utilizza il **controllo di flusso** per **prevenire** il **sovraffollamento** del **buffer** di ricezione del **destinatario**, garantendo che il mittente non invii dati più velocemente di quanto il destinatario possa elaborare. Questo meccanismo **si basa** sulla "finestra di ricezione" (receive window), che rappresenta lo spazio disponibile nel buffer di ricezione del destinatario.

Per tenere **traccia** dello **stato** del **buffer**, definiamo le seguenti variabili:

- **LastByteRead**: il numero dell'**ultimo byte** nel **flusso di dati** che il processo applicativo su B ha **letto dal buffer**.
- **LastByteRcvd**: il numero dell'**ultimo byte** nel **flusso di dati** che è **arrivato** dalla rete e che è stato **copiato** nel buffer di ricezione di B.

Affinché il buffer di ricezione di B non venga sovraffollato, deve valere la seguente condizione:

$$\text{LastByteRcvd} - \text{LastByteRead} \leq \text{RcvBuffer}$$

La finestra di ricezione, indicata con **rwnd** (è dinamica), rappresenta lo spazio libero disponibile nel buffer di ricezione:

$$\text{rwnd} = \text{RcvBuffer} - (\text{LastByteRcvd} - \text{LastByteRead})$$

Il **mittente**, ossia l'Host A, tiene **traccia** di due variabili:

- **LastByteSent** (l'ultimo byte inviato)
- **LastByteAcked** (l'ultimo byte per cui ha ricevuto acknowledgment).

La differenza tra queste due variabili rappresenta la **quantità di dati inviati** ma **non ancora confermati**. Il mittente si assicura che questa quantità sia **inferiore o uguale** alla dimensione della finestra di ricezione del destinatario (**rwnd**), garantendo che non sovraffollchi il buffer di ricezione del destinatario:

$$\text{LastByteSent} - \text{LastByteAcked} \leq \text{rwnd}$$

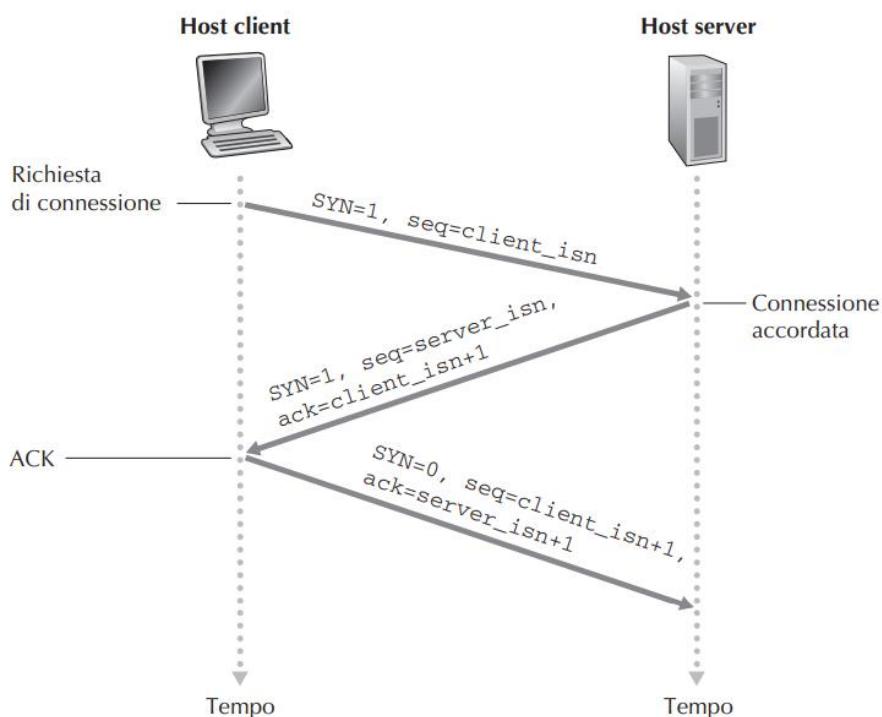
Se il buffer di ricezione di B si riempie completamente (**rwnd = 0**) e non c'è nulla da inviare, il mittente A continua a inviare segmenti contenenti un **singolo byte** di dati. In questo modo, il destinatario B può rispondere con un acknowledgment contenente un valore non nullo per rwnd quando il suo buffer si **svuota**, consentendo al mittente A di **riprendere l'invio di dati**.

Gestione della connessione TCP

Il **processo di stabilire e rilasciare una connessione** TCP è fondamentale per il funzionamento delle comunicazioni in rete.

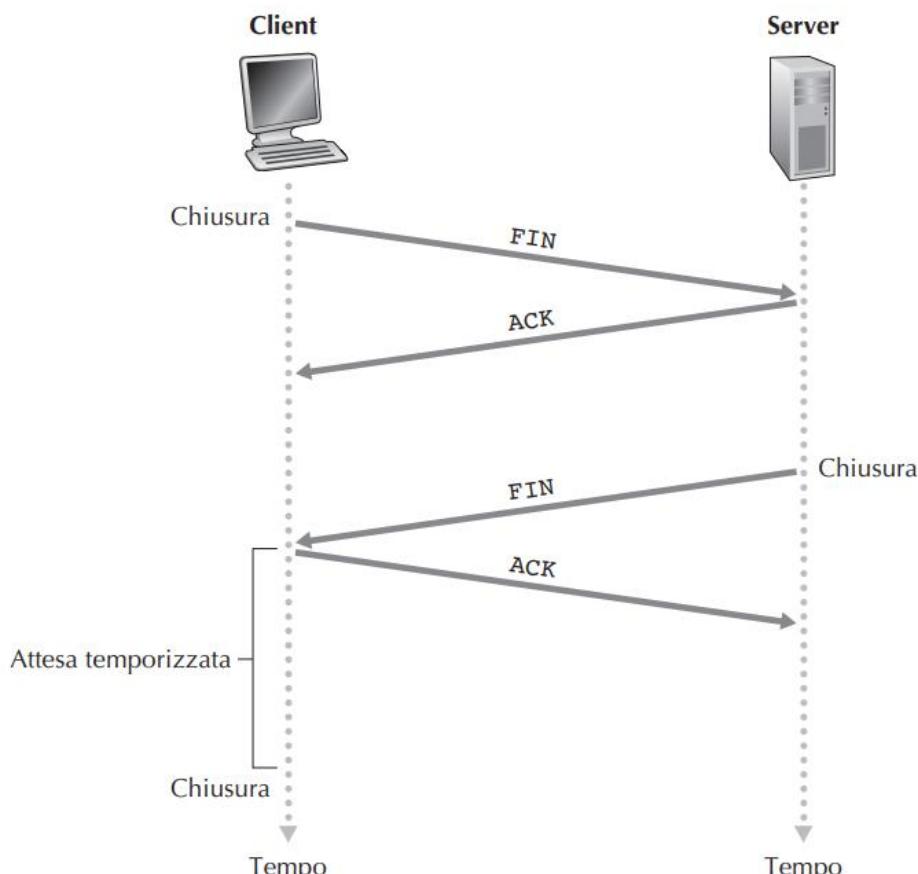
Inizializzazione della connessione

Durante la fase di **stabilizzazione**, chiamata "handshake a tre vie," un host **client** desideroso di iniziare una connessione **invia** un segmento **SYN** al server. Questo segmento speciale contiene il bit **SYN** impostato a **1** e un **numero di sequenza** iniziale generato **casualmente**. Il **server risponde** inviando un segmento **SYNACK**, con il bit **SYN** anche esso impostato a **1**, un campo **ACK** con il valore del numero di **sequenza iniziale** del client **incrementato di 1** e il suo proprio numero di sequenza iniziale. Infine, il **client invia** un **terzo segmento** con il bit **SYN** a **0** per confermare la connessione. Questa procedura, nota come "handshake a tre vie," richiede tre pacchetti per stabilire la connessione.



Chiusura della connessione

Per **chiudere** una **connessione**, uno degli **host invia** un segmento con il bit **FIN** impostato a **1**, indicando la sua intenzione di terminare la comunicazione. **L'altro host risponde** con un **acknowledgment**, e **poi** a sua volta **invia un segmento** con il bit **FIN**, completando così la chiusura della connessione. Durante questo processo, i due host passano attraverso vari stati TCP, come **FIN_WAIT_1**, **FIN_WAIT_2** e **TIME_WAIT**, prima di terminare completamente la connessione.



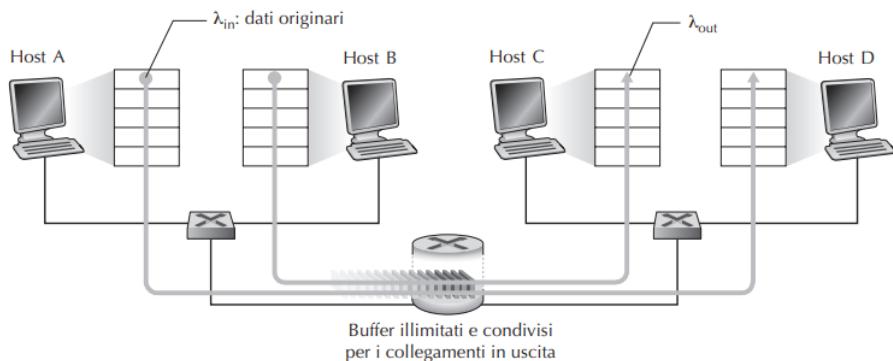
La chiusura di una connessione TCP segue questa sequenza:

1. L'host che **desidera chiudere** invia un segmento con il bit **FIN**.
2. L'host che **riceve** il **FIN** **risponde** con un **ACK** e continua a inviare dati se necessario.
3. Una volta completato l'invio dei dati, l'host che ha ricevuto il **FIN** **invia** anch'esso un **segmento FIN**.
4. Quando **entrambi** gli **host** confermano la **chiusura**, la connessione è terminata.
5. Lo stato **TIME_WAIT** previene problemi di segmenti residui.

Cause e costi della congestione

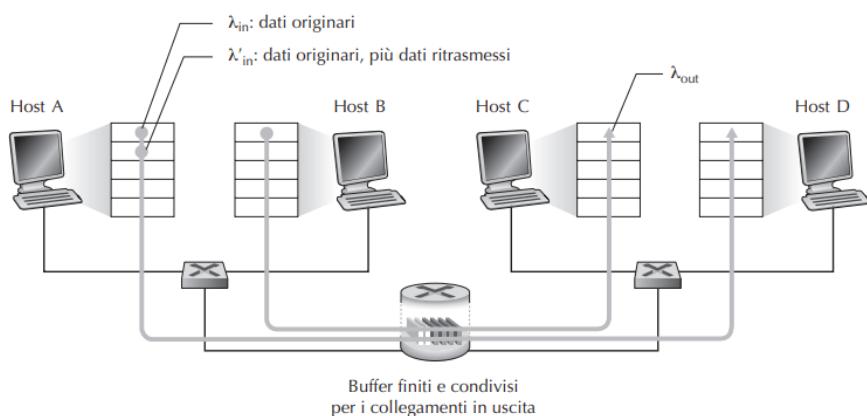
Scenario 1: due mittenti e un router con buffer illimitati

In questo scenario, **due host** (A e B) condividono **un router** intermedio con **buffer illimitati** e un **collegamento di uscita** con capacità R. Quando il tasso di invio dei dati da A e B supera **R/2**, il throughput **rimane limitato a R/2**. Questo limite causa **ritardi crescenti** e **congestione**, anche con buffer illimitati, quando il tasso di arrivo dei pacchetti si avvicina a R/2. In breve, la condivisione della capacità del collegamento può causare ritardi significativi e congestione quando i tassi di invio si avvicinano alla metà della capacità del collegamento.



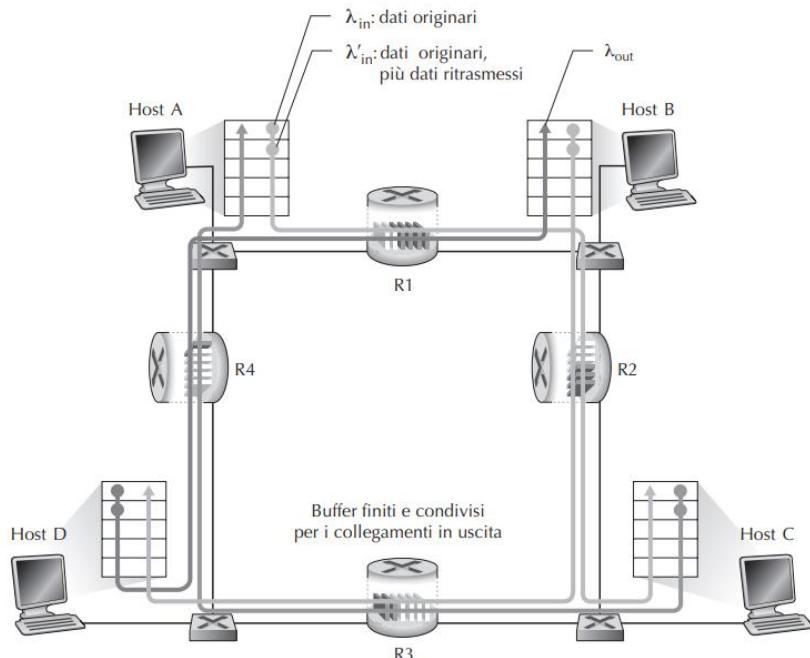
Scenario 2: due mittenti e un router con buffer limitati

In questo secondo scenario, **due mittenti** condividono un **router** con **buffer limitati**. Quando i buffer sono **pieni**, i pacchetti in eccesso vengono **scartati**. Le prestazioni dipendono dalla politica di ritrasmissione. In un caso ideale **senza perdite**, il **throughput** è **ottimale**. Nel caso di ritrasmissioni solo dopo la perdita, il throughput diminuisce con l'aumento del carico offerto. Ritrasmissioni premature possono sprecare **larghezza di banda**. In sintesi, le prestazioni sono influenzate dalla politica di ritrasmissione, causando **ritardi** e **inefficienze** con un aumento del carico offerto.



Scenario 3: quattro mittenti, router con buffer finiti e percorsi con più collegamenti

In questo scenario, **quattro host** comunicano tramite **collegamenti sovrapposti**. Quando il **traffico è leggero**, il **throughput aumenta** con l'aumento del traffico. Tuttavia, con un **traffico molto pesante**, il **throughput si riduce** perché i pacchetti vengono **scartati** a causa **dell'overflow** (raggiunge la sua capacità massima) dei buffer. Questo spreco di capacità trasmissiva porta a un compromesso tra il **traffico inviato** e il **throughput**, con un **decadimento del throughput** quando il **traffico è elevato**.



Approcci al controllo di congestione

Esistono **due approcci** principali al **controllo di congestione** nelle reti:

- **Controllo di congestione end-to-end:** La congestione viene **rilevata dagli host** basandosi su **segnali** come la **perdita** di pacchetti o i **ritardi**. **TCP** è un esempio che utilizza questo approccio.
- **Controllo di congestione assistito dalla rete:** I **router** o i componenti di rete forniscono **feedback esplicito** ai **mittenti** sulla congestione. Possono inviare un "pacchetto di strozzatura" (**chokepacket**) o impostare un campo nei pacchetti per **segnalare la congestione**.

Controllo di congestione TCP

TCP utilizza il controllo di congestione **end-to-end**, poiché il livello IP non fornisce **feedback esplicativi** sulla congestione. TCP regola la velocità di invio dei dati sulla connessione utilizzando la finestra di congestione (**cwnd**) e rilevando la congestione attraverso eventi di perdita, come **timeout** o ricezione di **tre ACK duplicati**.

TCP **aumenta la velocità di invio** quando riceve **acknowledgment non duplicati**, indicando che la **rete funziona bene**. In caso di **perdita di pacchetti**, dovuta alla congestione, TCP **diminuisce la velocità** di invio.

L'algoritmo di controllo di congestione TCP comprende tre fasi principali:

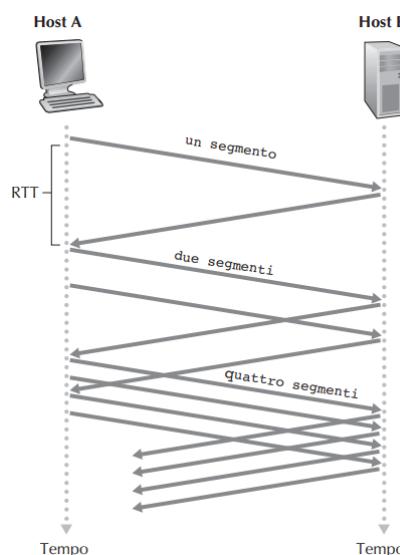
- **slow start**: aumenta **rapidamente** la finestra di congestione,
- **congestion avoidance**: aumenta la finestra di congestione **gradualmente**
- **fast recovery** (presente solo in TCP Reno).

Slow Start

La fase di **Slow Start** in TCP inizia con una finestra di congestione (**cwnd**) di **1 MSS**. Durante questa fase, la **cwnd** viene **raddoppiata** crescendo in modo **esponenziale** ogni volta che un segmento riceve un **ACK**.

La fase di Slow Start **termina** in due modi principali:

1. Se si verifica un **evento di perdita**, come un **timeout** o la ricezione di **tre ACK duplicati**, la **cwnd** viene **ripristinata** a **1 MSS** ed il valore di **ssthresh** (soglia di Slow Start) viene impostato a **metà del valore di cwnd**.
2. Quando **cwnd** raggiunge o **superà** il valore di **ssthresh**, la fase di Slow Start termina e TCP entra nella modalità di **Congestion Avoidance**.



Congestion Avoidance

Quando TCP entra nella fase di **Congestion Avoidance**, il valore di **cwnd** (Window di Congestione) è ridotto a circa la **metà del suo valore precedente**. Questo approccio più **conservativo** è progettato per **evitare** il rapido **aumento di cwnd** che caratterizza la fase di **Slow Start** e **prevenire congestioni**.

Durante la Congestion Avoidance, TCP **aumenta cwnd di 1 MSS** (Maximum Segment Size) **ogni** Round-Trip Time (**RTT**).

La fase di Congestion Avoidance può **terminare** in diverse situazioni:

1. In caso di **timeout**, simile a quanto avviene in Slow Start, **cwnd** viene impostato a **1 MSS** e **ssthresh** (soglia di Slow Start) è impostato a **metà** del valore di **cwnd**.
2. Se si verificano **tre acknowledgment duplicati**, che indicano la perdita di pacchetti ma non un collasso completo della rete, **TCP dimezza cwnd** e imposta **ssthresh a metà** del valore **di cwnd**.

Fast Recovery

Nella fase di **Fast Recovery** di TCP, la finestra di congestione (**cwnd**) **aumenta di 1 MSS** per ogni **acknowledgment duplicato** ricevuto per il segmento perso. Quando arriva un **acknowledgment** per il **segmento perso**, **TCP** passa alla fase di **Congestion Avoidance**, riducendo **cwnd** per evitare congestioni eccessive. In caso di **timeout**, TCP ritorna alla fase di **Slow Start**.

Esistono **due** principali **versioni di TCP** che gestiscono la congestione in modo leggermente diverso:

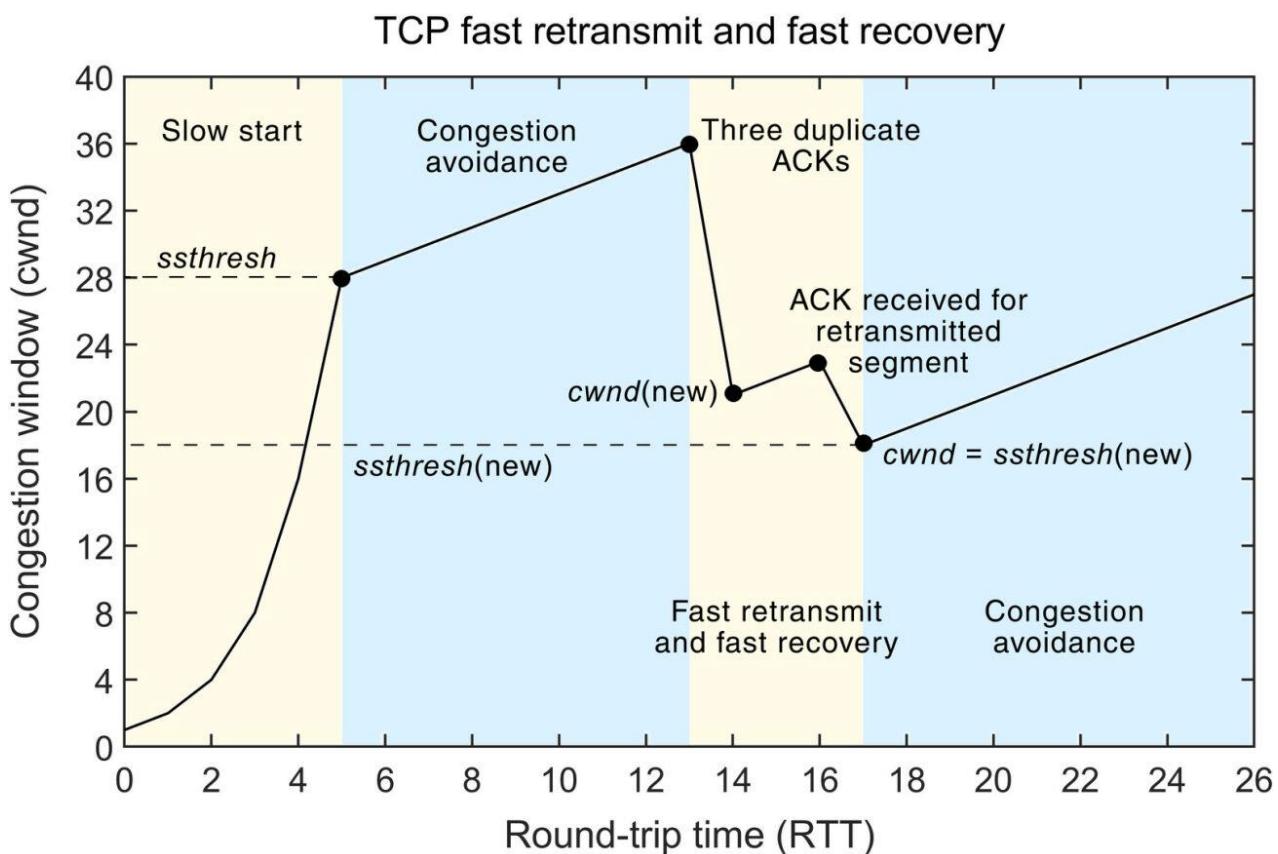
- **TCP Tahoe**
- **TCP Reno**

TCP Reno utilizza **Fast Recovery** per un recupero più rapido dalle perdite rispetto a **TCP Tahoe**.

Retrospettiva sul controllo di congestione di TCP

Il controllo di congestione di TCP può essere descritto come un processo di "incremento additivo, decremento moltiplicativo" (AIMD). Questo significa che TCP **aumenta linearmente** la dimensione della finestra di congestione (**cwnd**) e quindi il tasso di trasmissione, incrementando cwnd di **1 MSS per RTT**. Quando si verifica un evento di **triplice ACK duplicato** (indicando una perdita), TCP **diminuisce cwnd** della **metà**.

Questo approccio di controllo di congestione **AIMD** crea un modello a **dente di sega** nel comportamento di TCP, dove la finestra di congestione **cresce linearmente** fino a un evento di perdita e poi viene dimezzata, riprendendo **gradualmente** a crescere per rilevare eventuali opportunità di aumento del tasso di trasmissione.



TCP Tahoe VS TCP Reno

In sintesi, **TCP Reno** utilizza una politica di **ritrasmissione selettiva** e un meccanismo di fast recovery per garantire una maggiore efficienza nella trasmissione dei dati sulla rete.

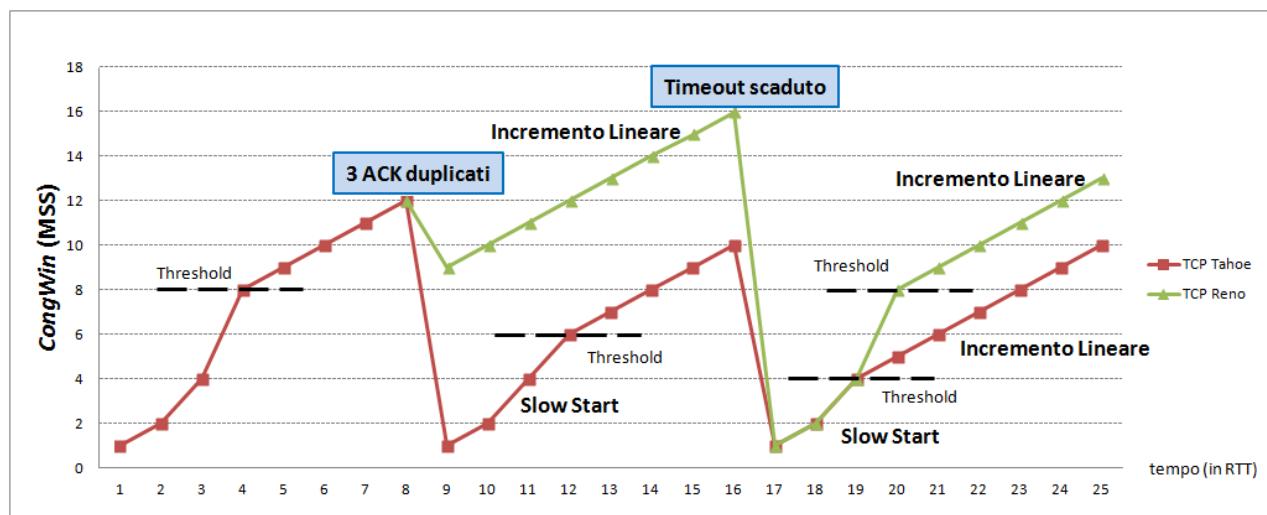
TCP Tahoe utilizza una politica di **ritrasmissione binaria** e un meccanismo di slow start per garantire una maggiore stabilità nella trasmissione dei dati sulla rete.

TCP-Tahoe implementa:

- **Slow start,**
- **Congestion avoidance**
- **Fast retransmit**

TCP-Reno implementa:

- **Slow start,**
- **Congestion avoidance**
- **Fast retransmit**
- **Fast recovery**



Livello Di Rete

Data la sua complessità e la quantità di argomenti, tratteremo il livello di rete in due capitoli. Vedremo che può essere diviso in due parti interagenti: il piano dei dati (**data plane**) e il piano di controllo (**control plane**).

Inoltro ed Instradamento

Il livello di rete ha due importanti funzioni nel piano dei dati: **l'inoltro** (forwarding) e **l'instradamento** (routing).

- **Inoltro (Forwarding):** Coinvolge il **trasferimento** di pacchetti da un router all'altro attraverso gli appropriati **collegamenti di uscita**. In altre parole, quando un router riceve un pacchetto, deve **decidere** su quale interfaccia di uscita **inoltrarlo**.
- **Instradamento (Routing):** Questa funzione riguarda la determinazione del **percorso** che i pacchetti devono seguire attraverso una **rete**. Gli algoritmi di instradamento (o **algoritmi di routing**) sono utilizzati per determinare il **percorso ottimale** dei pacchetti dalla sorgente alla destinazione.

Per l'inoltro dei pacchetti, i router utilizzano informazioni estratte dall'intestazione dei pacchetti per consultare una tabella di inoltro (**forwarding table**). Questa tabella indica a quale **interfaccia** di uscita il pacchetto deve essere **invitato**.

Piano di controllo: Tradizionale VS SDN

Nel piano di **controllo tradizionale**, le funzioni di **inoltro** e **instradamento** sono **integrate** nei **router**. Gli algoritmi di instradamento determinano le tabelle di inoltro nei router e richiedono comunicazioni tra di loro per garantire la corretta configurazione. **L'approccio SDN (Software Defined Network) separa** queste **funzioni**, con un **controller remoto** che **calcola** e **distribuisce** le tabelle di **inoltro** ai router. Questo rende la rete più **flessibile** e **gestibile**, consentendo nuove politiche di rete. Il controller SDN è una parte chiave del paradigma SDN, spostando il controllo dalla rete hardware dei router a un **livello software centralizzato**.

Modelli di servizio

Il **livello di rete** può offrire diversi servizi, tra cui

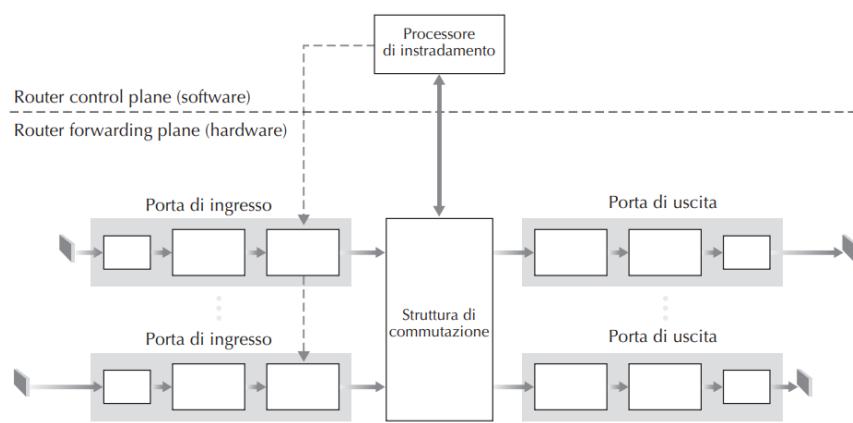
- **consegna garantita**
- **consegna garantita con ritardo limitato**
- **consegna ordinata**
- **banda minima garantita**
- **servizi di sicurezza.**

Tuttavia, il servizio predominante su Internet è il "**servizio best-effort**," che offre il **massimo impegno possibile** ma senza ulteriori garanzie. Il modello best-effort con una buona larghezza di banda ha dimostrato di **supportare** una vasta gamma di applicazioni con **successo**.

Che cosa si trova all'interno di un router?

Un **router** è composto da **quattro componenti** principali:

1. **Porte di ingresso (input port)**: Queste porte terminano **fisicamente** i collegamenti in **ingresso** al router. Possono anche gestire pacchetti di controllo, come quelli contenenti informazioni sui protocolli di **instradamento**, inviandoli al **processore** di instradamento.
2. **Struttura di commutazione (switching fabric)**: Questa struttura **connette** fisicamente le **porte di ingresso** alle **porte di uscita** all'interno del router
3. **Porte di uscita (output port)**: Queste porte **memorizzano** i pacchetti provenienti dalla **struttura di commutazione** e li **trasmettono** sui collegamenti in **uscita**.
4. **Processore di instradamento (routing processor)**: Questo componente esegue le funzioni del **piano di controllo**.



Elaborazione alle porte di ingresso e inoltro basato sull'indirizzo di destinazione

Le **porte di ingresso** di un router sono responsabili della **terminazione fisica** e **dell'elaborazione** a livello di collegamento per i collegamenti in ingresso. Ogni porta ha una copia locale della **tabella di inoltro**, utilizzata per determinare la **porta di uscita** dei pacchetti. La ricerca nella tabella è basata su corrispondenze di **prefissi** di indirizzi **di destinazione**. Questa operazione di inoltro è fondamentale e deve essere eseguita in nanosecondi, quindi richiede un'implementazione hardware efficiente, spesso con l'uso di memorie specializzate come le **TCAM** (Complessità: **O(1)**).

L'inoltro basato sulla "regola di corrispondenza a prefisso più lungo" avviene quando un router deve **determinare** la porta di **uscita** per un pacchetto basandosi **sull'indirizzo di destinazione** nel pacchetto. Ecco come funziona:

1. Il router **riceve** un pacchetto e **controlla** l'indirizzo di destinazione.
2. Quindi, il router **confronta** il prefisso dell'indirizzo di destinazione con le voci nella sua tabella di inoltro.
3. Il router **seleziona** la voce con il **prefisso** di indirizzo di **destinazione** più **lungo** che corrisponde all'indirizzo del pacchetto.
4. Una volta **individuata** la voce corrispondente, il router **utilizza** la **porta di uscita specificata** in quella voce per **inoltrare** il pacchetto alla **prossima fase** del percorso di rete.

The slide is titled "Longest prefix matching" and includes the logo of Università di Catania. It defines "longest prefix match" as finding the forwarding table entry for a given destination address by using the longest address prefix that matches the destination address.

Destination Address Range	Link interface
11001000 00010111 00010*** *****	0
11001000 00010111 00011000 *****	1
11001000 00010111 00011*** *****	2
otherwise	match!

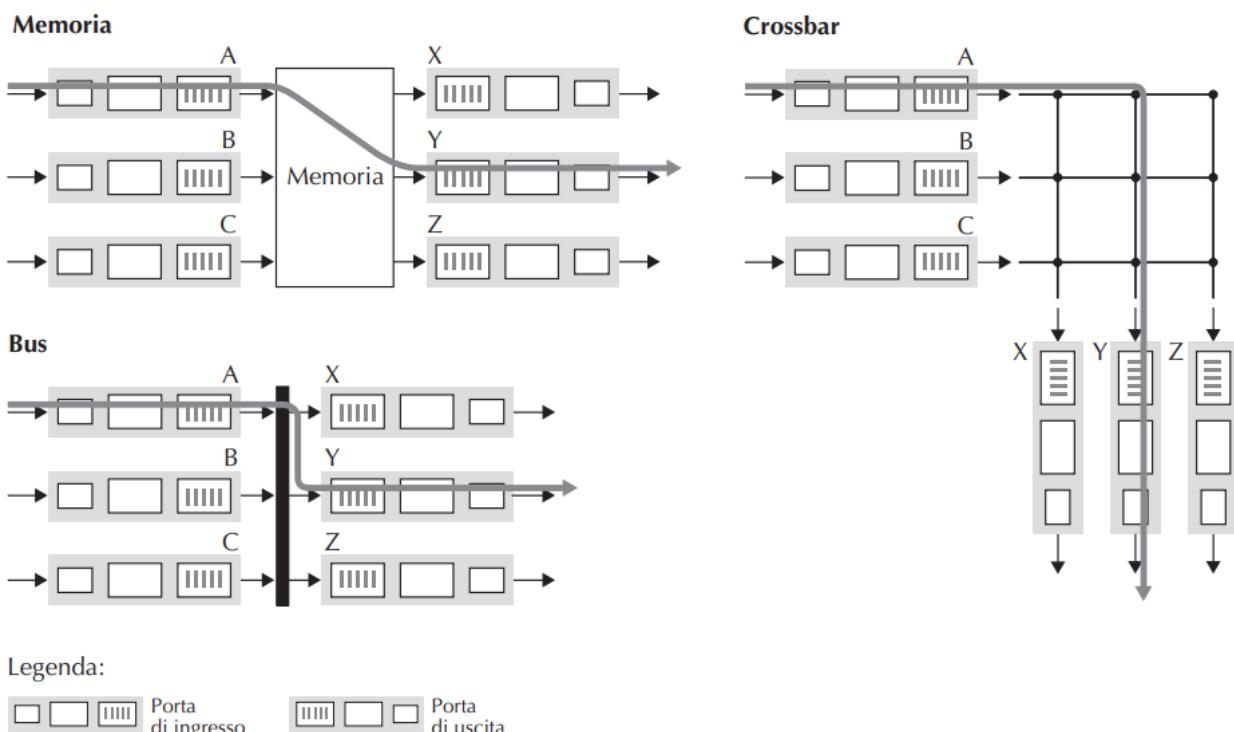
examples:

11001000 00010111 00010110 10100001 which interface?
11001000 00010111 00011000 10101010 which interface?

Struttura di commutazione

La **commutazione** all'interno dei router può avvenire in **diversi modi**:

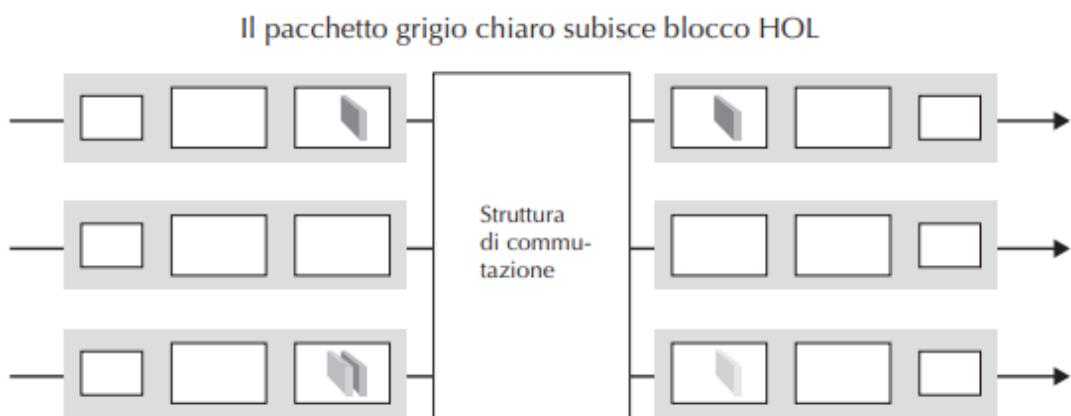
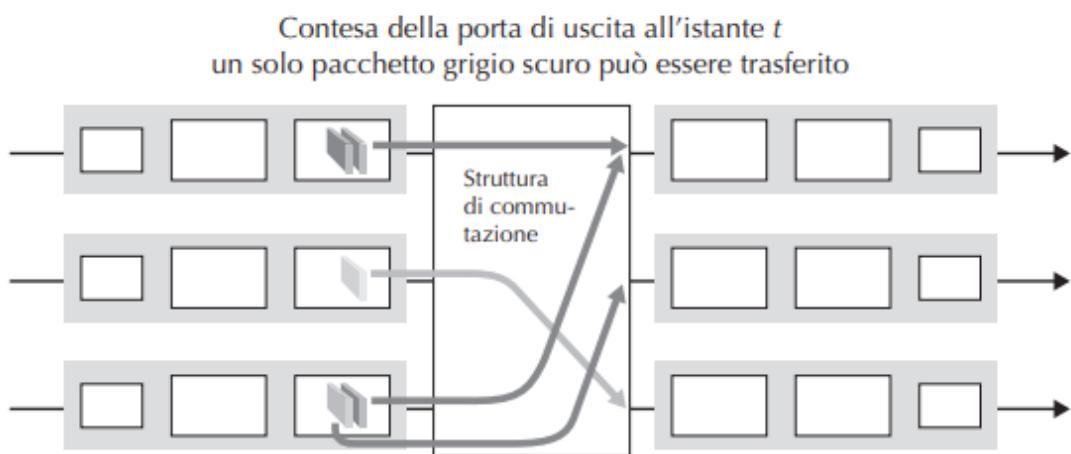
- **Commutazione in memoria:** Inizialmente, i **router** erano **calcolatori tradizionali**, e la **CPU gestiva** direttamente la **commutazione** tra le porte di **ingresso e uscita**. I pacchetti venivano **copiati** nella memoria della CPU e poi **instradati** in base alle informazioni di destinazione. Questo metodo era **limitato** dalla larghezza di banda della memoria e consentiva **solo un'operazione alla volta**.
- **Commutazione tramite bus:** In questo approccio, le **porte** di ingresso **trasferiscono** i pacchetti direttamente alle porte di uscita tramite un **bus condiviso**. Tuttavia, **solo un pacchetto** alla volta **può attraversare il bus**, limitando la larghezza di banda di commutazione.
- **Commutazione attraverso una rete di interconnessione:** Questo metodo utilizza una matrice di commutazione (**crossbar switch**) composta da bus verticali e orizzontali. Quando un **pacchetto** deve essere **instradato** da una porta di **ingresso** a una porta di **uscita**, il controller chiude l'incrocio corrispondente nella matrice, **consentendo** al **pacchetto** di **essere instradato**. Questo approccio permette di **instradare più pacchetti contemporaneamente**.



Dove si verifica l'accodamento?

Accodamento in ingresso

Se la **struttura di commutazione** non è **abbastanza veloce** nel trasferire i pacchetti senza ritardo, possono formarsi **code di pacchetti** alle porte di ingresso. Quando ciò accade, **solo un pacchetto alla volta** può essere **trasferito**, indipendentemente dalla velocità di commutazione. Questo fenomeno è chiamato "**blocco in testa alla coda**" (HOL, head-of-the-line blocking), dove i pacchetti devono attendere il loro turno per essere trasferiti, anche se la porta di uscita è disponibile.



Legenda:

- destinato alla porta di uscita superiore destinato alla porta di uscita centrale destinato alla porta di uscita inferiore

Accodamento in uscita

Quando una **struttura di commutazione** è molto più **veloce** delle linee di ingresso e i pacchetti da ogni porta di ingresso sono destinati alla stessa porta di uscita, possono formarsi **code di pacchetti** alle porte di **uscita**. Questo può accadere quando la velocità di commutazione è molto più rapida delle velocità di linea delle porte.

Per **calcolare** la **capacità** del **buffer** necessaria, si può utilizzare la formula:

$$\text{Buffering} = \frac{\text{RTT} * C}{\sqrt{N}}$$

Dove:

- **RTT** è il ritardo di propagazione medio
- **C** è la capacità del collegamento
- **N** è il numero di flussi TCP che attraversano il collegamento.

Questa formula tiene conto del numero di flussi TCP e può essere più efficiente rispetto a stime basate solo su RTT, specialmente quando ci sono molti flussi sulla rete.



Schedulazione dei pacchetti

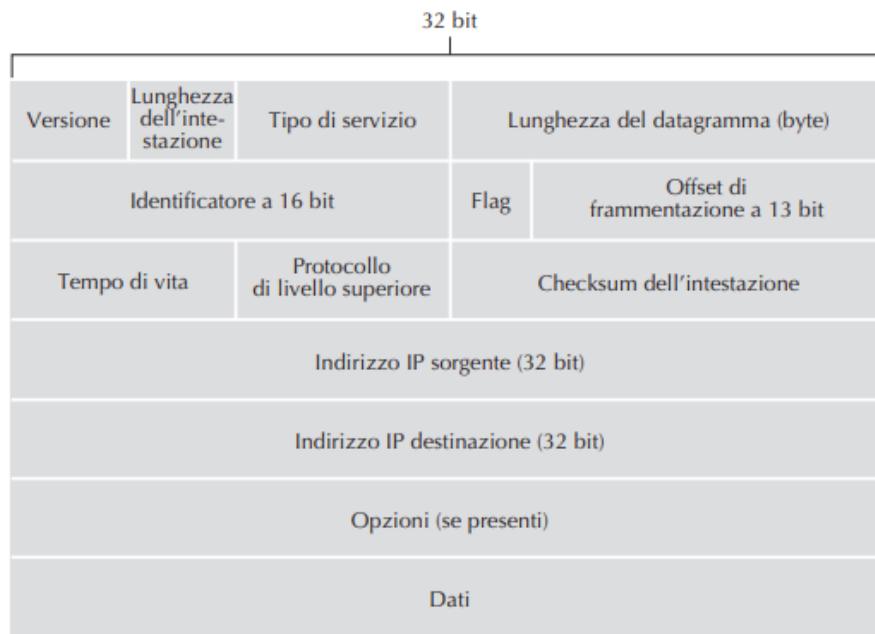
Questa sezione tratta della pianificazione dell'**ordine** in cui i **pacchetti** vengono **trasmessi** da una porta di **uscita** di un router.

Tipologie di algoritmi di schedulazione dei pacchetti

- **Primo-Arrivato-Primo-Servito (FIFO):** I pacchetti **vengono trasmessi** nell'**ordine** in cui sono **arrivati** nella coda di uscita. Questa strategia è simile a fare la fila in un negozio.
- **Code con Priorità:** I pacchetti sono **suddivisi** in **classi** basate sulla **priorità**. I pacchetti con **priorità** più **alta** vengono **trasmessi prima** di quelli con priorità inferiore. Ad esempio, i pacchetti di gestione di rete possono avere priorità più alta rispetto al traffico utente.
- **Round Robin:** I pacchetti sono divisi in classi e **vengono trasmessi** ciclicamente in **modo equo** tra le classi. Ad esempio, un pacchetto da ciascuna classe viene trasmesso a turno prima di iniziare un nuovo ciclo.
- **Accodamento Equo Ponderato (WFQ - Weighted Fair Queuing):** Simile al round robin, ma **ogni classe** ha un **peso** assegnato. Le classi con **pesi** più **alti** ricevono una parte **maggior** del **tempo** di **trasmissione**. Ciò garantisce che le classi con pesi più alti ricevano una parte equa della larghezza di banda, anche se altre classi hanno pacchetti in coda.

Questi algoritmi di schedulazione dei pacchetti sono utilizzati nei router per gestire l'ordine di trasmissione dei pacchetti in base a varie considerazioni, come la priorità, l'equità e la capacità del collegamento.

Formato dei datagrammi IPv4



Principali campi dei datagrammi IPv4:

- **Numero di Versione:** Questi quattro bit indicano la **versione** del protocollo IP utilizzato nel datagramma. Ad esempio, **IPv4** utilizza la versione 4, mentre **IPv6** utilizza la versione 6.
- **Lunghezza dell'Intestazione:** Questi 4 bit indicano la **lunghezza dell'intestazione** IP. Questa lunghezza può variare a seconda delle opzioni incluse nel datagramma.
- **Tipo di Servizio (TOS):** Questi bit consentono di distinguere i diversi tipi di **servizio** desiderati per il datagramma, come **basso ritardo** o **alta affidabilità** (non usato in IPv4).
- **Lunghezza del Datagramma:** Questo campo rappresenta la lunghezza totale del **datagramma** IP, inclusa l'intestazione e i dati, misurata in byte.
- **Identificatore, Flag, Offset di Frammentazione:** Questi campi gestiscono il processo di **frammentazione** dei datagrammi, che può essere necessario quando il datagramma è troppo grande per essere trasmesso in una sola volta.
- **Tempo di Vita (TTL):** Questo campo **impedisce** ai **datagrammi** di **rimanere indefinitamente** nella **rete**. Il TTL viene decrementato di uno ad ogni passaggio attraverso un router, e quando raggiunge zero, il datagramma viene scartato.
- **Protocollo:** Questo campo indica il **protocollo di trasporto** al quale i dati del datagramma devono essere consegnati quando raggiungono la destinazione. Ad esempio, il valore 6 indica **TCP**, mentre il valore 17 indica **UDP**.

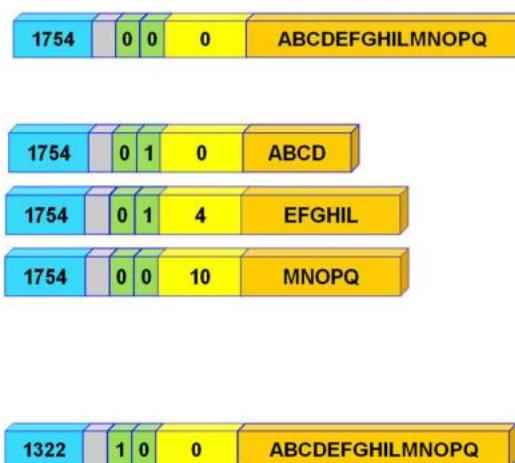
- **Checksum dell'Intestazione:** Questo checksum consente ai router di **rilevare errori** nei datagrammi ricevuti. Ogni router calcola il checksum per verificare la correttezza **dell'intestazione**.
- **Indirizzi IP Sorgente e Destinazione:** Questi campi indicano gli **indirizzi IP del mittente** e del **destinatario** del datagramma.
- **Opzioni:** Questi campi consentono di **estendere l'intestazione IP con informazioni aggiuntive**, se necessario.
- **Dati (Payload):** Questo campo contiene i **dati effettivi** del datagramma, che possono essere un segmento a livello di trasporto (ad esempio, TCP o UDP) o altri tipi di dati.

I datagrammi IP hanno **un'intestazione** di base di **20 byte**, escludendo le opzioni. Se il **datagramma contiene un segmento TCP**, l'intestazione totale sarà di **40 byte**, poiché includerà anche l'intestazione TCP.

Frammentazione dei datagrammi IPv4

La **frammentazione** in IPv4 è il **processo** di **suddivisione** di **datagrammi** in pezzi più piccoli quando sono troppo grandi per essere trasmessi su un collegamento di rete con una capacità limitata, nota come MTU (Maximum Transmission Unit).

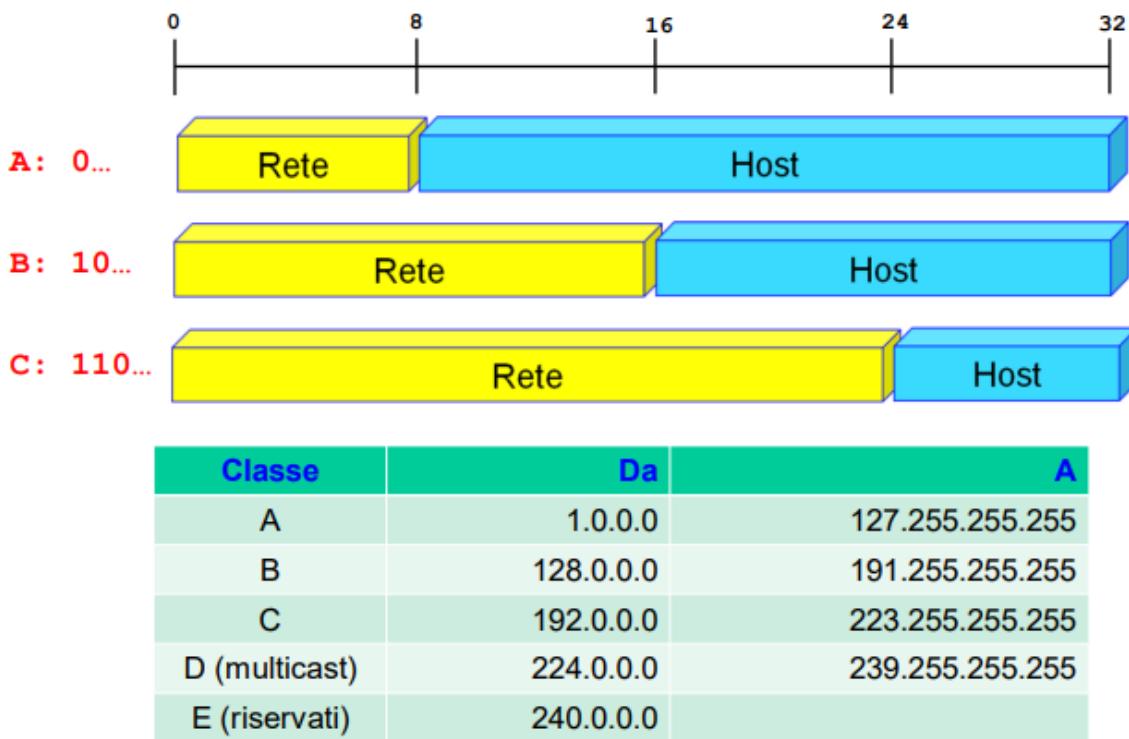
- Il bit "**Don't Fragment**" nel datagramma IPv4 **serve** a **impedire** la **frammentazione**. Se questo bit è impostato a 1 e il datagramma supera la MTU del collegamento, il router scarterà il datagramma e invierà un messaggio di errore all'host mittente, causando la potenziale perdita di dati.
- Il bit "**More Fragments**" indica se ci sono altri frammenti associati allo stesso datagramma. Se è impostato a 1, **significa** che ci sono **ulteriori frammenti da ricevere**.



Indirizzamento IPv4

L'indirizzamento IPv4 è **fondamentale** per **Internet** ed è basato su un sistema di indirizzi a 32-bit (4 byte), scritti in notazione decimale puntata.

In passato, l'indirizzamento IPv4 seguiva uno schema di **classi**, noto come "classful addressing," che **divideva** gli indirizzi in **categorie**:



Questo **schema** si è rivelato **inefficiente** per organizzazioni con esigenze variabili, portando alla creazione di blocchi di indirizzi non utilizzati. Per affrontare questa problematica, è stato introdotto il **CIDR** (Classless Inter-Domain Routing), che offre una **gestione più flessibile** degli **indirizzi**. La notazione CIDR, come "**a.b.c.d/x**," indica il numero di bit destinati alla parte di rete (x) nell'indirizzo IP. Questo approccio consente di assegnare in modo più efficiente gli indirizzi, evitando sprechi.

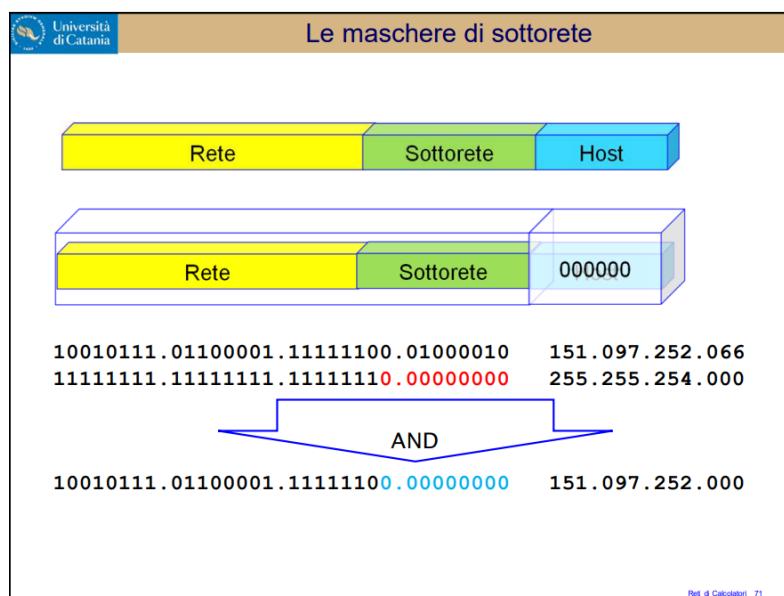
Inoltre, è importante menzionare le **maschere di sottorete** (subnet mask), che identificano la parte di rete e di host di un indirizzo IP. Le maschere di sottorete vengono spesso indicate con la notazione "**/x**," dove "**x**" **rappresenta** il numero di **bit** riservati alla **rete**. Ad esempio, una maschera di sottorete "/24" indica che i primi 24 bit sono dedicati alla rete e i rimanenti 8 bit ai dispositivi.

Inoltre, c'è un indirizzo IP speciale, **255.255.255.255**, che rappresenta un **broadcast**, ossia **l'invio** di un **pacchetto** a **tutti** gli **host** nella stessa **sottorete**.

LAN Interconnesse

Come determinare l'indirizzo di una rete

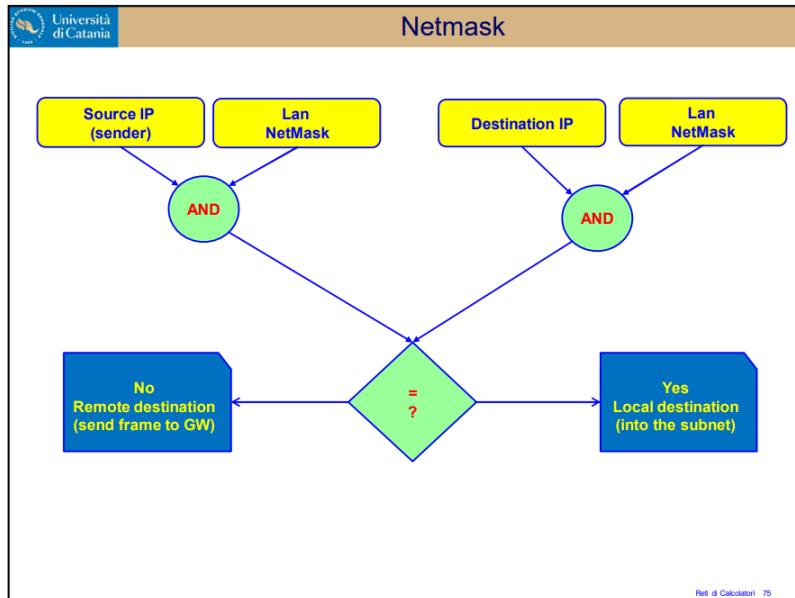
- Converti in binario:** Trasforma l'indirizzo IP e la subnet mask in **formato binario**.
- AND logico bit per bit:** Esegui un'operazione AND bit per bit tra l'indirizzo IP binario e la subnet mask binaria. I bit risultanti rappresentano **l'indirizzo di rete**.
- Risultato in forma decimale:** **Converti** il risultato dell'AND in forma **decimale** per ottenere l'indirizzo di rete.



Come determinare se due indirizzi fanno parte della stessa rete

Per determinare se due indirizzi IP fanno parte della stessa **LAN** (Local Area Network), è importante considerare la **subnet mask** e confrontare **l'indirizzo di rete** risultante. Ecco come procedere:

- Converti in binario:** Trasforma entrambi gli indirizzi IP in **formato binario**.
- Calcola l'indirizzo di rete:** Utilizza l'operazione logica **AND** tra ciascun indirizzo IP e la **subnet mask** corrispondente per ottenere **l'indirizzo di rete** per ciascun indirizzo.
- Confronta gli indirizzi di rete:** Se gli indirizzi di rete risultanti sono **identici**, allora i due indirizzi IP fanno parte della **stessa LAN**. In caso **contrario**, se **gli indirizzi** di rete **sono diversi**, i due dispositivi si trovano in **reti diverse**.



Se due **dispositivi** si trovano in **LAN diverse** e desiderano comunicare tra loro, dovranno passare attraverso il **gateway**.

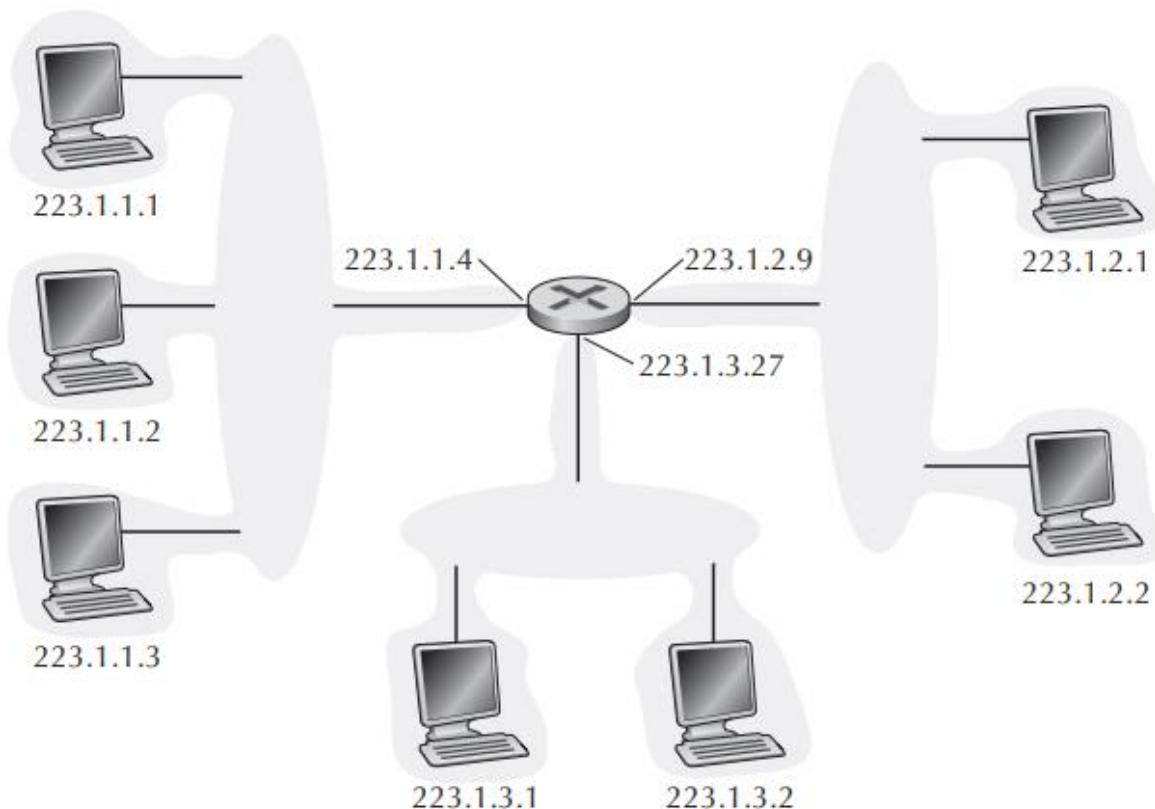
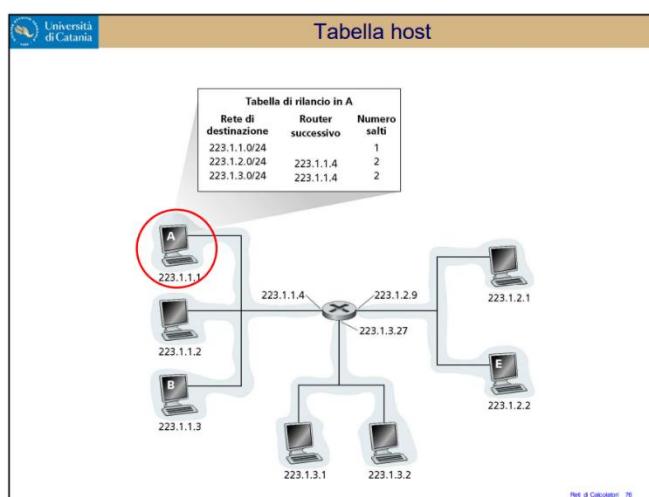


Tabelle host e tabelle di routing

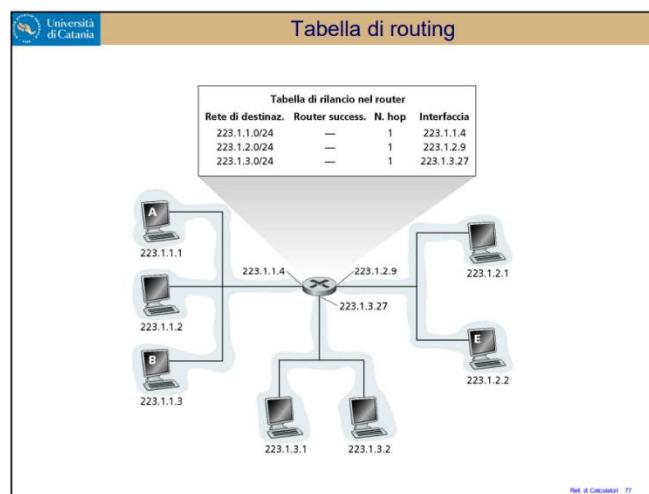
Le **tabelle host** contengono i seguenti campi:

- **Rete di destinazione:** Specifica quali **reti** possono essere **raggiunte** da questo host.
- **Router successivo:** Questo campo è **vuoto** se l'host mira a raggiungere un dispositivo all'interno della **stessa LAN**, altrimenti contiene **l'indirizzo IP del router** necessario per instradare il traffico verso una **rete esterna**.
- **Numero di salti:** Indica il **numero di dispositivi intermedi** (salti) che devono essere attraversati per raggiungere la destinazione desiderata.



Le **tabelle di routing** contengono i seguenti campi:

- **Rete di destinazione:** Indica le **reti raggiungibili** tramite il **router**.
- **Router successivo:** Questo campo è **vuoto** quando il router **non** è direttamente **connesso** ad **altri router**. Altrimenti contiene **l'indirizzo IP del router successivo**.
- **Numero di salti:** Indica il **numero di dispositivi intermedi** (salti) che devono essere **attraversati** per raggiungere la **destinazione** desiderata.
- **Interfaccia:** È il **punto di connessione** fisica o logica **attraverso** cui il **router invia i pacchetti** verso la destinazione.



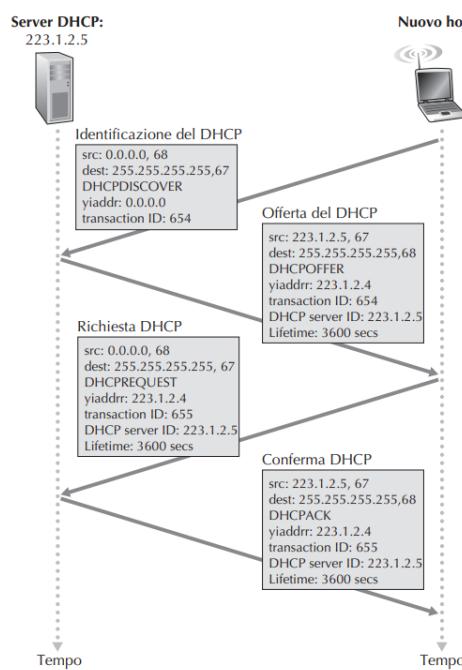
Come ottenere l'indirizzo di un host: DHCP

Il protocollo **DHCP** (Dynamic Host Configuration Protocol) consente a un host di **ottenere in modo automatico** un **indirizzo**.

Di seguito in dettaglio le **4 fasi** del protocollo **DHCP**:

- Individuazione del server DHCP:** Quando un host si collega a una rete, invia un messaggio DHCP "discover" in broadcast per individuare un server DHCP disponibile. Questo messaggio viene inviato con l'indirizzo IP sorgente "**0.0.0.0**" e l'indirizzo IP di destinazione "**255.255.255.255**" (broadcast) sulla porta **67** tramite **UDP**.
- Offerta del server DHCP:** Il server DHCP risponde con un messaggio DHCP "offer". Questo messaggio **contiene** un **indirizzo IP proposto** per l'host, insieme a informazioni sulla maschera di sottorete e la durata della concessione (**lease time**) dell'indirizzo IP.
- Richiesta DHCP:** L'host risponde con un messaggio DHCP "request", **confermando la richiesta** di configurazione.
- Conferma DHCP:** Il server DHCP, **conferma la concessione** fornendo tutte le informazioni di configurazione richieste. Questo messaggio è chiamato DHCP "ACK". Una volta ricevuto il DHCP ACK, **l'host utilizza l'indirizzo IP** assegnato e le altre impostazioni di rete.

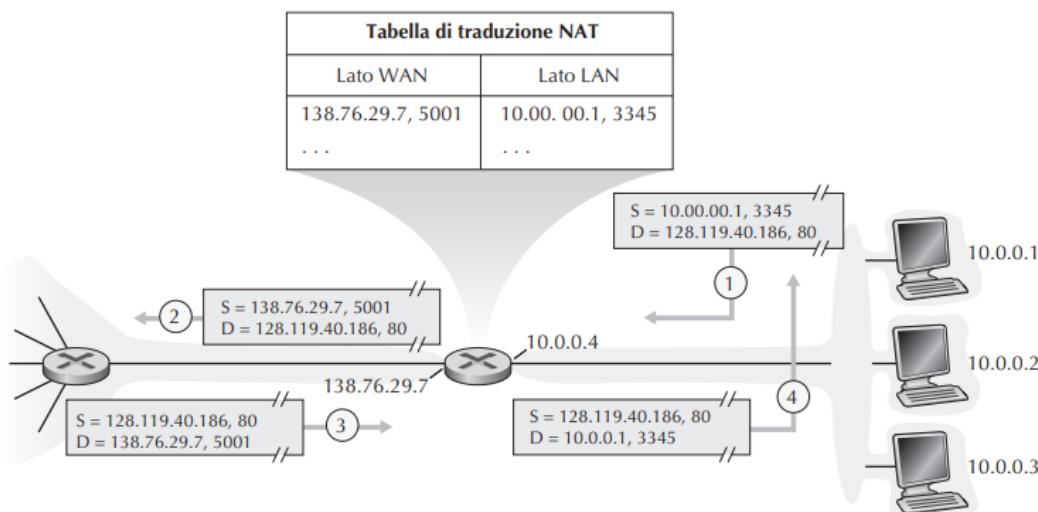
DHCP **semplifica** la **connessione** degli host alla rete **fornendo** loro **automaticamente configurazioni** di rete, come **l'indirizzo IP** e altre impostazioni. È utile in reti con dispositivi che si collegano spesso. **Tuttavia**, in scenari di mobilità, **DHCP può cambiare l'indirizzo IP** quando un dispositivo si sposta tra sottoreti, **interrompendo connessioni TCP**.



NAT (Network Address Translation)

Il **NAT** (Network Address Translation) è una **tecnica fondamentale** nel **networking** che **risolve** alcune **problematiche** legate alla **scarsità** degli **indirizzi IP pubblici** e alla protezione della privacy delle reti private. Tuttavia, questa soluzione **presenta** alcune **problematiche** e **richiede soluzioni** aggiuntive, tra cui **UPnP** (Universal Plug and Play).

Il **NAT** funziona attraverso un **router** abilitato al **NAT**, che agisce come **intermediario** tra la rete **privata** e **Internet**. Quando i dispositivi interni inviano pacchetti a Internet, **il router NAT modifica** gli **indirizzi IP** e i **numeri di porta** all'interno dei pacchetti in modo che sembrino provenire dal router stesso. Questo consente a molti dispositivi interni di **condividere lo stesso indirizzo IP pubblico**. Il NAT utilizza una **tabella di traduzione NAT** per **tenere traccia** delle associazioni tra **gli indirizzi IP** interni e quelli pubblici, compresi i **numeri di porta**, consentendo al router NAT di **intradare** correttamente le **risposte** da Internet ai **dispositivi interni**.



Il **problema** principale del **NAT** è la **gestione** e l'**aggiornamento** della **tabella di traduzione NAT**, poiché richiede **un'azione manuale** da parte **dell'amministratore di rete** per configurare le **regole di inoltro**. Grazie a **UPnP**, le macchine possono **configurare automaticamente** le regole di **NAT** quando necessario, senza **intervento manuale**. Le regole **UPnP** nella tabella **NAT hanno** una **scadenza** e vengono **rimosse** se non **utilizzate**, migliorando la **sicurezza**. Se più dispositivi cercano di utilizzare la **stessa porta** UPnP contemporaneamente, si **verifica** un **errore**.

Internet Control Message Protocol (ICMP)

Il protocollo **ICMP** (Internet Control Message Protocol) è utilizzato per scambiare **informazioni di controllo** e notificare **errori** tra host e router a livello di rete. È spesso associato alla notifica di errori, ad esempio quando un router non può trovare una route verso una destinazione, viene inviato un messaggio ICMP di tipo **3** per segnalare **l'errore**.

ICMP è considerato **parte** del livello di **rete**, ma funziona sopra il protocollo IP. I messaggi **ICMP** contengono un **campo tipo** e un **campo codice**, e includono l'intestazione e i **primi 8 byte** del datagramma IP che ha generato il messaggio ICMP, per **consentire al mittente** di **identificare l'errore**.

Tuttavia, **ICMP** non è utilizzato solo per **segnalare errori**. Ad esempio, il comando "**ping**" invia messaggi **ICMP** di tipo **8** per testare la connettività tra due **host**. Un altro uso interessante di ICMP è il comando "**traceroute**", che utilizza messaggi ICMP per determinare il percorso tra due host, identificando i nomi e gli indirizzi IP dei router lungo il percorso.

ICMP è **essenziale** per il funzionamento di **Internet** e ha una versione specifica per **IPv6** che introduce **nuovi tipi e codici** di **messaggio**.

	Bit 0–7	Bit 8–15	Bit 16–23	Bit 24–31
0	Tipo	Codice		Checksum
32	Dati dell'intestazione			

Tipo ICMP	Codice	Descrizione
0	0	risposta echo (a ping)
3	0	rete destinazione irraggiungibile
3	1	host destinazione irraggiungibile
3	2	protocollo destinazione irraggiungibile
3	3	porta destinazione irraggiungibile
3	6	rete destinazione sconosciuta
3	7	host destinazione sconosciuto
4	0	riduzione (controllo di congestione)
8	0	richiesta echo
9	0	annuncio di un router
10	0	scoperta di un router
11	0	TTL scaduto
12	0	intestazione IP errata

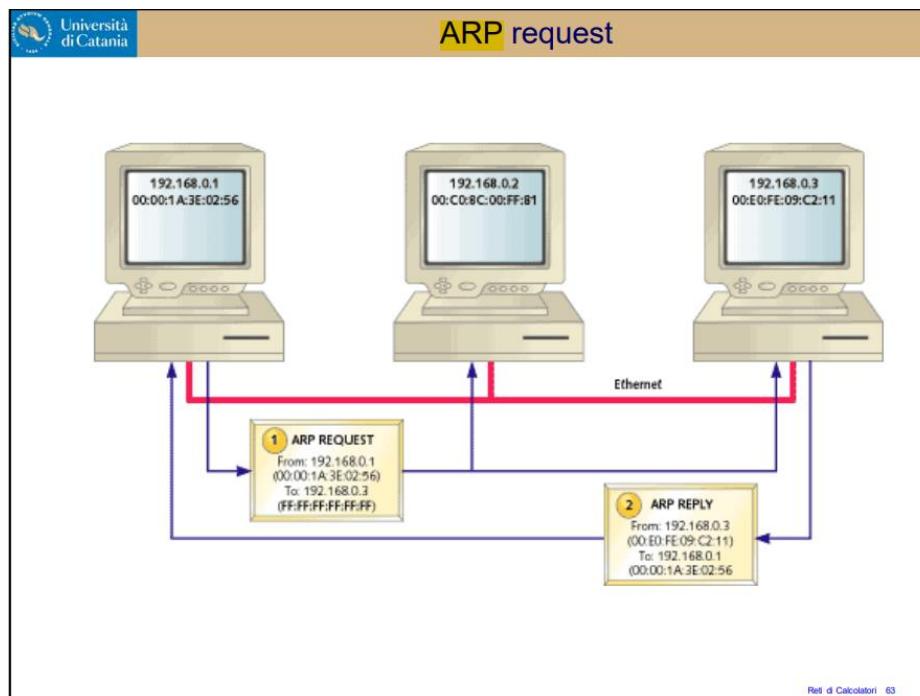
ARP, RARP, BOOTP

Protocollo ARP

Il Protocollo **ARP** (Address Resolution Protocol) è un componente fondamentale delle reti Ethernet e svolge il ruolo di **mappare** gli indirizzi **IP** ai corrispondenti indirizzi **MAC** (Media Access Control) all'interno di una rete locale.

Si basa su due passaggi: **ARP Request** ed **ARP Reply**

- 1. ARP Request:** Quando un dispositivo in una rete deve comunicare con un altro dispositivo, ma **non conosce** l'indirizzo **MAC** corrispondente all'indirizzo IP di destinazione, invia un pacchetto **ARP Request**. Questo pacchetto **chiede** agli altri **dispositivi** nella stessa rete di **rispondere** con il **loro** indirizzo **MAC se hanno la corrispondenza** cercata.
- 2. ARP Reply:** Quando un dispositivo riceve un ARP Request con l'indirizzo IP corrispondente al proprio, **risponde** con un **pacchetto** ARP Reply **contenente il proprio indirizzo MAC**. Questa risposta viene memorizzata nella tabella ARP del dispositivo mittente per future comunicazioni.



La **tabella ARP** è una tabella **utilizzata** per **memorizzare** temporaneamente le **associazioni** tra **indirizzi IP** e indirizzi **MAC** nella rete locale. Questa cache viene **aggiornata periodicamente** o quando vengono ricevute risposte ARP da altri dispositivi. La tabella ARP **permette ai dispositivi** di **evitare** di effettuare **richieste** ARP per ogni pacchetto inviato.

Indirizzo IP	Indirizzo LAN	TTL
222.222.222.221	88-B2-2F-54-1A-0F	13:45:00
222.222.222.223	5C-66-AB-90-75-B1	13:52:00

Rete di Calcolatori - 64

Tuttavia, il Protocollo ARP presenta alcune **vulnerabilità**:

- **Mancanza di autenticazione:** ARP **non prevede l'autenticazione** dei dispositivi che inviano richieste o risposte ARP. Questo significa che un dispositivo malevolo può **falsificare informazioni** ARP, inducendo altri dispositivi a **instradare il traffico** verso di esso.
- **ARP è senza stato:** ARP **non tiene traccia** degli stati precedenti **delle richieste ARP**. Questo significa che un dispositivo può rispondere a una richiesta ARP senza una corrispondente richiesta ARP precedente, aprendo la possibilità di **attacchi ARP falsi**.
- **Aggiornamento della ARP Table:** Un **host** che **riceve un pacchetto ARP** è tenuto ad **aggiornare** la sua **ARP table** con l'associazione IP-MAC fornita nel pacchetto, anche se questa **informazione è falsa**.

Le **vulnerabilità** dell'ARP possono essere **sfruttate** da attaccanti per effettuare **attacchi di intercettazione** del **traffico** o per **deviare il traffico** legittimo a loro vantaggio.

Protocollo RARP

Il protocollo **RARP** è stato utilizzato storicamente per risolvere un indirizzo **MAC** in un **indirizzo IP** in una rete locale. Tuttavia, il protocollo **RARP** è diventato **obsoleto** ed è stato in gran parte sostituito da **DHCP**, che offre funzionalità più avanzate e flessibili per l'assegnazione dinamica degli indirizzi IP e altre configurazioni di rete.

Protocollo BOOTP

Il protocollo **BOOTP** è stato **sviluppato** per fornire una **configurazione** iniziale e un **avvio** remoto dei **dispositivi** su una **rete**. BOOTP consente a un dispositivo di ottenere un **indirizzo** IP, un indirizzo del **server** di avvio e altre informazioni di configurazione.

IPv6

IPv6 nasce principalmente dal fatto che lo spazio di indirizzamento **IPv4** a 32 bit stava esaurendo a causa della **crescente connettività** di nuove sottoreti e nodi IP a Internet. IPv6 è stato **progettato** per affrontare la crescente **domanda** di **indirizzi IP** univoci e per **introdurre** altre **migliorie rispetto a IPv4**.

Indirizzi IPv6	
0000 0000	Riservati (includono IPv4)
0000 001	Indirizzi OSI NSAP (deprecated)
0000 010	Indirizzi Novell IPX
001	Indirizzi unicast globali
010	Indirizzi per service providers
100	Indirizzi geografici
1111 1110 10	Indirizzi unicast link-local
1111 1110 11	Indirizzi unicast site-link (deprecated)
1111 1111	Multicast

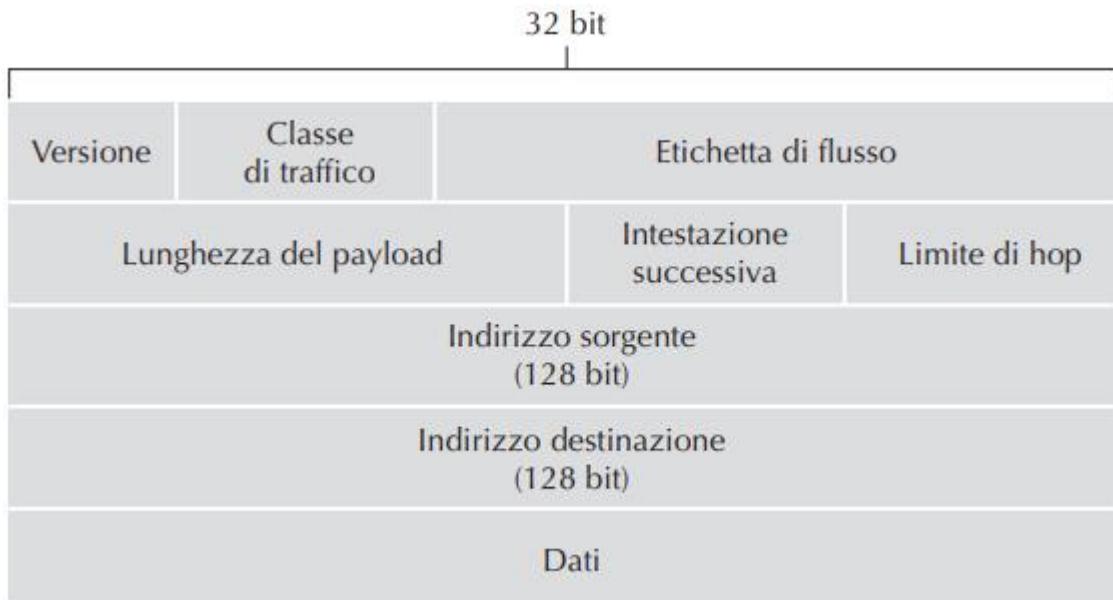
Formato dei datagrammi IPv6

Il **formato** dei **datagrammi** IPv6 presenta alcune **modifiche** significative rispetto a IPv4:

- **Indirizzamento Esteso:** IPv6 **estende** la dimensione degli indirizzi IP da **32** a **128** bit, fornendo una **quantità** praticamente **illimitata** di **indirizzi**. Oltre agli indirizzi **unicast** e **multicast**, IPv6 supporta anche gli indirizzi **anycast**, che consentono di inviare dati a **qualsiasi host** all'interno di un gruppo.
- **Intestazione Ottimizzata:** L'**intestazione** IPv6 è **fissa** a **40 byte** e contiene **solo i campi essenziali**. Le **opzioni** sono state **rimosse** dall'intestazione standard ma possono essere incluse come **intestazioni successive**.
- **Etichettatura dei Flussi:** IPv6 introduce il concetto di "**flusso**", che può essere utilizzato per **etichettare pacchetti** appartenenti a flussi speciali che richiedono un **trattamento diverso**, ad esempio, per la **qualità del servizio** o il servizio in tempo reale.

Il formato **IPv6** include i seguenti **campi principali**:

- **Versione** (4 bit): Identifica la **versione** IP (6 per IPv6).
- **Classe di Traffico** (8 bit): Simile al campo **Type of Service** (TOS) di IPv4, utilizzato per attribuire **priorità** a determinati datagrammi.
- **Etichetta di Flusso** (20 bit): Utilizzata per **identificare** un **flusso** di **datagrammi**.
- **Lunghezza del Payload** (16 bit): Indica la **lunghezza** dei **dati** nel datagramma IPv6.
- **Intestazione Successiva** (8 bit): **Identifica** il **protocollo** a cui verranno consegnati i dati del datagramma.
- **Limite di Hop** (8 bit): Decrementato da ogni router che inoltra il datagramma; **quando** raggiunge **0**, il **datagramma** viene **eliminato**.
- **Indirizzi Sorgente e Destinazione:** Definiscono i **mittenti** e i **destinatari** del datagramma.
- **Dati:** Il **payload** passato al protocollo specificato nel campo **Intestazione Successiva**.



Alcuni dei principali cambiamenti includono **l'eliminazione** della **frammentazione** e del **riassemblaggio** nei **router intermedi** (queste operazioni sono ora gestite **solo** da **sorgente e destinazione**) e l'assenza del **checksum** dell'intestazione, poiché i protocolli di livello superiore (come TCP e UDP) gestiscono i propri checksum.

Tutte queste modifiche mirano a semplificare l'elaborazione dei pacchetti IP e a migliorare l'efficienza complessiva della rete IPv6.

Riepilogo differenze tra IPv4 ed IPv6

Sono stati **rimossi** da IPv4 i seguenti campi

- **ID, Flags, Offset**: Venivano utilizzati per la **frammentazione**, non più necessari in IPv6
- **ToS** (inutilizzato)
- **HLEN**: Sappiamo per certo che la **dimensione dell'header** è fissa ed è **40 byte**
- **Checksum: Controllo superfluo** a questo livello di rete, è più affidabile se fatto a livello data link
- **Option**: Non del tutto rimosso, è una delle possibili "intestazioni successive" a cui punta l'header IPv6

Sono stati **modificati** i seguenti campi:

- **Total length** => Lunghezza del payload
- **Protocol** => Header successivo
- **TTL** => Hop Limit

Header opzionali

Gli **header** opzionali in **IPv6** sono **parti aggiuntive dell'intestazione** di un pacchetto IPv6 che possono essere utilizzate per fornire informazioni aggiuntive o **opzionali**. Questi **header** vengono **inseriti tra l'intestazione** principale IPv6 e il **carico utile** dei **dati**. Hanno un funzionamento simile a quello di una **lista semplicemente linkata**. In breve, gli header opzionali in IPv6 **offrono** una **flessibilità maggiore** rispetto a IPv4, consentendo di **includere informazioni** aggiuntive **solo quando necessario**, **semplificando** così la **gestione** dei **pacchetti** nella **rete**.

Indirizzi globali unicast, locali, di sito

- **Global Unicast:**
 - Indica una **singola macchina**. I primi 48 bit indicano il sito: 3 dicono il tipo di **indirizzo**, 45 il **sito** a cui si vuole fare riferimento
- **Indirizzi Locali:**
 - Iniziano per **FE80::/64** e sono utilizzabili solo tra nodi dello **stesso link**. Non possono essere usati su internet.
- **Indirizzi di sito:**
 - Iniziano per **FEC0::/48** e sono utilizzabili solo tra nodi dello **stesso sito**. Non possono essere usati su internet.

Esempio:

2001:760:4804:0:0:0:105 | Indirizzo **IPv6** che indica la **singola macchina**

2001:760:4804::/48 | **Sottorete** con maschera

Unicast, Multicast ed Anycast

- **Unicast:** Tutti quegli indirizzi che **iniziano** per **2 o 3**
- **Multicast:** Iniziano per **FF**, si differenzia dal broadcast perché è **ristretto** ad un **determinato sottogruppo** di host indirizzabili (broadcast non fa distinzione)
- **Anycast:** Finiscono con una **serie** di 0, sono macchine **indirizzabili** in **Internet** tramite router che si occuperanno dello smistamento.

Firewall e DMZ

Firewall

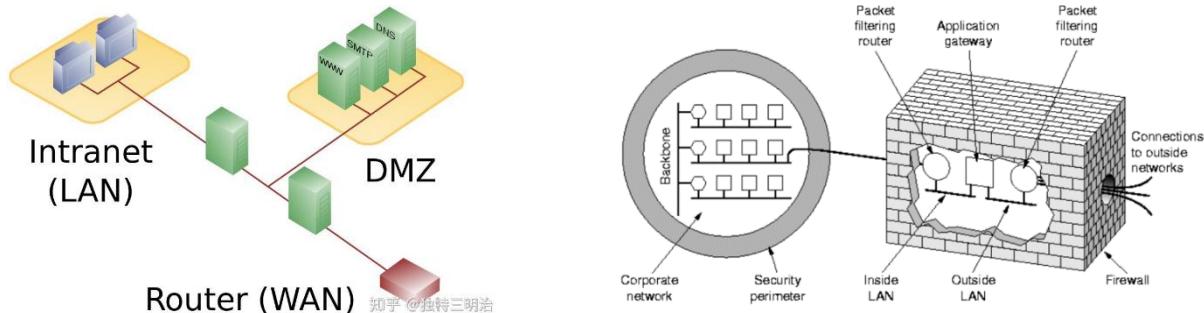
Il **firewall** è uno strumento importante per la **sicurezza** informatica, e ne esistono due tipi principali: **firewall software** e **firewall hardware**.

- Un **firewall software** è **un'applicazione** che gira su un dispositivo e si basa su un **insieme** di **regole** per decidere quali pacchetti di dati possono **entrare** o **uscire** dal dispositivo.
- Un **firewall hardware**, d'altro canto, è un **dispositivo** dedicato progettato per **filtrare** il **traffico** di **rete**. Questo li rende più **difficili** da **violare** perché contengono meno complessità rispetto a un sistema operativo completo.

Tuttavia, è importante notare che i **firewall hardware** sono più **costosi rispetto** ai **firewall software**. Quindi, la scelta tra i due dipenderà dalle esigenze specifiche di **sicurezza** e dal budget disponibile.

DMZ

La **DMZ** (Demilitarized Zone) è una zona nella rete informatica che, dal punto di vista della **sicurezza**, si trova in una **posizione intermedia** tra la rete **interna** e quella **esterna**. Nel contesto informatico, la DMZ viene spesso utilizzata per ospitare **server** o servizi che devono essere **accessibili dall'esterno**, come **server web** o server di posta elettronica. Tuttavia, questi server sono separati dalla rete interna per **garantire** un ulteriore strato di **sicurezza**. In sistemi più avanzati, è possibile creare una rete DMZ separata a cui collegare **fisicamente** le macchine **esposte** a **Internet**. In questo caso, la sicurezza non viene delegata completamente al **firewall** del router, ma è necessario avere **regole** di sicurezza **specifiche** per la DMZ per **proteggere** le **risorse interne**.



Algoritmi di Routing

Per formulare i problemi di **instradamento**, si utilizza un **grafo** in cui i **nodi** rappresentano i **router** e gli **archi** i **collegamenti** tra di essi, con ogni **arco** associato a un **costo**. L'obiettivo principale di un algoritmo di instradamento è **trovare** il **percorso** con il costo **minimo** tra sorgente e destinazione in questo grafo.

Gli algoritmi di instradamento possono essere **classificati** in diverse **categorie**:

- **Centralizzati** vs. **Decentralizzati**: Gli algoritmi **centralizzati** calcolano **percorsi minimi** con una **conoscenza globale** della **rete**, mentre quelli **decentralizzati** operano in modo **distribuito** e **iterativo**, con ogni nodo che elabora informazioni solo sui collegamenti diretti. Esempio link-state (centralizzato) e distance-vector (decentralizzati).
- **Statici** vs. **Dinamici**: Gli algoritmi **statici** cambiano raramente e spesso **richiedono intervento umano** per **modificare i percorsi**. Gli algoritmi **dinamici** si **adattano** alle **variazioni del traffico** o della topologia di rete e possono essere eseguiti periodicamente o in risposta a cambiamenti.

La scelta tra algoritmi centralizzati o decentralizzati, statici o dinamici, sensibili o insensibili al carico **dipende** dalle **esigenze** specifiche della **rete** e dalle **condizioni** di **traffico**.

Flooding

Il **flooding** è una **tecnica di instradamento** che ha **vantaggi** e **svantaggi** specifici, ed è importante considerarli quando si decide di utilizzarla in una rete.

Vantaggi del Flooding

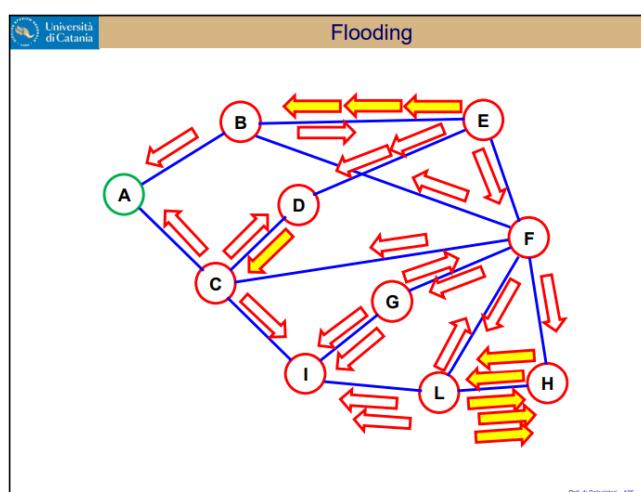
- **Trova un percorso verso la destinazione, se esiste:** Uno dei principali vantaggi del flooding è che è **garantito trovare** un **percorso** verso la **destinazione**, a condizione che esista un percorso possibile nella rete.
- **Trova il percorso migliore:** Sebbene il flooding trasmetta il messaggio a tutti i nodi, è possibile che uno dei primi nodi a ricevere il messaggio sia il nodo destinatario, quindi il **percorso più breve** potrebbe essere scoperto rapidamente.

Svantaggi del Flooding

- **Crea un numero elevato di pacchetti duplicati:** Il principale svantaggio del flooding è che genera un grande numero di pacchetti duplicati, poiché il messaggio viene inviato a tutti i nodi, inclusi quelli che potrebbero aver già ricevuto lo stesso messaggio. Questo può **saturare la rete troppo velocemente**.

Come Risolvere i Problemi del Flooding:

- **Usare un ID per i pacchetti:** Ogni pacchetto può contenere un identificatore univoco in modo che i **nodi possano riconoscere** e **scartare i pacchetti duplicati**.
- **Usare un HOP limit:** Si può limitare il numero di hop (salti) che un pacchetto può fare attraverso la rete. Quando il contatore di **hop** raggiunge il **limite**, il **pacchetto** viene **scartato**.
- **Eliminare i cicli dal grafo:** Un'altra soluzione è **assicurarsi** che la **rete** sia **priva** di **cicli**. Questo può essere ottenuto utilizzando algoritmi di rilevamento e prevenzione dei cicli.
- **Inoltrare solo lungo uno Spanning Tree:** Limitare la propagazione dei pacchetti a un albero di copertura (spanning tree) della rete può aiutare a evitare loop e ridurre la congestione (**trasforma** un **grafo** in un **albero**).



Distance Vector (DV)

L'algoritmo di instradamento a Vettore Distanza (**DV**) è un approccio **decentralizzato e asincrono** per determinare le rotte ottimali all'interno di una rete. Ecco un riassunto dei principali concetti relativi all'algoritmo DV:

Asincrono e Distribuito

- Nell'instradamento DV, ogni **nodo** in una rete **comunica** solo con i suoi **vicini** direttamente **connessi**.
- Non è necessario che tutti i nodi operino in modo sincronizzato; ogni **nodo** può **aggiornare** le sue **informazioni** in modo **indipendente**.

Auto-Terminante

- L'algoritmo DV è auto-terminante, il che significa che **non richiede** un **segnale** specifico per **interrompere** i **calcoli**. Si blocca quando non avvengono ulteriori scambi informativi tra i vicini.

Formula di Bellman-Ford

- L'algoritmo DV utilizza la formula di **Bellman-Ford** per calcolare i percorsi a costo minimo tra i nodi.
- Questa formula si basa su stime di costo tra i nodi e viene utilizzata per aggiornare le stime di costo nei nodi.

Aggiornamento dei Vettori delle Distanze

- Inizialmente, ogni **nodo** ha una **stima** dei **costi** ai suoi **vicini** direttamente **connessi** e **mantiene** un **vettore** delle **distanze** per tutte le **destinazioni** nella rete.
- I **nodi scambiano i loro vettori** delle **distanze** con i **vicini** e utilizzano la formula di Bellman-Ford per aggiornare i costi minimi per le destinazioni.
- Questo **processo** di **aggiornamento** si **ripete** fintanto che ci sono **cambiamenti** nei costi minimi.

Tabella di Instradamento

- Ogni nodo mantiene una **tabella di instradamento** che indica il **router successivo** per raggiungere una determinata destinazione.
- La tabella di instradamento viene aggiornata in base alle informazioni ricevute dai vicini.

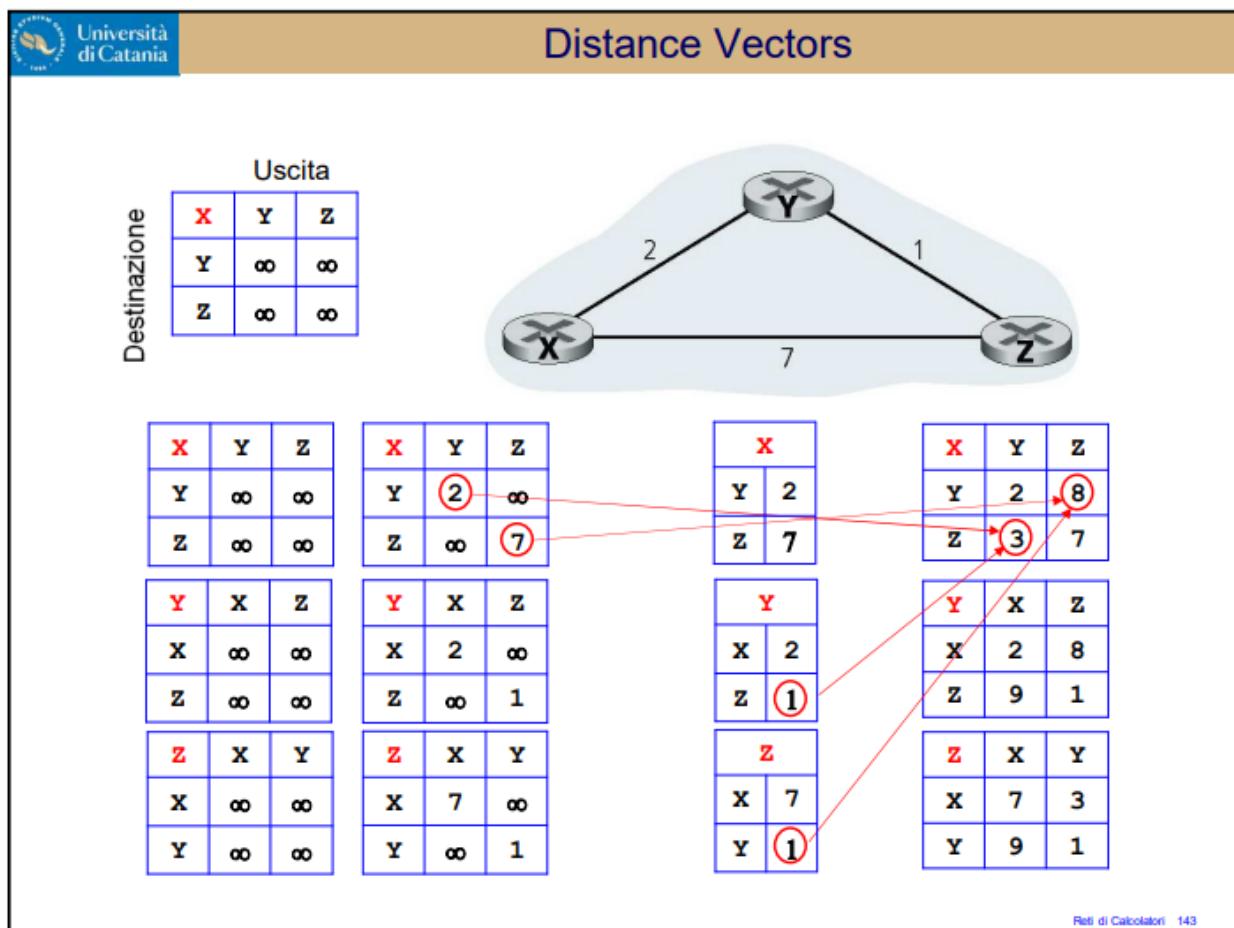
Convergenza

- Nonostante **l'asincronicità** degli aggiornamenti, l'algoritmo DV converge verso i costi minimi effettivi dei percorsi.
- Ciascuna **stima** dei costi si **avvicina** progressivamente ai **costi minimi** reali.

Applicazioni

- L'instradamento DV viene utilizzato in vari protocolli di instradamento del mondo reale, tra cui **RIP**.

Il riassunto evidenzia come l'algoritmo DV **consenta** ai **nodi** di **collaborare** per **determinare** i **percorsi** a costo **minimo** in modo **distribuito** e **asincrono**, senza richiedere un coordinamento **centralizzato**.



Caso di decremento del costo dei collegamenti:

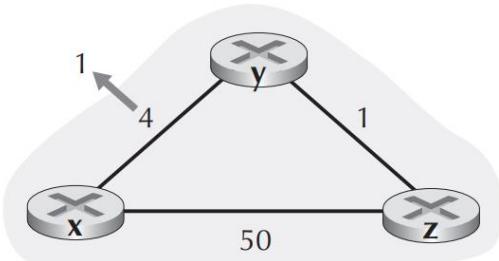
Quando il costo di un collegamento tra nodi **diminuisce** (ad esempio, da **4 a 1**) nel protocollo di **instradamento distance-vector**, il processo procede senza problemi:

- **Cambio di costo:** Il nodo Y **rileva il cambiamento** e **aggiorna** il suo vettore delle **distanze**.
- **Aggiornamento dei vicini:** Y **comunica il cambiamento** ai suoi vicini se influisce sul percorso a costo minimo.

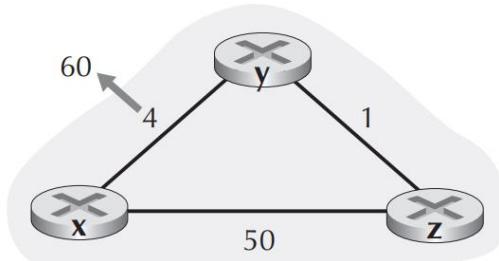
Caso di aumento del costo dei collegamenti:

Quando il costo di un collegamento **aumenta** notevolmente (ad esempio, da **4 a 60**), sorge il problema del "**conteggio all'infinito**" e **l'inversione avvelenata** offre una soluzione:

- **Calcolo del nuovo percorso:** Y calcola il nuovo percorso a costo minimo verso X, **considerando** ancora il **percorso attraverso Z** (costo 5), creando un **ciclo**.
- **Conteggio all'infinito:** Si verifica il problema del "conteggio all'infinito" poiché Y e Z continuano a instradare **l'uno** attraverso **l'altro**, creando un **ciclo infinito**.
- **Inversione avvelenata:** Per risolvere il problema, Z comunica a Y che la sua **distanza** verso X è **infinita** ($+\infty$), nonostante Z conosca il vero costo. Questo **impedisce** a Y di continuare ad **instradare** attraverso Z.
- **Risoluzione:** Il ciclo si **interrompe** quando Y riceve l'informazione da Z che la distanza è **infinita**. Successivamente, Y preferirà il percorso diretto da Y a X.



a.



b.

Link State Routing (LSR)

L'instradamento **Link-State** (LS) è un approccio all'instradamento di rete in cui ogni nodo **conosce la topologia completa** della rete e i **costi** dei **collegamenti**

Funzionamento dell'Instradamento Link-State (LS):

- **Conoscenza della topologia:** Ogni nodo **dispone** di una **visione completa** della **rete**, con informazioni dettagliate su tutti i collegamenti e i loro **costi**. Queste informazioni sono raccolte attraverso **scambi** di pacchetti tra i **nodi**.
- **Calcolo dei percorsi:** L'algoritmo utilizzato nell'instradamento **LS** è spesso l'algoritmo di **Dijkstra**. Esso calcola il percorso a costo minimo da un nodo di origine a tutti gli altri nodi, basandosi sulla topologia completa.

Differenze rispetto all'instradamento Distance-Vector

A differenza dell'instradamento **Distance-Vector**, in cui i nodi **condividono** solo **informazioni** limitate sui loro **vicini**, nell'instradamento **LS** ogni nodo ha una **visione completa** della **rete**. Non ci sono aggiornamenti periodici delle tabelle di routing; invece, le **tabelle** vengono **aggiornate** solo quando c'è un **cambiamento** nella **topologia**.

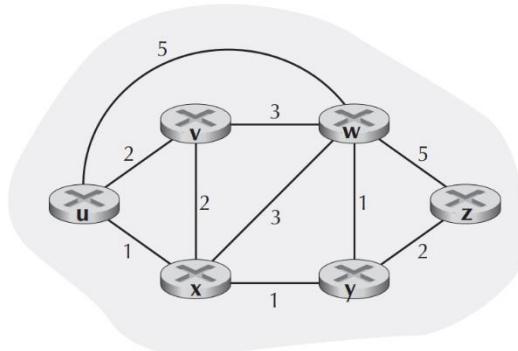


Tabella 5.1 Esecuzione dell'algoritmo Dijkstra sulla rete della Figura 5.3.

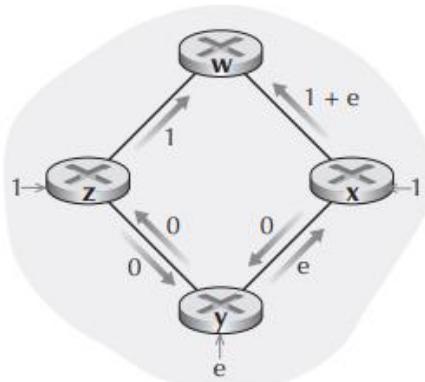
Passo	N'	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					

Problema delle oscillazioni

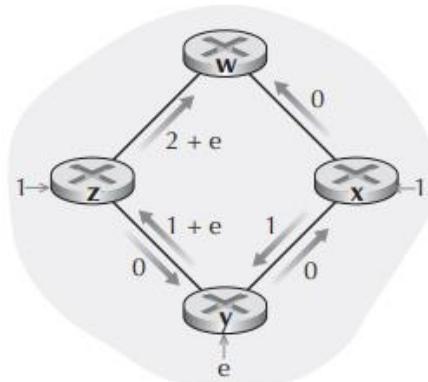
In questa situazione, i nodi possono **alternare** continuamente i loro **percorsi a costo minimo** a causa delle fluttuazioni nei costi dei collegamenti.

Supponiamo che ci siano quattro nodi: x , y , z e w , e che ci siano collegamenti tra di loro con **costi basati** sulla **quantità** di **traffico** trasportato. Inizialmente, x e z inviano un'unità di traffico ciascuno a w , mentre y ne **invia una quantità maggiore**, rappresentata da "e".

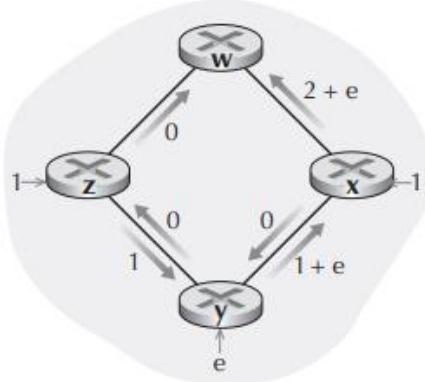
- **Prima Iterazione:** Nella prima iterazione dell'algoritmo LS, ogni nodo **calcola** il **percorso a costo minimo** per raggiungere w . Il nodo y decide che il percorso in **senso orario** ha un costo **inferiore**, mentre x decide lo stesso. Di conseguenza, entrambi x e y scelgono il percorso in senso orario.
- **Iterazioni Successive:** Nelle iterazioni successive, x , y e z possono continuare a **cambiare** il loro **percorso preferito** tra **orario** e **antiorario** a causa delle fluttuazioni nei costi dei collegamenti. Questo causa **oscillazioni continue** nei percorsi a costo minimo e può portare a **instabilità nell'instradamento**.



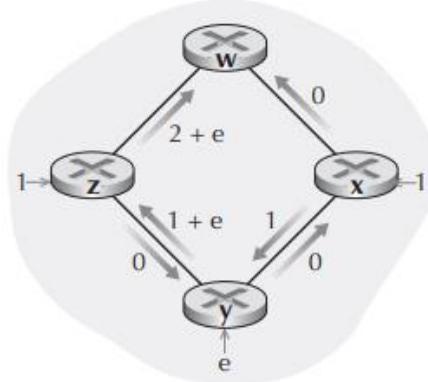
a. Instradamento iniziale



b. x, y determinano un percorso migliore verso w , in senso orario



c. x, y, z determinano un percorso migliore verso w , in senso antiorario



d. x, y, z determinano un percorso migliore verso w , in senso orario

Link State Routing VS Distance Vector

Il confronto tra gli instradamenti **Link-State** (LS) e **Distance-Vector** (DV) evidenzia le differenze chiave tra i due approcci:

- **Complessità dei messaggi:** In LS, ogni **nodo** deve **conoscere** i costi di tutti i collegamenti nella rete, il che richiede l'invio di un **elevato numero** di **messaggi**, approssimativamente $O(|N| \cdot |E|)$. Inoltre, ogni cambiamento nei costi dei collegamenti deve essere **comunicato** a **tutti i nodi**. In DV, i nodi comunicano **solo** con i loro **vicini** direttamente connessi, quindi **richiede meno messaggi**.
- **Velocità di convergenza:** LS ha una complessità computazionale di $O(|N|^2)$, richiedendo $O(|N| \cdot |E|)$ messaggi. DV può **convergere lentamente** e può manifestare **cicli di instradamento**
- **Robustezza:** In caso di **guasti, problemi** di funzionamento o **sabotaggi**, LS può consentire a un nodo di comunicare **costi errati solo** per i collegamenti direttamente **connessi**, poiché i **calcoli** di instradamento sono **isolati**. Con DV, un nodo può comunicare **percorsi** a costo minimo **errati a tutte le destinazioni**, rendendo la rete più **vulnerabile** agli **errori** di instradamento e permettendo **dirottamenti del traffico**.

DV	LSR
Decentralizzato	Globale
Messaggi solo per i vicini	Messaggi in Broadcast
Informazioni riguardanti tutte le destinazioni	Informazioni riguardanti solo i propri link
Conteggio all'infinito	Problemi di sincronia (oscillazioni)
Messaggi solo per variazioni sui link	Maggiore traffico di messaggi (invii periodici)
Poco robusto nei confronti di nodi maliziosi	Robusto agli attacchi

Internet Routing

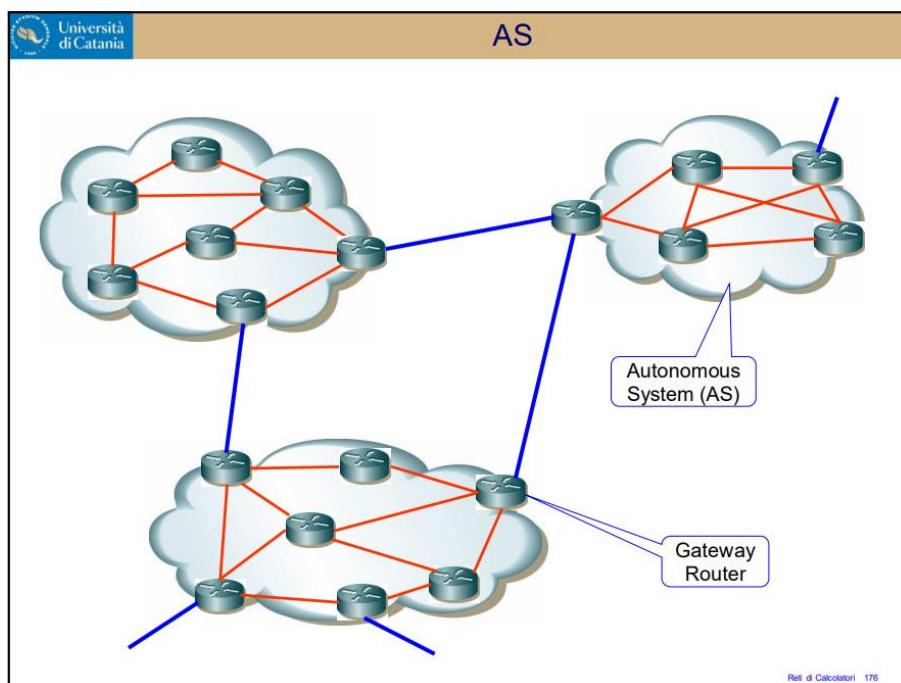
In Internet, il **routing** è principalmente **dinamico**, il che significa che le rotte e le decisioni di instradamento possono **cambiare** in risposta a modifiche nella **topologia** di rete o nei **costi** dei collegamenti.

Intra-AS e Inter-AS Routing

Internet è divisa in "autonomous systems" (AS), che sono unità di politica di instradamento. Gli AS possono essere costituiti da una **singola rete** o da un **gruppo** di **reti** controllate da un **amministratore** di rete. All'interno di ciascun AS, viene utilizzato **l'intra-AS routing** per determinare le rotte all'interno dell'AS stesso. Per collegare tra loro gli AS e instradare il traffico tra di essi, viene **utilizzato l'inter-AS routing**, che è responsabile dell'interscambio di informazioni di routing tra AS diversi. All'interno di ciascun AS, le reti e i router sono **fortemente connessi**, il che significa che esiste una **connettività diretta** e completa tra i componenti all'interno dell'AS. Questo consente un **instradamento** più **efficiente** e **preciso** all'interno dell'AS.

Border Gateway Protocol (BGP)

Il Border Gateway Protocol (BGP) è un protocollo di instradamento utilizzato per gestire il traffico tra i diversi domini autonomi (AS) su Internet. Funziona basandosi su politiche di instradamento personalizzate, annunciando rotte e garantendo la connettività globale. BGP è essenziale per il funzionamento di Internet.



Routing Internet Protocol (RIP)

RIP v1

RIP v1 è una **versione iniziale** del protocollo di instradamento **RIP** basato sul concetto di **Distance Vector**. Utilizza una metrica di instradamento denominata "**HOP**" (numero di salti effettuati) per determinare i percorsi più brevi. Tuttavia, RIP v1 ha alcune **limitazioni** significative. Ad esempio, **supporta** solo **reti** con un massimo di **15 hop** e non è in grado di gestire **subnetting** o **routing** basato su **classi**.

RIP v1 aggiorna periodicamente le **tabelle** di **instradamento**, inviando **messaggi** di routing ogni **30 secondi**. Se un percorso non viene aggiornato entro **180 secondi**, la sua distanza viene considerata **infinita**, e dopo ulteriori **120 secondi** (garbage-collection timer), il percorso viene **rimosso** dalla tabella di **instradamento**.

Per evitare **loop** nell'instradamento, **RIP v1 utilizza** una tecnica chiamata "**poisoned reverse**." In caso di perdita di un percorso, il router annuncerà la distanza a **infinito** (**16 hop**) per quel percorso ai suoi vicini, indicando che il percorso non è più valido. Questo impedisce ai **router** di continuare a **instradare pacchetti** verso un percorso **irraggiungibile**.

Le **tabelle** di instradamento in RIP v1 **contengono** le **seguenti informazioni** per ogni percorso:

- **Indirizzo di destinazione.**
- **Distanza** dalla destinazione in **hop** (numero di salti effettuati).
- **Next hop:** l'indirizzo del router adiacente al quale inviare i pacchetti per raggiungere la destinazione.
- **Timeout:** il tempo dopo il quale considerare il percorso come obsoleto.
- **Garbage-collection timer:** il tempo dopo il quale eliminare completamente il percorso dalla tabella.

RIP v1 utilizza **due** tipi di **messaggi**:

- **REQUEST:** Usato per **richiedere** informazioni di instradamento ai nodi adiacenti.
- **RESPONSE:** Usato per **inviare** informazioni di instradamento ai nodi adiacenti.

Struttura del datagramma RIP v1

Command	Version	Must Be Zero
Address Family Identifier		Must Be Zero
IP Address		
Must Be Zero		
Must Be Zero		
Metric		

- **Command:** 1 (**Richiesta**) o 2 (**Aggiornamento**).
- **Version:** Versione del protocollo (sempre **1** per RIP v1).
- **Address Family Identifier:** Sempre **2** per il protocollo IP.
- **IP Address:** Indirizzo di **destinazione** (rete o sottorete).
- **Metric:** Numero di **hop** (valore compreso tra 1 e 15).

RIP v2

RIP v2 è un'evoluzione di RIP v1 che supera alcune delle limitazioni della versione precedente. Introduce il supporto per **CIDR** e **VLSM**, consentendo una pianificazione più flessibile delle **subnet** e l'indirizzamento più efficiente delle reti.

Altre caratteristiche di RIP v2 includono **l'autenticazione** dei messaggi per garantire l'integrità delle informazioni di routing, la specifica del prossimo hop (**next hop**) per migliorare la precisione dell'instradamento e l'implementazione di "**split horizon**" per evitare loop nell'instradamento.

- **Poisoned Reverse:** Un router comunica ai suoi vicini la distanza a **infinito** per un percorso che ha subito una **perdita**. Questo **impedisce** ai vicini di **utilizzare** nuovamente il **percorso** in questione per l'invio dei pacchetti.
- **Split Horizon:** **Impedisce** a un router di **inviare** informazioni di **routing** a un **vicino** se quelle informazioni sono state **apprese** dal **vicino** stesso. Questa tecnica **evita** che un pacchetto venga **inoltrato nuovamente** verso il **router** da cui è stato **ricevuto**, contribuendo a **prevenire loop** nell'instradamento.

Struttura del datagramma RIP v2:

0	7	15	31
Command	Version	0000	
0xFFFF		Autentication Type	
Autentication			
Address Family Ident.		Route Tag	
IP Address			
Subnet Mask			
Next Hop			
Metric			

- **Command:** 1 (Richiesta) o 2 (Aggiornamento).
- **Version:** Versione del protocollo (sempre 2 per RIP v2).
- **Address Family Identifier:** Sempre 2 per il protocollo IP.
- **Route Tag:** Identificazione di **route** esterni.
- **IP Address:** Indirizzo di **destinazione** (rete o sottorete).
- **Metric:** Numero di **hop** (valore compreso tra 1 e 15).

RIPng

RIPng è una variante di RIP basata su **RIPv2**, ma è progettata esclusivamente per **IPv6**, non supportando IPv4. Riprende molte delle caratteristiche di RIPv2, come **CIDR** e **VLSM**. **Non include l'autenticazione** dei messaggi.

0	7	15	31
Command	Version	0000	
IPv6 Prefix			
Route Tag		Prefix len.	Metric
...			

Open Shortest Path First (OSPF)

L'**OSPF** (Open Shortest Path First) è un **protocollo** di instradamento progettato per **afrontare** le **limitazioni** di protocolli più vecchi come **RIP**.

- **Sostituto di RIP:** OSPF è stato sviluppato per **superare** le **limitazioni** di **RIP** adottando l'approccio **Link State Routing**, offrendo una maggiore **scalabilità** e **flessibilità** per reti di grandi dimensioni.
- **Link State Database:** Ogni router all'interno di un sistema autonomo mantiene un database detto "**link state database**" contenente informazioni sui **collegamenti** verso tutti gli altri router nella rete. Ogni record prende il nome di **Link State Record** (LSR).
- **Autenticazione dei Messaggi:** I messaggi OSPF possono essere **autenticati**, garantendo che solo router **autorizzati** possano partecipare al protocollo e **influenzare** le tabelle di **instradamento**.
- **Utilizzo di Metriche Diverse:** OSPF consente agli amministratori di rete di utilizzare metriche diverse per **determinare i percorsi ottimali**.

Il protocollo OSPF è diviso in **tre parti principali**:

- **HELLO:** Questa fase coinvolge la **scoperta** e la **verifica** dei router **adiacenti**. I router inviano messaggi "HELLO" per **rilevare** i loro **vicini** e stabilire **connessioni** con essi.
- **EXCHANGE:** Durante questa fase, avviene la **sincronizzazione iniziale** dei **database** link-state tra i **router adiacenti**. I router scambiano informazioni sui loro **collegamenti** e stabiliscono un'immagine coerente del **network**.
- **FLOODING:** In questa fase, gli aggiornamenti del database link-state vengono **diffusi** nell'intera **area** di OSPF.

Gli **LSA** (Link State Advertisement), che rappresentano gli stati dei collegamenti, **vengono emessi** quando si verifica **uno** dei seguenti **eventi**:

- Quando un router rileva un **nuovo router adiacente**.
- Quando si verifica un **guasto** su un **collegamento**.
- Quando il costo di un **collegamento cambia**.
- Periodicamente, tipicamente ogni **30 minuti**.

Livello DLL

Describe nodi e collegamenti, un esempio di trasmissione di dati in una rete aziendale. Successivamente, offre un analogo nel mondo dei trasporti per spiegare meglio questi concetti.

Servizi offerti dal livello di collegamento

I **servizi** offerti dal livello di **collegamento** sono:

- **Framing:** La maggior parte dei protocolli **incapsula** i **datagrammi** del **livello di rete** all'interno di un **frame** a livello di **collegamento**. Un frame è composto da un campo dati che contiene il datagramma e vari campi di intestazione, con la struttura specificata dal protocollo.
- **Accesso al collegamento:** I protocolli di controllo dell'accesso al mezzo trasmisivo **stabiliscono** le **regole** per l'**invio** dei **frame** nel collegamento. Questo è importante soprattutto nei collegamenti condivisi da più nodi, dove un protocollo coordina la trasmissione dei frame.
- **Consegna affidabile:** Alcuni protocolli a livello di collegamento forniscono un **servizio** di **consegna affidabile**, garantendo il trasporto senza errori di ogni datagramma. Questo servizio può essere realizzato attraverso **ACK** e **ritrasmissioni** ed è utile soprattutto in collegamenti con elevati tassi di errore.
- **Rilevazione e correzione degli errori:** Molti protocolli di collegamento includono **meccanismi** per **rilevare** la presenza di **errori** nei dati trasmessi. Questo è fatto attraverso l'aggiunta di bit di controllo di errore nel frame e l'esecuzione di un controllo da parte del nodo ricevente. Alcuni protocolli possono anche **correggere** gli errori, determinando il punto esatto del frame in cui si è verificato l'errore e correggendolo.

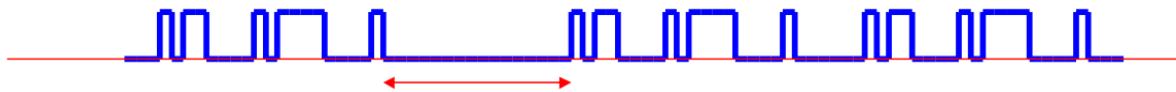
Framing dei dati

Problema del Framing

Riguarda il problema **dell'ambiguità** nel riconoscere una lunga **assenza di luce** in un **cavo**, dove la **luce** rappresenta **1** e **l'assenza** di luce rappresenta **0**. Questa ambiguità può causare problemi nella comunicazione.

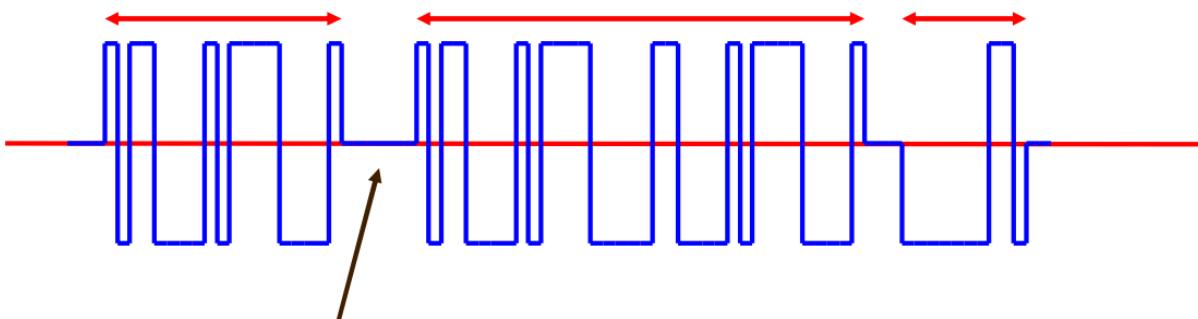
Codifica a due livelli (Binary Encoding):

In questa modalità, l'informazione viene rappresentata utilizzando solo **due livelli** o stati distinti, di solito **0** e **1**. Questo è il tipo più comune di codifica ed è alla base dell'Ethernet che trasmette dati utilizzando segnali elettrici che possono essere interpretati come 0 o 1. Il limite della codifica a due livelli è la sua **incapacità** di distinguere tra **l'assenza** di un segnale e una sequenza di **zeri binari**.



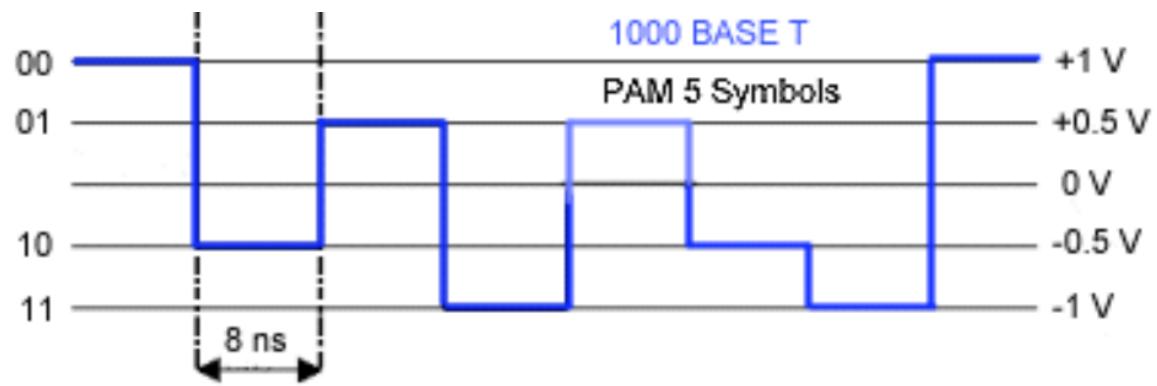
Codifica a tre livelli (Ternary Encoding):

Qui, l'informazione è rappresentata utilizzando **tre** livelli distinti, di solito **-1, 0 e 1**. Questo tipo di codifica è meno comune di quella binaria ma è utilizzata in alcune **applicazioni specifiche**, ad esempio nei computer quantistici o in alcune reti neurali artificiali. L'aspetto critico della codifica a tre livelli è la presenza di una **ridondanza** equivalente a **un terzo** della larghezza di **banda**.



Codifica a cinque livelli (Quinary Encoding):

In questo caso, l'informazione è rappresentata utilizzando **cinque livelli** distinti. Questo tipo di codifica è ancora **meno comune** e viene utilizzato in situazioni molto specifiche dove è necessaria una maggiore capacità di rappresentazione dei dati rispetto a binaria o ternaria. La codifica a cinque livelli permette di rappresentare le **coppie** di **bit** 00, 01, 10 e 11. **Ridondanza** pari ad **1/5** della banda.



Codifica 4b5b

In questa codifica, ogni gruppo di **4 bit** di dati viene convertito in un **codeword** di **5 bit**. Questo viene fatto per garantire che ci siano abbastanza transizioni tra 0 e 1 nei dati, aiutando nella **sincronizzazione** e nella **correzione** degli **errori**. In questa codifica, **non esistono codeword** che contengono più di **quattro zeri consecutivi**, il che indicherebbe **l'assenza di segnale**. La codifica 4B5B è utilizzata principalmente per migliorare la sincronizzazione e la rilevazione degli errori nei segnali di trasmissione. È comunemente usata nell'Ethernet a 100 Mbps.

Ridondanza pari ad **1/5** della banda.

Framming dei dati - Codifica 4B5B			
Nome	4B	5B	Descrizione
0	0000	11110	hex data 0
1	0001	01001	hex data 1
2	0010	10100	hex data 2
3	0011	10101	hex data 3
4	0100	01010	hex data 4
5	0101	01011	hex data 5
6	0110	01110	hex data 6
7	0111	01111	hex data 7
8	1000	10010	hex data 8
9	1001	10011	hex data 9
A	1010	10110	hex data A
B	1011	10111	hex data B
C	1100	11010	hex data C
D	1101	11011	hex data D
E	1110	11100	hex data E
F	1111	11101	hex data F
I	-NONE-	11111	Idle
J	-NONE-	11000	SSD #1
K	-NONE-	10001	SSD #2
T	-NONE-	01101	ESD #1
R	-NONE-	00111	ESD #2
H	-NONE-	00100	Halt

Reti di Calcolatori - 8

Codifica 8B10b

La codifica **8B10B** è un metodo utilizzato in standard come SATA, USB 3.0, HDMI e Gigabit Ethernet per trasmettere **dati** ad alta **velocità**. È progettata per mantenere un **bilanciamento** tra bit 1 e 0 (il che significa che cerca di mantenere il numero di bit **1** il più possibile **uguale** al numero di bit **0** trasmessi.)

Il processo inizia con 8 bit, che vengono suddivisi in **due gruppi**: uno di **5** bit e uno di **3** bit. Questi gruppi vengono quindi convertiti in 6 bit e 4 bit, tramite le codifiche **5b6b** e **3b4b**, rispettivamente. **Ridondanza** pari ad **1/5** della banda.

Codifica 8B10B		
3b Decimal	3b Binary (HGF)	4b Binary (fghi)
0	000	0100 or 1011
1	001	1001
2	010	0101
3	011	0011 or 1100
4	100	0010 or 1101
5	101	1010
6	110	0110
7	111	0001 or 1110 or 1000 or 0111

Reti di Calcolatori - 12

Codifica 8B10B		
5b Decimal	5b Binary (EDCBA)	6b Binary (abcde)
0	00000	100111 or 011000
1	00001	011101 or 100010
2	00010	101101 or 010010
3	00011	110001
4	00100	110101 or 001010
5	00101	101001
6	00110	011001
7	00111	111000 or 000111
8	01000	111001 or 000110
9	01001	100101
10	01010	010101
11	01011	110100
12	01100	001101
13	01101	101100
14	01110	011100
15	01111	010111 or 101000
16	10000	011011 or 100100
17	10001	100011
18	10010	010011
19	10011	110010
20	10100	001011
21	10101	101010
22	10110	011010
23	10111	111010 or 000101
24	11000	110011 or 001100
25	11001	100110
26	11010	010110
27	11011	110110 or 001001
28	11100	001110
29	11101	101110 or 010001
30	11110	011110 or 100001
31	11111	101011 or 010100

Reti di Calcolatori - 13

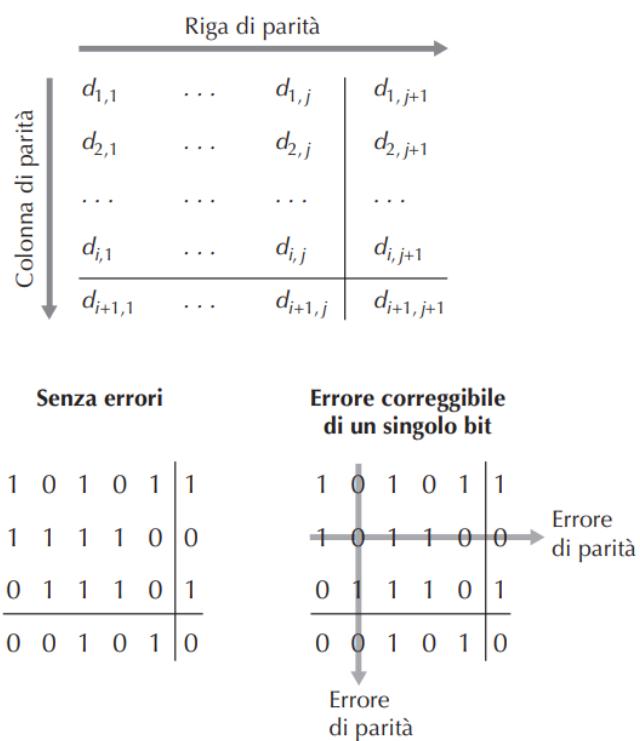
Tecniche di rilevazione e correzione degli errori

Controllo di Parità

Il controllo di parità è una **tecnica** di base per **rilevare errori** nei dati durante la trasmissione. In questa tecnica, un **bit** di **parità** viene aggiunto ai dati da inviare, **in modo** da **rendere pari** o dispari il **numero totale** di **bit 1** nei dati originali più il bit di parità, a seconda dello schema scelto (parità pari o parità dispari). Il ricevente può quindi contare il numero di bit a 1 nei dati ricevuti e, se il numero è dispari, sa che si è verificato almeno un errore nei bit.

Tuttavia, il controllo di parità ha **limitazioni**. Se si verifica un **numero pari** di **errori** nei dati, questo schema **non può rilevarli**. Inoltre, il controllo di parità è vulnerabile agli errori a raffica, in cui gli errori tendono a verificarsi in gruppo. La probabilità di non rilevare errori a raffica con un solo bit di parità può essere elevata.

Per **afrontare** queste **limitazioni**, il testo introduce una **generalizzazione bidimensionale** del **controllo di parità**, in cui i dati sono **suddivisi** in **righe** e **colonne**, ognuna con il suo bit di parità. **Questo schema** bidimensionale **consente** al ricevente non solo di **rilevare** gli **errori**, ma anche di **identificare** la **posizione** del bit errato e **correggerlo**. Questa capacità di rilevare e correggere gli errori è nota come "forward error correction" (FEC) ed è ampiamente utilizzata in dispositivi audio e reti. Le tecniche FEC possono migliorare l'affidabilità delle comunicazioni, riducendo la necessità di ritrasmissioni.



Checksum

Il **checksum** è una tecnica di **rilevazione** degli **errori** che tratta i **d** bit di dati come sequenze di numeri interi con **k** bit ciascuno. Per calcolare il checksum, si **sommano** questi numeri interi da **k** bit e si utilizzano i bit del risultato come bit di **rilevazione** degli errori. Nel caso del checksum di Internet, i dati vengono trattati come interi di **16 bit** e **sommati**. Il **complemento a 1** di questa somma costituisce il checksum di Internet, che viene inserito nell'intestazione dei segmenti dati. Il **ricevente** può quindi **controllare** il **checksum** calcolando il complemento a 1 della somma dei dati ricevuti, compreso il checksum stesso, e **verificando** che **tutti i bit** del risultato **siano 1**. Se non è così, viene segnalato un **errore**.

In sintesi, il **checksum** è una tecnica di rilevazione degli errori basata su **somme** di numeri **interi**, utilizzata principalmente a livello di trasporto, come in TCP e UDP. Tuttavia, il **CRC** è preferito a livello di collegamento per la sua maggiore efficienza nella rilevazione degli errori, poiché è gestito da **hardware** dedicato nelle schede di **rete**.

(Vedi video esplicativo: <https://www.youtube.com/watch?v=AtVWnyDDaDI>)

1	1	1	1	1				
1	0	0	0	0	1	0	0	
0	0	1	0	0	1	0	0	
1	1	1	0	0	0	1	0	
1	0	0	1	1	0	0	1	
<hr/>								
0	0	1	0	0	0	1	1	
<hr/>								
0	0	1	0	0	1	0	1	
1	1	0	1	1	0	1	0	

(Mittente)

1	1	1	1	1	1	1		
1	0	0	0	0	1	0	0	
0	0	1	0	0	1	0	0	
1	1	1	0	0	0	1	0	
1	0	0	1	1	0	0	1	
1	1	0	1	1	0	1	0	
<hr/>								
1	1	1	1	1	1	0	1	
<hr/>								
1	1	1	1	1	1	1	1	

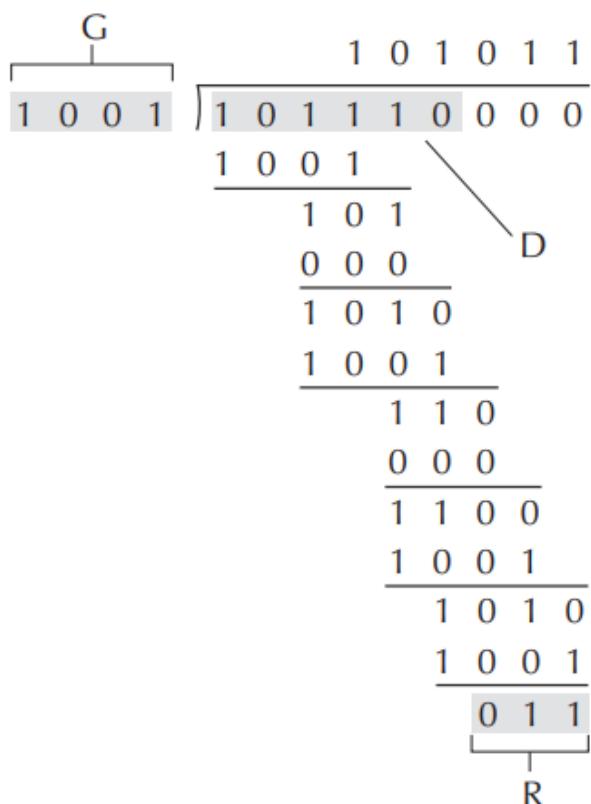
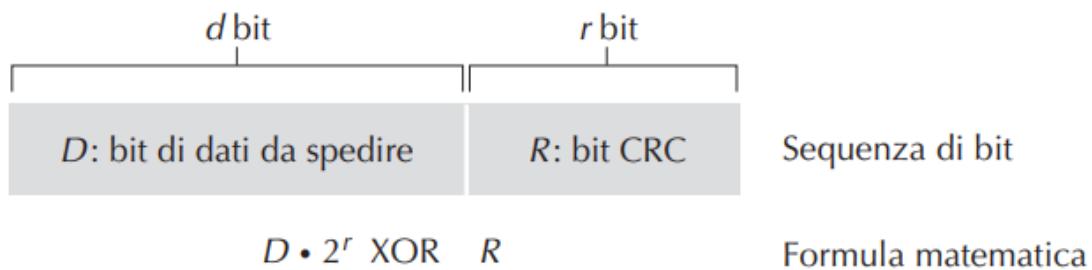
(Destinatario)



Controllo a ridondanza ciclica (CRC)

I codici **CRC**, noti anche come codici **polinomiali**. Questi codici si basano su un approccio in cui la **stringa** di bit da **trasmettere** è vista come un **polinomio**, con i coefficienti rappresentati dai bit della stringa. Le operazioni sui bit vengono interpretate come operazioni di aritmetica polinomiale.

Nel processo di calcolo dei CRC, si parte da un **blocco** di **dati (D)** costituito da **d bit** e si utilizza una stringa di bit chiamata **generatore (G)** di **lunghezza $r + 1$** , con il **bit** più **significativo** di G uguale a **1**. L'obiettivo è **unire** i dati D con una sequenza aggiuntiva di **r bit** (R) in modo che la **somma** dei due formi una **stringa** di **$d + r$ bit** che sia **esattamente divisibile** per **G** nell'aritmetica modulo 2. Se la **divisione** $(d + r) / G$ ha un **resto diverso da zero**, si sa che si è verificato un **errore** nei dati trasmessi.



Distanza di Hamming

La distanza di **Hamming** è una misura utilizzata in teoria dell'informazione e **correzione** degli errori per valutare la differenza tra due sequenze di simboli di uguale **lunghezza**. In altre parole, **indica** quanti **bit** o simboli devono essere **modificati** o **sostituiti** in una sequenza per trasformarla in un'altra sequenza di uguale **lunghezza**. La distanza di **Hamming** è definita come il **numero** di **posizioni** in cui le due **sequenze** differiscono. Ad esempio, se abbiamo due sequenze:

Sequenza 1: 1100101

Sequenza 2: 1001101

La **distanza di Hamming** tra queste due sequenze è **2**, perché ci sono due posizioni in cui i bit sono diversi.

La **distanza di Hamming** è una misura fondamentale in vari contesti, tra cui:

- **Correzione degli errori:** Nelle comunicazioni digitali, i codici a correzione di errore utilizzano la distanza di Hamming per determinare quante **modifiche** possono essere **apportate** a una **sequenza** senza **compromettere** la corretta **ricezione** dei dati.
- **Rilevazione degli errori:** La distanza di Hamming è **utilizzata** anche per **rilevare errori** nelle **sequenze** di dati **trasmesse**. Se la distanza tra la sequenza trasmessa e la sequenza ricevuta **supera** una certa **soglia**, si può concludere che ci sono stati **errori di trasmissione**.

Vocabolario

In questo contesto, il "**vocabolario**" rappresenta **l'insieme** delle **codeword valide** per una macchina.

Rilevazione e Correzione degli errori

Con una distanza di **Hamming** "d=3", è possibile **correggere** un singolo **errore**. Ciò significa che, se una sequenza **subisce** una modifica di un bit **durante la trasmissione**, è possibile **identificare** quale bit è stato **alterato** e **ripristinarlo** al suo stato originale.

- **Rilevazione di "e" errori:** Per **rilevare** "e" errori, è sufficiente un vocabolario con una distanza di Hamming "**d=e+1**". In questo caso, il sistema può rilevare la presenza di errori nella sequenza di dati, ma non può correggerli.
- **Correzione di "e" errori:** Per **correggere** "e" errori, è necessario un vocabolario con una distanza di Hamming "**d=2e+1**". Questo significa che il vocabolario deve essere progettato in modo da garantire una distanza sufficientemente ampia tra le sequenze in modo che possano essere identificati e corretti più errori.

Quanti bit di ridondanza 'r' servono?

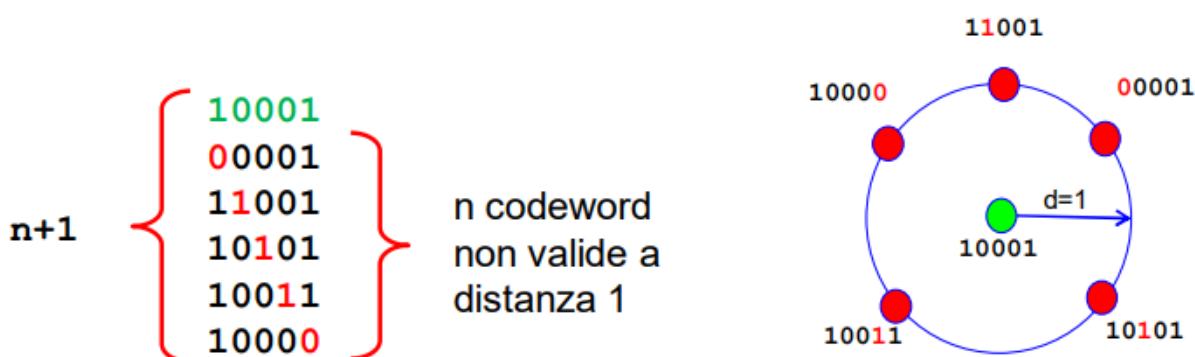
Per **determinare** quanti bit di **ridondanza** (r) sono necessari per **costruire** un **codice** in grado di **correggere** errori **singoli**, possiamo seguire i seguenti passaggi:

Definire la struttura del **vocabolario**: Iniziamo con **m** bit di **dati**, e vogliamo aggiungere **r** bit di controllo/**ridondanza**. Quindi, ogni **codeword** avrà una **lunghezza** di

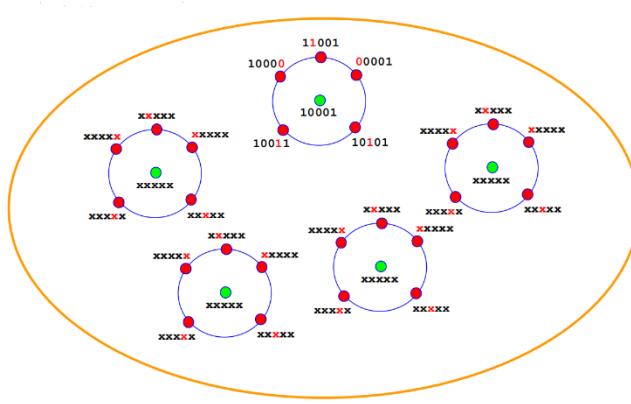
$$n = m + r \text{ bit}$$

Le **combinazioni** possibili con n bit sono 2^n , ma solo 2^m di queste sono **valide**.

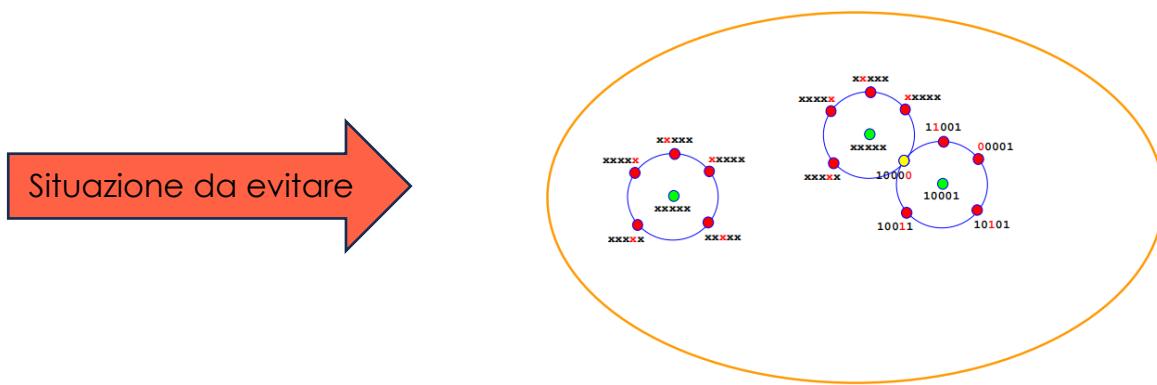
Prendiamo come esempio una **codeword valida**, ad esempio, "10001". Tutte le **codeword** a **distanza 1** da questa sono **errate**. Modificando un bit alla volta, possiamo creare altre **codeword**, ma **tutte** saranno **errate** e avranno la **stessa distanza** da quella originale. Possiamo **rappresentare** queste **codeword** errate su un **cerchio**, dato che sono a **distanza uguale** da quella **valida**.



Questo ci porta a un modello in cui ci sono **2^m cerchi** circondati da **n bit ciascuno**.



In altre parole, è **essenziale** evitare che ci siano **codeword** a **distanza 1** simultaneamente tra due altre **codeword**. Se ciò **accadesse**, il ragionamento che stiamo seguendo non **funzionerebbe**. Garantendo una **distanza minima** di **3**, ci assicuriamo che questa **situazione** non si **verifichi**. Per semplificare, possiamo immaginare questi **cerchietti** come entità separate poiché stiamo presupponendo che la distanza minima nel vocabolario sia di 3.



Se vogliamo correggere errori singoli ($d=3$), otteniamo **l'equazione**:

$$(n+1) * 2^m \leq 2^n$$

Semplificando l'equazione, ovvero sostituendo **$n=m+r$** , otteniamo:

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

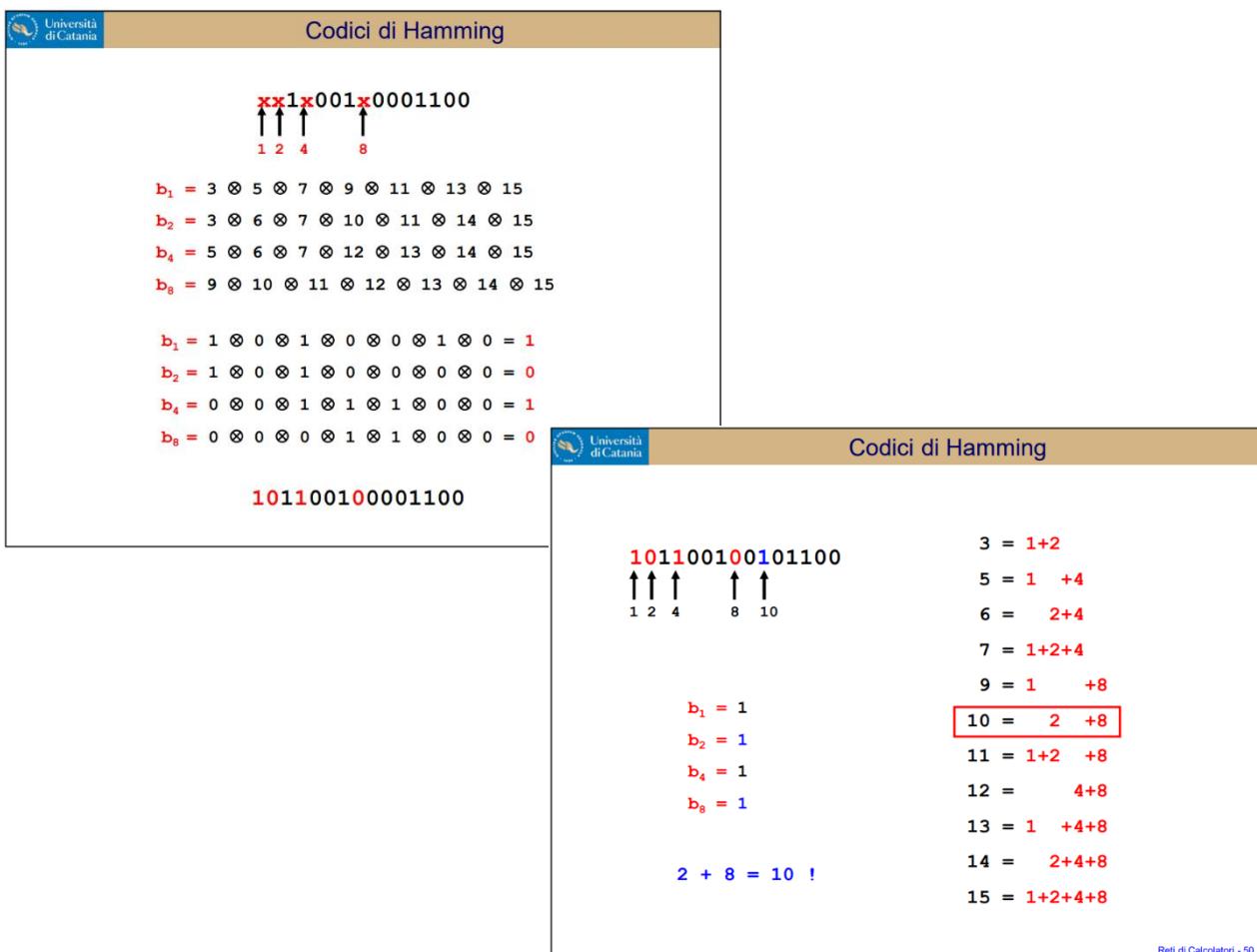
Quindi, per garantire la **correzione di errori singoli**, otteniamo l'espressione:

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

Codice di Hamming

Il **codice** di **Hamming** è un tipo di codice a **correzione** di **errori** che sfrutta l'aggiunta di **bit** di **controllo** alle sequenze binarie per **rilevare** e **correggere** errori. Ecco un riassunto dei concetti chiave: Per creare il codice di **Hamming**, si aggiungono bit di **controllo** alle sequenze binarie. Questi bit sono **posizionati** in modo **strategico**, in particolare in posizioni che sono **potenze** di **due**. I bit di controllo vengono calcolati in modo che riflettano il numero di bit a valore 1 in gruppi specifici all'interno della sequenza. Questi bit di controllo sono ottenuti utilizzando operazioni **XOR** sui bit di dati.

- **Rilevazione degli errori:** Durante la **trasmissione** o la memorizzazione dei dati, i **bit** di **controllo** vengono **inclusi insieme** ai bit di **dati**. Durante la **ricezione**, i bit di controllo **vengono** nuovamente **calcolati**. Se i bit di controllo calcolati **non corrispondono** a quelli ricevuti, ciò indica la presenza di **errori** nella sequenza.
- **Correzione degli errori:** Il codice di **Hamming** è in grado di **correggere** gli errori. Utilizzando le informazioni fornite dai bit di controllo, è possibile **identificare** la **posizione esatta** dei bit **errati** e correggerli.



Gestione di errori in sequenza

Il codice di **Hamming** può anche **gestire errori concentrati in sequenza**. Organizzando le codeword in modo da trasmetterle **colonna per colonna**, gli **errori** vengono **distribuiti** su diverse **codeword**, rendendo possibile la loro correzione. Tuttavia, la **trasmissione** in **colonna** richiede **bufferizzazione** prima della trasmissione e l'operazione inversa a destinazione, **causando** un certo **ritardo** di trasmissione.

The slide has a blue header bar with the university logo and the title "Codicci di Hamming". The main content area contains two blocks of binary code. The first block is a long string of binary digits:
**xx1x110xx0x011xx0x110xx1x000xx0x011xx1x100xx0x001xx0x11
0xx1x011xx1x110xx1x001xx0x011**
The second block is a list of 10 binary strings:
**xx1x110
xx0x011
xx0x110
xx1x000
xx0x011
xx1x100
xx0x001
xx0x110
xx1x011
xx1x110
xx1x001
xx0x011**

Reti di Calcolatori - 51

Protocolli del Data Link per il MAC

Esistono due **tipi** di **collegamenti** di rete: il collegamento **punto a punto**, che coinvolge un solo **trasmettente** e un unico **ricevente**, e il collegamento **broadcast**, che può avere **più nodi trasmittenti e riceventi** connessi allo stesso **canale condiviso**.

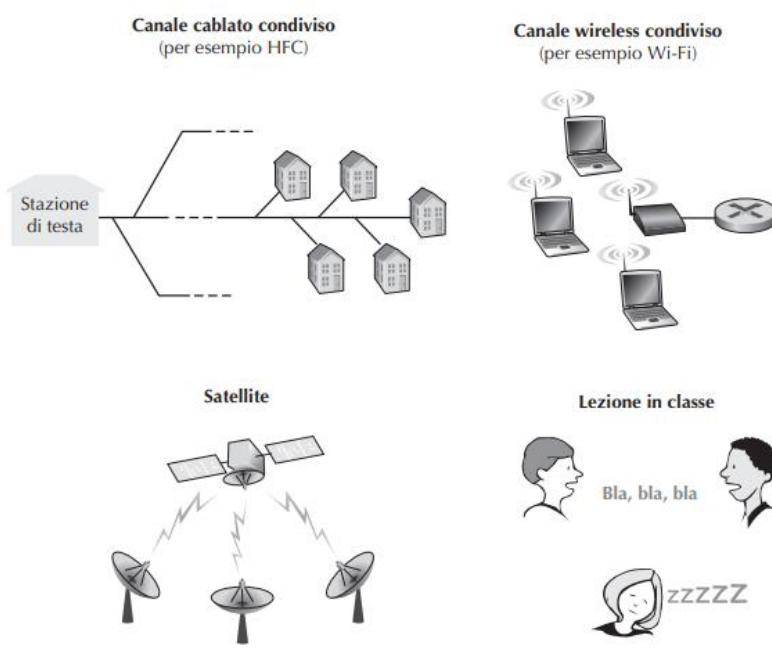
Si sottolinea **l'importanza** di **coordinare l'accesso** di più nodi in un **canale broadcast** condiviso, noto come **problema dell'accesso multiplo**. Questo problema è analogo a situazioni in cui molte persone condividono lo stesso canale di comunicazione, come una festa o una classe. I **protocolli** di accesso multiplo **regolamentano** le **trasmissioni** dei nodi su un canale broadcast condiviso.

Il **problema principale** dei **canali condivisi** è il rischio delle **collisioni**, dove **più nodi trasmettono contemporaneamente**, causando una **perdita** di frame e un **utilizzo inefficiente** del **canale**. Per ottimizzare l'uso del canale broadcast, sono necessari protocolli di accesso multiplo che coordinino le trasmissioni dei nodi.

Le **caratteristiche** ideali di un buon **protocollo** di accesso multiplo **sono**:

- **Garantire** un **throughput** di R bit al secondo per un singolo nodo
- **Suddividere equamente** la **larghezza di banda** tra M nodi attivi
- Essere **decentralizzato**
- **Semplice** da **implementare**.

Questi **criteri** sono **fondamentali** per garantire un **funzionamento efficiente** delle reti di calcolatori.

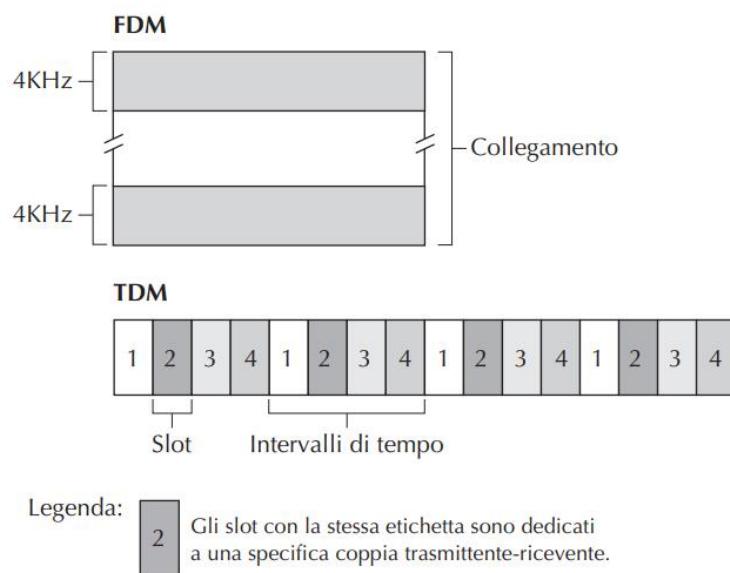


Protocolli a suddivisione del canale

Le tre **tecniche** di multiplexing per **suddividere la larghezza di banda** di un canale broadcast tra i nodi condivisi sono: Divisione di Tempo (**TDMA**), Divisione di Frequenza (**FDMA**) e Divisione di Codice (**CDMA**).

- **TDMA suddivide il tempo** in **intervalli di tempo**, ciascuno dei quali è diviso in **slot** temporali assegnati a ciascun nodo. **Ogni** volta che un **nodo** ha dati da inviare, **trasmette durante** il suo **slot** temporale. TDMA è equo ma obbliga i nodi a rispettare un limite di trasmissione anche quando altri nodi non stanno trasmettendo.
- **FDMA divide il canale** in diverse **frequenze**, ciascuna con una larghezza di banda di R/N, assegnando a ciascun nodo una di queste frequenze. Questo **crea N canali separati**. Anche FDMA è equo ma limita la larghezza di banda di ciascun nodo a R/N, anche quando altri nodi non trasmettono.
- **CDMA assegna** un **codice univoco** a ciascun **nodo**, che codifica i dati trasmessi. Questo permette a **nodi diversi** di **trasmettere simultaneamente** e ai **destinatari di ricevere i dati correttamente**, anche se ci sono interferenze da parte degli altri nodi. CDMA è **utilizzato** nelle **reti wireless** e nella telefonia cellulare per la sua capacità di evitare le interferenze.

Ognuna di queste tecniche ha vantaggi e svantaggi, e la scelta tra di esse dipende dalle specifiche esigenze della rete e dalle circostanze. TDMA e FDMA sono adatti per situazioni in cui la larghezza di banda può essere suddivisa in modo equo tra i nodi, mentre CDMA è preferito in ambienti con interferenze in cui è necessario consentire la trasmissione simultanea dei dati da parte di nodi diversi.



Esempi di TDM e FDM con quattro nodi.

Protocolli ad accesso casuale

I protocolli di accesso casuale permettono ai nodi di trasmettere a velocità massima (R bps) e, in caso di collisioni, ritrasmettono dopo un ritardo casuale. Questo ritardo casuale è scelto in modo indipendente dai nodi coinvolti, riducendo le collisioni. Esempi includono ALOHA e i protocolli CSMA come Ethernet.

ALOHA vs SLOTTED ALOHA

ALOHA e Slotted ALOHA sono due protocolli di accesso al canale utilizzati nelle reti di comunicazione, in particolare nelle reti a pacchetto come quelle utilizzate per la trasmissione dati. La principale differenza tra i due riguarda la sincronizzazione dei pacchetti trasmessi e la gestione dei conflitti.

- **ALOHA** è un protocollo di accesso al canale semplice e non sincronizzato. In ALOHA, i **nodi** della rete **possono trasmettere** dati in **qualsiasi momento** senza preoccuparsi della sincronizzazione con gli altri nodi. Quando un nodo desidera trasmettere un pacchetto, lo invia direttamente sul canale. **Se più nodi tentano di trasmettere** contemporaneamente, può verificarsi una **collisione**, e tutti i pacchetti coinvolti nella collisione vengono persi. **Dopo una collisione**, i **nodi ritentano** la **trasmissione** in modo casuale **dopo** un certo **periodo di attesa**.
- **Slotted ALOHA** è una **versione migliorata** di ALOHA che **introduce una sincronizzazione a slot di tempo**. Il tempo è suddiviso in slot di uguale durata, e i nodi possono trasmettere solo all'inizio di un determinato slot. Questo sistema di slot rende più efficiente l'accesso al canale e riduce il rischio di collisioni. **Se due nodi tentano di trasmettere** nello **stesso slot**, si verifica comunque una **collisione**, ma il sistema è più efficiente perché la probabilità di collisione è ridotta rispetto ad ALOHA non sincronizzato.

In sintesi, la principale differenza tra ALOHA e Slotted ALOHA è la presenza di una sincronizzazione a slot di tempo in Slotted ALOHA, che riduce le collisioni e rende l'accesso al canale più efficiente rispetto a ALOHA. Slotted ALOHA è quindi una versione migliorata di ALOHA che ha contribuito a ottimizzare le prestazioni delle reti di comunicazione a pacchetto.

CSMA: accesso multiplo con rilevamento della portante

Il comportamento "maleducato" di ALOHA può causare collisioni e inefficienze nella trasmissione dei dati. Esistono due regole per migliorare l'efficienza delle reti di comunicazione:

- **Ascoltare prima di parlare (Carrier Sensing):** I nodi devono ascoltare il canale prima di iniziare una trasmissione. Se il canale è già occupato da un altro nodo, devono attendere prima di iniziare la loro trasmissione. Questo processo è chiamato "rilevamento della portante."
- **Smettere di parlare se qualcun altro inizia (Collision Detection):** Se un nodo sta trasmettendo e un altro nodo inizia a trasmettere contemporaneamente, il primo nodo deve interrompere la sua trasmissione, attendere un periodo casuale e quindi riprovare a trasmettere. Questa pratica è chiamata "rilevamento della collisione."

Queste due **regole sono alla base** dei **protocolli CSMA** (Carrier Sense Multiple Access, accesso multiplo con rilevamento della portante) e **CSMA/CD** (CSMA with Collision Detection, CSMA con rilevamento della collisione), che sono stati **sviluppati** per **migliorare l'efficienza** delle reti di comunicazione. Il **ritardo di propagazione** dei segnali lungo il canale ha un **ruolo importante** nel determinare le **prestazioni del protocollo**, poiché può **influenzare** la **possibilità** di **collisioni**.

- **CSMA 1-persistent** prevede la trasmissione non appena scompare la portante.
- **CSMA p-persistent** la stazione, dopo aver rilevato il termine della precedente trasmissione, trasmette con probabilità p.
- **CSMA non-persistent** la stazione aspetta un tempo random prima di ricontrizzare il canale.

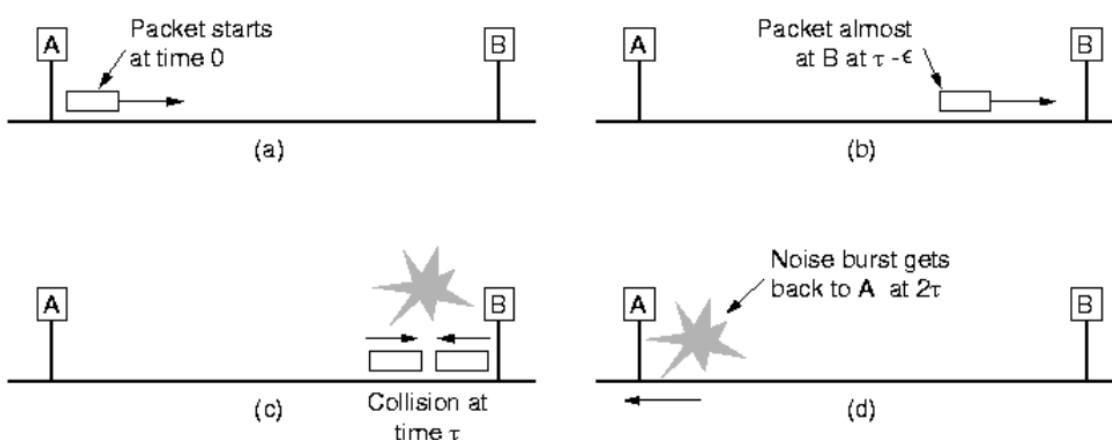
Accesso multiplo a rilevazione della portante con rilevamento delle collisioni (CSMA/CD)

CSMA/CD introduce la **rilevazione** di **collisione** che migliora le prestazioni del protocollo, poiché **evita** la **trasmissione inutile** di un frame danneggiato dall'interferenza con un altro frame.

Le **operazioni** di una scheda di rete collegata a un canale broadcast in un protocollo **CSMA/CD**:

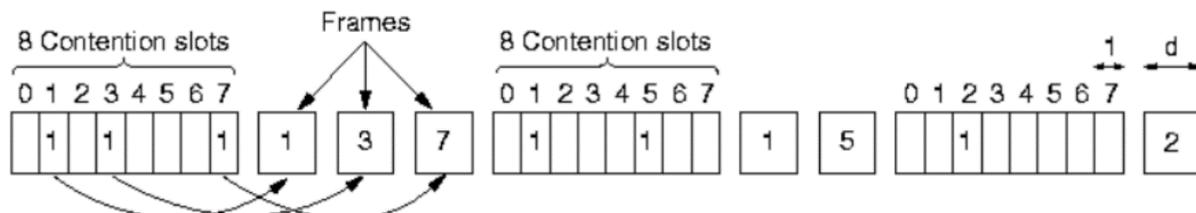
1. La **scheda di rete prepara** un **frame** a livello di collegamento quando ottiene un datagramma dal livello di rete e **lo mette** in un **buffer**.
2. La **scheda di rete verifica** se il **canale** è **libero prima** di **iniziare** la **trasmissione**. Se il canale è **occupato**, **attende finché** non rileva che il canale è **libero per iniziare la trasmissione**.
3. Durante la **trasmissione**, la scheda di rete **verifica** la **presenza** di **segnali provenienti da altre schede** di rete sul canale broadcast.
4. Se la scheda di rete **completa la trasmissione** del frame **senza rilevare segnali da altre schede**, il suo **lavoro** è **finito**. In caso contrario, se rileva segnali durante la trasmissione, **interrompe** immediatamente la **trasmissione del frame**.
5. Dopo l'interruzione della trasmissione, la **scheda di rete attende** un **periodo casuale** prima di tornare al **passo 2**.

L'importanza di attendere un **intervallo di tempo casuale dopo** una **collisione** anziché un periodo fisso, al fine di **evitare** che i **nodi continuino a collidere contemporaneamente**.



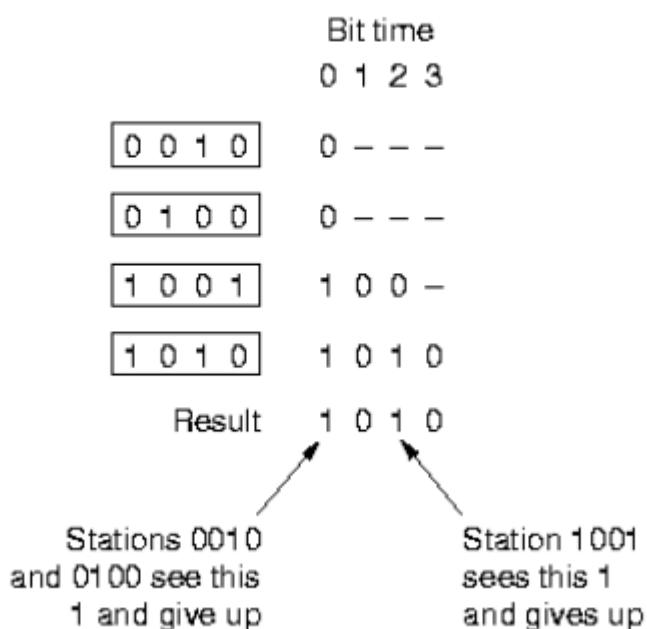
Protocolli senza collisioni

- **Bitmap** prevede che **ogni dispositivo abbia** un proprio "slot" nel quale può **trasmettere**. Quando un dispositivo vuole parlare, mette un segnale nel suo **slot**. Dopo un periodo di **attesa**, i dispositivi sanno chi ha vinto e trasmettono uno alla volta. Tuttavia, ha problemi: può essere **lento con molte stazioni** e richiede di sapere in anticipo quante stazioni ci sono.



- **CSMA/BA CanBus risolve** questi **problemi**. Invece di **slot**, i dispositivi **inviano** "1" o restano in silenzio ("0"). Se due dispositivi dicono "1" contemporaneamente, il risultato è "1", altrimenti vince il "0". **Ogni dispositivo** ha un **numero** e mette il suo bit più **significativo** sul **canale**. Solo uno con un bit diverso trasmette alla volta. Questo metodo usa meglio la **banda** e non richiede conoscere il numero esatto di dispositivi. **Quelli con numeri più alti hanno la priorità.**

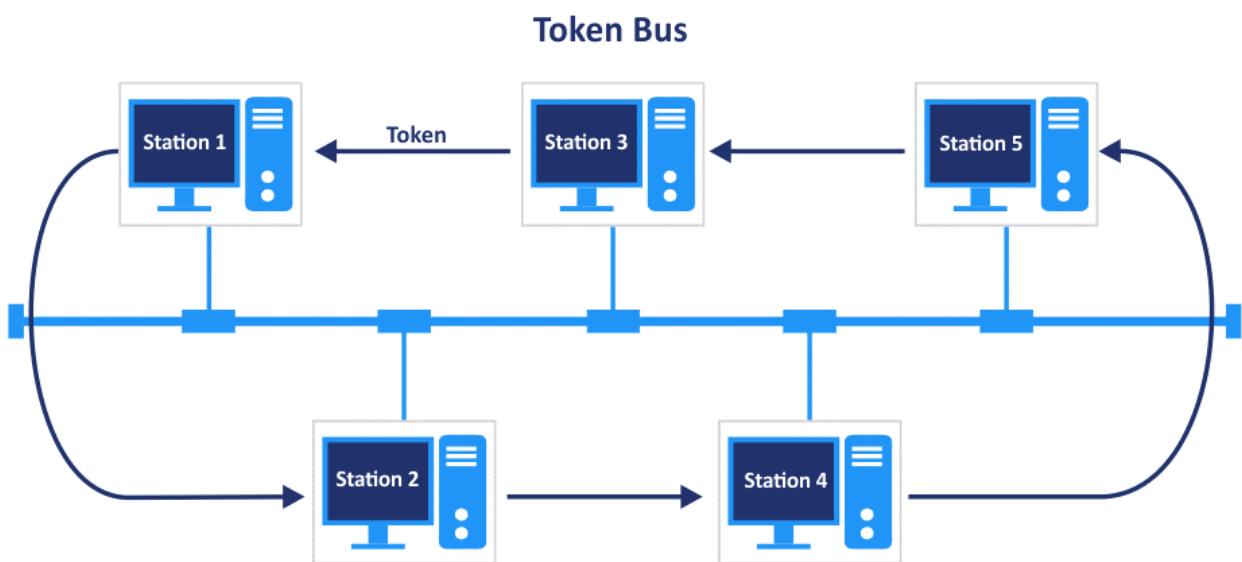
Conteggio binario a ritroso (CSMA/BA CanBus)



Protocolli a turno

- **Protocollo di polling:** In questo **protocollo**, un nodo designato come "**principale**" interroga "a **turno**" gli altri nodi. Il nodo principale invia un **messaggio** a un **nodo alla volta**, consentendogli di trasmettere un numero massimo di frame. Dopo la trasmissione, il nodo principale passa al nodo **successivo**. Il protocollo **elimina collisioni** e slot vuoti ma introduce **ritardi** nel polling e può causare **inattività** del canale se il nodo principale si **guasta**.
- **Protocollo token-passing:** In questo caso, non c'è un nodo principale. Invece, circola un **frame** di controllo chiamato "**token**" tra i nodi in un **ordine prestabilito**. Ogni nodo riceve il token, può trasmettere un numero massimo di frame e poi inoltra il token al nodo **successivo**. Questo protocollo è altamente **efficiente** ma può avere **problemi** se un nodo si **guasta** o non riesce a inoltrare il **token**.

Entrambi i protocolli a turno cercano di migliorare l'efficienza rispetto ai protocolli ad accesso casuale come **ALOHA** e **CSMA**, ma ognuno ha le proprie limitazioni.



IEEE 802.1 / 802.2 / 802.3

IEEE 802 è una famiglia di **standard** che definisce le **regole** per le reti locali (**LAN**) e altre tecnologie di comunicazione. Tra questi, ci sono gli standard **802.1**, **802.2** ed **802.3**, che hanno scopi diversi:

- **IEEE 802.1:** Questo standard definisce **regole** e **protocolli** per la **gestione della rete**. In altre parole, si occupa di come le diverse apparecchiature di una rete **comunicano** tra **loro** e di come vengono **gestite**. È come un "**gestore del traffico**" per garantire che i dati vadano dove devono andare in modo **efficiente**.
- **IEEE 802.2:** Questo standard è conosciuto anche come **LLC** (Logical Link Control), ed è uno strato di **controllo logico** che si trova nel livello di collegamento dati. Si occupa di **controllare** come i dati vengono **invia**ti e **ricevuti** tra i dispositivi su una rete LAN. Funziona come una sorta di "**guardiano**" che verifica che i dati siano **invia**ti nel formato **corretto** e che raggiungano il **destinatario** giusto.
- **IEEE 802.3:** Questo è uno degli standard più famosi ed è comunemente noto come **Ethernet**. Definisce come i dati vengono **fisicamente trasmessi** su un cavo Ethernet, inclusi i dettagli sul tipo di **cavi** e i segnali elettrici utilizzati. In sostanza, l'802.3 stabilisce come i **computer** e gli altri dispositivi si **collegano** e **comunicano** su una rete **cablata**. Utilizza **CSMA/CD**.

In breve, l'**802.1** **gestisce** la gestione della **rete**, l'**802.2** **controlla** il **flusso di dati** e l'**802.3** si occupa della **trasmissione fisica** dei **dati** su una rete **Ethernet**. Questi standard lavorano insieme per creare una rete LAN affidabile e ben gestita.

Cablaggi

I cablaggi **Ethernet** sono importanti componenti delle reti **cablate** che consentono la **trasmissione** di dati tra dispositivi. Ci sono **diverse varianti** di cablaggio Ethernet, o **standard**, ciascuna progettata per soddisfare requisiti specifici di **velocità**, **distanza** e **ambiente** di rete. Di seguito, spiegherò le differenze tra alcune varianti comuni di cablaggio Ethernet: **10BASE5**, **10BASE2**, **10BASE-T** e **10BASE-F**.

Ecco un riepilogo delle **differenze** tra le varianti di **cablaggio Ethernet**:

- **10BASE5:**
 - Cavo **coassiale** spesso.
 - Distanza massima di **500** metri.
 - **Obsoleta**.
- **10BASE2:**
 - Cavo **coassiale** sottile.
 - Distanza massima di **185-200** metri.
- **10BASE-T:**
 - Doppino **intrecciato in rame** (twisted pair).
 - Distanza massima di **100** metri.
 - Utilizza connettori **RJ-45**.
- **10BASE-F:**
 - Cavi a **fibra** ottica.
 - Supporta distanze maggiori fino a **2000m**.
 - Utilizza **connettitori** specifici per la **fibra ottica**.

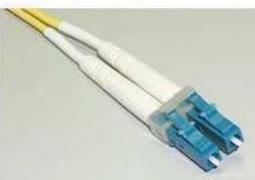
 Cablaggi

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









Reti di Calcolatori - 76

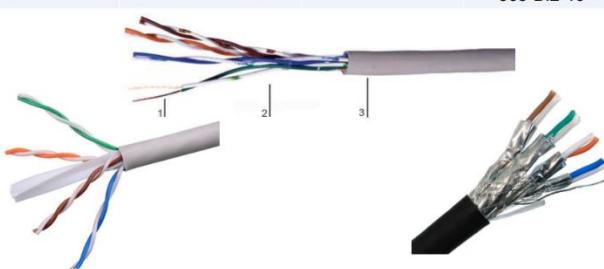
Categorie dei cavi Ethernet

I cavi Ethernet **CAT5**, **CAT5e**, **CAT6** e **CAT6a** sono tutti utilizzati per connettere dispositivi in reti **cablate**. Le principali differenze tra di loro includono le capacità di **trasmissione**, la **velocità** di trasferimento dei dati, la **qualità** della trasmissione e la **distanza** massima supportata:

- **CAT5 (Categoria 5):**
 - **Velocità:** Fino a **100 Mbps**.
 - **Qualità:** Non ideale per applicazioni Gigabit Ethernet.
 - **Applicazioni tipiche:** Comunemente utilizzato in reti locali a 100 Mbps.
- **CAT5e (Categoria 5e):**
 - **Velocità:** Fino a **1 Gbps** (Gigabit Ethernet).
 - **Qualità:** Migliorata rispetto a CAT5 per supportare reti Gigabit Ethernet e ridurre le interferenze.
 - **Applicazioni tipiche:** Utilizzato in reti Gigabit Ethernet e VoIP.
- **CAT6 (Categoria 6):**
 - **Velocità:** Fino a **10 Gbps** su **brevi** distanze (spesso 55 metri o meno)
 - **Qualità:** Migliorata ulteriormente per ridurre al minimo le interferenze e supportare reti a 10 Gbps.
 - **Applicazioni tipiche:** Utilizzato in reti aziendali e data center per supportare velocità di trasferimento elevate.
- **CAT6a (Categoria 6a):**
 - **Velocità:** Fino a **10 Gbps** su una distanza massima di **100 metri**.
 - **Qualità:** La migliore disponibile, con una maggiore schermatura per ridurre le interferenze.
 - **Applicazioni tipiche:** Ideale per reti aziendali e data center che richiedono una prestazione elevata e una maggiore riduzione delle interferenze.

CAT5e è una **scelta** solida per la **maggior parte** delle reti **Gigabit Ethernet**, mentre **CAT6** e **CAT6a** sono ideali per reti con requisiti di **prestazione** più **elevati** e maggiore **riduzione** delle **interferenze**.

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

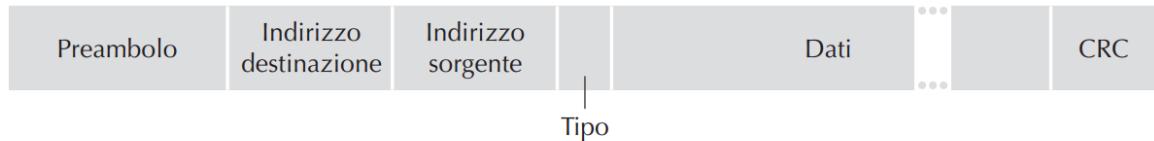


Reti di Calcolatori - 80

Ethernet

L'Ethernet è diventata la tecnologia principale per le reti **cablate** grazie alla sua semplicità, efficacia dei **costi** e alla continua evoluzione delle prestazioni, inclusa l'introduzione degli **switch** Ethernet che hanno eliminato le **collisioni** e migliorato l'**efficienza** delle reti.

Struttura del frame Ethernet



Il **frame Ethernet** è composto da **sei campi principali**:

- **Preamble (8 byte):** I primi sette byte contengono sequenze specifiche di bit utilizzate per la **sincronizzazione**. L'ottavo byte indica **informazioni importanti** da trasmettere.
- **Indirizzo di Destinazione (6 byte):** Contiene l'indirizzo **MAC** della scheda di rete di **destinazione**.
- **Indirizzo Sorgente (6 byte):** Contiene l'indirizzo **MAC** della scheda di rete che **ha trasmesso il pacchetto**.
- **Tipo (2 byte):** Indica il **tipo** di **protocollo** di rete utilizzato nel campo dati del frame, come **0x0800** per **IPv4**, **0x86DD** per **IPv6**. Questo campo aiuta il dispositivo di **ricezione** a sapere come **interpretare** il contenuto del campo **dati**.
- **Campo Dati (da 46 a 1500 byte):** Contiene il **payload** del frame, che può essere un datagramma IP o altri tipi di pacchetti di rete. Se il payload è più piccolo di 46 byte, il campo dati viene "**riempito**" di **0** fino a raggiungere questa dimensione.
- **Controllo a Ridondanza Ciclica (CRC) (4 byte):** Questo campo permette al dispositivo di **ricezione** di **rilevare errori** nei dati del frame. Viene utilizzato per verificare l'integrità dei dati trasmessi.

Codifica Manchester

La **codifica Manchester** è una tecnica di **codifica** dei dati **utilizzata** nella **trasmissione** digitale per garantire una **sincronizzazione** affidabile tra il **trasmettitore** e il **ricevitore**. Funziona **invertendo** il valore di un segnale durante la **metà** di ciascun intervallo di **bit**, consentendo al ricevitore di rilevare facilmente l'inizio e la fine di ciascun bit. Questa tecnica è chiamata così perché è stata sviluppata presso l'Università di Manchester nei primi anni '40.

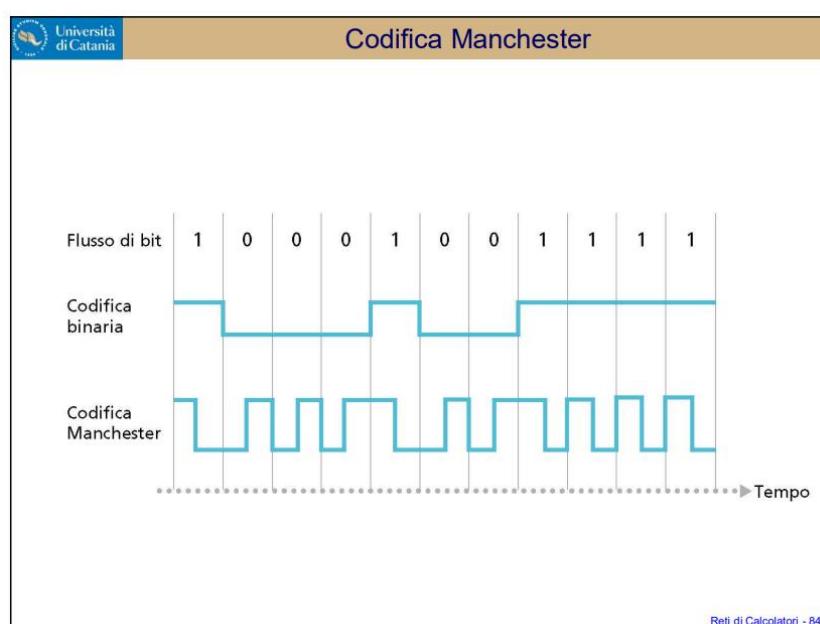
Ecco come **funziona** la codifica **Manchester**:

- **Bit "0":** Inizia con il livello **basso** (0) durante la **prima metà** del periodo di bit e passa al livello **alto** (1) nella **seconda metà**.
- **Bit "1":** Inizia con il livello **alto** (1) durante la **prima metà** del periodo di bit e passa al livello **basso** (0) nella **seconda metà**.

In altre parole, ogni bit è diviso in due **intervalli** di **tempo uguali**, e la transizione tra i livelli di segnale avviene esattamente a metà del periodo di bit.

La **codifica Manchester** è **utilizzata** in vari **contesti**, tra cui:

- **Ethernet:** In molte varianti di Ethernet, come **10BASE-T** e **100BASE-TX**, la codifica Manchester viene utilizzata per garantire la **sincronizzazione** e la **rilevazione** degli **errori** durante la trasmissione dei dati su cavi di rame.
- **Fibre ottiche:** Nelle comunicazioni su **fibre ottiche**, la codifica Manchester è spesso utilizzata per la **trasmissione affidabile** dei dati a lunghe distanze e ad **alte velocità**.



Fast Ethernet e Gigabit Ethernet

Fast Ethernet e **Gigabit Ethernet** sono standard di Ethernet che supportano diverse velocità di trasferimento dati:

Fast Ethernet (100BASE-TX)

Fast Ethernet è stato un aggiornamento rispetto all'Ethernet originale a **10 Mbps** e ha fornito una velocità di **100 Mbps**. È stato utilizzato ampiamente nelle reti locali. Ecco alcune caratteristiche principali:

- **Velocità di trasferimento dati:** **100 Mbps** (Megabit per secondo).
- **Tipo di cavo:** Doppino **intrecciato** (twisted pair), spesso utilizza CAT5 o CAT5e.
- **Modalità:** Full-duplex o half-duplex.
- **Distanza massima:** Fino a **100 metri**.

Alcune **tipologie** di cavi Fast ethernet sono:

- **100BASE-T4:** Questo standard è progettato per operare su cavi **CAT3**. Fornisce una velocità di **100 Mbps** attraverso quattro coppie di cavi.
- **100BASE-TX:** Questo standard utilizza la codifica **4B5B** ed è progettato per operare su cavi **CAT5** o **superiori**. Fornisce una velocità di **100 Mbps** attraverso due coppie di cavi in modalità **full-duplex**.
- **100BASE-FX:** Questo standard utilizza la **fibra ottica** per trasmettere dati a una velocità di **100 Mbps** su distanze più lunghe rispetto alle varianti basate su rame. È spesso utilizzato per connessioni su **lunghe distanze** all'interno di edifici.

Name	Cable	Max. segment	Advantages
100Base-T4	Twisted pair	100 m	Uses category 3 UTP
100Base-TX	Twisted pair	100 m	Full duplex at 100 Mbps
100Base-FX	Fiber optics	2000 m	Full duplex at 100 Mbps; long runs

Gigabit Ethernet (1000BASE-T)

Gigabit Ethernet è stato un importante passo avanti rispetto a Fast Ethernet, fornendo una velocità di **1 Gbps**. È ampiamente utilizzato in reti aziendali e data center.

Ecco alcune **caratteristiche principali**:

- **Velocità di trasferimento dati:** **1 Gbps** (Gigabit per secondo).
- **Tipo di cavo:** Doppino **intrecciato** (twisted pair), utilizza CAT5e o CAT6.
- **Modalità:** **Full-duplex**.
- **Distanza massima:** Fino a **100 metri**.

Alcune **tipologie** di cavi Fast ethernet sono:

- **1000BASE-SX e 1000BASE-LX:** Entrambi sono standard Gigabit Ethernet che utilizzano la **fibra ottica**. **SX** è progettato per **distanze** più **brevi** all'interno di un edificio, mentre **LX** supporta **distanze** più **lunghe**.
- **1000BASE-CX:** Questo standard Gigabit Ethernet utilizza cavi a coppie **torsate corte** per connessioni point-to-point su brevi distanze, come in un rack di server.
- **1000BASE-T:** Questo standard Gigabit Ethernet utilizza quattro coppie di cavi **CAT5e** o superiori per una velocità di **1 Gbps** in **modalità full-duplex** su una distanza massima di **100 metri**.

Name	Cable	Max. segment	Advantages
1000Base-SX	Fiber optics	550 m	Multimode fiber (50, 62.5 microns)
1000Base-LX	Fiber optics	5000 m	Single (10 μ) or multimode (50, 62.5 μ)
1000Base-CX	2 Pairs of STP	25 m	Shielded twisted pair
1000Base-T	4 Pairs of UTP	100 m	Standard category 5 UTP

Come raggiungere una velocità di 1 Gbps su cavi CAT5

1. **Rimuovere** la codifica **4B5B** (da 100 a 125 Mbps).
2. Utilizzare le **4 copie** di **cavi** simultaneamente (da 125 a 500 Mbps).
3. **Trasmissione full-duplex** (500 Mbps full-duplex).
4. Utilizzare **5 livelli per baud** (possibile variazione rispetto allo stato precedente) invece di **3** (MLT-3) per raggiungere **2 Gbps full-duplex**.
5. Utilizzare una correzione di errore avanzata (**FEC**) per recuperare ulteriori 6 dB di segnale per garantire la qualità della trasmissione a questa alta velocità.

Riepilogo Ethernet, Fast Ethernet e Gigabit Ethernet

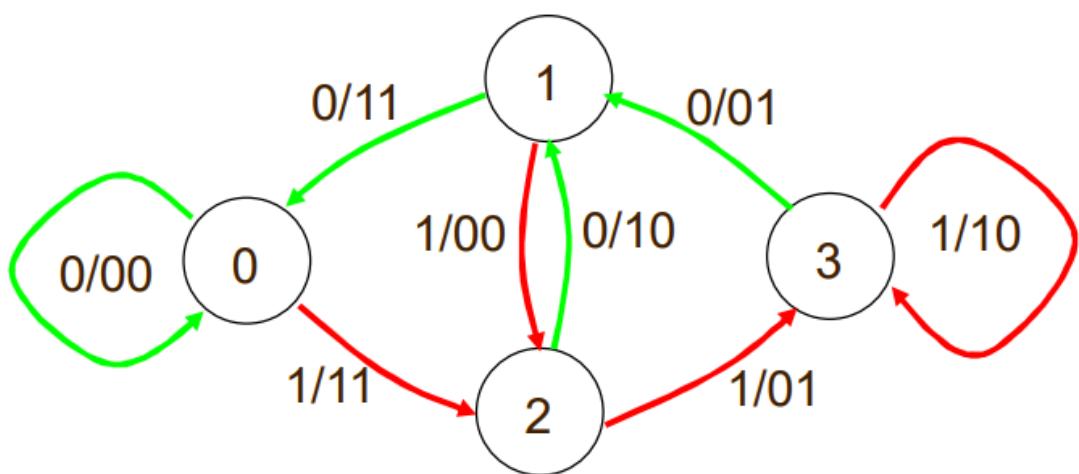
Riepilogo 10 – 100 – 1000

Tecnologia	Massima lunghezza del link	Codifica	Topologia del mezzo		Bit rate (bps)
10Base5	500 m	Manchester	bus	50-ohm coax	10 M
10Base2	185 m	Manchester	bus	50-ohm coax	10 M
10BaseT	100 m	Manchester	star	2 pair UTP cat. 3,4,5	10
100BaseFL 100BaseT2	2000 m 100 m	Manchester PAM 5x5	star star	Multi-mode fiber* 2 pairs UTP cat. 3,4,5	10 M 100 M
100BaseT4	100 m	8B/6T	star	4 pairs UTP cat. 3,4,5	100 M
100BaseTX	100 m	4B/5B with MLT-3	star	2 pairs UTP cat. 5	100 M
100BaseFX 1000BaseT	412/2000 m 100 m	4B/5B with NRZI PAM 5x5	star star	Multi-mode fiber* 4 pairs UTP Cat 5	100 M 1000 M
1000BaseSX	275 m	8B/10B	star	Multi-mode fiber†	1000 M
1000BaseLX	316/550 m	8B/10B	star	Multi-mode Fiber‡	1000 M
1000BaseCX	25 m	8B/10B	star	Twinax	1000 M

Reti di Calcolatori - 105

Schemi di Trellis e decodifica di Viterbi

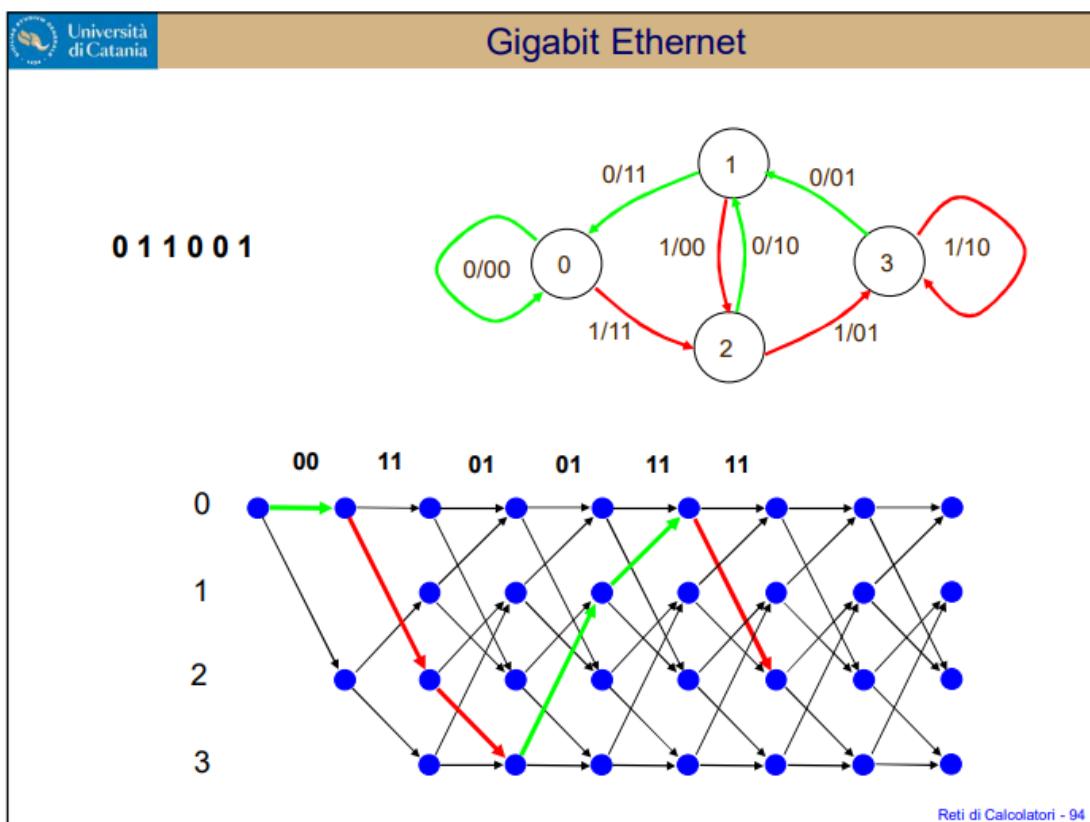
Gli schemi di **Trellis** sono rappresentazioni grafiche dei **codici convoluzionali**, mentre l'algoritmo di **decodifica di Viterbi** utilizza questi schemi per trovare la **sequenza di bit più probabile** in una trasmissione attraverso un canale rumoroso. Serve per la **correzione degli errori** nelle comunicazioni **wireless** e **satellitari**. Utilizzata in **Gigabit Ethernet**.



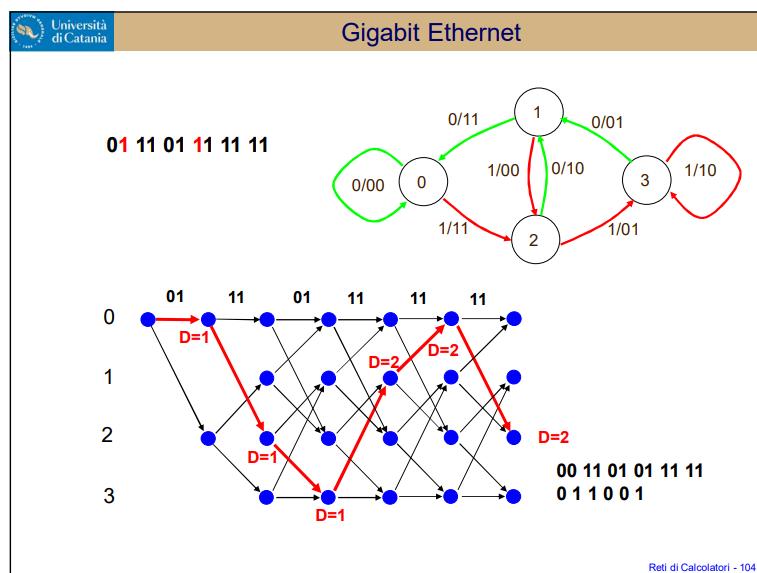
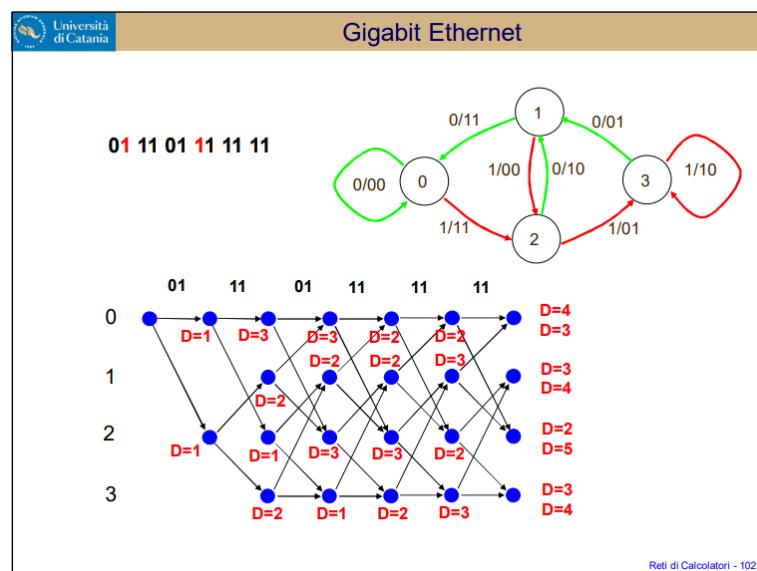
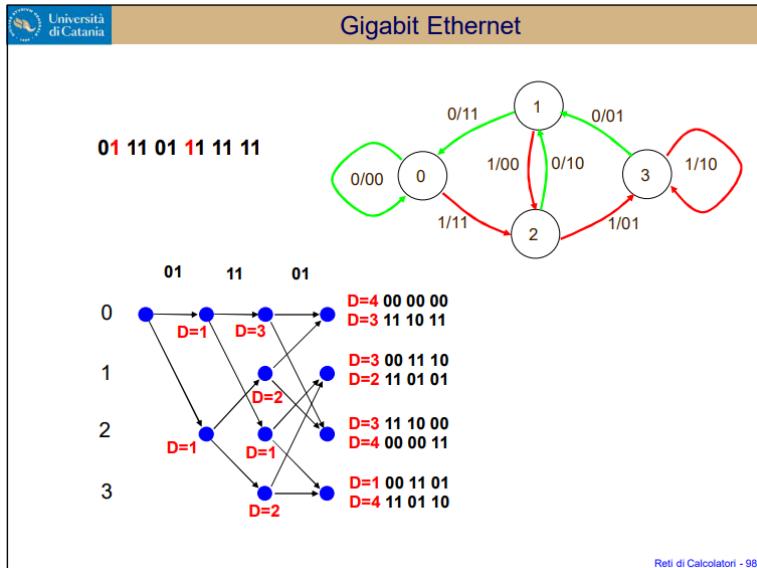
Come funziona la decodifica di Viterbi?

- **Inizializzazione:** L'algoritmo inizia dallo stato iniziale di un **diagramma** a grafo noto come "**Trellis**". Ogni stato rappresenta una possibile **sequenza** di **bit** trasmessi. Inizia con il calcolo di una "**metrica di distanza**" per ciascuno dei simboli di input possibili. La **metrica** di distanza rappresenta quanto il **simbolo** di input attuale **differisce** da quello **previsto** per ogni **stato**.
- **Passo iterativo:** L'algoritmo si sposta in **avanti** attraverso il **Trellis**, stato per stato, **iterativamente**. Ad ogni passo, **calcola** la metrica di **distanza** per **ciascuno** dei possibili **simboli** di input successivi e seleziona il **percorso** con la **metrica di distanza minima**. Questo percorso rappresenta la sequenza di bit più **probabile** fino a quel momento.

L'**obiettivo** dell'algoritmo di **Viterbi** è quello di identificare la **sequenza** di **bit** più **probabile** tra tutte le possibili **sequenze** che potrebbero essere state **trasmesse** attraverso il canale con **errori**. Questo è possibile grazie al calcolo delle metriche di **distanza** e alla selezione del percorso con la **distanza minima** ad ogni passo.



Reti di Calcolatori - 94

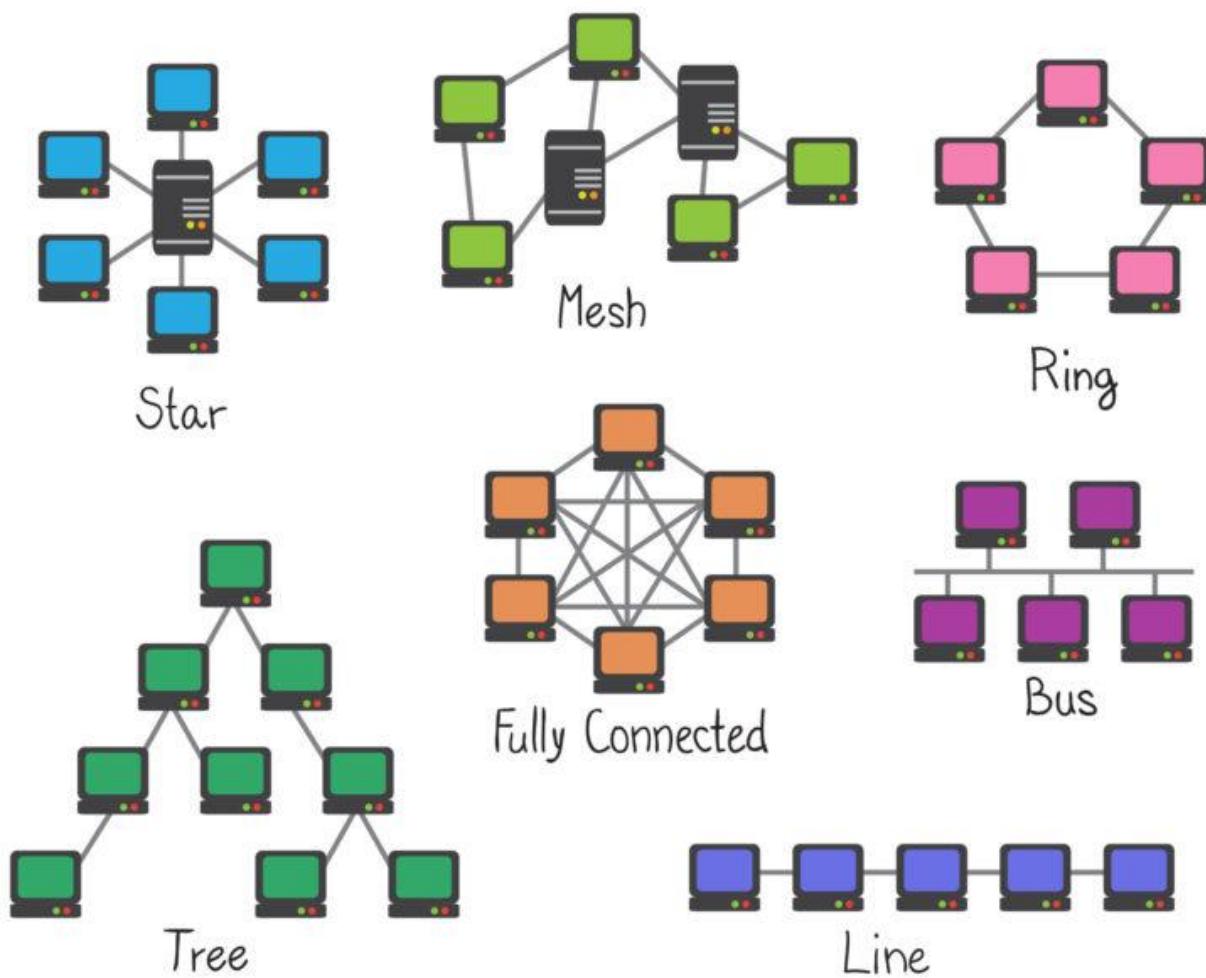


Schemi di interconnessione in Ethernet

Gli schemi di interconnessione in Ethernet si riferiscono alla topologia fisica utilizzata per collegare dispositivi in una rete Ethernet. Le topologie comuni includono:

- **Stella**: Tutti i dispositivi sono collegati a uno switch o hub centrale. È comune e efficiente.
- **Bus**: I dispositivi sono collegati a un cavo Ethernet condiviso (obsoleto).
- **Anello**: I dispositivi sono collegati in un anello fisico chiuso (meno comune).
- **Maglia** (Mesh): Ogni dispositivo è collegato a diversi altri, offrendo ridondanza.
- **Ibrida**: Una combinazione di topologie diverse nella stessa rete.

La scelta della topologia dipende dalle esigenze specifiche della rete, tra cui dimensioni, ridondanza, sicurezza e budget.



Repeater, Hub, Bridge, Switch

I **repeater**, **hub**, **bridge** e **switch** sono tutti dispositivi utilizzati nelle reti di computer per **trasmettere** dati, ma hanno **funzioni** e **caratteristiche diverse**. Ecco le principali differenze tra di loro:

Repeater (Ripetitore)

- **Funzione principale:** Il **repeater** è un dispositivo **passivo** che **amplifica** il **segnale** e lo **ritrasmette** sulla stessa rete senza effettuare alcuna analisi o elaborazione dei dati.
- **Utilizzo:** È utilizzato per **estendere la portata** di una **rete**, compensando la perdita di segnale lungo cavi o tratti di cavo molto lunghi.
- **Livello di funzionamento:** Lavora a livello **fisico** (livello 1) del modello OSI e opera semplicemente in base alla potenza del segnale.

Hub (Concentratore)

- **Funzione principale:** Ha lo stesso **funzionamento** del **repeater**. La differenza è che l'hub ha una struttura tipicamente a **stella**. Ha **più di due porte**.
- **Utilizzo:** È stato **utilizzato in passato** per creare reti locali (LAN), ma è ora considerato **obsoleto** a favore di **switch** più intelligenti.
- **Livello di funzionamento:** Opera a livello **fisico** (livello 1) e di **collegamento dati** (livello 2) del modello OSI.

Bridge (Ponte)

- **Funzione principale:** Il bridge è un dispositivo che **collega** due **segmenti** di rete locali (**LAN**) e può **analizzare il traffico** di rete per inoltrare i dati solo al segmento di destinazione corretto. Aiuta a dividere **fisicamente** una rete in segmenti **logici**. Serve quindi per **collegare LAN differenti**.
- **Utilizzo:** È stato utilizzato per **suddividere** le **reti** locali in segmenti per migliorare le **prestazioni** e il controllo del traffico.
- **Livello di funzionamento:** Opera principalmente a livello di **collegamento dati** (livello 2) del modello OSI.

In breve, il **repeater amplifica** il segnale, l'**hub ritrasmette** i dati a tutti i dispositivi, il **bridge connette segmenti** di rete e **filtra il traffico**, mentre lo **switch analizza** gli indirizzi **MAC** e instrada i dati solo al **destinatario corretto**, rendendo quest'ultimo la scelta più **avanzata ed efficiente** nelle reti moderne.

Switch (Commutatore)

- **Funzione principale:** Lo **switch** è un dispositivo attivo che analizza **l'indirizzo MAC** (Media Access Control) di ogni **frame** di dati e **inoltra** il frame solo alla porta del destinatario **corretto**. Questo consente una **trasmissione** più **efficiente** e **riduce** il **traffico** di rete non necessario.
- **Utilizzo:** È il dispositivo più comune nelle moderne reti locali (LAN) ed è **essenziale** per gestire il **traffico** in modo intelligente.
- **Livello di funzionamento:** Opera a livello di **collegamento dati** (livello 2) del modello OSI ed è in grado di funzionare anche a **livello di rete** (livello 3) quando necessario.

Switch: Inoltro e filtraggio

Lo **switch** a livello di **collegamento** svolge il ruolo di **ricevere** i frame in **ingresso** e **inoltrarli** ai nodi della rete. Questo processo avviene in modo **trasparente** per i nodi, che inviano direttamente i frame ai destinatari senza passare per lo switch. La **trasparenza** dello **switch** sta nel fatto che i nodi **non** sono **consapevoli** dell'**intermediazione** dello switch nella **trasmissione** dei loro **dati**.

Le operazioni principali dello switch sono il **filtraggio** e **l'inoltro**. Il filtraggio decide se un frame deve essere **inoltrato** o **scartato**, mentre l'inoltro determina **l'interfaccia** attraverso cui il frame deve essere **invia**to.

Queste operazioni sono gestite da una tabella di commutazione che contiene informazioni sugli indirizzi **MAC** dei nodi, le relative **interfacce** di uscita e i **tempi** di **aggiornamento**.

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

Quando un frame arriva, il suo indirizzo MAC di destinazione viene **confrontato** con la **tabella** e possono verificarsi tre **scenari**:

- **Broadcast:** Se uno switch riceve un frame e **non sa** dove si trova il **destinatario**, lo invia a tutte le altre porte (**interfacce**) tranne quella da cui è arrivato. È come urlare in una stanza piena di persone sperando che il destinatario lo senta.
- **Scarto:** Se uno **switch** riceve un **frame** e sa che il destinatario si trova sulla stessa "porta" da cui è arrivato il **frame**, allora lo **scarta**. È come ricevere una **lettera** che è stata inviata a te **stesso**; non c'è bisogno di consegnarla a te stesso.
- **Inoltro:** Se uno switch riceve un **frame** e sa che il destinatario si **trova** su una **diversa porta**, lo **inoltra solo** a quella **porta**. È come ricevere una lettera destinata a qualcun altro e **consegnarla** solo a quella **persona** anziché a tutti gli altri.

In breve, lo switch decide se **inviare** il frame a **tutte** le porte (broadcast), **scartarlo** (se il destinatario è sulla stessa porta da cui è arrivato), o **inoltrarlo** solo alla **porta giusta** (se il destinatario è su una diversa interfaccia).

Autoapprendimento

Gli **switch** hanno la capacità di **autoapprendere** autonomamente le informazioni necessarie per instradare i **frame** senza richiedere l'intervento dell'amministratore di rete. Questa capacità funziona nel seguente modo:

1. La **tabella** di commutazione dello switch è inizialmente **vuota**.
2. Ogni volta che lo switch riceve un frame, esso memorizza nella tabella:
 - a. L'indirizzo **MAC** del mittente del frame.
 - b. L'**interfaccia** da cui è arrivato il frame.
 - c. Il **momento** in cui il frame è stato ricevuto.
3. Dopo un certo periodo di **tempo**, chiamato "**aging time**" (tempo di invecchiamento), se lo switch non riceve più frame da un particolare indirizzo **sorgente**, lo **rimuove** automaticamente dalla sua **tabella**. Questo assicura che le informazioni siano sempre aggiornate.

Questo processo di autoapprendimento consente agli switch di costruire le proprie tabelle di **routing** in modo **automatico**, senza richiedere **configurazioni manuali**. È una caratteristica "**plug-and-play**", il che significa che basta collegare i segmenti di rete alle interfacce dello switch senza dover configurare manualmente le tabelle, **semplificando** notevolmente **l'installazione** e la **gestione** delle **reti**.

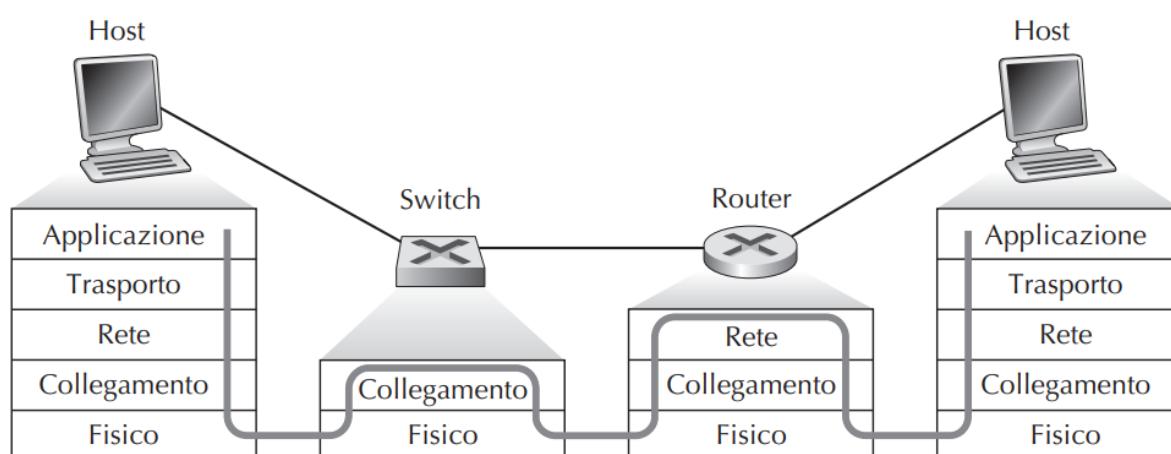
Proprietà della commutazione a livello di collegamento

Le **proprietà** della **commutazione** a livello di **collegamento**, rispetto ai collegamenti **broadcast** come i **bus** o le **topologie a stella** basate su **hub**, offrono diversi **vantaggi**:

- **Eliminazione delle collisioni:** Gli **switch** evitano il problema delle **collisioni**, il che significa che non c'è spreco di **banda** dovuto a conflitti di **trasmissione**. Questo migliora **significativamente** le **prestazioni** delle **LAN** rispetto ai collegamenti **broadcast**.
- **Collegamenti eterogenei:** Gli **switch** consentono a diversi collegamenti nella stessa **LAN** di funzionare a diverse **velocità** e di utilizzare mezzi **trasmissivi** diversi.
- **Gestione semplificata:** Gli switch **facilitano** la **gestione** della rete in diversi modi. Possono individuare **automaticamente** problemi come le schede di rete **difettose** e **disconnetterle** internamente, senza la necessità dell'intervento **fisico** dell'amministratore di rete. Inoltre, quando si verifica un problema, come un cavo **danneggiato**, solo il nodo che utilizzava quel cavo viene **disconnesso**, rendendo la **risoluzione** dei **problem**i più **efficiente**.

Switch e router a confronto

Nel confronto tra **switch** e **router**, emergono **differenze significative** nei loro ruoli e nelle caratteristiche:



- **Switch:**

- Operano a **livello 2** (livello di collegamento), **inoltrando pacchetti** utilizzando gli indirizzi **MAC**.
- Sono dispositivi **plug-and-play** con capacità di filtraggio e inoltro dei pacchetti relativamente elevate.
- Le reti di switch sono solitamente organizzate in una **topologia** ad **albero** per evitare cicli con i frame broadcast.
- Le reti di switch possono richiedere **tabelle ARP** di **grandi dimensioni**, causando traffico ARP significativo e richiedendo **notevoli risorse** di **elaborazione**.
- Non offrono protezione contro **tempeste di broadcast**, quindi se un host invia un flusso ininterrotto di pacchetti broadcast, può causare il collasso della rete.
- Solitamente utilizzati in reti con un **numero limitato** di **host** e segmenti.

- **Router:**

- Operano a **livello 3** (livello di rete), inoltrando pacchetti utilizzando gli **indirizzi IP**.
- Consentono una **topologia** di rete più **flessibile** e non sono vincolati a una struttura ad albero.
- **Proteggono** dalle **tempeste di broadcast** a livello 2.
- Richiedono la **configurazione** degli indirizzi IP, **non** sono **plug-and-play**.
- Possono avere **tempi di elaborazione** più lunghi rispetto agli **switch**, poiché devono elaborare fino ai campi del livello 3.

In sintesi, gli **switch** sono adatti per reti di dimensioni più limitate con un numero limitato di **host** e **segmenti**, offrendo prestazioni elevate e una configurazione semplificata. D'altra parte, i router sono più adatti per reti di **dimensioni** più grandi, con una **topologia** di rete complessa, protezione contro tempeste di **broadcast** e necessità di instradamento a livello 3. La scelta tra i due **dipenderà** dalle esigenze specifiche di **ciascuna rete**.

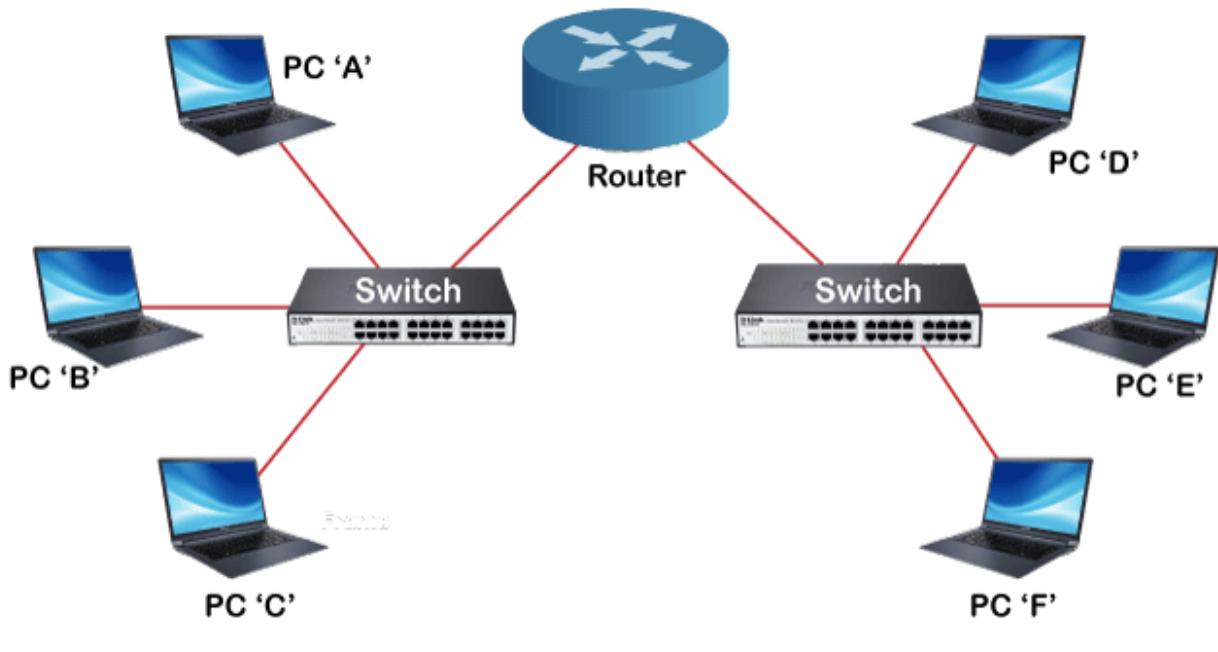
	Hub	Router	Switch
Isolamento del traffico	No	Sì	Sì
Plug and play	Sì	No	Sì
Instradamento ottimale	No	Sì	No

Schemi di indirizzamento: Piatto VS Gerarchico

Ci sono due livelli principali di instradamento nelle reti: il livello **Data Link Layer** (DLL) e il livello **Network Layer** (Livello di Rete).

- **Indirizzamento Flat a livello DLL:** A questo livello, gli indirizzi sono "flat", il che significa che non seguono alcuna **logica** di **distribuzione** nella rete. Ad esempio, gli indirizzi **MAC** vengono assegnati dalla **fabbrica** produttrice della scheda di rete. Questo schema permette ai dispositivi di essere **collegati** in qualsiasi punto della rete, poiché l'indirizzo stesso **non contiene** informazioni di **intradamento**.
- **Indirizzamento Gerarchico a livello di Rete:** A questo livello, gli indirizzi sono "gerarchici" e basati su una **struttura a livelli**. Questa struttura gerarchica consente di determinare il percorso di instradamento in base al valore dell'indirizzo **IP**.

Il **routing** in Internet funziona grazie alla **suddivisione** delle conoscenze sui vari router: ogni **router** non deve conoscere tutti gli **indirizzi** di Internet, ma solo una parte o deve delegare l'operazione ad altri router. L'indirizzamento "flat" è conveniente perché **non impone** regole di **posizionamento** fisso degli **indirizzi**, ma ha il **limite** delle **dimensioni** dello schema di **indirizzamento**.



LAN Virtuali (VLAN)

Le **LAN virtuali (VLAN)** consentono di dividere una rete fisica in più segmenti virtuali, ognuno dei quali può essere configurato e gestito in modo indipendente dagli altri. Il funzionamento di una VLAN si basa **sull'assegnazione di tag (etichette)** ai frame di dati all'interno della rete. In questo modo, i dispositivi in una stessa **VLAN** possono **comunicare** tra di loro come se fossero sulla stessa **rete fisica**, mentre i frame tra **VLAN diverse** richiedono un **router** o un dispositivo di livello superiore per l'instradamento. Per realizzare una VLAN lo switch viene partizionato in più LAN.

Le **VLAN** consentono di affrontare **tre principali problemi** nelle reti:

- **Mancanza di isolamento del traffico:** Con le **VLAN**, è possibile isolare il traffico all'interno di gruppi **specifici**, impedendo che il traffico **broadcast** si diffonda in tutta la rete. Ciò migliora le **prestazioni** e la **sicurezza** della LAN.
- **Uso inefficiente degli switch:** Invece di utilizzare un gran numero di **switch** separati per ogni gruppo, le **VLAN** permettono di definire gruppi virtuali all'interno di uno switch fisico. Ciò consente di utilizzare **meno switch**, **migliorando l'efficienza**.
- **Gestione degli utenti:** Con le **VLAN**, è facile configurare e gestire gli utenti che si spostano tra **gruppi** o **appartengono** a più **gruppi**. La configurazione delle porte sugli switch può essere **rapidamente** adattata alle esigenze degli **utenti**.

Le **VLAN** possono essere implementate basandosi sulle porte degli **switch**, sugli indirizzi **MAC**, sui protocolli di livello di rete o su altri criteri. Inoltre, le **VLAN** possono essere estese attraverso **router IP** per creare una rete virtuale più **ampia**. In definitiva, le **VLAN** sono una soluzione **flessibile** e **scalabile** per affrontare le sfide di isolamento del traffico, utilizzo efficiente degli **switch** e gestione degli utenti nelle **reti locali**.

VLAN Untagged e Tagged

- **VLAN Untagged:** In una VLAN "Untagged," i pacchetti **non** hanno un **tag speciale** quando lasciano una porta. La porta è associata a una VLAN specifica e **tutti** i pacchetti **inviai** attraverso quella porta **appartengono automaticamente** a quella **VLAN**.
- **VLAN Tagged:** In una VLAN "Tagged," i pacchetti sono **contrassegnati** con un tag **VLAN** speciale quando **lasciano** una porta. Questo tag identifica a quale **VLAN appartiene il pacchetto**. È utilizzato per consentire a più VLAN di **condividere** la stessa **connessione** fisica, come tra **switch** o dispositivi che supportano molte VLAN.

Standard IEEE 802.1Q

Questo è uno **standard** che definisce come aggiungere i **tag VLAN** ai **pacchetti**. Assicura che i dispositivi di rete possano comunicare tra VLAN diverse su una singola connessione **fisica** (trunk) e che i tag **VLAN** siano interpretati **correttamente**. È importante che tutti i dispositivi nel trunk supportino lo standard 802.1Q per un funzionamento corretto.

Reti di Calcolatori - 135