

Skippato nel riassunto:

- Introduzione fino ai primi protocolli
- Lan, Wan (non protocolli)

Skippato negli appunti interi (quindi anche riassunto):

- Gestione droni
- Sicurezza di Rete
- Bluetooth

Soffermato principalmente su Protocolli e qualunque cosa che aveva un acronimo

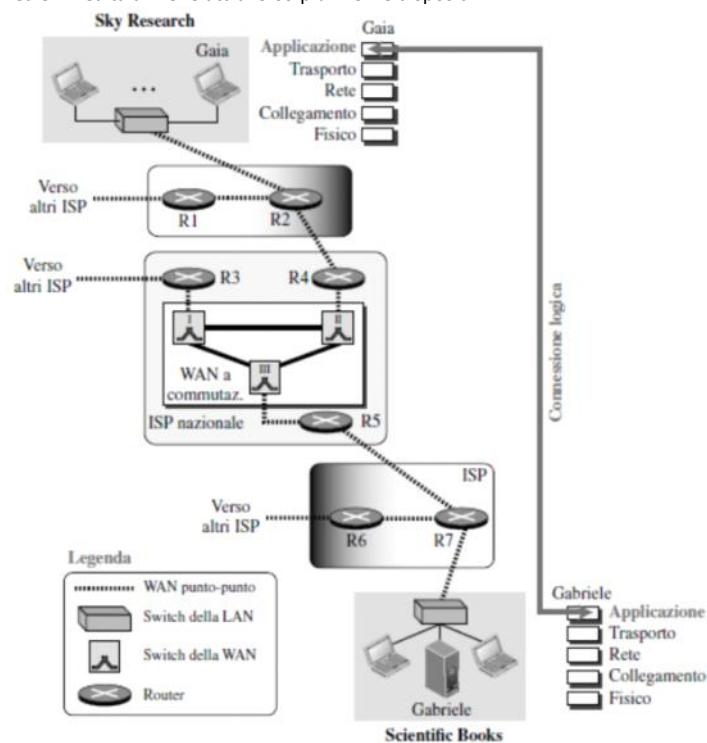
Comunque è un riassunto un po estensivo, praticamente sono i vari appunti normali ma ridotti togliendo/riducendo parti irrilevanti

Reti e Internet

giovedì 26 giugno 2025 20:25

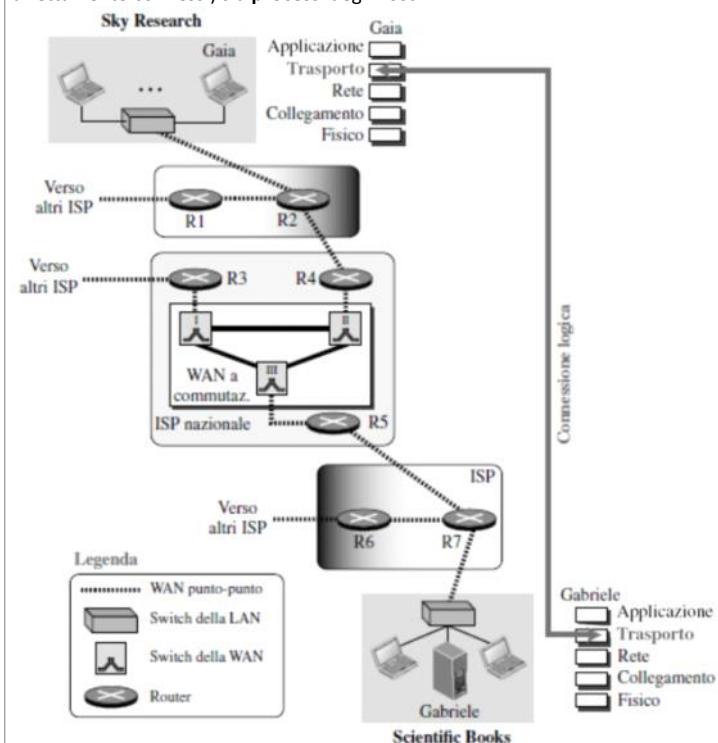
Comunicazione a Livello Applicazione

Possiamo immaginare una **connessione logica** tra i due utenti; la connessione reale in realtà avviene attraverso più livelli e dispositivi



Comunicazione a Livello Trasporto

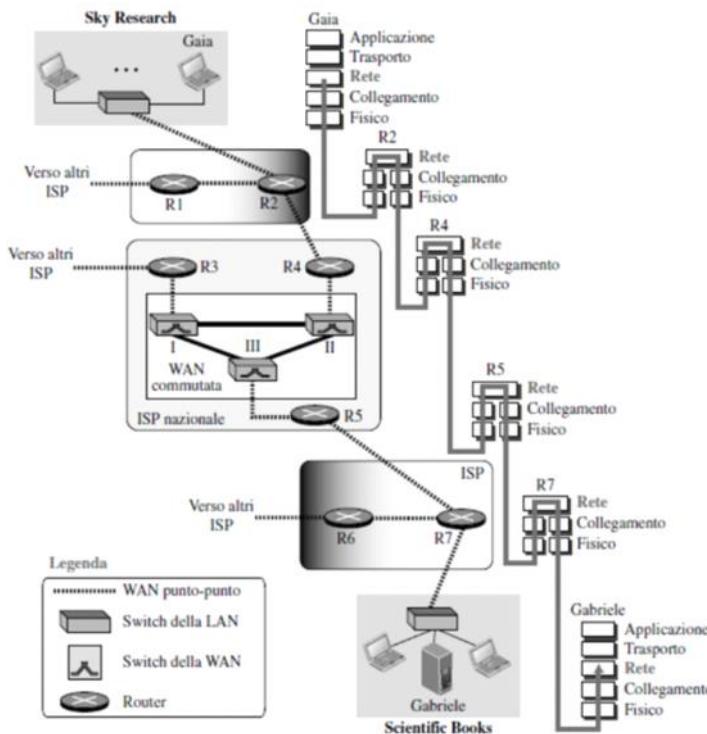
Protocolli di trasporto forniscono la **comunicazione logica** (come se fossero direttamente connessi) tra **processi** degli host



Comunicazione a Livello Rete

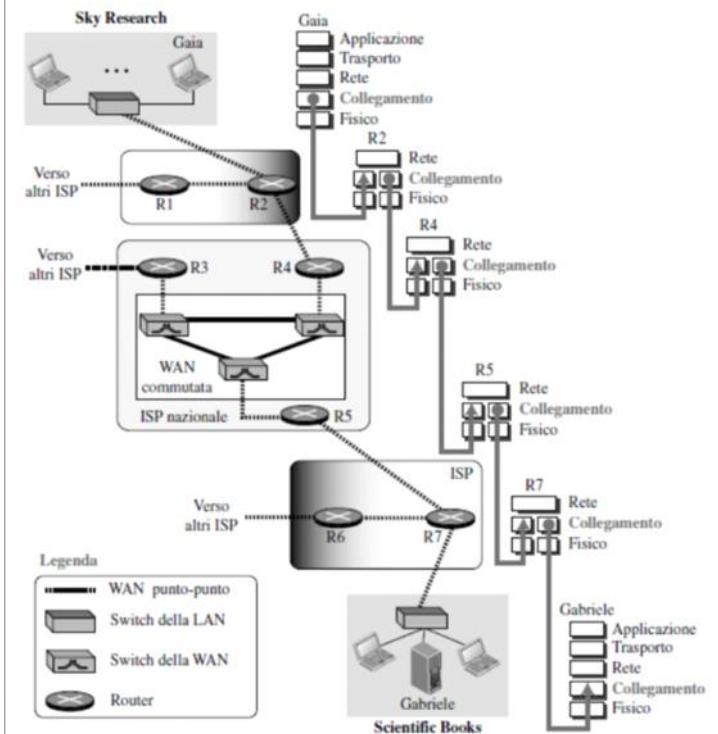
Comunicazione host-to-host

La comunicazione avviene tra host (end-to-end)



Comunicazione a Livello Collegamento

Internet è una combinazione di reti unite da dispositivi di collegamento (router e switch). La comunicazione a liv. di collegamento è **hop-to-hop**

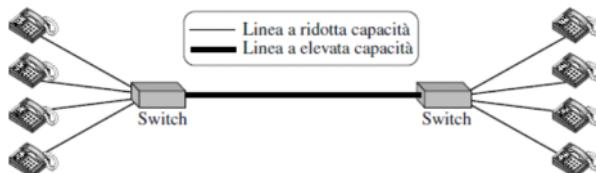


Terminologia:

- Host e router sono **nodi di rete** che creano/modificano/inoltrano il segnale
- I canali di comunicazione che collegano nodi adiacenti su un cammino sono i **collegamenti o link** (cablati, wireless o LAN)
- L'**host** è una macchina che esegue applicazioni, mentre il **server** è una macchina che esegue programmi che offrono servizi ad applicazioni utenti (es siti web)
- Gli **ISP** permettono all'utente di collegarsi all'Internet tramite **access network** (collegamenti agli ISP locali) per arrivare alla **backbone di internet** (diversi ISP (locali/regionali/nazionali/internazionali) di vario tipo che gestiscono il routing tra di loro)
- Un **protocollo** è un insieme di **regole** che definiscono come i dispositivi comunicano su una rete

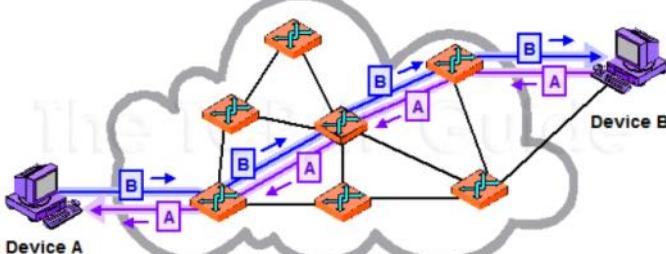
Commutazione

Reti a commutazione di circuito



Rimane sempre un collegamento aperto, chiamato **circuito o socket**, usato per la comunicazione tra i due dispositivi.

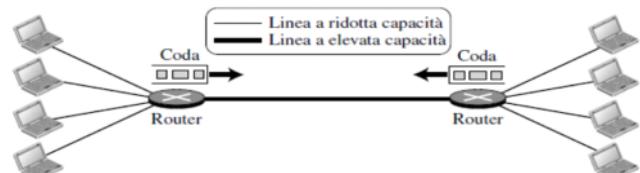
Le **risorse necessarie sono riservate** per tutta la durata della comunicazione e le **informazioni sul circuito sono mantenute dalla rete**.



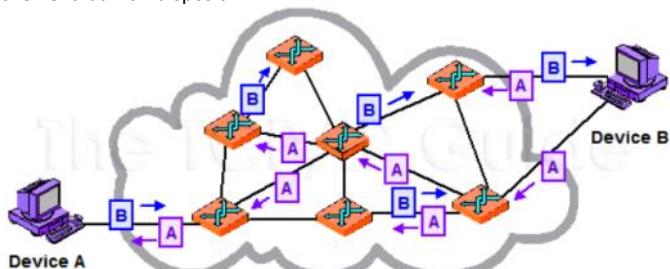
Le risorse di rete (ad es. ampiezza di banda/bandwidth) sono suddivise in "pezzi", dove vengono allocati ai vari collegamenti e rimangono **inattive** se non utilizzate (es. pause durante una conversazione telefonica)

Ad esempio la **banda** viene divisa in **frequenza (FDM)** e **tempo (TDM)**.

Reti a commutazione di pacchetto (store and forward)



I dispositivi si **scambiano blocchi di dati (pacchetti)** con una lunghezza max definita (messaggi divisi in più pacchetti) che viaggiano su percorsi diversi. Gli switch quindi **memorizzano (store)** e **inoltrano (forward)** i pacchetti provenienti dai vari dispositivi.



Il router **memorizza in una coda** i pacchetti in arrivo, creando un po' di ritardo ma permettendo a più utenti di usare la rete

Capacità e Prestazioni

In una rete a commutazione di **pacchetto** le prestazioni sono determinate da:

- **Bandwidth e bit rate**: misurata in **bit al secondo (bps)** indica quanti bit al secondo un link garantisce di trasmettere. Indica la **capacità di trasferire dati**
- **Throughput**: se il bit rate è la **velocità potenziale**, il throughput indica quanto velocemente riusciamo **effettivamente** a inviare i dati. Il throughput di un percorso è dato dal **throughput minimo** dei vari link.
Il link tra due router è condiviso tra i flussi dati, quindi se entrambi i router sono collegati a N host, il link tra i router è il throughput del link diviso N
Es: due router hanno entrambi 3 dispositivi collegati e un link tra di loro di 600kbps, il throughput end-to-end sarà $600/3 = 200\text{ kbps}$
- **Latenza (delay)**: indica quanto tempo serve ad un pacchetto per arrivare completamente a destinazione. Ci sono **4 cause di ritardo** dovute principalmente all'accodamento dei pacchetti nei buffer del router:
 - d_{proc}** = **ritardo di elaborazione (processing)**: controllo sui bit e canale di uscita del pacchetto
 - d_{queue}** = **ritardo di accodamento (queueing)**: attesa di trasmissione
 - d_{trans}** = **ritardo di trasmissione (transmission)**: tempo di invio di tutti i bit del pacchetto
$$\text{Ritardo di trasmissione} = \frac{\text{lung. pacchetto (bit)}}{\text{bit rate link (bps)}}$$
 - d_{prop}** = **ritardo di propagazione (propagation)**: tempo per un bit per propagarsi sul link (da punto A a punto B)
$$\text{Ritardo di propagazione} = \frac{\text{lungh. link fisico}}{\text{velocità prop. del link } (\sim 2 * 10^8 \frac{\text{m}}{\text{sec}}) \text{ (circa velocità luce)}}$$

$$d_{\text{nodo}} = d_{\text{proc}} + d_{\text{queue}} + d_{\text{trans}} + d_{\text{prop}}$$

- **Perdita pacchetti (packet loss)**: Se il buffer del router è pieno, i pacchetti in arrivo vengono **scartati** (persi). Questi possono essere **rtrasmessi** o no

Tramite traceroute possiamo calcolare il **ritardo** e possiamo definire la **quantità di dati che viaggiano nella rete** con il prodotto **bit rate * ritardo**

Esercizio

- Quanto impiega un pacchetto di 1000 byte per propagarsi su un link di 2500km, con velocità di propagazione pari a $2,5 \times 10^8 \text{ m/s}$ e bit rate di 2 Mbps?
- Questo ritardo dipende dalla lunghezza del pacchetto?

- Calcola ritardo di trasmissione

- Packet lenght: 1000 byte = 8000 bit
- Link lenght: 2500km = 2500000 m
- Velocità propagazione: $2,5 \times 10^8 \text{ m/s}$
- Bit rate: 2 Mbps = 2000000 bps

$$1) \text{ Ritardo di propagazione} = \frac{2500000}{2,5 \times 10^8} = \frac{2,5}{2,5 \times 10^2} = \frac{1}{1 \times 10^2} = \frac{1}{100} = 0,01 \text{ sec} = 10 \text{ ms}$$

2) No, il ritardo di propagazione dipende dalla lunghezza del link e dalla velocità di propagazione

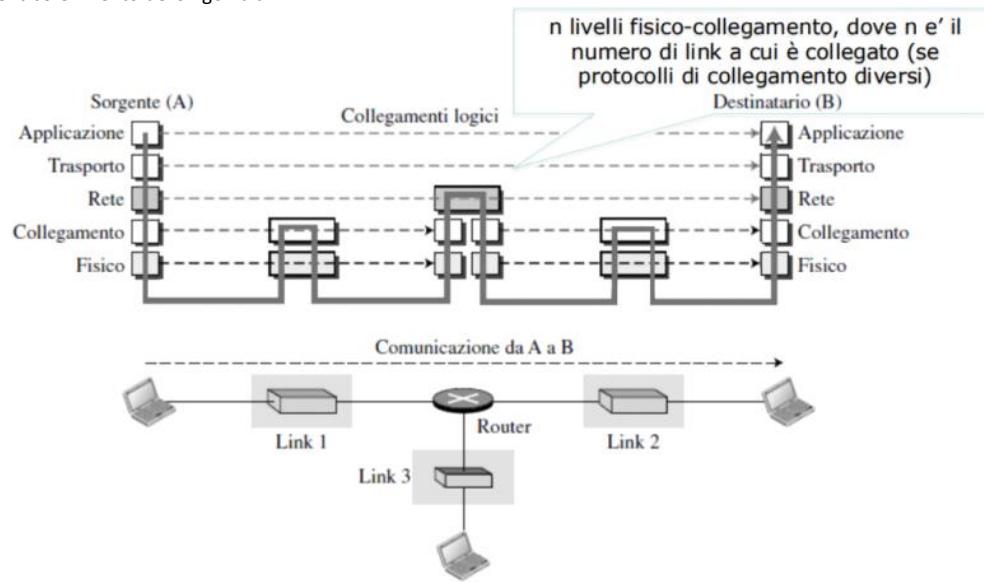
$$3) \text{ Ritardo di trasmissione} = \frac{8000}{2000000} = 0,004 \text{ sec} = 4 \text{ ms}$$

TCP/IP

In Internet si usa la **gerarchia di protocolli TCP/IP** (un protocollo di un liv. superiore è supportato dai servizi forniti dal liv. inferiore) che definisce una **pila di protocolli** divisi in livelli:

- **Applicazione**: sede dell'applicazione di rete
 - HTTP, SMTP, FTP, DNS
 - Pacchetti = **messaggi**
- **Trasporto**: trasferimento messaggi al liv. applicazione tra client e server di un applicazione
 - TCP, UDP
 - Pacchetti = **segmenti**
- **Rete**: instradamento dei segmenti dall'origine alla destinazione (**end-to-end**)
 - IP, instradamento, ICMP

- Pacchetti = **datagrammi**
- **Collegamento (link)**: trasmissione datagrammi da un nodo al successivo nel percorso (**hop-to-hop**)
 - MAC, WiFi, Ethernet, Aloha, CSMA, CDMA
 - Pacchetti = **frame**
- **Fisico**: trasferimento dei singoli bit



Applicazione

venerdì 27 giugno 2025 15:44

Il liv. applicazione fornisce servizi all'utente (es email, sito web, streaming video, condivisione file). Per creare una applicazione bisogna definire:

- Il tipo di architettura da creare (client-server, peer-to-peer, ecc)
- Il modo di comunicazione tra i processi dell'applicazione
- I tipi di servizi di rete che richiede l'applicazione (affidabilità, banda, ecc)

Paradigma client-server:

- Client: in esecuzione solo quando deve chiedere servizi al server
- Server: sempre in esecuzione in attesa dei client, può fornire servizi a più client

Per visitare un sito web si usa:

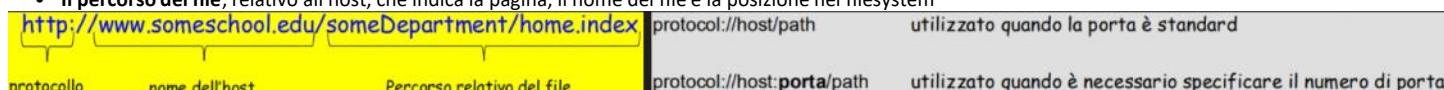
- Web client (es. browser): interfaccia dell'utente per accedere e visualizzare i siti
- Web server (es. Apache)
- HTML: linguaggio standard per le pagine web
- HTTP: uno dei protocolli per la comunicazione tra client e server

Sito web:

- Una pagina web è costituita da oggetti (file HTML, immagine, audio, ecc)
- Ogni pagina è formata da un file base HTML che include diversi oggetti referenziati
- Ogni oggetto è referenziato da un URL (Uniform Resource Locator)

L'URL è composto da 3 parti:

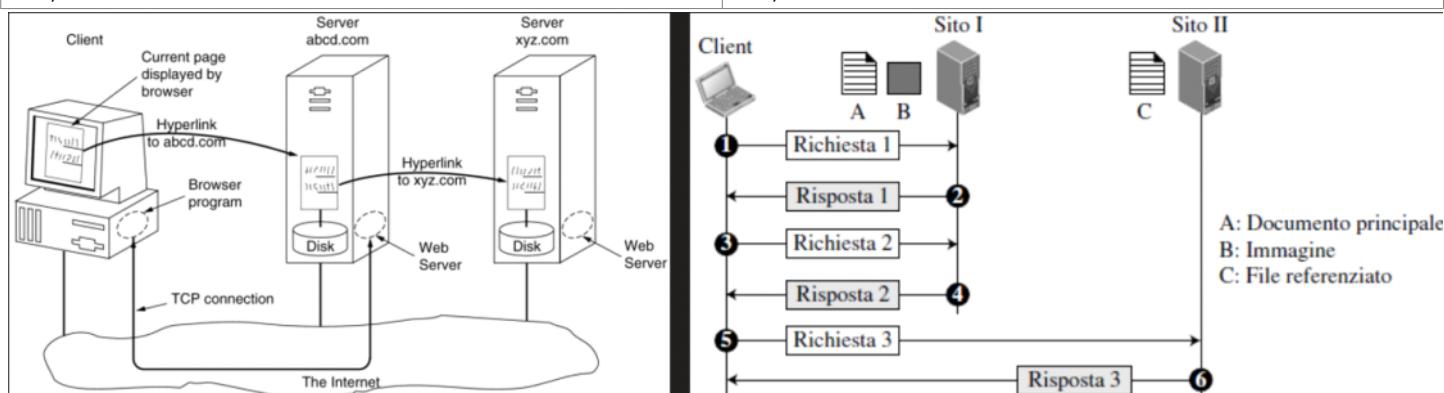
- Il protocollo
- Il nome dell'host in cui è situata la pagina
- Il percorso del file, relativo all'host, che indica la pagina, il nome del file e la posizione nel filesystem



HTTP (HyperText Transfer Protocol) e Web Cache

HTTP è un protocollo a liv. applicazione che definisce come i client richiedono pagine ai server e come questi le trasferiscono al client

Client	Server
Il browser richiede, riceve, "visualizza" gli oggetti del web. 1) Determina l'URL ed estrae host e filename 2) Esegue una connessione TCP alla porta 80 dell'host indicato nell'URL 3) Invia richiesta per i file 4) Riceve i file dal server 5) Chiude la connessione 6) Visualizza il file	Il server invia oggetti in risposta ad una richiesta. Può servire più richieste, provenienti anche da client diversi 1) Accetta una connessione TCP da un client 2) Riceve i nomi dei file richiesti 3) Recupera i file dal disco 4) Invia i file al client 5) Rilascia la connessione



Tempo di Risposta

RTT (Round Trip Time) è il tempo che impiega un pacchetto per andare dal client al server e tornare al client (include ritardi).

Il tempo di risposta di una trasmissione di un file si può definire come $2 * RTT + \text{trasmissione file}$

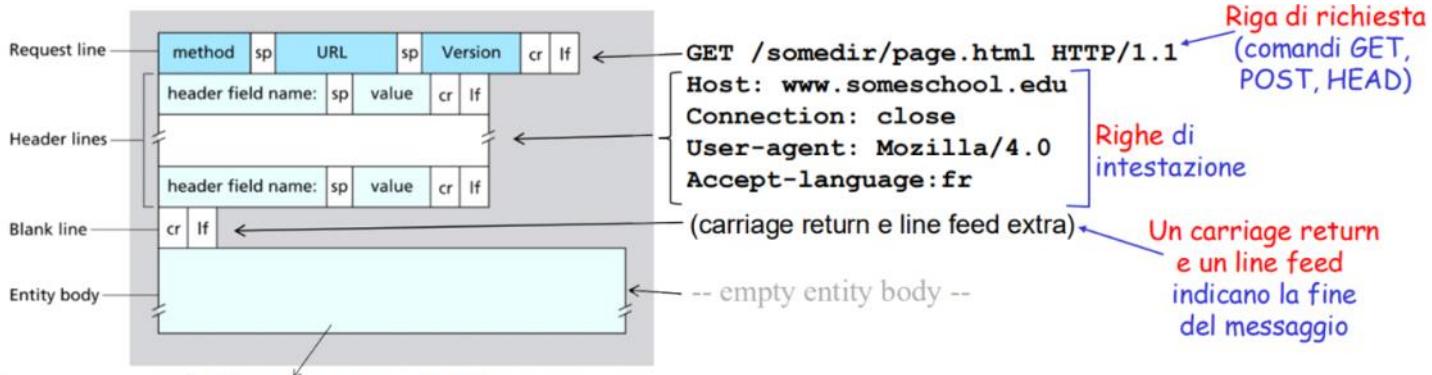
- 1) Un RTT per iniziare la connessione TCP
- 2) Un RTT per la richiesta HTTP e i primi byte di risposta
- 3) Tempo di trasmissione del file

Connessioni HTTP

Connessioni Persistenti	Connessioni Non Persistenti
<ul style="list-style-type: none">• Modalità di default• Più oggetti possono essere trasmessi su una singola connessione TCP tra client e server• La connessione viene chiusa quando rimane inattiva per un lasso di tempo (timeout) configurabile <p>Un RTT per ogni oggetto inviato/ricevuto (la connessione rimane aperta)</p>	<ul style="list-style-type: none">• Un solo oggetto viene trasmesso su una connessione TCP (file HTML, immagine, ecc)• Ciascuna coppia richiesta/risposta viene inviata su una connessione TCP separata• Prima di inviare una richiesta al server, bisogna stabilire una connessione <p>2 RTT per oggetto (deve riaprire la connessione ogni volta)</p>

Richiesta HTTP (Lato Client)

Per inviare/richiedere info al server si possono usare 4 tipi di comandi (e le relative informazioni da chiedere) nel messaggio di richiesta HTTP



Campo vuoto per il GET, utilizzato per il POST

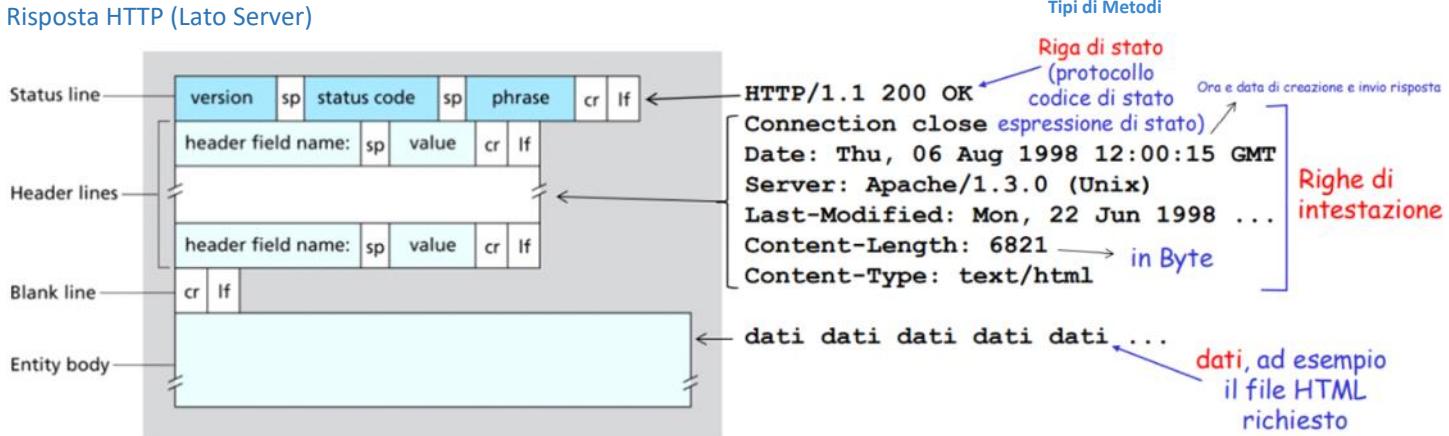
Intestazione	Descrizione
User-agent	Indica il programma client utilizzato
Accept	Indica il formato dei contenuti che il client è in grado di accettare
Accept-charset	Famiglia di caratteri che il client è in grado di gestire
Accept-encoding	Schema di codifica supportato dal client
Accept-language	Lingaggio preferito dal client
Authorization	Indica le credenziali possedute dal client
Host	Host e numero di porta del client
Date	Data e ora del messaggio
Upgrade	Specifica il protocollo di comunicazione preferito
Cookie	Comunica il cookie al server (verrà spiegato successivamente)
If-Modified-Since	Invia il documento solo se è più recente della data specificata

Intestazione della Richiesta

GET	E' usato quando il client vuole scaricare un documento dal server. Il documento richiesto è specificato nell'URL. Il server normalmente risponde con il documento richiesto nel corpo del messaggio di risposta.
HEAD	E' usato quando il client non vuole scaricare il documento ma solo alcune informazioni sul documento (come ad esempio la data dell'ultima modifica). Nella risposta il server non inserisce il documento ma solo degli header informativi.
POST	E' usato per fornire input al server (contenuto dei campi di un form). L'input arriva al server nel corpo dell'entità.
PUT	E' utilizzato per memorizzare un documento nel server. Il documento viene fornito nel corpo del messaggio e la posizione di memorizzazione nell'URL.

Per inviare info al server
E' possibile anche usare GET
GET www.somesite.com/animalsearch?monkeys&banana

Tipi di Metodi



Intestazione	Descrizione
Date	Data corrente
Upgrade	Specifica il protocollo preferito
Server	Indica il programma server utilizzato
Set-Cookie	Il server richiede al client di memorizzare un cookie
Content-Encoding	Specifica lo schema di codifica
Content-Language	Specifica la lingua del documento
Content-Length	Indica la lunghezza del documento
Content-Type	Specifica la tipologia di contenuto
Location	Chiede al client di inviare la richiesta a un altro sito
Last-modified	Fornisce data e ora di ultima modifica del documento

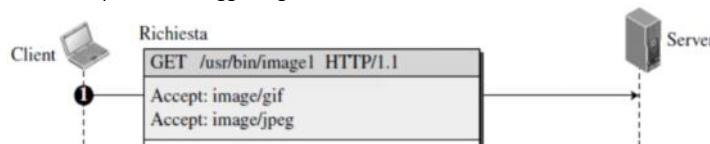
Intestazione della risposta

Code	Meaning	Examples
1xx	Information	100 = server agrees to handle client's request
2xx	Success	200 = request succeeded; 204 = no content present
3xx	Redirection	301 = page moved; 304 = cached page still valid
4xx	Client error	403 = forbidden page; 404 = page not found
5xx	Server error	500 = internal server error; 503 = try again later

Alcuni codici di stato della risposta

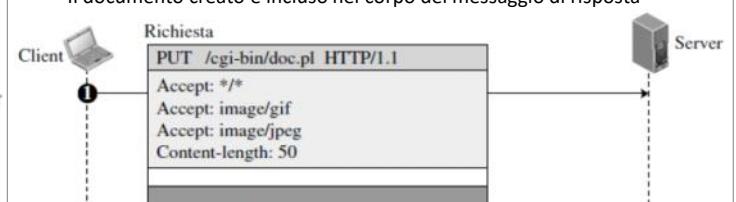
Esempio GET

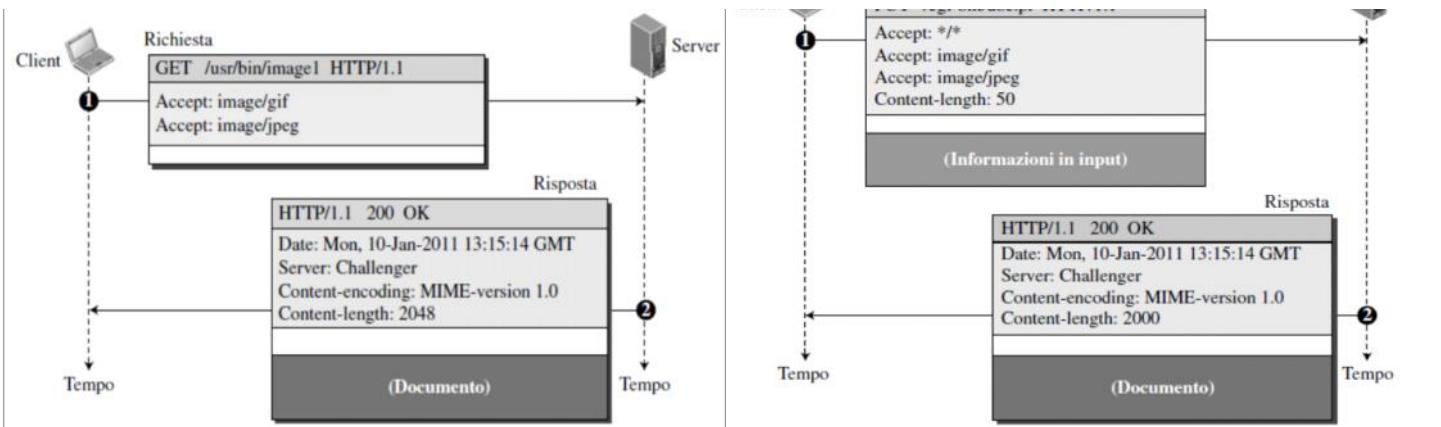
- Client usa **GET** per ottenere un immagine nel percorso "/usr/bin/image1".
- La riga di richiesta contiene il **metodo** (GET), l'**URL** e la **versione** (1.1) del protocollo HTTP. L'intestazione contiene le specifiche delle immagini accettate dal client (GIF e JPEG). Il messaggio di richiesta non ha corpo
- Il messaggio di risposta contiene la riga di stato e quattro righe di intestazione che contengono la data, il server, il metodo di codifica del contenuto (MIME) e la lunghezza del contenuto
- Il corpo del messaggio segue l'intestazione



Esempio PUT

- Client usa **PUT** per pubblicare una pagina Web da pubblicare sul server
- La riga di richiesta contiene il **metodo** (PUT), l'**URL** e la **versione** (1.1) del protocollo HTTP. L'intestazione è costituita da quattro righe e il corpo contiene la pagina Web inviata
- Il messaggio di risposta contiene la riga di stato e quattro di intestazione
- Il documento creato è incluso nel corpo del messaggio di risposta





Cookie

Siccome HTTP non **mantiene lo stato (stateless)** della connessione, i siti utilizzano il meccanismo **Cookie** per **tenere traccia** dell'utente. I **Cookie** si usano per **creare una sessione** di richieste/risposte HTTP "con stato" (es, carrelli di un venditore online) che ha una **vita breve**. Funzionamento:

- 1) Se il client visita un sito e non ha ancora un cookie, il server **genera un SID (Session ID)** e lo invia con **Set-Cookie: SID = abcd**
- 2) Il browser salva il cookie e lo **invia insieme alle richieste** al server
- 3) Il server usa il cookie per **riconoscere l'utente** e fornire risposte personalizzate consultando il suo database

Quindi servono:

- 1) **Header Set-Cookie** nella risposta HTTP dal server
- 2) **Header Cookie** nella richiesta HTTP dal client
- 3) Un file cookie mantenuto dal browser
- 4) Un database con l'associazione (SID, utente) sul server

Web Caching

Il **web caching** si usa per migliorare le prestazioni del server salvando copie delle pagine richieste così da non doverle richiedere al server ogni volta.

In questo modo riduce i tempi di risposta per l'utente, riduce il traffico sulla rete e aiuta a dividere il carico sui server

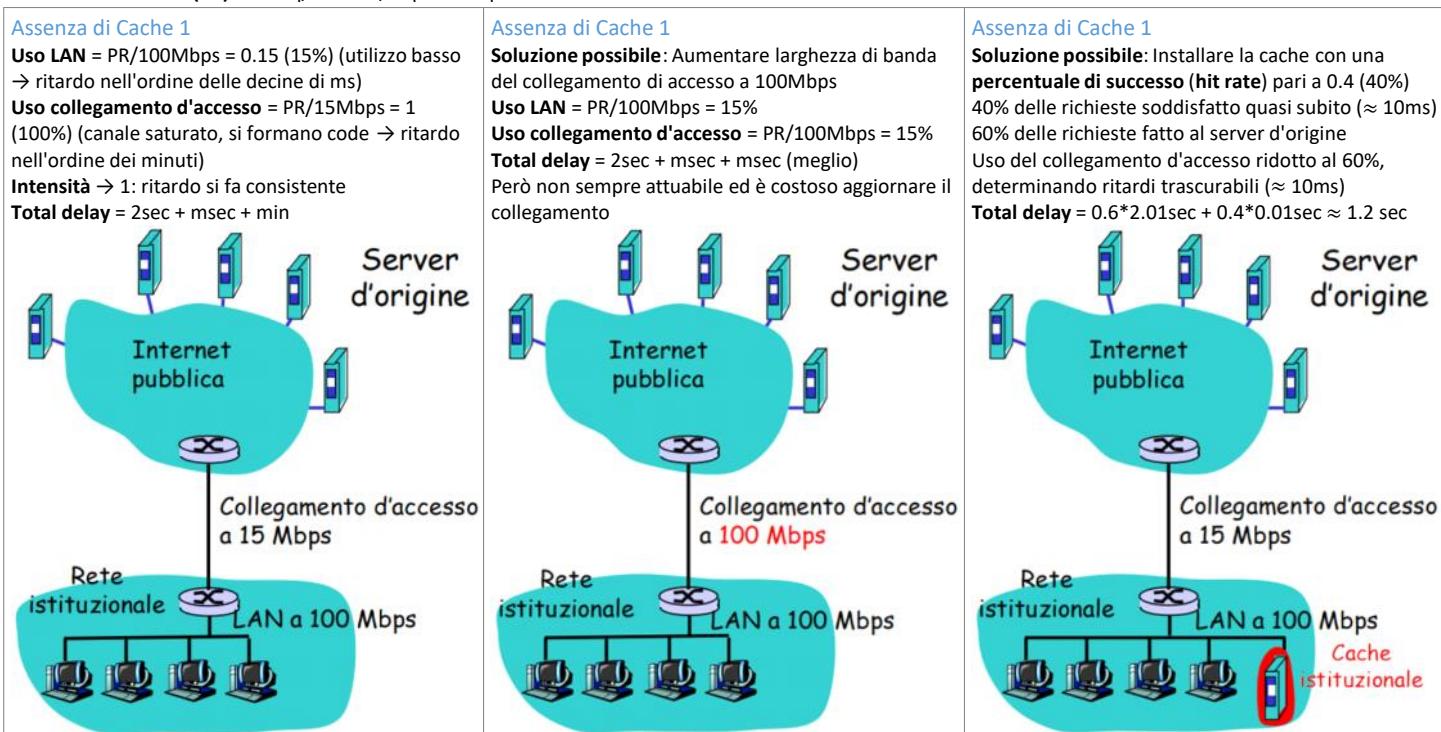
Browser Caching	Server Proxy
Il browser mantiene una cache locale delle pagine visitate, configurabile dall'utente (es. durata, scadenza). Si può usare l'intestazione HTTP Expires per decidere per quanto mantere i dati	Si interpone tra client e server e conserva pagine per più utenti. Se una pagina è già presente nella cache viene restituita direttamente al client senza contattare il server

Quindi, se il client invia una richiesta HTTP alla cache :

- E la cache **non ha quell'oggetto**, allora **invia** la richiesta HTTP al **server** e la cache **memorizza** la risposta HTTP (per richieste future) insieme all'**ultima data di modifica** e invia la risposta al client
- E la cache **ha l'oggetto**, verifica prima se è **aggiornato** inviando una **richiesta condizionale** al server per vedere se è stato modificato e se non è **scaduto** (modificato) lo invia al client

Esempio:

- Dimensione media oggetto: 1Mb
- Frequenza media richieste dai browser ai server = 15 richieste al secondo
- Ritardo per recuperare un oggetto su Internet (**Internet Delay**) = 2 sec
- Il tempo totale di risposta (**total delay**) = LAN delay + access delay + Internet delay
- **Peso richieste (PR)** = $15 \text{req/s} * 1\text{Mb}/\text{req} = 15\text{Mbps}$



DNS (Domain Name System)

Molti host in Internet hanno un **nome (hostname)** (es, google.com) che sono facili da ricordare, però per accederci si usa un **indirizzo IP** (32 bit) per indirizzare i datagrammi da inviare.

Il **DNS** è un sistema che **traduce** gli hostname in IP (es www.google.com → 192.168.1.1). Però esistono 2^{32} IP da memorizzare:

- Memorizzazione → **database distribuito**, implementato con una **gerarchia di server DNS**
- Accessibilità → **protocollo a liv. applicazione**, per consultare il database e risolvere gli hostname

Il protocollo DNS è eseguito dal client e usa il protocollo di **trasporto UDP** alla porta 53

Esempio:

- 1) Il browser estrae il nome dell'host (es www.google.com) dall'URL e lo passa al client DNS
- 2) Il client DNS invia una **query** con l'hostname ad un server DNS (es IP 8.8.8.8) (per richiedere la traduzione IP)
- 3) Riceve una risposta con l'indirizzo IP dell'hostname (es www.google.com → 142.251.209.46)
- 4) Il browser ora può avviare una connessione TCP al server HTTP localizzato a quell'IP

Aliasing

L'**host aliasing** permette di associare un **nome semplice (alias)** al posto di uno **più complesso (canonico)**.

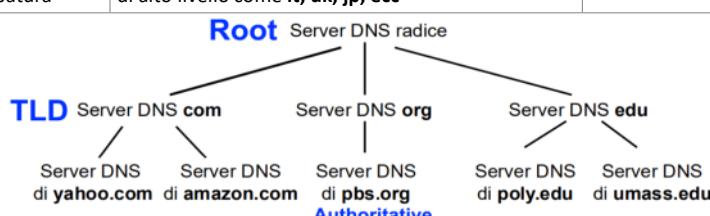
Esempio: Es. www.enterprise.com è un alias di www.relay1.west-coast.enterprise.com

Gerarchia Server DNS

Si usa una **struttura gerarchica** di server DNS dove i vari host vengono distribuiti su svariati server DNS. I vari server DNS salvano nella **cache** (per un certo periodo) la **mappatura** degli IP dei server TLD o di competenza per migliorare le prestazioni

Esistono 3 classi principali di server DNS (esistono anche i server DNS **locali** dei vari ISP):

Root (Server DNS Radice)	TLD (Top-Level-Domain)	Authoritative (Server di Competenza)
Viene contattato dal server DNS locale e indirizza verso i server TLD se non conosce la mappatura	Gestisce domini com, org, net, ecc e di domini locali di alto livello come it, uk, jp, ecc	Forniscono l'IP effettivo degli host



Esempio: Il client vuole l'IP di www.amazon.com:

- 1) Il client chiede al **DNS locale**
- 2) Il DNS locale interroga il **service radice (root)** per trovare il server TLD .com
- 3) Il TLD risponde con il **server di competenza** per amazon.com
- 4) Il server di competenza fornisce l'IP di www.amazon.com

DNS Record

La mappatura è mantenuta nei database come un **resource record (RR)** e vengono spediti tra server e host richiedenti all'interno dei **messaggi DNS**

Formato RR: (Name, Value, Type, TTL)

Tempo residuo di vita

Type=A

Hostname → **IP address**

- **name** è il nome dell'host
- **value** è l'indirizzo IP

Esempio: (relay1.bar.foo.com, 45.37.93.126, A)

Type=MX

Alias → **mail server canonical name**

- **value** è il nome canonico del server di posta associato a **name**

Esempio: (foo.com, mail.bar.foo.com, MX)

Type=NS

Domain name → **Name Server**

- **name** è il dominio (ad esempio foo.com)
- **value** è il nome dell'host del server di competenza di questo dominio

Esempio: (foo.com, dns.foo.com, NS)

Type=CNAME

Alias → **Canonical Name**

- **name** è il nome alias di qualche nome "canonico" (nome vero)
- **value** è il nome canonico

Esempio: (foo.com, relay1.bar.foo.com, CNAME)

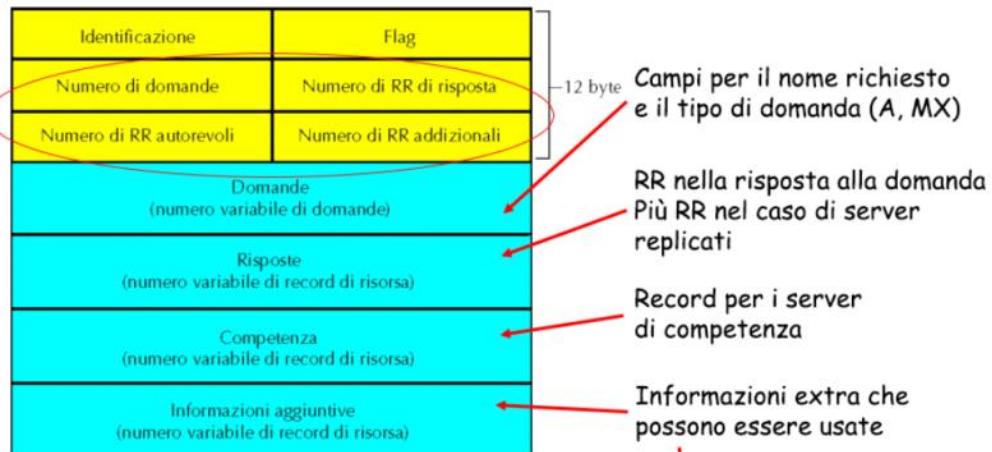
Esempio:

- Server di competenza per un hostname:
 - Contiene record **tipo A** per l'hostname (es: (corsi.di.uniroma1.it, 131.111.45.68, A))
- Server non di competenza per un hostname:
 - Contiene record **tipo NS** per il dominio che include l'hostname
 - Contiene record **tipo A** con l'IP del server DNS nel campo **value** del record NS
 - Esempio: Un server TLD **it** non è competente per l'host "corsi.di.uniroma1.it" e contiene:
 - (uniroma1.it, dns.uniroma1.it, NS)
 - (dns.uniroma1.it, 128.119.40.111, A)

Messaggi DNS

Protocollo DNS: **domande (query)** e messaggi di **risposta**, entrambi con lo stesso **formato**

- Identificazione:** numero di 16 bit per la domanda; la risposta alla domanda usa lo stesso numero
- Flag:**
 - domanda o risposta
 - richiesta di ricorsione
 - ricorsione disponibile
 - risposta di competenza (il server è competente per il nome richiesto)
- Numero di:** numero di occorrenze delle quattro sezioni di tipo dati che seguono



Nel caso di una risposta MX, il campo di risposta contiene il record MX con il nome canonico del server di posta, mentre la sezione aggiuntiva contiene un record di tipo A con l'indirizzo IP relativo all'hostname canonico del server di posta

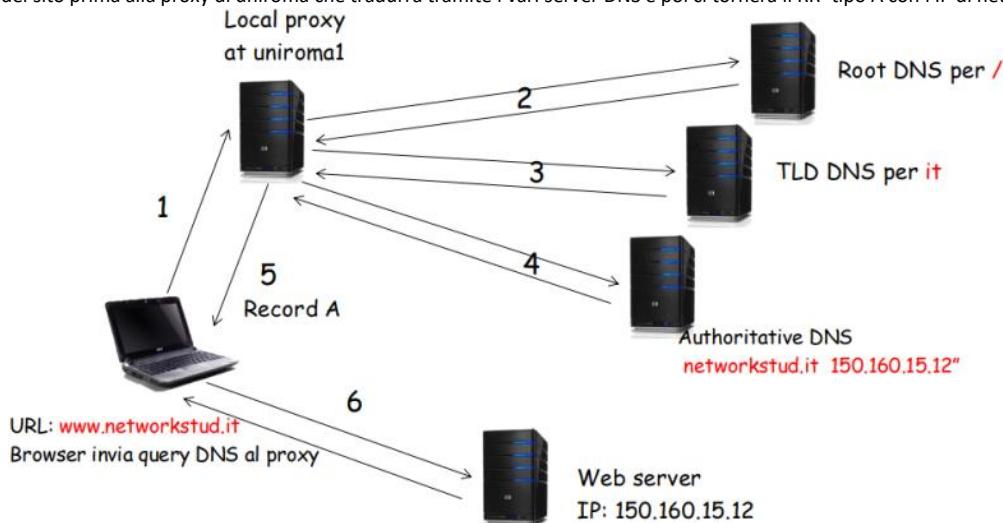
Inserire Record nel Database DNS

Esempio: vogliamo aggiungere un nuovo dominio della nostra società "Network Stud"

Si possono aggiungere nuovi domini al DNS contattando un **registrar** che verifica l'unicità del dominio richiesto e lo inserisce nel database

- Registriamo il nome networkstud.it** presso **registrar** (www.registro.it)
 - Inseriamo nel server di competenza (es. nostro server DNS www.dns1.networkstud.it) un record **tipo A** per www.networkstud.it e un record **tipo MX** per networkstud.it per indicare il server di posta elettronica
 - A record:** www.networkstud.it → 150.160.15.12 (indirizzo server web)
 - MX record:** networkstud.it → mail.networkstud.it (posta elettronica)
 - CNAME record:** mail.networkstud.it → mail.google.com (se usiamo Gmail)
 - Forniamo al registrar i nomi e gli IP dei server DNS di competenza (primario e secondario)
 - Registrar inserisce due RR nel server TLD it:
 - (networkstud.it, dns1.networkstud.it, NS)
 - (dns1.networkstud.it, 212.212.212.1, A)

Gli utenti otterranno l'IP del sito prima alla proxy di uniroma che tradurrà tramite i vari server DNS e poi ci tornerà il RR tipo A con l'IP di networkstud



FTP (File Transfer Protocol)

FTP è un protocollo per il **trasferimento file** da/a un host remoto. Il **client** è colui che **inizia il trasferimento** e il **server** è l'**host remoto**. Esecuzione connessione:

- Il client fornisce il nome dell'host remoto → **FTP client** stabilisce una **connessione TCP** sulla **porta 21** con il **FTP server**
- Stabilita la connessione, il client fornisce nome e password che vengono inviate sulla connessione come parte dei comandi
- Ottenuta l'autorizzazione dal server, il client può inviare/ricevere i file al server

Connessione di controllo	Connessione di dati
Connessione di controllo Si occupa delle info di controllo del trasferimento (es ID utente, cambio dir, ecc) <ul style="list-style-type: none"> Usa la porta 21 della connessione per inviare le info Parte quando si richiede sul client ftp HostName Connessione di controllo è "fuori banda" (out of band), quindi invia le informazioni separatamente dai dati trasferiti Il Server FTP mantiene lo "stato" (es: directory corrente, autotentificazione) 	Connessione di dati Si occupa del trasferimento dei file <ul style="list-style-type: none"> Connessione dati TCP sulla porta 20 Parte quando il server riceve un comando per trasferire file (es: get, put) Dopo il trasferimento di un file, il server chiude la connessione Quindi viene creata una connessione per ogni file trasferito nella sessione
Comandi comuni Invia come testo ASCII sul server remoto <ul style="list-style-type: none"> USER username PASS password LIST: elenca i file della dir corrente, la lista di file viene inviata dal server su una nuova connessione di dati RETR filename: recupera (get) un file dalla dir corrente remota 	Codici di ritorno comuni Codici di stato ed espressione (come in HTTP) che seguono il comando client <ul style="list-style-type: none"> 331 Username OK, password required 125 data connection already open; transfer starting 425 Can't open data connection 452 Error writing file <p>Le risposte sono composte da due parti:</p>

- STOR filename: memorizza (put) un file nella dir corrente remota

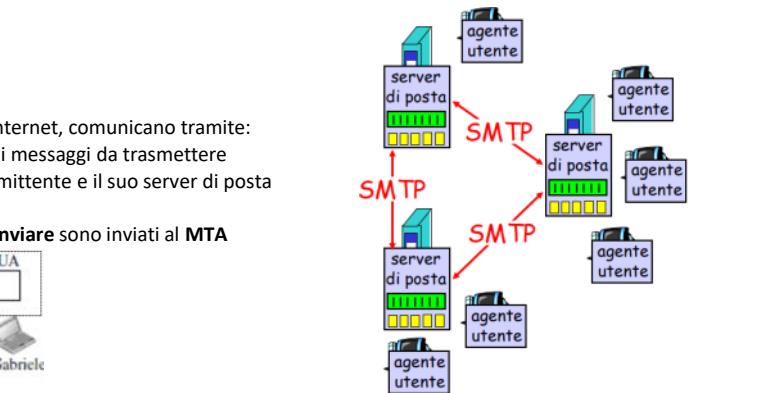
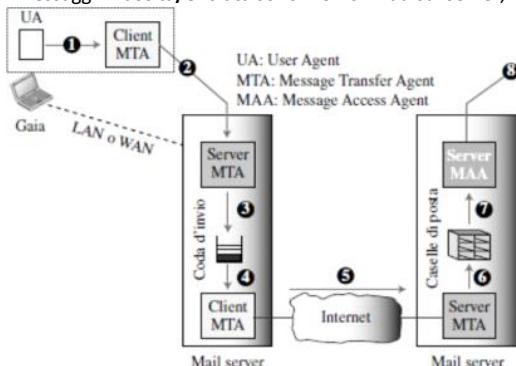
- Num. di 3 cifre: indica il codice della risposta
- Un testo: contiene parametri necessari o informazioni supplementari

Posta Elettronica

Ci sono tre componenti principali:

- **User Agent**: usato per scrivere/inviare/leggere un messaggio
- **Message Transfer Agent (MTA)**: usato per trasferire il messaggio su Internet, comunicano tramite:
 - **Server di posta**: caselle contenenti i messaggi in arrivo e **code** di messaggi da trasmettere
 - **Protocollo SMTP**: tra server di posta per inviare messaggi e tra mittente e il suo server di posta
- **Message Access Agent (MAA)**: usato per leggere la mail in arrivo

I messaggi in **uscita/entrata** sono memorizzati sul server, mentre quelli **da inviare** sono inviati al **MTA**



SMTP (Simple Mail Transfer Protocol)

È un protocollo che usa TCP su port 25 per trasferire i messaggi di posta tra client e server.

Avviene un trasferimento **diretto** tra client SMTP al server SMTP, diviso in tre fasi:

- 1) **Handshaking**: il client indica l'email del mittente e destinatario
- 2) **Trasferimento messaggi**: la connessione è **persistente** quindi può inviare più messaggi
- 3) **Chiusura trasmissione**

Formato Messaggi

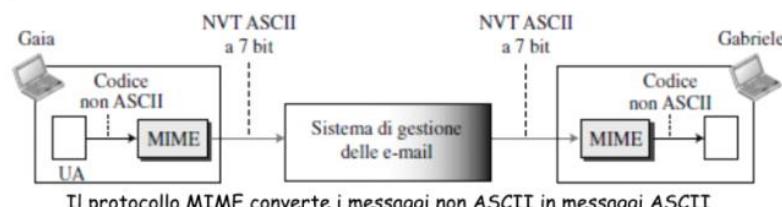
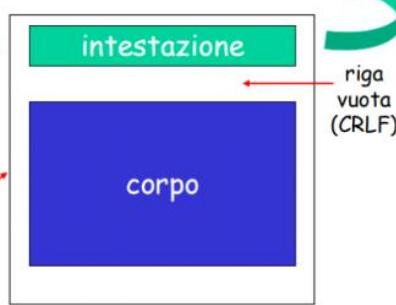
Formato dei messaggi di posta elettronica

To:	indirizzo di uno o più destinatari.
From:	indirizzo del mittente.
CC:	indirizzo di uno o più destinatari a cui si invia per conoscenza.
Bcc:	blind Cc: gli altri destinatari non sanno che anche lui riceve il messaggio.
Subject:	argomento del messaggio.
Sender:	chi materialmente effettua l'invio (ad es. nome della segretaria).

SMTP: protocollo per scambiare messaggi di posta elettronica

RFC 822: standard per il formato dei messaggi di testo:

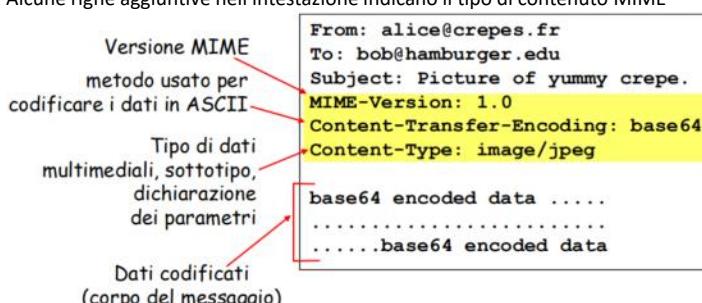
- Rigue di intestazione, per esempio
 - ❖ To:
 - ❖ From:
 - ❖ Subject:
 - differenti dai comandi SMTP!
- corpo
 - ❖ il "messaggio", soltanto caratteri ASCII



Estensioni di messaggi multimediali

Per inviare contenuti diversi dal testo ASCII si usano intestazioni aggiuntive usando le estensioni **MIME**.

Alcune righe aggiuntive nell'intestazione indicano il tipo di contenuto MIME



Formato del messaggio ricevuto

Un'altra classe di righe di intestazione viene inserita dal server di ricezione SMTP. Es: il server di ricezione aggiunge **Received**, specificando il nome del server che ha inviato (from) il messaggio e che lo ha ricevuto (by) e l'orario di ricezione

```
Received: from crepes.fr by hamburger.edu; 12 Oct 98 15:27:39 GMT
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg
base64 encoded data .....
.....base64 encoded data
```

Protocolli di Accesso alla Posta

SMTP è un protocollo **push** che consegna/memorizza i messaggi, per eseguire l'**operazione pull** l'user agent si affida ai protocolli di accesso alla posta:

- **POP3 (Post Office Protocol)**: apre una connessione TCP su porta 110 al server di posta per visualizzare i messaggi in modalita "scarica e cancella" (elimina i messaggi dopo la visualizzazione) o "scarica e mantieni" (mantiene i messaggi). Quando stabilisce la connessione richiede l' **autorizzazione** (tra agente e server), si recuperano i messaggi e alla chiusura della connessione vengono eliminati
- **IMAP (Internet Mail Access Protocol)**: più funzioni di POP3 e permette di manipolare i messaggi e salvarli in cartelle locali
- **HTTP**: es Gmail, Hotmail

Trasporto

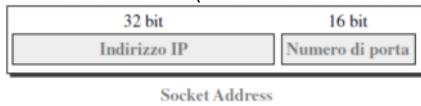
sabato 28 giugno 2025 16:29

Il liv. di rete invia i segmenti tra gli host e il liv. di **trasporto** traduce i segmenti e li invia ai **processi** dell'host (**comunicazione tra processi**)

Per poter stabilire una comunicazione tra due processi bisogna usare l'**indirizzo IP** per individuare l'**host** e il **num. di porta** per individuare il **processo**

- Host → IP
- Processo → num. porta

IP + porta = **socket address** (indirizzo che individua l'host e il processo tramite il **demultiplexing** del segmento ricevuto) (**una porta** può avere **più socket**)



Il campo num. di porta è di 16 bit con val. da 0 a 65535, e le porte da 1 a 1023 si chiamano "**well known port number**" (es HTTP 80, POP3 100, SMTP 25, ecc)

- 0: non usato
- 1-255: riservati per processi conosciuti
- 256-1023: riservati per altri processi
- 1024-65535: usati da app utente

Una comunicazione tra processi è identificata da una **coppia di socket address: locale** (mittente) e **remoto** (destinatario)

Socket Lato Client	Socket Lato Server
Serve un socket address locale (client) e remoto (server) per comunicare	Serve un socket address locale (server) e remoto (client) per comunicare
Socket address locale: <ul style="list-style-type: none">• l'IP del client• N° di porta è assegnato temporaneamente dall'OS	Socket address locale: <ul style="list-style-type: none">• IP del server• N° di porta noto (assegnato dal progettista)
Socket address remoto: <ul style="list-style-type: none">• IP del server fornito dal DNS• N° porta noto in base all'app (es: HTTP porta 80)	Socket address remoto: <ul style="list-style-type: none">• Il socket address locale definito del client che si connette <p>Il socket address remoto varia ad ogni interazione con client diversi</p>

I servizi di trasporto più usati sono:

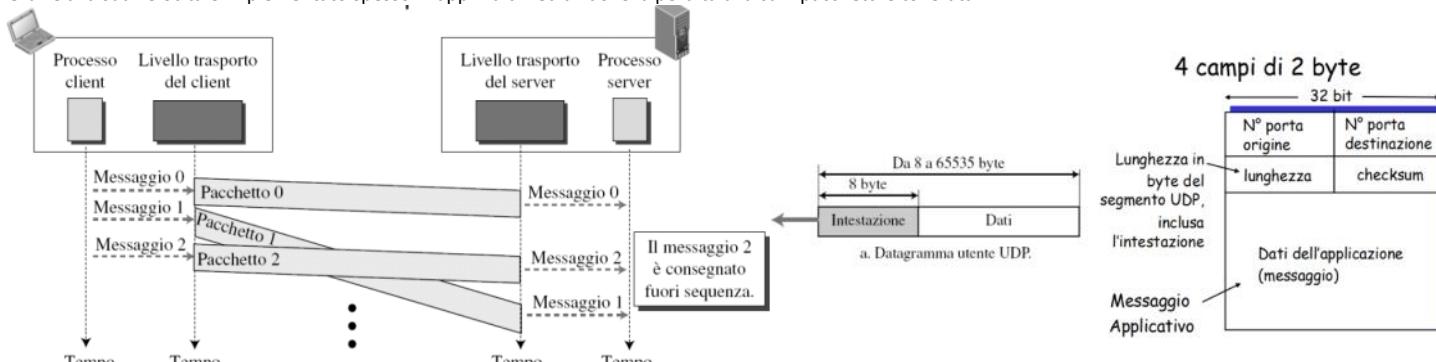
- **UDP: non affidabile** (ma più veloce) poiché due messaggi potrebbero arrivare in tempi diversi rispetto a quelli di invio
- **TCP: affidabile e orientato alla connessione** (richiede un setup iniziale), mantenendo l'ordine dei bit e controllando la quantità di dati inviati

UDP (User Datagram Protocol)

UDP non implementa una connessione e non fornisce controllo di flusso o congestione sui dati, però è **molto più veloce di TCP** perché non esegue questi controlli

I datagrammi inviati **non sono numerati**, quindi possono arrivare al destinatario in **ordine diverso** da quello di partenza.

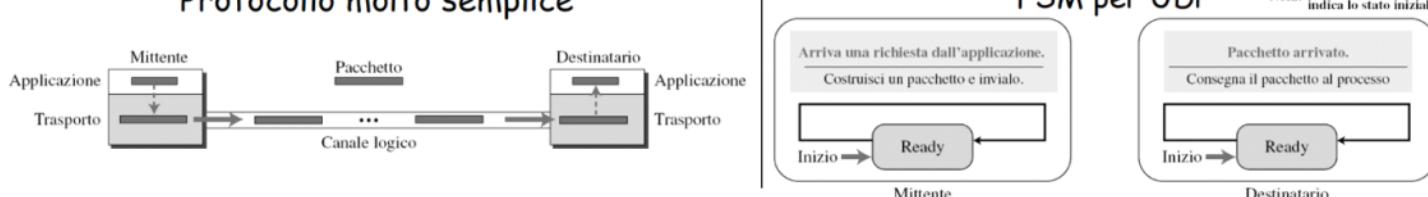
Grazie alla sua velocità è implementato spesso in app multimediali dove la perdita di alcuni pacchetti è tollerata



Rappresentazione con FSM (Finite State Machine)

Il comportamento di un protocollo si può rappresentare da un **automa a stati finiti**, che rimane in uno stato finché **non avviene un evento** che modifica lo stato

Protocollo molto semplice



TCP (Transmission Control Protocol)

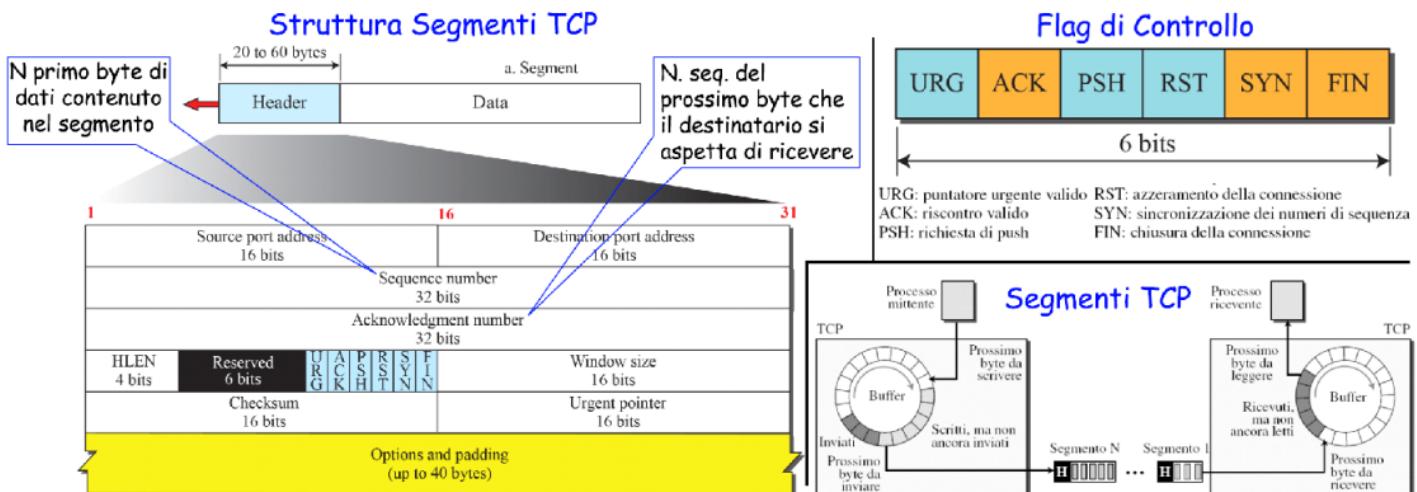
TCP è un protocollo:

- **Affidabile (controllo errori)**
- **Orientato al flusso di dati e alla connessione**
- **Esegue il controllo del flusso e della congestione**

TCP riceve uno **stream** di byte dal processo mittente che deve raggruppare in un certo num. di byte in **segmenti**, aggiungere un **intestazione** e passarlo al liv. di rete

Convenzione:

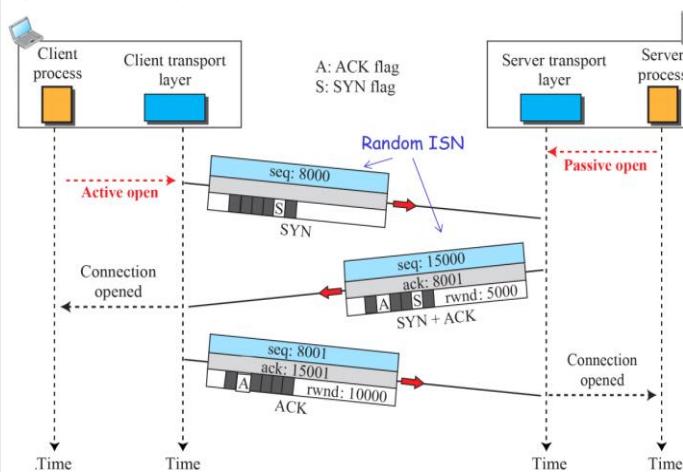
- Ogni pacchetto ha un **num. di sequenza** nell'header (se usiamo m bit → num. da 0 a $2^m - 1$) usato per capire la sequenza dei pacchetti in arrivo.
Il numero indica il "numero" del **primo byte** di segmento nel flusso di byte
- Il num. di **riscontro** (ACK) indica il num. di **sequenza del prossimo byte atteso**
Es: se il destinatario ha ricevuto correttamente il pacchetto 0, invia un riscontro con val 1 (ACK 1), cioè il prossimo pacchetto atteso ha num. di sequenza 1
- **ACK cumulativo (AckNo):** tutti i pacchetti fino al num. di sequenza indicato nell'ack sono **stati ricevuti correttamente**
Es: AckNo = 7 → i pacchetti fino a 6 sono stati ricevuti e si attende il 7



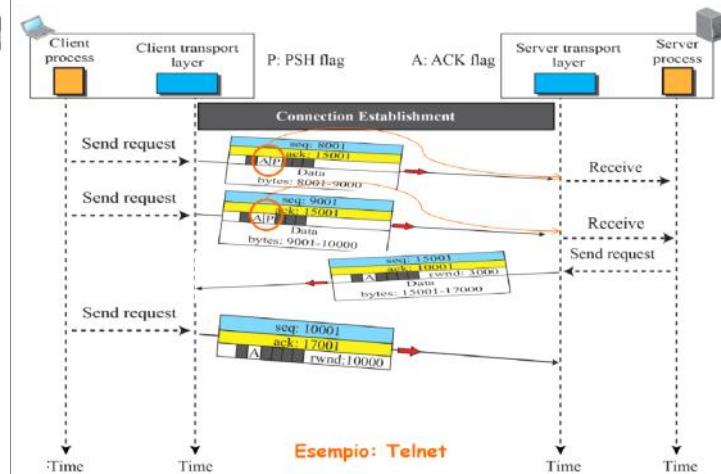
Connessione TCP

È un percorso virtuale tra mittente e destinatario, diviso in 1. apertura della connessione, 2. trasferimento dati, 3. chiusura della connessione

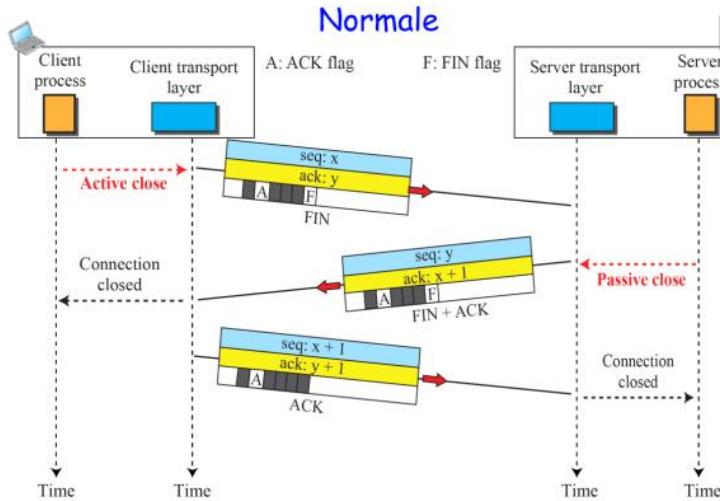
Apertura: 3 way handshake



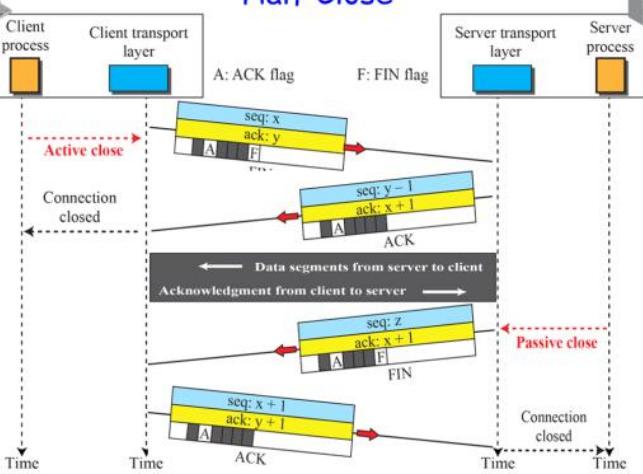
Trasferimento Dati



Chiusura della Connessione



Half Close



Meccanismi di Trasferimento Dati Affidabili

Convenzione:

- Finestra di invio:** concetto astratto che definisce una porzione di dim. x che indica quali num. di sequenza il mittente ha inviato/riscontrato. Si usano tre variabili: S_f (primo pacchetto non riscontrato), S_n (prossimo pacchetto da inviare nella finestra), S_{size} (dimensione della finestra)
- Finestra di ricezione:** come la finestra di invio, è una porzione di dim. x che indica quali num. di sequenza il ricevente è in ascolto (scarta quelli fuori sequenza) Si usano le variabili: R_n (prossimo pacchetto atteso) e R_{size} (dimensione della finestra)
- Pipelining:** il mittente ammette più pacchetti in transito, ancora da notificare (richiede una finestra di invio più grande)
- Piggybacking:** quando un pacchetto trasporta dati, può trasportare anche i riscontri relativi ai pacchetti ricevuti

Stop-and-Wait

Protocollo orientato alla connessione, controllo di flusso e errori, dove il mittente invia un solo pacchetto (finestra di invio/ricezione di dim. 1) e ne attende l'ack prima di spedire il successivo.

Se il pacchetto è:

- Corretto:** il destinatario invia un ACK
- Corrotto/perso:** il mittente non riceve ACK, scatta il timer e alla scadenza ritrasmette il pacchetto

Siccome la finestra è di dim. 1, usiamo 0 e 1 come num. di sequenza per i pacchetti. Abbiamo 3 casi quando il mittente invia un pacchetto:

1. Pacchetto arriva correttamente → scorre la finestra (da 0 a 1 o viceversa) e invia l'ACK per il prossimo pacchetto.
 2. Pacchetto perso o danneggiato → scade il timer e il mittente rinvia il pacchetto
 3. Pacchetto arriva correttamente (scorre la finestra e invia ACK) ma ACK perso → il pacchetto viene rinvia (scade il timer) ma il destinatario vede che il num. di sequenza è diverso, quindi scarta il pacchetto e rinvia l'ACK
- Però se il rate è elevato e il ritardo è consistente → stop-and-wait molto inefficiente

Go-back-N

È un protocollo con **pipelining e ack cumulativo** dove i num. di seq. sono calcolati in modulo 2^m ($m = \text{dim. "num di seq." in bit}$)

Finestra di Invio

La finestra di **invio** ha dim. $2^m - 1$ e può scorrere **una o più pos.** quando viene ricevuto un **AckNo $\geq S_f$** e **AckNo $< S_n$**

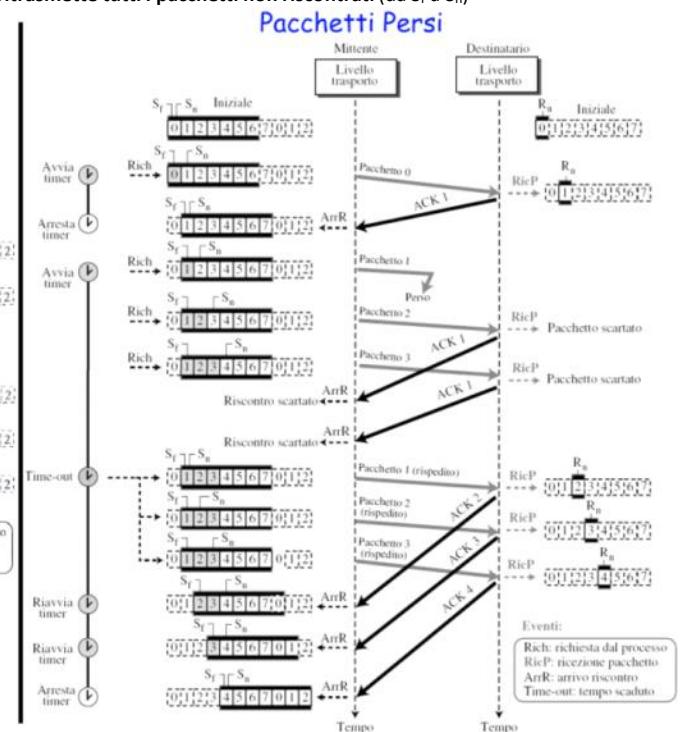
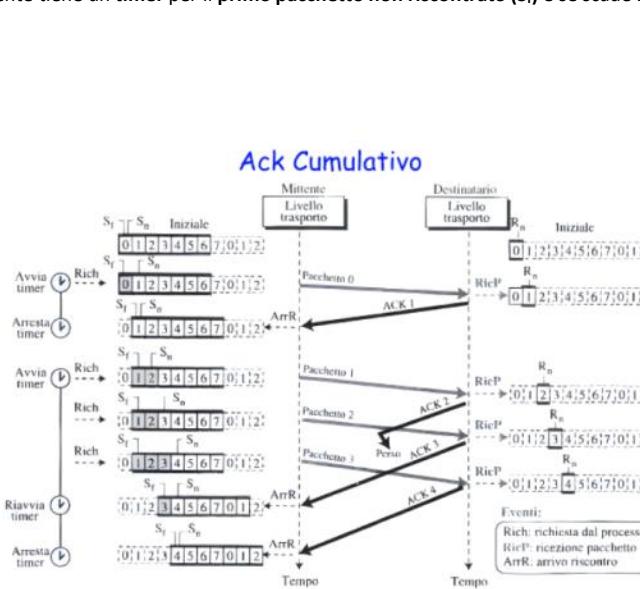


Finestra di Ricezione

La finestra di **ricezione** invece ha dim. 1, quindi il destinatario è in attesa di uno specifico pacchetto (scarta quelli fuori sequenza)

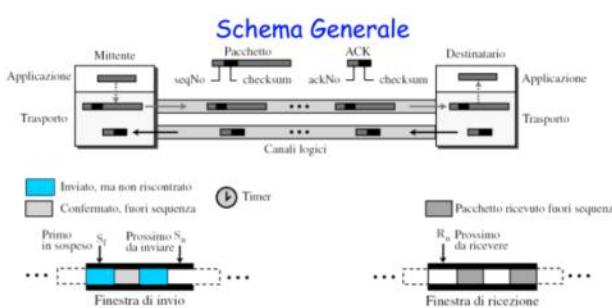


Il mittente tiene un **timer** per il **primo pacchetto non riscontrato (S_f)** e se scade **rtrasmette tutti i pacchetti non riscontrati** (da S_f a S_n)



Ripetizione Selettiva

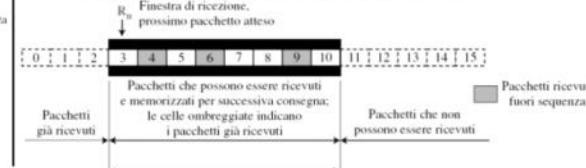
È un protocollo simile a Go-back-N ma invece di risedere tutti i successivi già inviati, **risedisce solo quelli non riscontrati** (mantiene un **timer per ogni pacchetto**) Il ricevente **accetta pacchetti fuori ordine** e invia **ACK specifici** per ogni pacchetto ricevuto. Utilizza le finestre di dim. 2^{m-1}

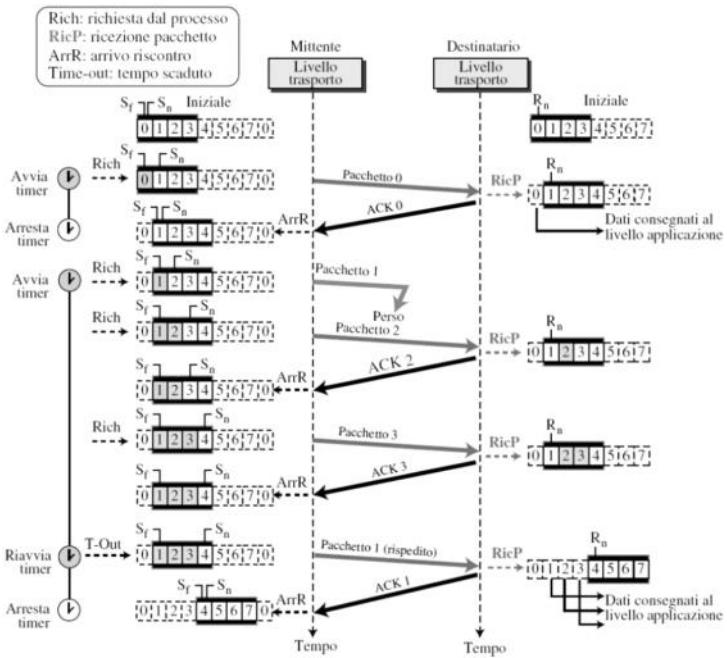


Finestra di Invio e Ricezione



Finestra di invio e ricezione hanno la stessa dimensione





Riassunto

- Stop-and-Wait:** no pipeline, finestra invio/ricezione dim. 1, num. seq = 0 e 1
Il mittente invia un solo pacchetto e ne attende l'ACK prima di spedire il successivo
- Go-back-N:** si pipeline, finestra invio dim. $2^m - 1$, finestra ricezione dim. 1, num. seq in modulo 2^m , timer per pacchetto più vecchio non riscontrato
Il mittente invia più pacchetti insieme e il ricevente ne accetta uno alla volta, se scade il timer rinvia tutti i pacchetti non riscontrati
- Ripetizione selettiva:** si pipeline, finestra invio/ricezione dim. 2^{m-1} , num. seq in modulo 2^m , timer per ogni pacchetto, no AckNo
Il mittente invia più pacchetti insieme (se scade il timer di un pacchetto lo rinvia) e il ricevente accetta anche pacchetti fuori ordine inviando ACK specifici

Meccanismo	Uso
Checksum	Per gestire errori nel canale
Acknowledgment	Per gestire errori nel canale
Numero di sequenza	Ack con errori Perdita pacchetti
Timeout	Perdita pacchetti
Finestra scorrevole, pipeling	Maggior utilizzo della rete

Gestione inaffidabilità della rete
Miglioramento prestazioni

Esercizio

Esercizio 1

Usando num. di seq. a 5 bit, qual è la dim. max delle finestre di invio e ricezione per ciascuno dei meccanismi visti?

- Stop-and-Wait:
 - Dim. Invio = 1
 - Dim. Ricezione = 1
- Go-back-N:
 - Dim. Invio = $2^5 - 1 = 31$
 - Dim. Ricezione = 1
- Ripetizione selettiva:
 - Dim. Invio = $2^{5-1} = 16$
 - Dim. Ricezione = $2^{5-1} = 16$

Esercizio 2

In una rete con Go-back-N con $m=3$ e dimensione della finestra di invio = 7, i val. delle var sono $S_f = 0$, $S_n = 4$ e $R_n = 2$.

Si ipotizzi che la rete non duplichi e non alteri l'ordine dei pacchetti

- Qual è il num. di seq. dei pacchetti in transito?
- Qual è il num. di riscontro dei pacchetti ack in transito?

Siccome $S_n = 4$ allora abbiamo inviato 4 pacchetti (0, 1, 2, 3) e poiché S_f è ancora a 0 significa che il mittente non ha ricevuto un singolo ack per i pacchetti inviati, quindi questi pacchetti sono in transito.

Poi siccome $R_n = 2$ allora vuol dire che sono arrivati in sequenza correttamente i primi 2 pacchetti (0, 1) quindi invia ACK2 per ricevere il prossimo pacchetto 2, quindi questo ack è in transito

TCP: Controllo Errori

Per una connessione **affidabile** il liv. di trasporto (TCP) implementa un **controllo degli errori** sui pacchetti tramite:

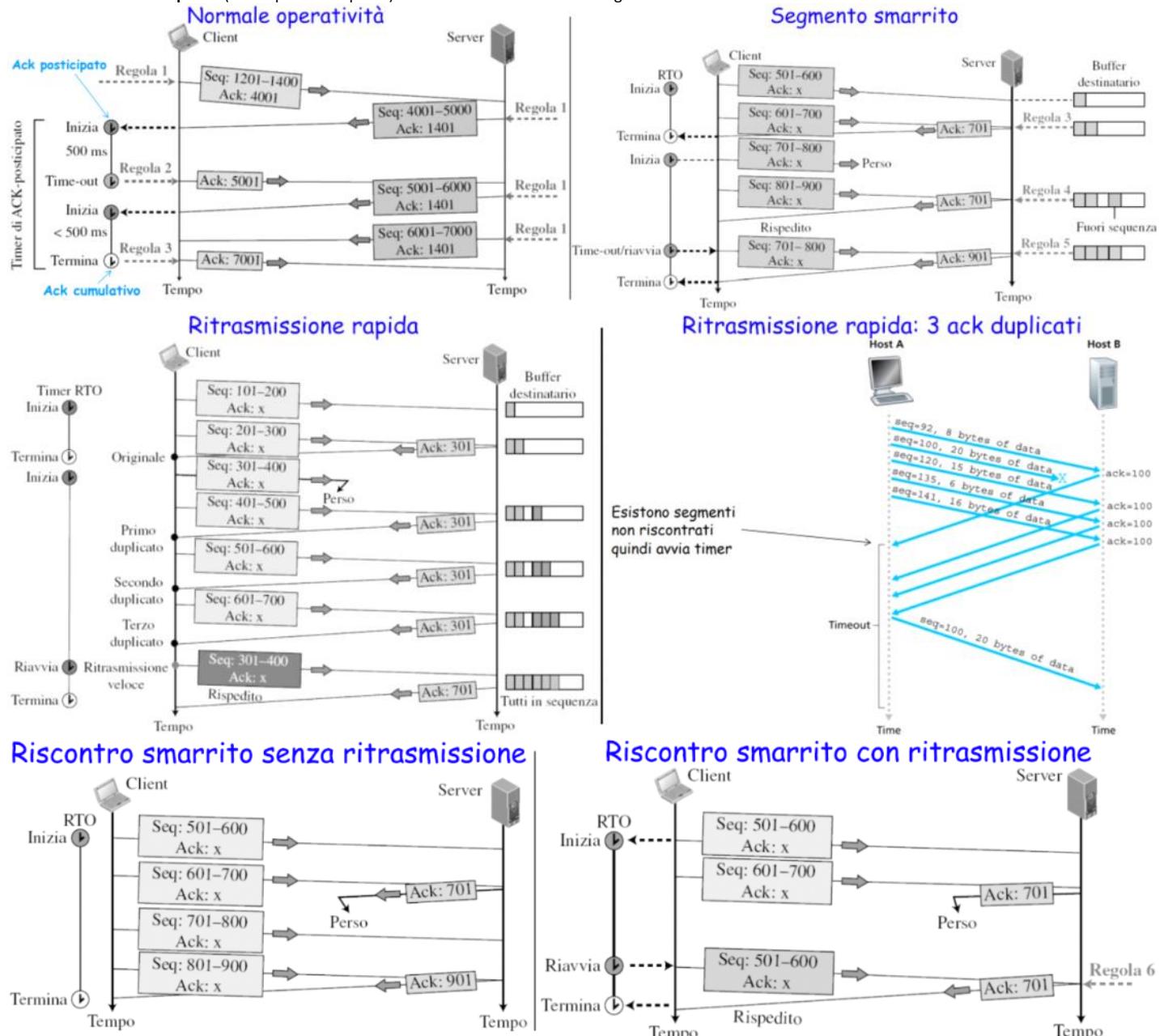
- Checksum:** scarta pacchetti corrotti
- ACK + timer ritrasmissione (RTO):** gestisce pacchetti persi o duplicati
- Ritrasmissione:** ritrasmette il segmento all'inizio della coda di spedizione

Rule	Evento	Azione
1	Arrivo segmento in ordine	Invia ACK (piggybacking) col prossimo segmento atteso
2	Arrivo segmento in ordine, ma i dati precedenti sono già stati riscontrati	ACK delayed. Attende max 500ms per il prossimo segmento per rispondere con un solo ACK cumulativo .
3	Arrivo segmento in ordine. Un altro segmento è in attesa di trasmissione	Invia subito un ACK cumulativo , riscontrando i segmenti ordinati

	dell'ACK (vedi precedente)	
4	Arrivo segmento fou ordine con num. seq. superiore a quello atteso. Viene rilevato un bucco	Invia immediatamente un ACK duplicato , indicando il num. seq. del byte perso (ritrasmissione rapida)
5	Arrivo di un segmento mancante (uno o più dei successivi è stato ricevuto)	Invia immediatamente un ACK per aggiornare il mittente sul flusso in ordine
6	Arrivo di un segmento duplicato	Invia immediatamente un ACK con num. seq. atteso

Se un segmento **non è riscontrato**:

1. **Scade il timer** → ritrasmette il segmento più vecchio non riscontrato e riavvia il timer
2. **Ricevuti 3 ACK duplicati** (indica pacchetto perso) → **ritrasmissione veloce** del segmento mancante



Riassunto Meccanismi

- **Pipeline** (approccio ibrido tra GBN e Ripetizione Selettiva): **non reinvia tutto** dopo un errore e **non gestisce ogni pacchetto separatamente**. Invia più segmenti senza aspettare un ACK uno per uno (**pipeline**) ma può **ritrasmettere solo il pacchetto perso**
- **Num. seq:** indica il **primo byte** del segmento
- **ACK:**
 - **Cumulativo:** conferma tutti i byte ricevuti in ordine fino a un certo punto (es. ACK 501 indica che si hanno ricevuti i byte fino al 500)
 - **Ritardato:** dopo aver ricevuto un segmento in ordine può **aspettare fino a 500ms** per vedere se arrivano i prossimi, così da inviare un **ACK cumulativo**
- **Timeout basato su RTT:** usa un solo timer per la **ritrasmissione**, associato al **segmento non riscontrato più vecchio** (non tutti), se riceve un **ACK parziale**, **riavvia il timer** sul segmento non confermato
- **Ritrasmissione:**
 - **Singola:** ritrasmette **solo il pacchetto perso** (non tutti quelli successivi)
 - **Rapida:** se riceve **3 ACK duplicati**, ritrasmette **subito** il pacchetto mancante, **senza aspettare il timeout (ritrasmissione)**

TCP: Controllo di Flusso

Il **controllo di flusso** serve a garantire che il **produttore non produce più velocemente di quanto il consumatore li elabora**.

Per il controllo si fa uso di **buffer** (aree di memoria temporanee) e il destinatario **invia segnali di controllo** in base alla **saturazione** dei buffer

- Se **velocità produzione mittente < velocità elaborazione mittente** → buffer liv. trasporto del **mittente è saturo** → segnala al liv. applicazione di sospendere l'invio e quando si libera spazio segnala di riprendere l'invio
- Se **velocità produzione mittente > velocità elaborazione destinatario** → buffer liv. trasporto del **destinatario è saturo** → segnala al liv. trasporto del mittente di

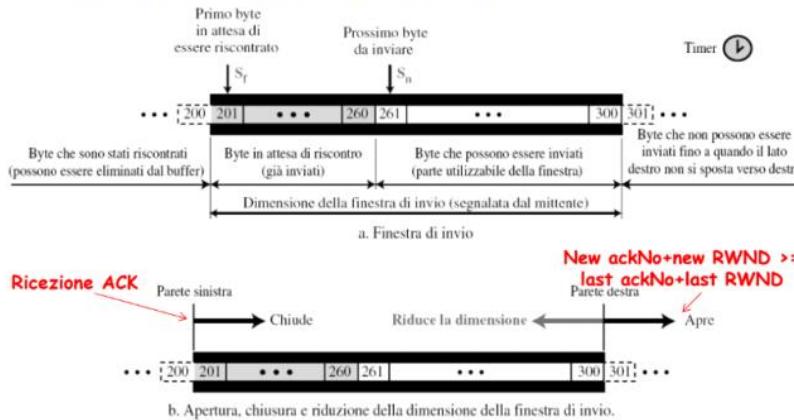
sospendere l'invio di messaggi e quando si libera spazio segnala di riprendere l'invio

Si usa il campo nell'header **RcvWindow (RWND)** per comunicare quanto spazio libero ha il destinatario nel buffer (feedback esplicito)

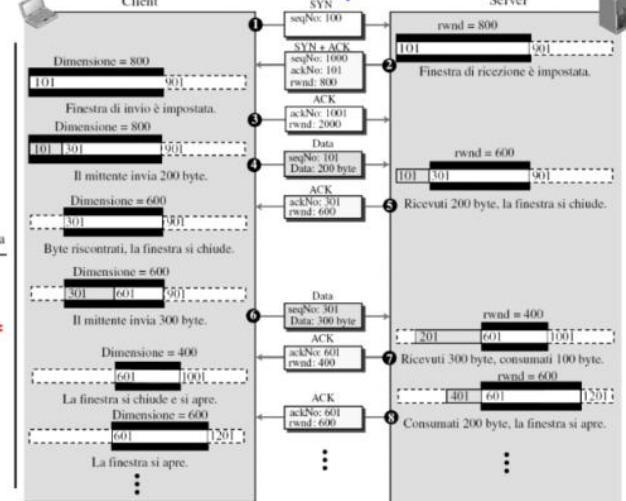


Finestra di Invio

L'apertura, chiusura e riduzione della finestra di invio sono controllate dal destinatario



Esempio



TCP: Controllo della Congestione

La **congestione** avviene quando il num. di pacchetti inviati alla rete **supera la sua capacità** di elaborazione, quindi i buffer dei router/switch si riempiono causando perdita di pacchetti e lunghi ritardi. Quindi il controllo della congestione regola il traffico nella **rete di mezzo** (es. router). I due approcci principali sono:

- **End-to-End** (usato da TCP): congestione dedotta da **perdita di pacchetti** e ritardi nei sistemi terminali
- **Assistito dalla rete**: i router inviano **feedback** ai terminali comunicando la freq. max di trasmissione consentita

Finestra di Congestione

Si usa la var. **CWND** (Congestion Window) per regolare la quantità di dati che il mittente può inviare in base allo **stato di rete**.

Si usa insieme a **RWND** (che regola la congestione del ricevente) per indicare la finestra di invio effettiva del mittente:

$$\text{Finestra invio} = \min(\text{CWND}, \text{RWND})$$

Rilevare la Congestione

Gli **ACK duplicati** e **timeout** sono segno di **perdita** (congestione), quindi TCP regola dinamicamente il flusso di dati:

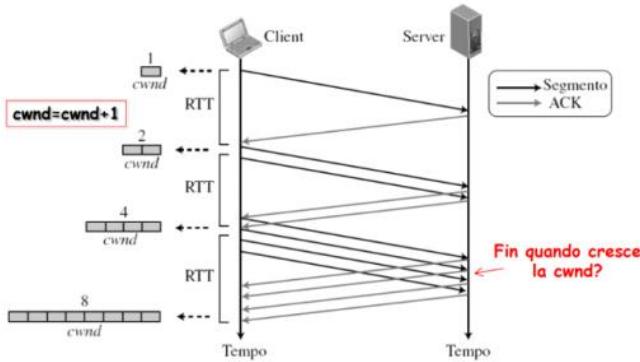
- Se **ACK regolari** → **aumenta CWND** (può incrementare la quantità di segmenti inviati)
- Se **perdita** → **riduce CWND** (deve ridurre i dati inviati senza riscontri)

Controllo della Congestione

L'algoritmo di controllo si basa su 3 componenti:

1. **Slow Start: crescita esponenziale** ($\text{CWND} += 1$) (inizio = 2^0 , 1 RTT = 2^1 , 2 RTT = 2^2 , ...)
 - CWND inizia a **1 MSS** (dim. max del segmento)
 - Incrementa CWND di 1 MSS per ogni ACK fino ad una soglia (**ssthresh**)
2. **Congestion Avoidance: crescita lineare dopo perdita** ($\text{CWND} = i + 1$) (inizio = i , 1 RTT = $i + 1$, 2 RTT = $i + 2$, ...)
 - Se viene perso un pacchetto (si pone $\text{ssthresh} = \text{CWND}/2$) o se $\text{CWND} \geq \text{ssthresh}$, inizia congestion avoidance
 - Incrementa CWND di 1 MSS per ogni RTT, fino ad una **perdita** (ACK duplicati o timeout)
 - Perdita → $\text{ssthresh} = \text{CWND}/2$ e $\text{CWND} = 1$ (torna a slow start)
3. **Fast Recovery (solo in TCP avanzati): crescita esponenziale dopo perdita leggera** (ACK duplicati)
 - Interpreta 3 ACK duplicati come **congestione leggera** (manca un pacchetto precedente es ACK 40, ACK 40, ACK 40) e invece di tornare a slow start (come congestion avoidance) continua con crescita esponenziale per ogni ACK duplicato (ricevuto pacchetto oltre quello mancante)
 - Quando riceve un **ACK nuovo maggiore** (es invece di ACK 40 (quello mancante) arriva ACK 44) allora è un **ACK cumulativo** quindi il pacchetto mancante è **finalmente arrivato**, quindi è un **ACK valido**

Incremento Esponenziale

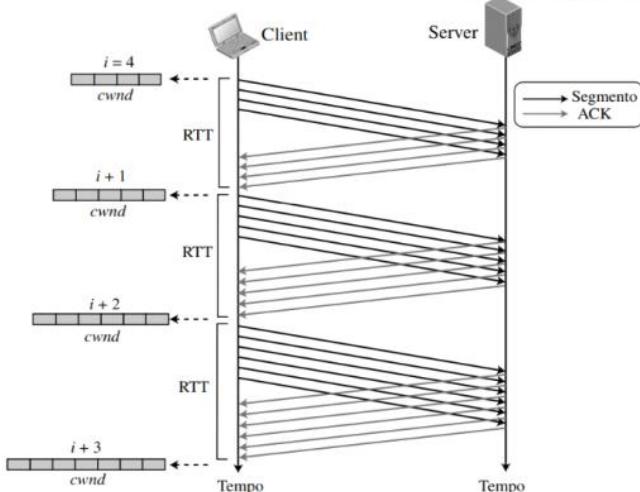


Se arriva un riscontro, $cwnd = cwnd + 1$

Inizio	\rightarrow	$cwnd = 1 \rightarrow 2^0$
Dopo 1 RTT	\rightarrow	$cwnd = cwnd + 1 = 1 + 1 = 2 \rightarrow 2^1$
Dopo 2 RTT	\rightarrow	$cwnd = cwnd + 2 = 2 + 2 = 4 \rightarrow 2^2$
Dopo 3 RTT	\rightarrow	$cwnd = cwnd + 4 = 4 + 4 = 8 \rightarrow 2^3$

La dimensione della finestra di congestione nell'algoritmo slow start viene aumentata esponenzialmente fino al raggiungimento di una soglia (ssthresh).

Incremento Lineare



Se arriva un riscontro, $cwnd = cwnd + (1 / cwnd)$

Inizio	\rightarrow	$cwnd = i$
Dopo 1 RTT	\rightarrow	$cwnd = i + 1$
Dopo 2 RTT	\rightarrow	$cwnd = i + 2$
Dopo 3 RTT	\rightarrow	$cwnd = i + 3$

Con l'algoritmo congestion avoidance, la dimensione della finestra di congestione viene aumentata linearmente fino alla rilevazione della congestione.

Riassunto

- Finestra di congestione = $\min(\text{CWND}, \text{RWND})$ (limita la freq. di invio del traffico)
 - **CWND**: regola la congestione di rete
 - **RWND**: regola la congestione del ricevente
- Rilevare la congestione: rileva la congestione in base alle **perdite** (ACK duplicati o timeout)
 - ACK regolari \rightarrow incrementa **CWND** (può inviare più dati)
 - perdite \rightarrow riduce **CWND** (meno dati inviati)
- Controllo della congestione: usa 3 componenti:
 - Slow Start: CWND cresce **esponenzialmente** (1 MSS per ACK \rightarrow inizio = 2^0 , 1 RTT = 2^1 , 2 RTT = 2^2 , ...) fino a ssthresh.
 - Congestion Avoidance: se $\text{CWND} \geq \text{ssthresh}$ cresce **linearmemente** (1 MSS per RTT \rightarrow inizio = i , 1 RTT = $i+1$, 2 RTT = $i+2$, ...)
 - Fast Recovery: se arrivano 3 ACK duplicati (congestione leggera) continua con **crescita esponenziale**, invece di tornare a slow start
se arriva un **nuovo ACK** ritorniamo a Congestion Avoidance con $\text{CWND} = \text{ssthresh}$
se **timeout** \rightarrow $\text{ssthresh} = \text{CWND}/2$ e $\text{CWND} = 1$ (torna slow start)

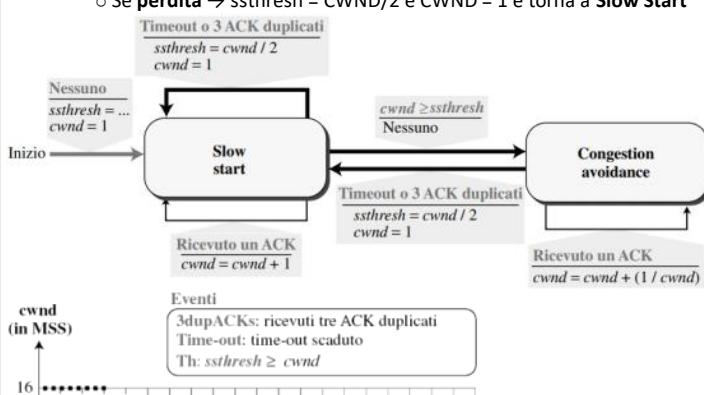
Versioni TCP

Le versioni più comuni di TCP col controllo della congestione sono **TCP Tahoe** e **TCP Reno** (simile a Tahoe ma aggiunge il **fast recovery**).

TCP Tahoe

Considera timeout e 3 ack duplicati come congestione e riparte da **Slow Start** con 1 con $\text{ssthresh} = \text{CWND}/2$ e $\text{CWND} = 1$

- **Slow Start**: crescita **esponenziale** ($1, 2, 4, 8, \dots$)
 - Se **perdita** \rightarrow $\text{ssthresh} = \text{CWND}/2$ e $\text{CWND} = 1$
 - Se $\text{CWND} \geq \text{ssthresh} \rightarrow$ **Congestion Avoidance**
- **Congestion Avoidance**: crescita **lineare** ($i+1, i+2, i+3, \dots$)
 - Se **perdita** \rightarrow $\text{ssthresh} = \text{CWND}/2$ e $\text{CWND} = 1$ e torna a **Slow Start**

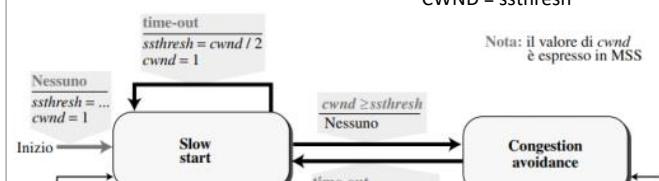


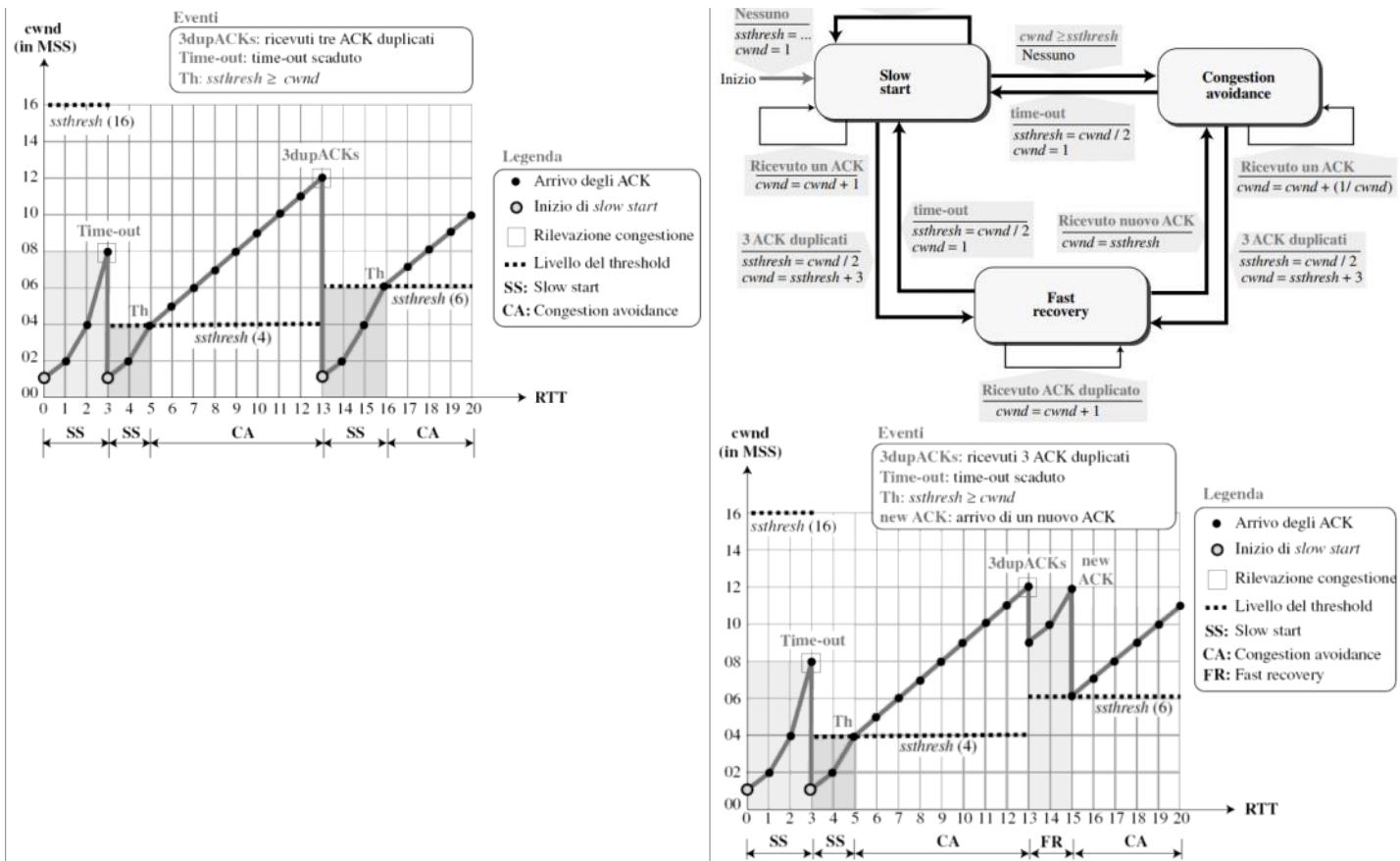
TCP Reno

3 ACK duplicati (pacchetto precedente mancante): **congestione lieve** \rightarrow **Fast Recovery** con $\text{ssthresh} = \text{CWND}/2$ e $\text{CWND} = \text{ssthresh} + 3$

Timeout: **congestione importante** \rightarrow riparte da **Slow Start** con $\text{ssthresh} = \text{CWND}/2$ e $\text{CWND} = 1$

- **Slow Start**: crescita **esponenziale** ($+1, +2, +4, +8, \dots$)
 - Se **timeout** \rightarrow riparte **Slow Start**
 - Se $\text{CWND} \geq \text{ssthresh} \rightarrow$ **Congestion Avoidance**
 - Se 3 ACK dup \rightarrow **Fast Recovery**
- **Congestion Avoidance**: crescita **lineare** ($i+1, i+2, i+3, \dots$)
 - Se **timeout** \rightarrow torna a **Slow Start**
 - Se 3 ACK duplicati \rightarrow **Fast Recovery**
- **Fast Recovery**: crescita **esponenziale** ($+1, +2, +4, +8, \dots$)
 - Se **timeout** \rightarrow riparte **Slow Start**
 - Se **nuovo ACK** (ACK cumulativo) \rightarrow **Congestion Avoidance** e $\text{CWND} = \text{ssthresh}$





Comparazione

Evento	Tahoe	Reno
Inizio	CWND = 1, Slow Start	CWND = 1, Slow Start
ACK regolare	Esponenziale (SS), Lineare (CA)	Esponenziale (SS e FR), Lineare (CA)
Timeout	Slow Start con $ssthresh = CWND/2$, CWND = 1	Slow Start con $ssthresh = CWND/2$, CWND = 1
3 ACK duplicati	Slow Start con $ssthresh = CWND/2$, CWND = 1	Fast Recovery con $ssthresh = CWND/2$, CWND = $ssthresh + 3$
Fast Recovery		CWND += 1 per ogni ACK dup ACK nuovo → Congestion Avoidance con CWND = $ssthresh$

Stima del RTT

Siccome il val. del timeout $\geq RTT$, bisogna stimare l'RTT poiché varia (cioè il tempo max da aspettare per ricevere un ACK prima di scattare timeout)
Si usa un val. stimato EstimatedRTT aggiornato in base al val di SampleRTT cioè un RTT che ignora le ritrasmissioni e si usa un solo SampleRTT per più segmenti inviati. Però SampleRTT varia in base alla congestione, quindi si esegue una media mobile ponderata influenzata dalle misure passate

- 1° val: $EstimatedRTT_0 = SampleRTT_0$
 - Val. successivi: $EstimatedRTT_{t+1} = (1 - \alpha) * EstimatedRTT_t + \alpha * SampleRTT_{t+1}$
- (val tipico $\alpha = 0.125$, quindi $(1-\alpha) = 0.875$)

Si usa un margine di sicurezza per calcolare la deviazione tra SampleRTT e EstimatedRTT:

- 1° val: $DevRTT_0 = SampleRTT_0 / 2$
 - Val. successivi: $DevRTT_{t+1} = (1 - \beta) * DevRTT_t + \beta * |SampleRTT_{t+1} - EstimatedRTT_{t+1}|$
- (val tipico $\beta = 0.25$, quindi $(1-\beta) = 0.75$)

Ora possiamo impostare il timeout:

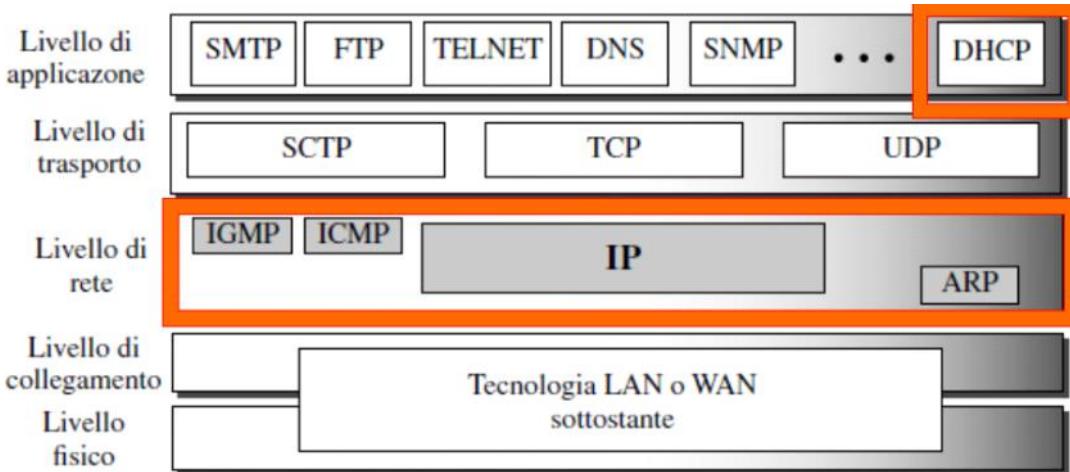
- Val. iniziale: 1 sec
- Se timeout → raddoppia
- Se nuovo ACK → aggiorna EstimatedRTT + margine: $TimeoutInterval = EstimatedRTT + 4 * DevRTT$

Riassunto

- SampleRTT: tempo reale misurato $\rightarrow SampleRTT = RTT$
- EstimatedRTT: stima stimata del RTT $\rightarrow EstimatedRTT_{t+1} = (1 - \alpha) * EstimatedRTT_t + \alpha * SampleRTT_{t+1}$
- DevRTT: deviazione tra RTT reale e stimato $\rightarrow DevRTT_{t+1} = (1 - \beta) * DevRTT_t + \beta * |SampleRTT_{t+1} - EstimatedRTT_{t+1}|$
- Timeout: tempo massimo per ricevere ACK $\rightarrow TimeoutInterval = EstimatedRTT + 4 * DevRTT$

Rete

domenica 4 maggio 2025 20:43



Al livello di rete abbiamo:

- **Router** (processing di milioni di flussi di pacchetti)
- **Algoritmi di routing:** si usano gli algoritmi sui grafi (le reti) per trovare i **percorsi a costo minimo**
 - **Approccio distribuito**
 - **Approccio centralizzato**
 - **Routing gerarchico (ISP e AS):** ogni AS è indipendente
- **Protocolli di supporto:**
 - **IP:** Internet Protocol v4 (anche v6)
 - **IGMP:** Internet Group Management Protocol (multicasting)
 - **ICMP:** Internet Control Message Protocol (gestione errori)
 - **ARP:** Address Resolution Protocol (associazione indirizzo IP - ind collegamento)

Routing

Vettore Distanza (DV)

Il **vettore distanza** è un array che rappresenta l'albero a costo minimo dalla radice verso gli altri dispositivi di rete, fornendo solo i costi minimi per arrivare alle varie destinazioni. I vantaggi sono:

- **Iterativo, asincrono:** ogni iterazione è causata dal cambio del costo di un collegamento o dalla ricezione di un DV aggiornato dal vicino
- **Distribuito:** Ogni nodo aggiorna i suoi vicini **solo** se il suo DV cambia

Costruzione vettore distanza:

1. Ogni dispositivo crea un DV iniziale con solo le distanze ai suoi **dispositivi vicini**.
2. Invia poi una copia ai vicini e quando un vicino x riceve un DV, usa la formula di **Bellman-Ford** per aggiornare il suo DV:
$$D_x(y) = \min_v \{c(x, v) + D_v(y)\} \text{ per ciascun nodo } y \text{ in } N$$

Se il suo DV è cambiato, allora manda una copia ai suoi vicini, che aggiorneranno a loro volta il loro DV

Quindi ogni dispositivo:

1. **Attende** (un DV dal vicino)
2. **Effettua l'aggiornamento**
3. Se il DV cambia, **notifica** i vicini e ricomincia dal punto 1

Siccome potrebbe crearsi un ciclo, dove due dispositivi si aggiornano a vicenda in loop, sul costo per arrivare ad un nodo, usiamo:

Split Horizon	Poisoned Reverse
Ciascun dispositivo invia solo una parte della sua tabella. Se il disp. B ritiene che il percorso migliore passa per A, NON deve inviare questa informazione ad A (siccome arriva da A e quindi la conosce già)	Si pone a ∞ il val del costo del percorso che passa attraverso il vicino a cui si invia il vettore

RIP

Protocollo a **vettore distanza** con distanza **misurata in hop** (max = 15, 16 = ∞). I router RIP si scambiano periodicamente la tabella di routing per aggiornarsi sulle reti raggiungibili. Se un router apprende da un altro che riesce a raggiungere una rete con N hop, allora la considera raggiungibile con N+1 hop.

Messaggi RIP	Timer RIP:
<ul style="list-style-type: none">• Request: inviato da nuovi router per info di routing o per diagnostica• Response: inviato in risposta ad una request o ogni 30 secondi	<ul style="list-style-type: none">• Periodico: invia aggiornamenti ogni 25/35 sec• Scadenza: se scade il timer (180 sec) di un percorso senza ricevere aggiornamenti, il suo costo è impostato a 16• Garbage collection: rimuove percorsi scaduti dopo 120 sec annunciando agli altri router il percorso non valido con costo 16

Guasto collegamento: Se un router non riceve notizie dal vicino per 180 sec, il percorso è considerato guasto, allora il router:

- 1) Modifica la tabella d'istradamento
- 2) La invia ai suoi vicini
- 3) I vicini inviano nuovi messaggi
- 4) L'utilizzo della **poisoned reverse** evita i cicli

Caratteristiche RIP:

- **Split horizon con Poisoned reverse:** Evita cicli mettendo a 16 il costo della rotta che passa dal vicino a cui si manda la risposta
- **Triggered updates:** quando cambia un percorso si inviano immediatamente le info ai vicini senza aspettare il timeout, riducendo il tempo di convergenza
- **Hold-down:** Alla info di un percorso non più valido, inizia un timer e ogni risposta che arriva da quella rotta entro il timeout vengono ignorati

Implementazione RIP: Implementato come app sopra UDP porta 520 come processo **routed** (route daemon) che mantiene le info d'istradamento e scambia

messaggi con i processi **routed** dei router vicini. Siccome è a liv. app, invia/riceve su socket standard e usa protocollo di trasporto standard

Protocollo OSPF

Esempio Navigazione Web

lunedì 24 febbraio 2025 17:16

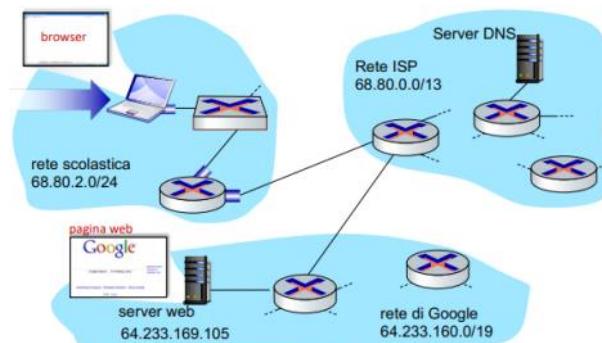
Programma

- Capacità e prestazioni delle reti
- Livello di Applicazione
 - Applicazione Web, Protocollo HTTP, Cookie
 - Web Caching. Protocollo DNS
 - Protocollo FTP. Protocolli di posta elettronica
- Livello di trasporto
 - Servizi del livello di trasporto. Protocollo UDP. Protocollo TCP
 - Interfaccia Socket
- Livello network
 - Protocollo IP. DHCP, NAT, Forwarding, ICMP
 - Algoritmi di Routing, Protocollo RIP, OSPF, BGP
- Livello di collegamento
 - Protocolli MAC. Protocollo ARP, Ethernet, switch, VLAN, PPP
 - LAN wireless Protocollo CDMA, Bluetooth, RFID
- Sicurezza delle reti

Collegamento

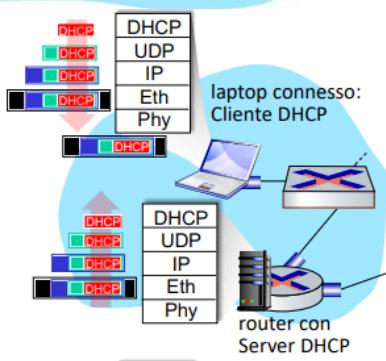
Scenario:

- un client si collega alla rete scolastica
- richiede la pagina web: www.google.com



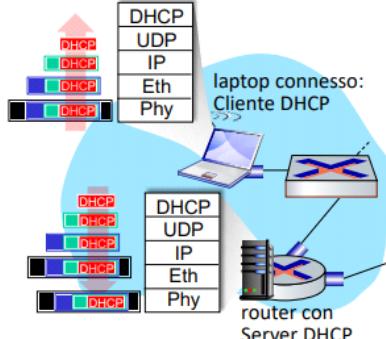
Richiesta su LAN

- Il laptop che si connette deve ottenere il proprio **indirizzo IP**, l'indirizzo del **router first-hop** e del **server DNS**: usiamo **DHCP**
- Richiesta **DHCP** **incapsulata in UDP**, **incapsulata in IP**, **incapsulata in 802.3 Ethernet**
- **Broadcast** frame Ethernet (dest: FFFFFFFFFFFF) su LAN, ricevuta dal **router** con server **DHCP**
- **Demux** da Ethernet > IP > UDP > DHCP



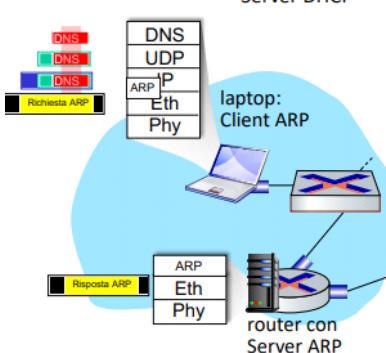
Ricevimento dell'ACK DHCP

- Il server **DHCP** manda l'**ACK DHCP** contenente l'indirizzo IP del **client**, l'IP del **router first-hop**, il nome e l'IP del **server DNS**
- Incapsulamento sul server DHCP, frame inoltrato (**switch learning**) tramite LAN, **demultiplexing sul client**
- Il client DHCP riceve la **risposta DHCP ACK**

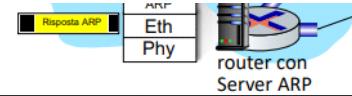


Query ARP (prima del DNS, prima dell'HTTP)

- Prima di inviare la richiesta **HTTP**, è necessario l'IP **google.com: DNS**
- **Query DNS** creata, incapsulata in UDP > IP > Eth. Per inviare frame al router, è necessario l'indirizzo **MAC** dell'interfaccia del router: **ARP**
- **Query ARP**, ricevuta dal router, che risponde con una **risposta ARP** fornendo l'indirizzo MAC dell'interfaccia del router
- Il client ora conosce l'indirizzo **MAC del first-hop router**, quindi ora può inviare un frame contenente una **query DNS**

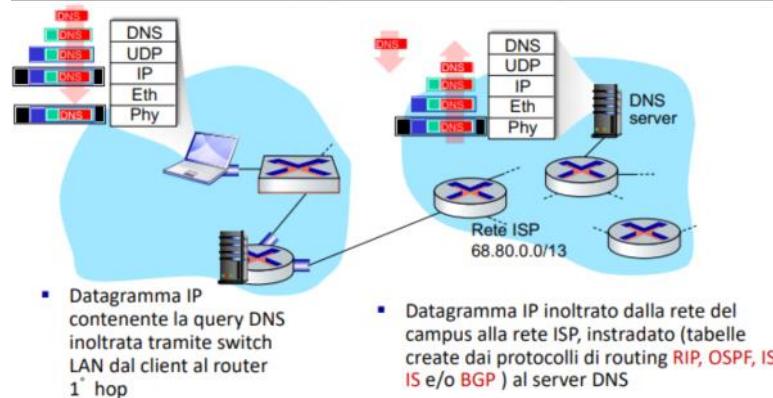


- Il client ora conosce l'indirizzo **MAC del first-hop router**, quindi ora può inviare un frame contenente una **query DNS**



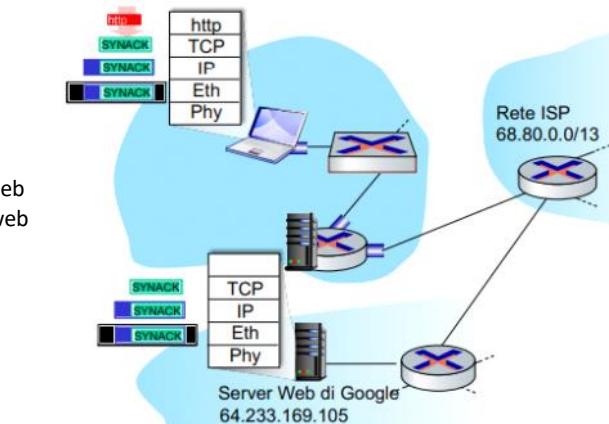
DNS

- Demuxed al DNS**
- Il DNS risponde al client con l'IP di www.google.com



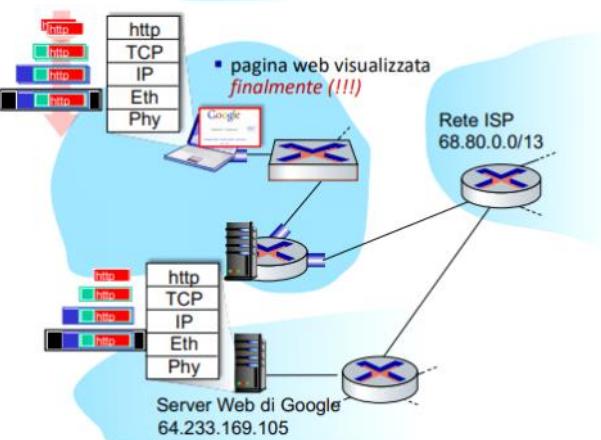
Connessione TCP che trasporta HTTP

- Per inviare la richiesta **HTTP**, il client deve aprire il **socket TCP** con il server web
- Segmento **TCP SYN** (step 1 nell'**handshake TCP a 3 vie**) instradato al server web
- Il server **Web** risponde con **TCP SYNACK** (step 2 nell'**handshake TCP a 3 vie**)
- Connessione TCP stabilita!**



Richiesta/Risposta HTTP

- Richiesta HTTP** inviata al socket **TCP**
- Datagramma IP** contenente la richiesta HTTP instradata a www.google.com
- Il server **Web** risponde con una **risposta HTTP** (contenente la pagina web)
- Datagramma IP** contenente la risposta HTTP reinstradata al client



Struttura di Internet e Link

sabato 1 marzo 2025 16:11

La rete è composta da dispositivi in grado di **scambiarsi informazioni**, tra cui sistemi terminali (end system, ad es computer, server) e dispositivi di **interconnessione**.
Sistemi Terminali

- **Host**: macchina dedicata ad eseguire applicazioni (Es. computer, cellulare, ...)
- **Server**: macchina con elevate prestazioni che esegue programmi che offrono servizi ad applicazioni utente (es siti web)

Dispositivi di Interconnessione

I dispositivi di interconnessione **rigenerano/modificano/inoltrano** il segnale che ricevono:

- **Router**: dispositivi che collegano una rete ad altre reti
- **Switch**: collegano più end system a livello locale

Collegamenti di Comunicazione

I dispositivi di rete sono collegati usando mezzi trasmissivi **cablati** (rame, fibra) o **wireless** (onde radio, satellite) chiamati **link (collegamenti)**

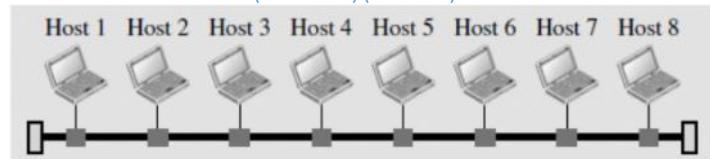
Cablati	Wireless
Mezzi fisici tra un trasmittente e ricevente <ul style="list-style-type: none">• Ethernet: 8 cavi rame• Cavo coassiale: due cavi rame<ul style="list-style-type: none">◦ Usato per cablatura locale ad alta velocità◦ Molto resistente alle interferenze• Fibra ottica:<ul style="list-style-type: none">◦ Materiale che conduce impulsi di luce◦ Alta frequenza trasmissiva (da 10 a 100 Gbps)◦ Basso tasso di errore, ripetitori distanziati, immune a interferenze radio	Mezzi ad onda libera dove i segnali si propagano nell'atmosfera <ul style="list-style-type: none">• Trasportano segnali nello spettro elettromagnetico• Non richiede installazione di cavi fisici• Effetti da interferenze, riflessione e ostruzione da ostacoli Tipi di canali radio : <ul style="list-style-type: none">• Microonde terrestri (es. canali fino a 45 Mbps)• LAN (es. WiFi-IEEE802.11) da 11 a 54 Mbps• Wide-area (es cellulari), es 3G: ~ 1 Mbps• Satellitari canali fino a 45 Mbps con ritardo punto-punto di 270 msec

Scale	Type	Example
Vicinity	PAN (Personal Area Network)	Bluetooth (e.g., headset)
Building	LAN (Local Area Network)	WiFi, Ethernet
City	MAN (Metropolitan Area Network)	Cable, DSL
Country	WAN (Wide Area Network)	Large ISP
Planet	The Internet (network of all networks)	The Internet!

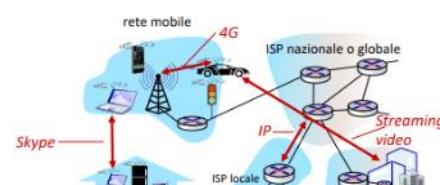
LAN: Local Area Network

Rete **privata** che collega i sistemi terminali in un singolo ufficio (azienda, università). Ogni sistema terminale ha un indirizzo che lo **identifica univocamente** nella rete

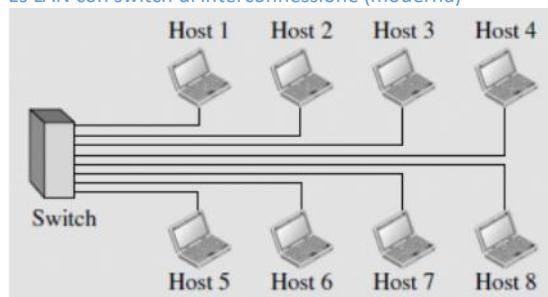
Es LAN con cavo condiviso (broadcast) (obsoleta)



- Il pacchetto inviato da un dispositivo viene **ricevuto da tutti gli altri**
- Solo il destinatario elabora il pacchetto, tutti gli altri lo ignorano



Es LAN con switch di interconnessione (moderna)



- Ogni dispositivo è collegato allo **switch**
- Lo switch riconosce l'**indirizzo di destinazione** e invia il pacchetto solo al destinatario **senza** inviarlo agli altri dispositivi
- Lo switch **riduce il traffico** nella LAN e consente a più coppie di dispositivi di comunicare contemporaneamente fra di loro

WAN: Wide Area Network

Rete geografica - interconnessione di dispositivi in grado di comunicare.

Può servire una **città**, una **regione**, o una **nazione** e interconnette dispositivi di connessione tra switch, router e modem.

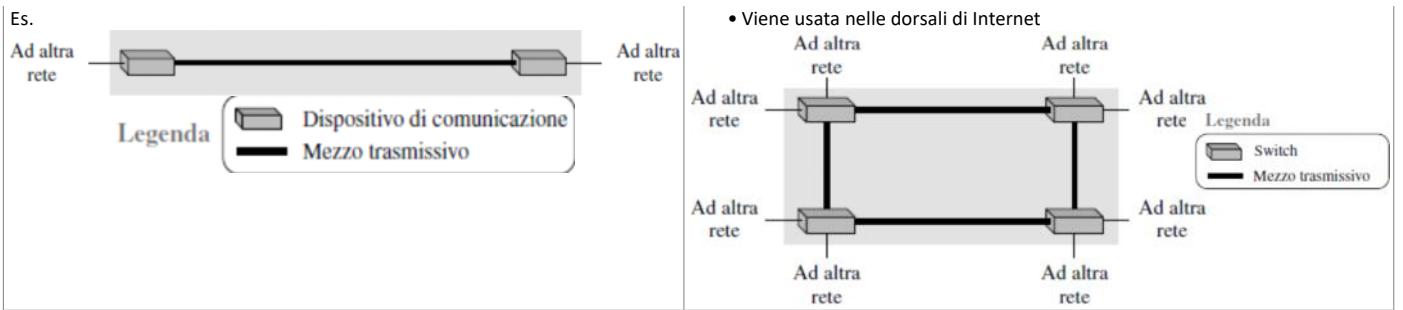
È gestita da un **operatore di telecomunicazioni** (Internet Service Provider - ISP) che fornisce i suoi servizi alle organizzazioni che ne fanno uso:

WAN punto-punto

Collega **due mezzi** di comunicazione tramite un **mezzo trasmissivo**

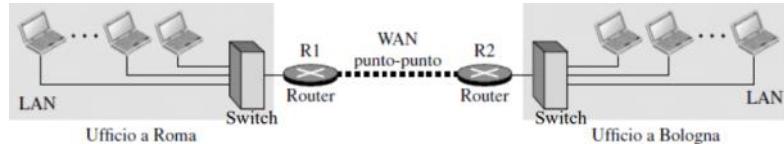
WAN a commutazione

- Rete con più di due punti di terminazioni

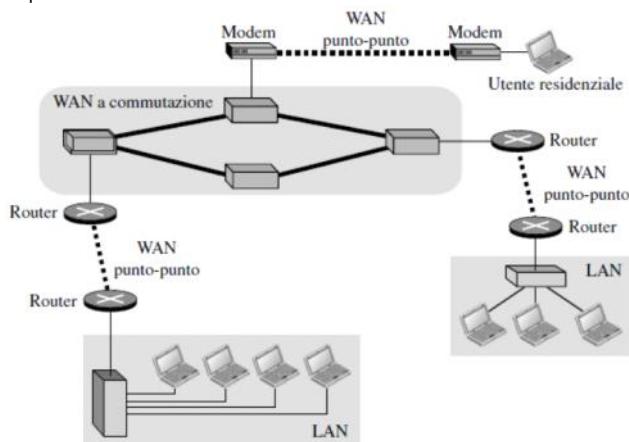


L'internet (con la i minuscola) è un insieme di LAN o WAN connesse fra di loro per formare un internetwork o internet

Es. un'azienda collega due uffici tra diverse città collegando tramite una WAN punto-punto di un ISP le due LAN degli uffici, creando un internet privato dove i router instradano i pacchetti da una LAN all'altra



Altro esempio di rete eterogenea composta da quattro WAN e tre LAN



f

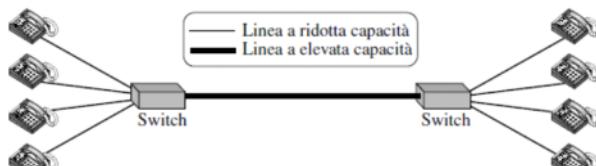
Commutazione

sabato 1 marzo 2025 20:55

Nell'internet i sistemi terminali della rete comunicano tra loro attraverso **switch e router** che si trovano nel percorso. Ci sono due tipi di reti basate su switch:

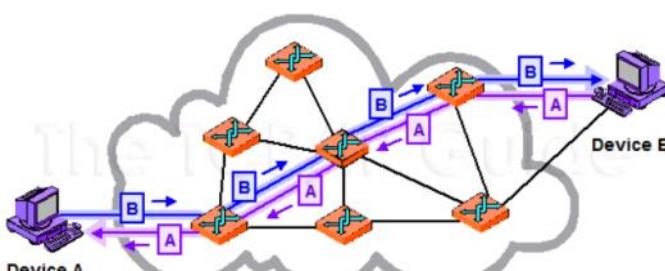
- Reti a commutazione di circuito (**circuit-switched network**)
- Reti a commutazione di pacchetto (**packet-switched network**)

Reti a commutazione di circuito



Tra due dispositivi vi è sempre aperto un collegamento dedicato, chiamato **circuito o socket**, usato per la comunicazione tra i due.

Le risorse necessarie sono **riservate** per tutta la durata della comunicazione e le informazioni sul circuito sono **mantenute dalla rete**.

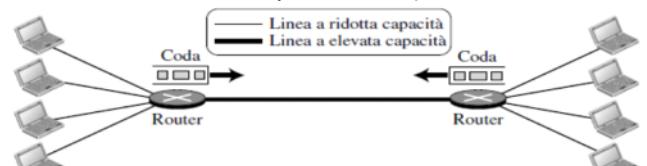


- Viene utilizzato **un solo percorso** per la comunicazione tra i dispositivi
- Comunicazioni **diverse** tra gli stessi dispositivi possono usare **percorsi diversi**

Le **risorse di rete** (ad es. **ampiezza di banda/bandwidth**) sono **suddivise in "pezzi"**, dove vengono allocati ai **vari collegamenti** e rimangono **inattive** se non utilizzate (es. pause durante una conversazione telefonica)

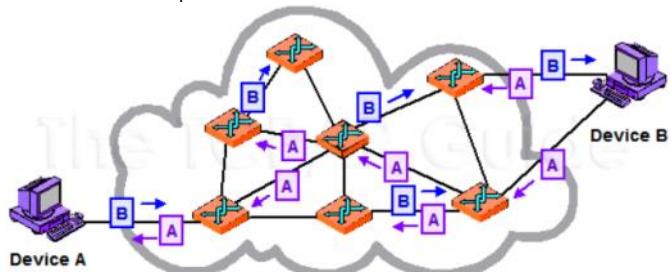
Ad esempio la **banda** viene divisa in **frequenza (FDM)** e **tempo (TDM)**.

Reti a commutazione di pacchetto (store and forward)



Invece di una comunicazione continua, i dispositivi si **scambiano blocchi di dati** chiamati **pacchetti** (i messaggi sono divisi in blocchi con una lunghezza max definita)

Gli switch quindi **memorizzano (store)** e **inoltrano (forward)** i pacchetti provenienti dai vari dispositivi.



- I blocchi di dati (anche dello stesso file) possono prendere **percorsi diversi** per arrivare a destinazione e arrivare in **ordine diverso**
- Non c'è un **percorso specifico** usato per il trasferimento dei dati

Se sono solo due dispositivi nella linea allora **non c'è tempo di attesa**. Ma con più dispositivi, se la linea non ha la capacità di inviare tutti i pacchetti che arrivano, allora il router li **memorizza in una coda**, creando quindi un po' di **ritardo** nell'invio di messaggi. Però questo permette a più utenti di usare la rete.

FDM e TDM

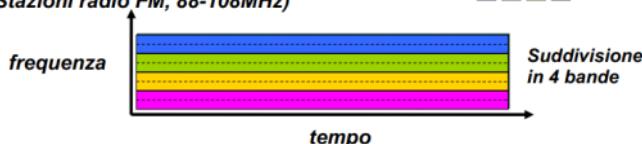
FDM e TDM sono due tecniche **multiplexing** per la trasmissione **simultanea** di più segnali su un solo canale

FDM (Frequency Division Multiplexing)

Per condividere le risorse, l'intero canale trasmissivo è diviso in **sottocanali** separati costituiti ognuno da una **banda di frequenza**.

Questo permette di condividere lo stesso canale a più dispositivi usando diverse regioni di frequenza, permettendo di comunicare contemporaneamente senza interferire tra di loro.

FDM: Frequency Division Multiplexing
(es. Stazioni radio FM, 88-108MHz)



Esempio:

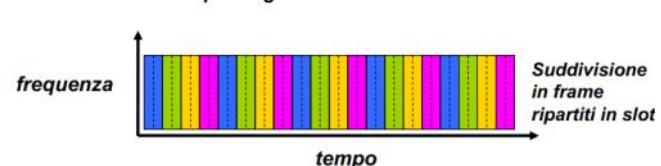
4 utenti



TDM (Time Division Multiplexing)

Per condividere le risorse, ogni dispositivo ottiene esclusivamente **a turno** le risorse della banda **per un breve lasso di tempo** ognuno della stessa durata.

TDM: Time Division Multiplexing



Internet e ISP

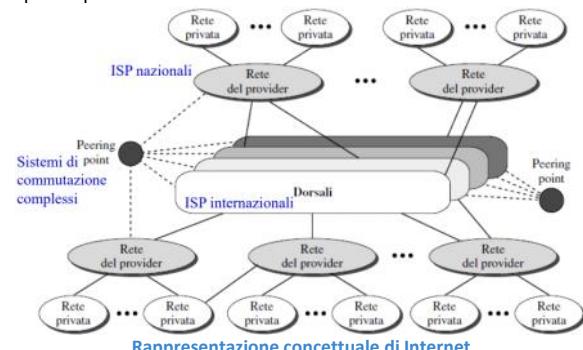
lunedì 3 marzo 2025 13:49

L'Internet (con la i maiuscola) che usiamo tutti i giorni è costituito da **migliaia di reti interconnesse** ed è una rete a commutazione di **pacchetto**. Per accedere all'Internet l'utente deve essere collegato ad un **ISP**, solitamente mediante una WAN punto-punto.

Il collegamento che connette al primo router di Internet è detto **rete di accesso o Access Point**.

L'accesso si può effettuare tramite:

- **Rete telefonica**
 - Servizio dial-up (via modem)
 - Servizio DSL (Digital Subscriber Line)
- **Reti wireless**
 - Wi-Fi
 - Cellulare
- **Collegamento diretto**
 - Un'azienda può divenire ISP locale, affittando delle reti WAN da un operatore

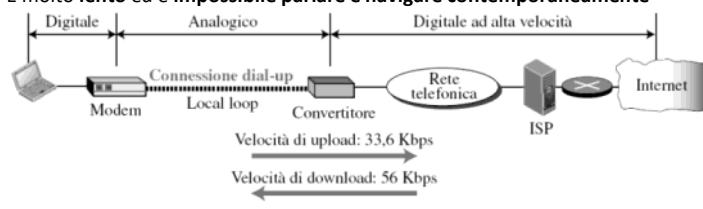


Accesso via Rete Telefonica

Ci si può collegare a Internet modificando la linea telefonica fra la sede del dispositivo (es. casa, azienda, etc) e la centrale telefonica con una WAN punto-punto

Servizio Dial-Up

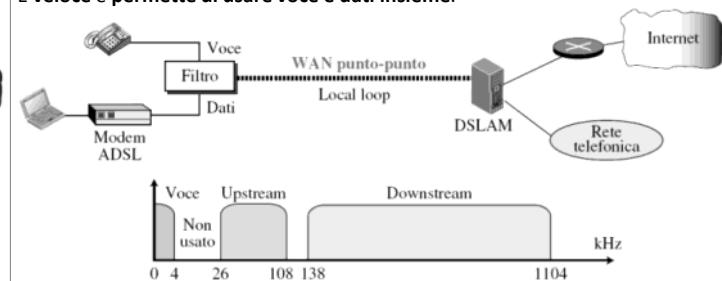
Si inserisce un modem (**modulatore-demodulatore**) che converte i **dati digitali** (del computer) in **analogici**, per trasmetterli sulla linea telefonica, e viceversa. È molto **lento** ed è **impossibile parlare e navigare contemporaneamente**.



Servizio DSL (Digital Subscriber Line)

Il collegamento tra abitazione e ISP è **diviso in tre bande** di frequenza non sovrapposte.

È **veloce** e permette di usare **voce e dati insieme**.



L'accesso si può effettuare tramite:

Ethernet

Nelle reti aziendali e universitarie le tecnologie per l'accesso a Internet è:

- **Ethernet**: lo switch (Ethernet) della LAN è collegato ad un router istituzionale che è connesso a quello della dorsale

Wireless

• Wi-Fi:

- **Access Point (locale)** connesso all'Ethernet cablato
- Raggio di azione di qualche decina di metri

• Cellulari:

- Si usa la **rete cellulare**
- **Access Point (base station)** della compagnia telefonica, con raggio di azione di qualche decine di chilometri

ISP

L'Internet è formato dalle **reti di accesso** (collegamenti che connettono le reti ai diversi ISP locali) + **backbone Internet** (diversi ISP che si occupano di gestire il routing dei pacchetti)

Access Network

Sono dei collegamenti fisici che connettono un sistema al primo **edge router**, che è il router che collega a Internet tramite le ISP.

L'access network può essere **residenziale** (punto-punto, modem via cavo), **aziendale** (LAN) o **wireless** (wifi, satellite)

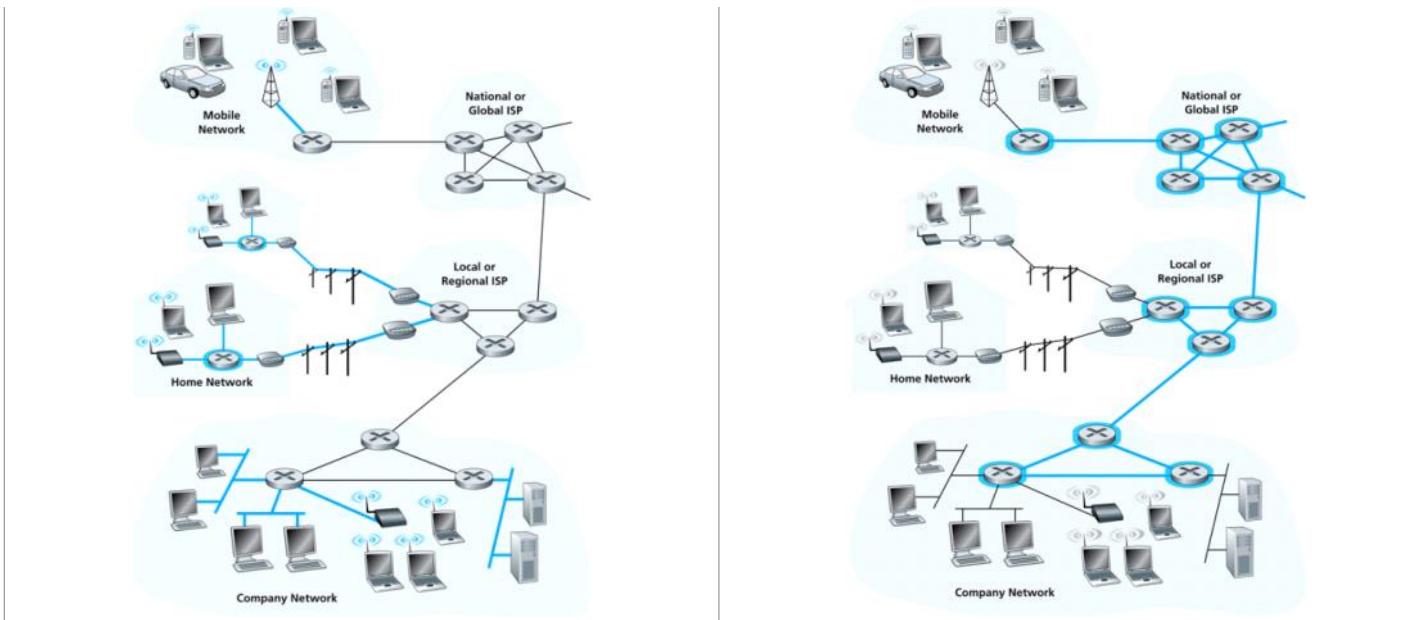
Eseguono diverse elaborazioni di pacchetti, firewalling, ecc

Backbone Internet

Sono i diversi ISP (locali/regionali/nazionali/globali) che permettono il collegamento degli utenti su tutto l'Internet.

Si utilizza principalmente la **commutazione di pacchetto**

I router degli ISP hanno larghezza di banda elevata e spostano semplicemente i pacchetti



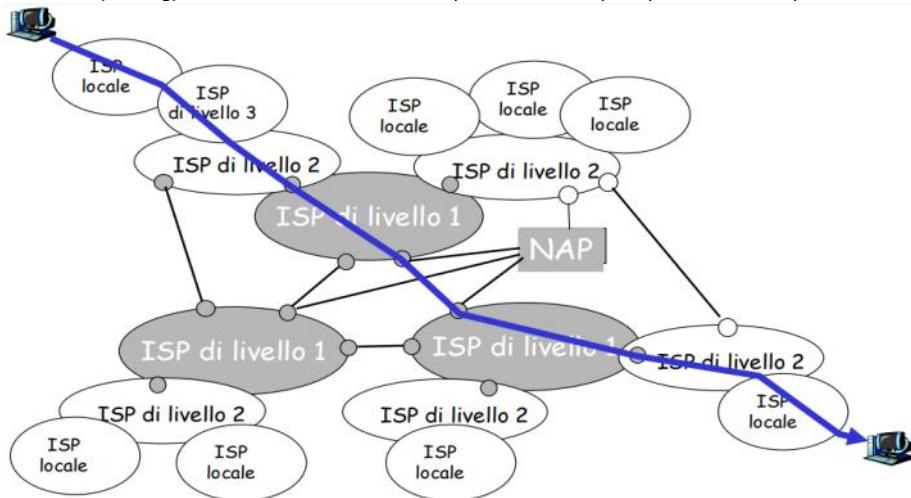
Struttura di Internet

La struttura della **backbone** dell'Internet è **gerarchica** con 3 livelli di ISP:

- **ISP di livello 1:** (es: Verizon, AT&T, ecc) che hanno una **copertura nazionale/internazionale** e comunicano tra di loro come "pari"
- **ISP di livello 2:** sono ISP **più piccoli** e hanno una **copertura nazionale/distrettuale** e possono connettersi solo ad alcuni ISP liv 1 e possibilmente a ISP di liv 2
 - Un ISP di liv 2 paga l'ISP di liv 1 che gli **fornisce la connettività** per il resto della rete, quindi sono clienti di un ISP di liv 1
 - Quando due ISP sono direttamente interconnessi vengono detti di **pari grado (peer)**
- **ISP di livello 3:** sono ISP **locali (o di accesso)** e sono dette reti di "**ultimo salto**" (last hop network) perché sono le più vicine ai **sistemi terminali** (utenti)
 - Gli ISP locali sono **clienti** degli ISP di livello superiore (liv 2) che li collegano all'intera Internet

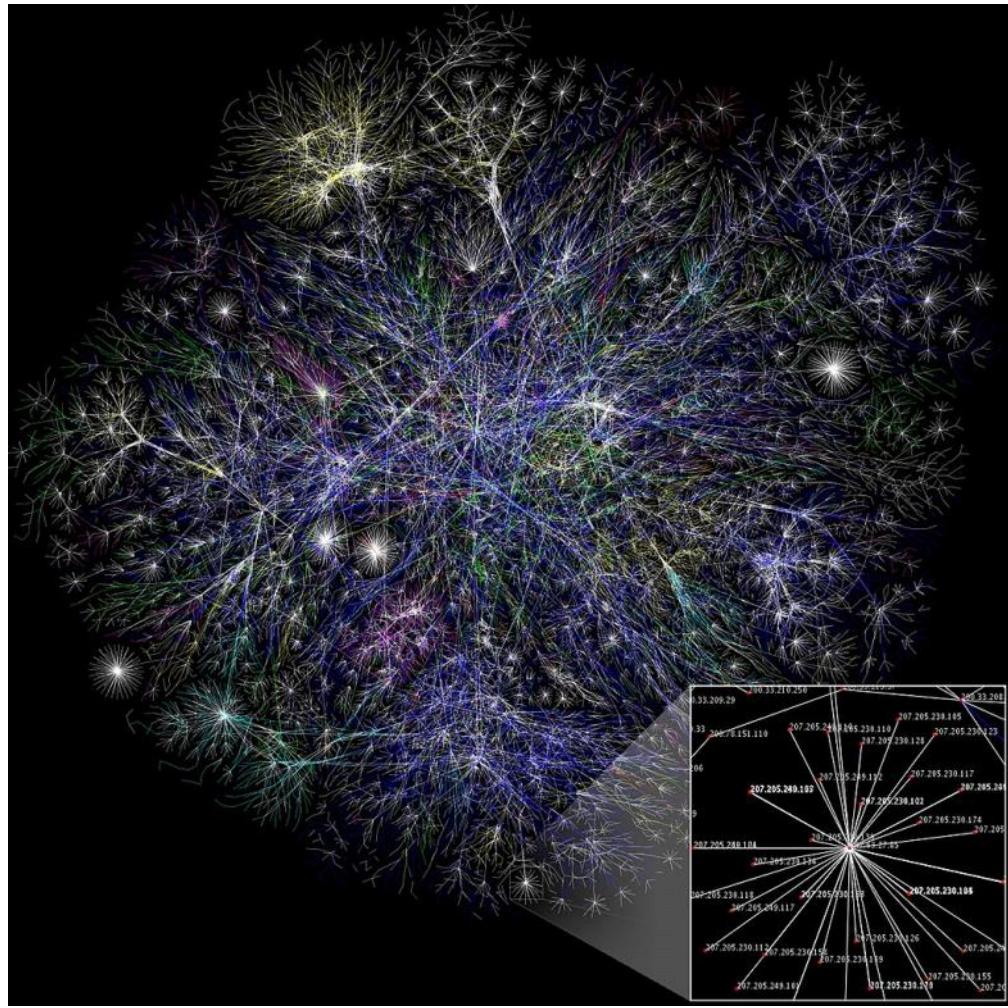
Un pacchetto passa attraverso molte reti passando per i percorsi minori dal sistema sorgente a quello di destinazione.

Viene eseguito quindi un **instradamento (routing)** dai modem/switch dei vari ISP per decidere su quale porta inviare un pacchetto di rete.



Internet oggi

su internet-map.net/ vi è anche una visualizzazione di migliaia di siti e del loro numero di accessi da parte degli utenti su Internet



Capacità e Prestazioni

martedì 4 marzo 2025 11:48

Il concetto di **velocità della rete o di collegamento**, che indica quanto velocemente si trasmettono/ricevono i dati, è molto ampio e coinvolge **vari fattori** che influiscono sulle **prestazioni della rete**.

Nel caso di una rete a commutazione di **pacchetto**, i fattori determinanti per le prestazioni sono:

- **Bandwidth (ampiezza di banda)** e Bit rate
- **Throughput**
- **Latenza (ritardo)**
- **Perdita di pacchetti**

Bandwidth e Bit Rate

Con **bandwidth (ampiezza di banda)** si indicano due concetti strettamente legati:

- 1) **Caratterizzazione del canale o del sistema trasmittivo**: misurata in **hertz (Hz)**, rappresenta l'**intervallo di frequenze** che un mezzo fisico (es. cavo) consente di trasmettere senza danneggiare il segnale in maniera irrecuperabile.

Maggiore il **bandwidth**, maggiore la **quantità di informazioni** che può essere veicolata attraverso il mezzo trasmittivo

- 2) **Caratterizzazione di un collegamento**: misurata in **bit al secondo (bps)**, detta anche **bit, rate o transmission rate** (velocità di trasmissione) e indica la **quantità di bit al secondo** che un link garantisce di trasmettere

Il **bit rate** dipende sia dalla banda (Hz) che dalla tecnica di trasmissione o formato di modulazione usato. Esso è **proporzionale** alla banda in Hz.

Si utilizza per fornire un'indicazione della **capacità** della rete di trasferire dati.

Per **banda di un tipo di rete** si intende il **bit rate garantito** (nominalmente) dai suoi link

Esempio: il bit rate di un link **Fast Ethernet** è di 100 Mbps, quindi tale rete può inviare **al massimo** 100 Mbps

Throughput

Il **throughput** indica quanto velocemente riusciamo **effettivamente** a inviare i dati su una rete.

Esso indica il **numero di bps che passano attraverso un punto della rete**.

Quindi rispetto al **bit rate** che misura la **potenziale velocità** di un link, il **throughput** è una misura dell'**effettiva velocità** (velocità reale).

Perciò il **throughput** sarà sempre **minore uguale al bit rate**. Se abbiamo un bit rate di B bps, ma possiamo inviare solo T bps abbiamo $T \leq B$

Esempio: se avessimo una strada progettata per 1000 auto/min ma col traffico si riduce a 100 auto/min. Allora il bit rate è 1000 e il throughput è 100 auto/min.

Il pacchetto normalmente passa per vari link, ognuno con **throughput diverso**. Il **throughput dell'intero percorso (end to end)** è il **throughput minimo** dei link.

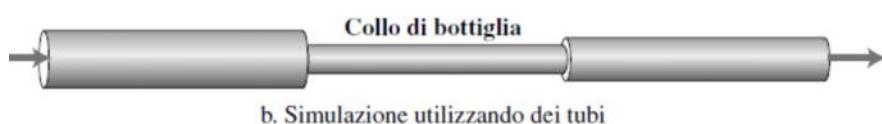
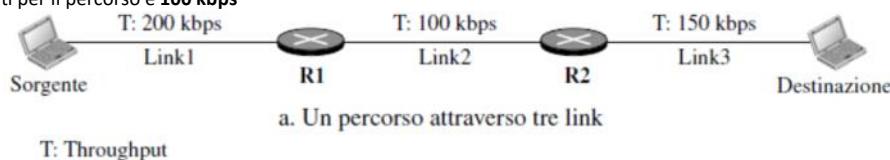
In generale in un percorso con **n link in serie** abbiamo:

$$\text{Throughput} = \min(T_1, T_2, \dots, T_n)$$

Collo di Bottiglia

Il **collo di bottiglia** è una situazione che si verifica quando un link con **throughput basso** vincola il throughput di tutto il collegamento end to end.

Nell'esempio il throughput dei dati per il percorso è **100 kbps**



Su Internet i dati passano attraverso due reti di accesso e la backbone di Internet che ha un **throughput molto alto** (Gb/s), quindi il throughput totale è definito dal minimo tra i due link di accesso che collegano la sorgente/destinazione a Internet.

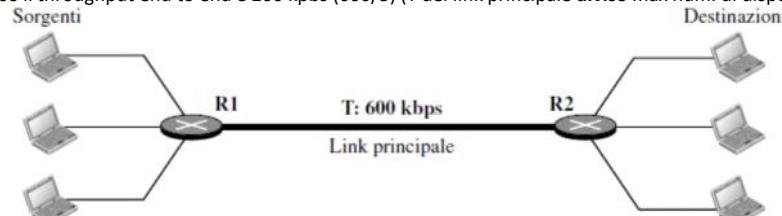
In questo esempio il throughput è il minimo tra T_1 e T_2 .

Se T_1 è 100 Mbps (Fast Ethernet LAN) e T_2 è 40 kbps (linea telefonica), il throughput è 40 kbps

Link Condivisi

Spesso il link tra due router raccoglie il flusso da varie sorgenti e/o distribuisce a varie destinazioni. Quindi il bit rate tra i due router è **condiviso** tra i flussi di dati.

Nell'esempio, siccome il link è condiviso il throughput end to end è 200 kbps ($600/3$) (T del link principale **diviso** max num. di dispositivi tra sorgente e destinazione)

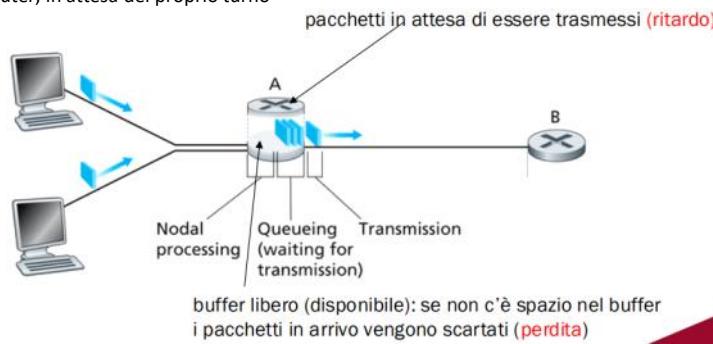


Latenza

La **latenza (ritardo o delay)** indica quanto tempo serve affinché un pacchetto arrivi completamente a destinazione dal momento in cui il primo bit parte dalla sorgente.

Nella commutazione di pacchetto i pacchetti si accodano nei buffer del router.

- Il tasso di arrivo dei pacchetti **eccede la capacità** del collegamento di evaderli (inviarli a destinazione)
- I pacchetti quindi si accodano nel router, in attesa del proprio turno



Ci sono 4 cause di ritardo per i pacchetti

1) Ritardo di Elaborazione del Nodo (processing delay)

- Controllo errori sui bit
- Determinazione del canale di uscita
- Tempo di ricezione dall'input alla consegna all'output

2) Ritardo di Accodamento (queueing delay)

- Attesa di trasmissione (nella coda di input e quella di output del router)
- Livello di congestione del router
- Varia da pacchetto a pacchetto

3) Ritardo di Trasmissione (transmission delay)

Tempo richiesto per **trasmettere tutti i bit** del pacchetto (far uscire il pacchetto).

Se il primo bit del pacchetto è trasmesso al tempo t_1 e l'ultimo bit viene ricevuto al tempo t_2 , il **ritardo di trasmissione** del pacchetto è $t_2 - t_1$.

I pacchetti sono trasmessi tramite **coda FCFS** (First Come First Serve)

Funzione della lunghezza del pacchetto e bit rate:

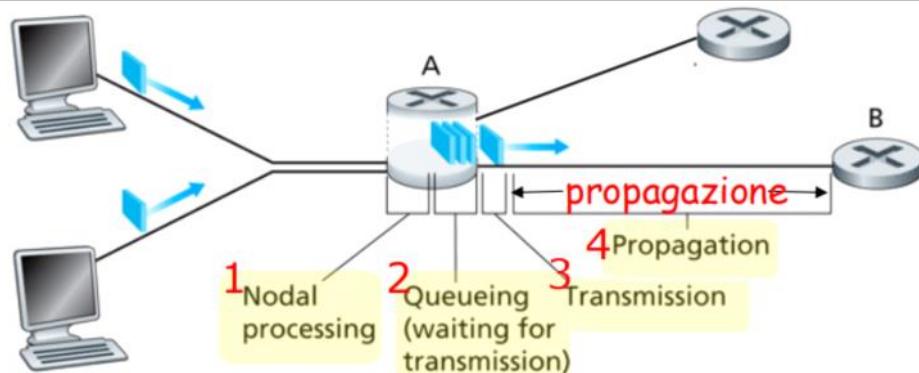
$$\text{Ritardo di trasmissione} = \frac{\text{lunghezza pacchetto (bit)}}{\text{bit rate link (bps)}}$$

4) Ritardo di Propagazione (propagation delay)

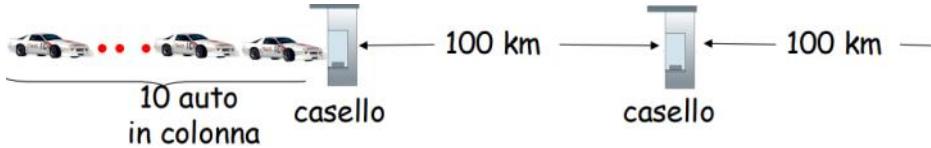
Tempo che un bit impiega per **propagarsi sul link** (da punto A a punto B)

Funzione della **distanza** del link

$$\text{Ritardo di propagazione} = \frac{\text{lunghezza link fisico}}{\text{vel.propagazione del link } (\sim 2 \times 10^8 \text{ m/sec}) \text{ (circa velocità luce)}}$$



Analogia con Casello Autostradale



- Le automobili viaggiano ("si propagano") alla **velocità** di 100 km/h
- Il casello serve ("trasmette") un'auto ogni 12 secondi (**bit rate**)
Quindi 5 auto/min
- Auto ~ bit; colonna auto ~ pacchetto

Quanto tempo serve perché le 10 auto in carovana abbiano superato il secondo casello?

- Tempo richiesto al casello per trasmettere l'intera colonna:
 $\frac{10}{(5 \text{ auto/min})} = 2 \text{ min}$
 Corrisponde al **tempo di trasmissione**
- Tempo richiesto a un'auto per viaggiare dall'uscita di un casello fino a quello successivo:
 $\frac{100 \text{ km}}{100 \text{ km/h}} = 1 \text{ ora}$
 Corrisponde al **tempo di propagazione**

Quindi il tempo che intercorre da quando l'intera colonna di vetture si trova di fronte al casello di partenza fino al momento in cui raggiunge quello successivo è:
Ritardo di trasmissione + Ritardo di propagazione = 2 min + 60 min = 62 minuti

Altro esempio

- Le automobili "si propagano" alla **velocità** di 1000 km/h
- Il casello "trasmette" un'auto al minuto

Le prime auto arriveranno al secondo casello prima che le ultime auto della colonna lascino il primo?

- Tempo di **trasmissione**: 1 min

- Tempo richiesto a un'auto per viaggiare dall'uscita di un casello fino a quello successivo:
 $1000 \text{ km} : 1000 \text{ km/h} = 1 \text{ min}$

Dopo 7 minuti, la prima auto sarà al secondo casello. Poiché passano 6 minuti dal tempo di arrivo al secondo casello, il primo casello avrà fatto passare già 6 macchine. Quindi al primo casello sono rimaste da trasmettere ancora **tre auto** (10 - 6 - 1(quella arrivata al 2° casello))

Il **primo bit** può arrivare al secondo router **prima** che il pacchetto sia stato **interamente trasmesso** dal **primo router**

Ritardo di un Nodo

$$d_{node} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

- d_{proc}** = ritardo di **elaborazione** (processing delay): in genere pochi microsecondi, o anche meno
 - Tempo richiesto per **esaminare il pacchetto e determinare dove dirigerlo**
- d_{queue}** = ritardo di **accodamento** (queueing delay): dipende dalla **congestione** dei pacchetti

- Dipende dal **num. di pacchetti precedentemente arrivati**
- d_{trans} = ritardo di trasmissione (**transmission delay**): significativo sui collegamenti a **bassa velocità**
 - Ritardo di trasmissione = $\frac{\text{lunghezza pacchetto (bit)}}{\text{bit rate link (bps)}}$
- d_{prop} = ritardo di propagazione (**propagation delay**): da pochi **microsecondi** a centinaia di **millisecondi**
 - Ritardo di propagazione = $\frac{\text{lunghezza link fisico}}{\text{vel.propagazione del link } (\sim 2 \times 10^8 \text{ m/sec})}$

Ritardo di Accodamento

Il ritardo di **accodamento** varia da pacchetto a pacchetto e dipende dal **tasso di arrivo**, dal **bit rate**, e dalla **lunghezza** dei pacchetti. Si fa quindi uso di **misure statistiche** per calcolare il ritardo medio.

- R = bit rate (bps)
- L = packet length (bit)
- a = tasso medio di arrivo dei pacchetti (pkt/s)
- $\frac{L \cdot a}{R} = \text{intensità di traffico}$
- $\frac{L \cdot a}{R} \approx 0$: poco ritardo
- $\frac{L \cdot a}{R} \rightarrow 1$: il ritardo si fa consistente
- $\frac{L \cdot a}{R} > 1$: più "lavoro" in arrivo di quanto possa essere svolto, ritardo medio infinito

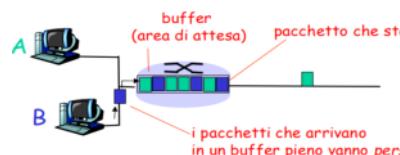


Quanto più l'intensità di traffico si avvicina a 1, tanto più rapidamente cresce il ritardo medio di accomodamento

Perdita di Pacchetti (Packet Loss)

Se il **buffer** del router ha **capacità finita**, quando il pacchetto trova il buffer **pieno**, viene **scartato** (va perso). Il pacchetto perso può:

- Essere **ritrasmesso dal nodo precedente**
- Essere **ritrasmesso dal sistema terminale che lo ha generato**
- **Non essere ritrasmesso**



Traceroute

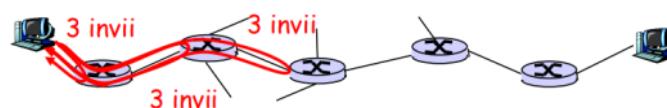
Il **Traceroute** (tracert) è un programma diagnostico usato per fornire una **misura del ritardo** dalla sorgente a tutti i router lungo il percorso Internet punto-punto verso la destinazione.

- Invia **gruppi di tre pacchetti**, ogni gruppo con **tempo di vita incrementale** (da 1 a n, max val = 30) che raggiungeranno il **router i** ($i = 1, n$) sul percorso verso la destinazione
- Il **router i** restituirà i pacchetti al mittente
- Il mittente calcola l'**intervallo tra trasmissione e risposta dei pacchetti**

Il tempo di andata e ritorno (round trip time - RTT) include i **4 ritardi** visti in precedenza

L'output presenta **6 colonne**:

- 1) Num. router sulla rotta
- 2) Nome router
- 3) Indirizzo router
- 4) RTT 1° pacchetto
- 5) RTT 2° pacchetto
- 6) RTT 3° pacchetto



Può accadere (se c'è **congestione** o se si segue un percorso diverso) che il **RTT** del router **n** sia **maggior** del **RTT** del router **n+1** a causa dei ritardi di accodamento.

Se la sorgente **non riceve risposta** da un **router intermedio** (o riceve meno di tre pacchetti) allora pone un asterisco al posto del **tempo RTT**

Esempio

traceroute: da gaia.cs.umass.edu a www.eurecom.fr

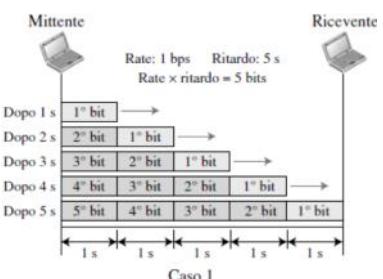
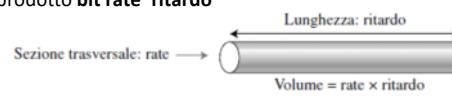
Tre misure di ritardo da gaia.cs.umass.edu a cs-gw.cs.umass.edu						
umass	1 cs-gw (128.119.240.254) 1 ms 1 ms 2 ms					
	2 border1-rt-fa5-1-0.gw.umass.edu (128.119.3.145) 1 ms 1 ms 2 ms					
	3 cht-vbns.gw.umass.edu (128.119.3.130) 6 ms 5 ms 5 ms					
vbns	4 jn1-at1-0-0-19.wor.vbns.net (204.147.132.129) 16 ms 11 ms 13 ms					
	5 jn1-so7-0-0.wae.vbns.net (204.147.136.136) 21 ms 18 ms 18 ms					
geant	6 abilene-vbns.abilene.ualcd.edu (198.32.11.9) 22 ms 18 ms 22 ms					
	7 nycm-wash.abilene.ualcd.edu (198.32.8.46) 22 ms 22 ms 22 ms					
	8 62.40.103.253 (62.40.103.253) 104 ms 109 ms 106 ms					
	9 de2-1.de1.de.geant.net (62.40.96.129) 109 ms 102 ms 104 ms					
	10 de2-1.de1.de.geant.net (62.40.96.50) 113 ms 121 ms 114 ms					
renater	11 renater-gw.fr1.fr.geant.net (62.40.103.54) 112 ms 114 ms 112 ms					
	12 nio-n2.cssi.renater.fr (193.51.206.13) 111 ms 114 ms 116 ms					
	13 nice.cssi.renater.fr (195.220.98.102) 123 ms 125 ms 124 ms					
	14 r3t2-nice.cssi.renater.fr (195.220.98.110) 126 ms 126 ms 124 ms					
	15 eurecom-valbonne.r3t2.ft.net (193.48.50.54) 135 ms 128 ms 133 ms					
	16 194.214.211.25 (194.214.211.25) 126 ms 128 ms 126 ms					
eurecom	17 ***					
	18 *** * significa nessuna risposta (risposta persa, il router non risponde)					
	19 fantasia.eurecom.fr (193.55.113.142) 132 ms 128 ms 136 ms					

Rate*Ritardo

Supponiamo di avere un link con **bit rate 1bps** e **ritardo di 5 sec**. Avremmo che:

- Il **max num. di bit** che possono riempire il collegamento è 5
- Quindi non possono esserci **più di 5 bit contemporaneamente nel link**

Possiamo pensare al link tra due punti come ad un tubo, la cui **sezione trasversale** rappresenta il **bit rate** e la **lunghezza** rappresenta il **ritardo**. Il volume del tubo è il prodotto **bit rate*ritardo**



Esercizio

- 1) Quanto impiega un pacchetto di 1000 byte per propagarsi su un link di 2500km, con velocità di propagazione pari a $2,5 \times 10^8$ m/s e bit rate di 2 Mbps?
- 2) Questo ritardo dipende dalla lunghezza del pacchetto?
- 3) Calcola ritardo di trasmissione

- Packet lenght: 1000 byte = 8000 bit
- Link lenght: 2500km = 2500000 m
- Velocità propagazione: $2,5 \times 10^8$ m/s
- Bit rate: 2 Mbps = 2000000 bps

1) Ritardo di **propagazione** = $\frac{2500000}{2,5 \times 10^8} = \frac{2,5}{2,5 \times 10^2} = \frac{1}{1 \times 10^2} = \frac{1}{100} = 0.01$ sec = 10 ms

2) No, il ritardo di propagazione dipende dalla lunghezza del link e dalla velocità di propagazione

3) Ritardo di **trasmissione** = $\frac{8000}{2000000} = 0.004$ sec = 4 ms

Stack TCP/IP e Protocolli

domenica 13 aprile 2025 19:19

Protocolli

Un protocollo **definisce le regole** che verranno rispettate in tutti i sistemi intermedi o finali coinvolti nella comunicazione.

Per situazioni complesse è opportuno suddividere i compiti in **più livelli (layer)**, in cui è richiesto un protocollo per ciascun livello, si esegue un **layering di protocolli**

La strutturazione dei protocolli in livelli consente di suddividere un compito complesso in compiti più semplice e rendendo **modulari i livelli** (indipendenti)

Quindi nel caso in cui si dovesse modificare un livello, non si vanno a intaccare gli altri. Quindi si crea una separazione tra servizi e implementazione: un livello usa servizi dal livello inferiore e offre servizi a quello superiore

Vantaggi modularità:

- Semplicità di design
- Possibilità di modificare un modulo in modo trasparente se le interfacce con gli altri livelli rimangono le stesse
- Possibilità di ciascun costruttore di adottare la propria implementazione di un livello purchè soddisfi i requisiti delle interfacce del liv. superiore

Svantaggi modularità:

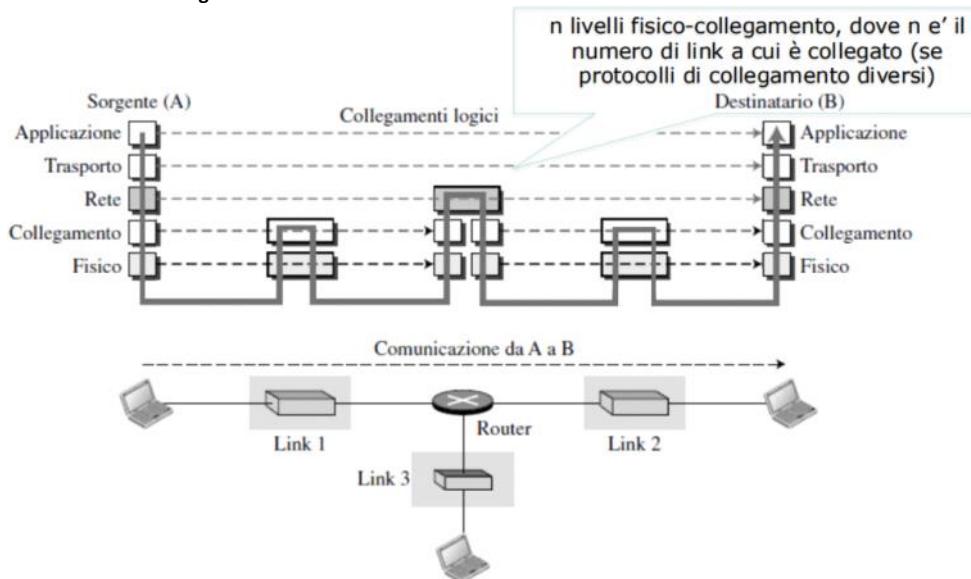
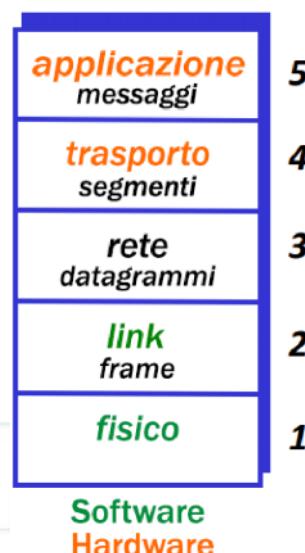
- A volte è necessario uno scambio di informazioni tra livelli non adiacenti (es: per ottimizzare app funzionante su wireless) non rispettando il **principio della stratificazione**

TCP/IP

In internet si usa una famiglia di protocolli chiamata **TCP/IP**, che è una **gerarchia di protocolli** formata da moduli interagenti, ciascuno con funzionalità specifiche. Con **gerarchia** si intende che ogni protocollo di livello superiore è supportato dai servizi forniti da quello al livello inferiore.

Pila dei protocolli TCP/IP (stack protocollare)

- **Applicazione**: sede dell'applicazione di rete
 - HTTP, SMTP, FTP, DNS
 - I pacchetti si chiamano **messaggi**
- **Trasporto**: trasferimento dei messaggi a liv. applicazione tra modulo client e server di un'applicazione
 - TCP, UDP
 - I pacchetti si chiamano **segmenti**
- **Rete**: instradamento dei segmenti dall'origine alla destinazione
 - IP, protocolli di instradamento
 - I pacchetti si chiamano **datagrammi**
- **Link (collegamento)**: trasmette datagrammi da un nodo a quello successivo nel percorso
 - Ethernet, Wi-Fi, PPP
 - Lungo un percorso, un datagramma può essere gestito da protocolli diversi
 - I pacchetti si chiamano **frame**
- **Fisico**: trasferimento dei **singoli bit**



Gerarchia dei protocolli

- La rete è organizzata a **livelli** e lo scopo di ogni livello è di **offrire determinati servizi al livello superiore**
- Il livello N di un pc è in comunicazione logica col livello N di un altro pc
- Le regole/convenzioni usate nella comunicazione si chiamano **protocolli** del livello N
- Le entità che formano i livelli si chiamano **pari (peer)**
- I peer comunicano usando il protocollo
- I dati non sono trasferiti direttamente dal livello N di un pc al livello N di un altro pc

Servizi e protocolli

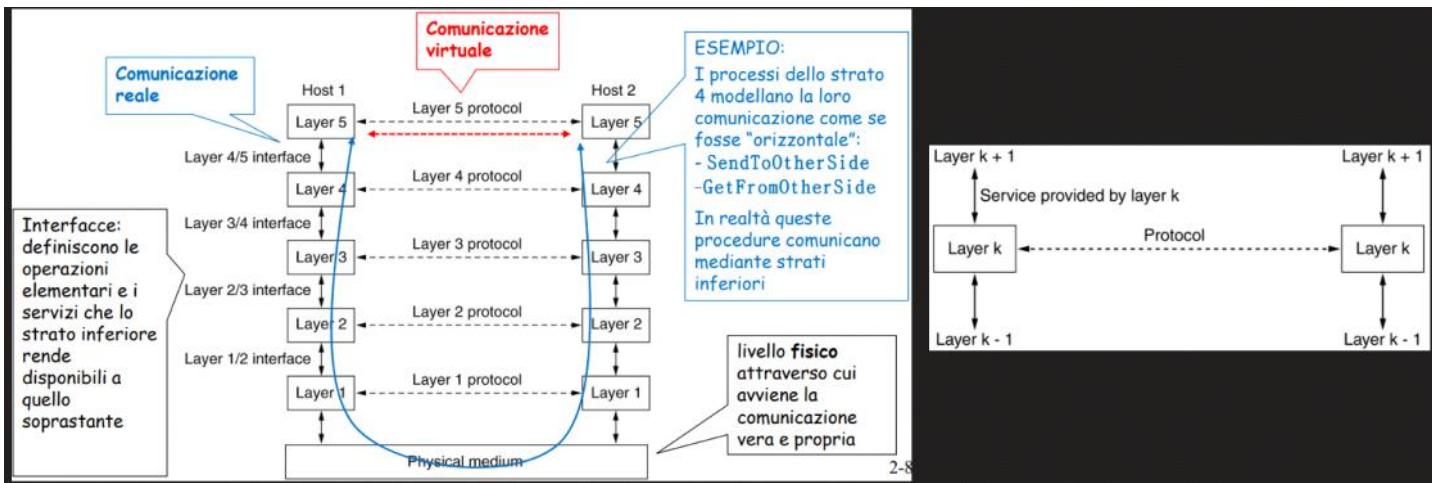
Servizio

È un **insieme di primitive** che uno strato offre a quello superiore

- Definisce quali op. lo strato è in grado di offrire, ma non dice come sono implementate
- È correlato all'interfaccia tra due strati, dove quello inferiore è il provider del servizio mentre quello superiore è l'utente

Protocollo

È un **insieme di regole** che controllano il formato e il significato dei pacchetti, o messaggi scambiati tra le entità pari all'interno di uno strato.



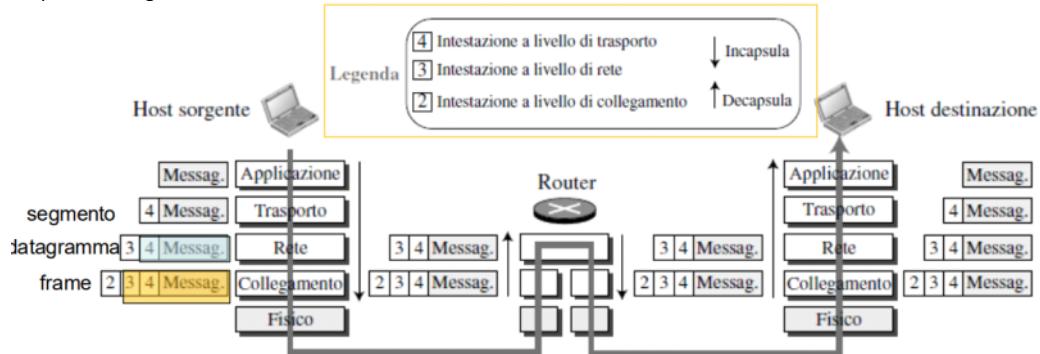
Incapsulamento e Decapsulamento

La sorgente effettua l'**incapsulamento**: prende il pacchetto dal liv. superiore, lo considera come **payload** (carico dati) e aggiunge un **header** (intestazione)

- **Messaggio (Applicazione)** = messaggio (nessuna intestazione)
- **Segmento (Trasporto)** = header trasporto + messaggio
- **Datagramma (Rete)** = header rete + segmento
- **Frame (Link)** = header collegamento + datagramma

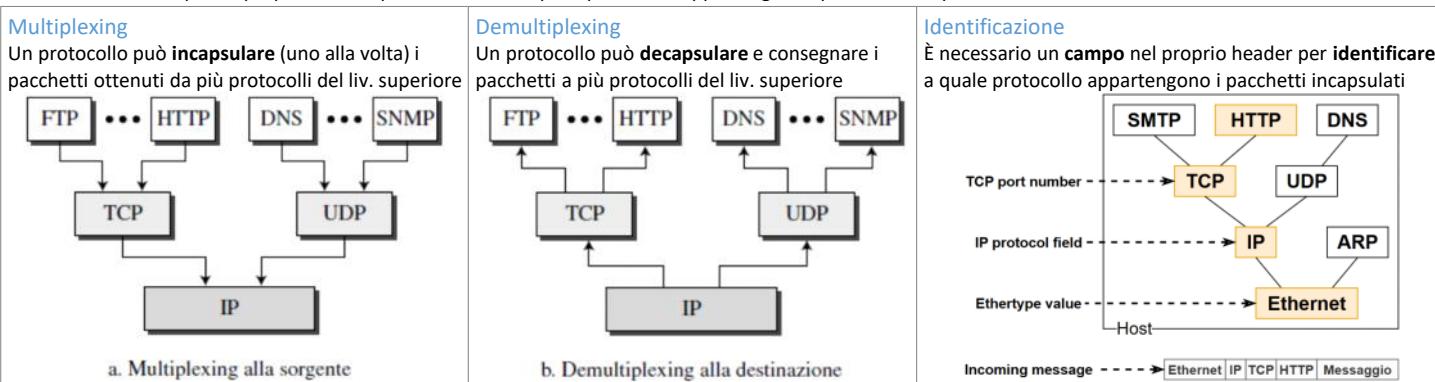
Il destinatario effettua il **decapsulamento**.

Il router effettua entrambi poiché collegato a due link.



Multiplexing e Demultiplexing

Dato che lo stack TCP/IP prevede più protocolli nello stesso livello, è necessario eseguire il **multiplexing** alla sorgente e il **demultiplexing** alla destinazione.
È necessario un campo nel proprio header per identificare a quale protocollo appartengono i pacchetti incapsulati



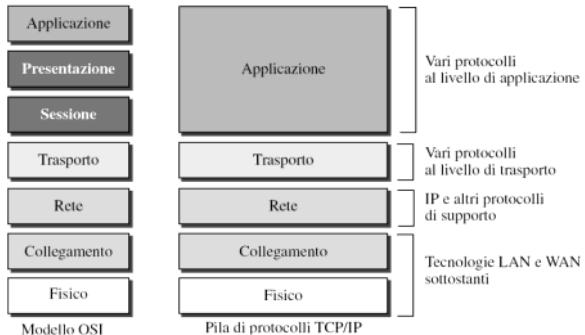
Indirizzamento

Poiché il TCP/IP prevede una comunicazione logica tra coppie di livelli, è necessario avere un indirizzo sorgente e uno destinazione **ad ogni livello**



Modello OSI

L'ISO (International Organization for Standardization) ha definito il **modello OSI** (Open System Interconnection) come alternativa al TCP/IP. Esso è un **framework stratificato** per il progetto di sistemi di rete che consentono la comunicazione fra qualsiasi tipo di dispositivo. Contiene due livelli in più rispetto ai 5 del TCP/IP



Solo che poiché TCP/IP era già ampiamente diffuso e i livelli di presentazione e sessione non erano mai stati completamente specificati, non riuscirono a dimostrare delle prestazioni tali da convincere le autorità di Internet a sostituire il TCP/IP

Livello Applicazione

domenica 13 aprile 2025 19:20

Il livello applicazione fornisce servizi all'utente.

Si può immaginare che esista una **connessione logica**: quindi i livelli applicazione della sorgente/destinazione, agiscono come se esistesse un collegamento diretto attraverso il quale poter inviare/ricevere messaggi.

La comunicazione **reale** avviene attraverso più livelli e più dispositivi e vari canali fisici

Dato che il liv. applicazione è quello che fornisce i servizi agli utenti, si possono facilmente aggiungere nuovi protocolli. Oggi infatti ne vengono costantemente aggiunti di nuovi.

I protocolli che sono standardizzati e documentati dagli enti che gestiscono l'Internet si chiamano **protocolli standard** (es. applicazioni web specificate dal protocollo HTTP)

Esistono anche **protocolli non standard** che non sono standardizzati. Non è necessario chiedere autorizzazioni, quindi qualunque azienda può sviluppare il proprio protocollo di livello applicazione.

Applicazione di Rete

Esempi di **applicazioni di rete**:

- Posta elettronica
- Sito web
- Messaggistica istantanea
- Condivisione file P2P
- Streaming di video
- Telefonia e videoconferenze via Internet

Per creare un'applicazione di rete bisogna scrivere programmi che:

- Girano su sistemi terminali diversi
- Comunicano attraverso la rete
- Es: software server Web comunica col software di un browser

Poiché sono software in grado di funzionare su più macchine (es email):

- Non occorre predisporre programmi per i dispositivi del nucleo della rete (es router o commutatori)
- I programmi applicativi sono indipendenti dalla tecnologia che c'è sotto

Architettura di un Applicazione di Rete

Per poter creare una nuova applicazione di rete bisogna avere un **piano architetturale** per definire:

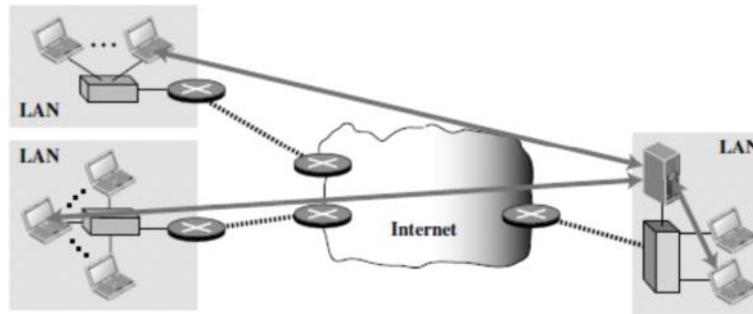
- Il **tipo di architettura** che si vuole creare (client-server, peer-to-peer, ecc)
- Il modo di **comunicazione** tra i processi dell'applicazione
- I **tipi di servizi** (di rete) che richiede l'applicazione (affidabilità, banda, ecc)

Bisogna definire se i due programmi applicativi devono entrambi richiedere/offrire servizi, oppure se ciascuno si occupa di uno dei due compiti.

Paradigma:

- Client-server
- Peer-to-peer (P2P)
- Architetture ibride (client-server e P2P)

Paradigma client-server

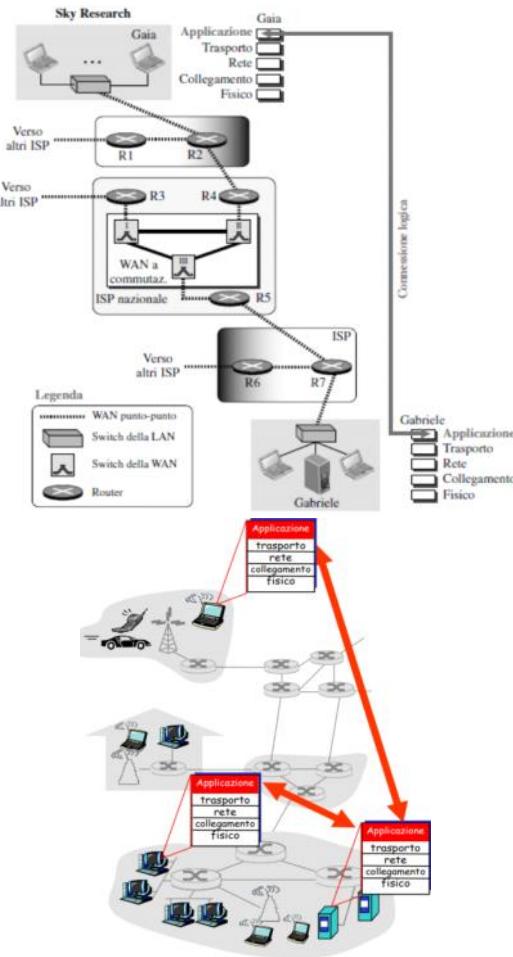


Client	Server
<ul style="list-style-type: none">• Richiede servizi• In esecuzione solo quando serve il servizio• Numerosi client richiedono il servizio	<ul style="list-style-type: none">• Fornisce i servizi• Sempre in esecuzione, in attesa di richieste dal client• Num. limitato di processi server pronti a offrire uno specifico servizio

Il ruolo di entrambi è **totalmente differente**: non si può eseguire un client come programma server e viceversa

Svantaggi:

- Carico di comunicazione centrato sul server (deve essere molto potente)
- Server farm per creare un potente server virtuale
- Costi di gestione per offrire servizio



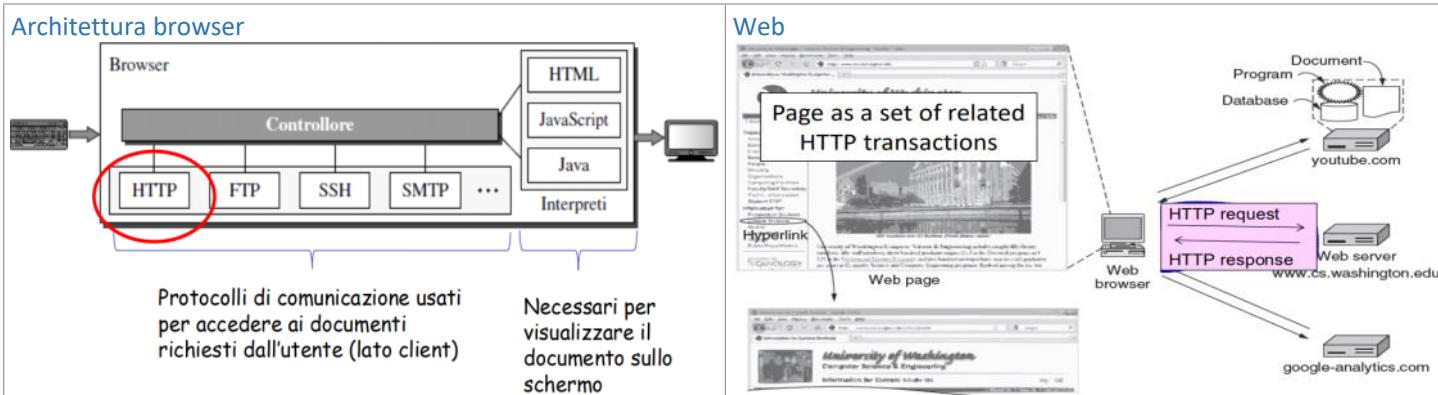
HTTP, Cookie, Web cache

lunedì 14 aprile 2025 17:01

World Wide Web (WWW)

Componenti del WWW:

- **Web client** (es. browser): interfaccia con l'utente
- **Web server** (es. Apache)
- **HTML**: linguaggio standard per pagine web
- **HTTP**: protocollo per la comunicazione tra client e server Web



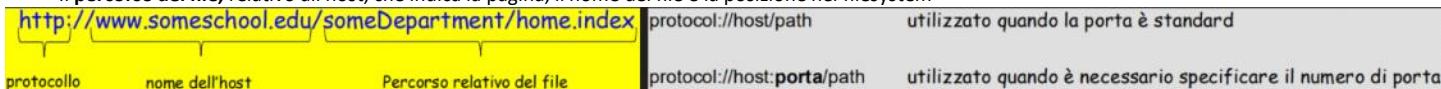
Terminologia:

- Una pagina web è costituita da **oggetti**
- Un oggetto può essere un file HTML, un'immagine, un file audio, ecc
- Una pagina web è formata da un **file base HTML** che include diversi oggetti referenziati
- Ogni oggetto è referenziato da un **URL (Uniform Resource Locator)**

URL (Uniform Resource Locator)

Per accedere ad un oggetto su internet si utilizza il suo **URL**, che è un nome univoco per accedere all'oggetto ed è composto da 3 parti:

- Il **protocollo**
- Il **nome dell'host** in cui è situata la pagina
- Il **percorso del file**, relativo all'host, che indica la pagina, il nome del file e la posizione nel filesystem

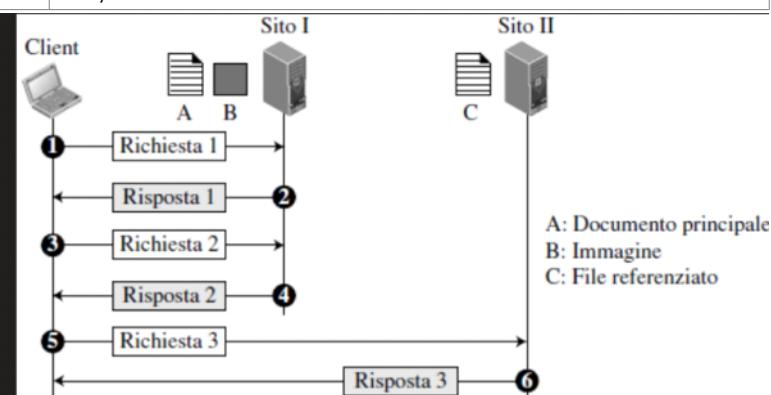
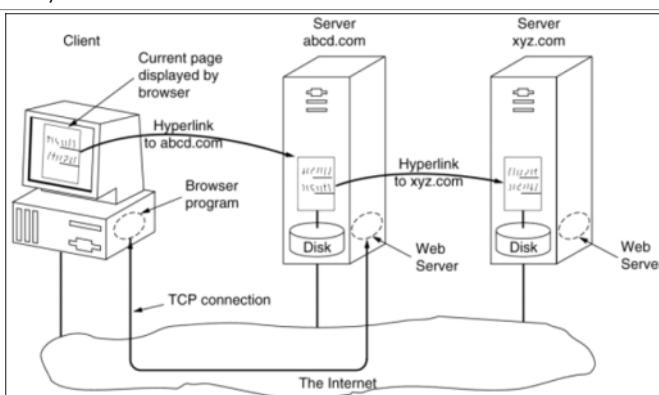


HTTP (Hypertext Transfer Protocol)

Protocollo a livello di applicazione del Web che definisce come i client richiedono pagine ai server e come questi le trasferiscono ai client

Modello client-server:

Client	Server
<p>Il browser richiede, riceve, "visualizza" gli oggetti del web.</p> <ol style="list-style-type: none"> 1) Determina l'URL ed estrae host e filename 2) Esegue una connessione TCP alla porta 80 dell'host indicato nell'URL 3) Invia richiesta per il file 4) Riceve il file dal server 5) Chiude la connessione 6) Visualizza il file 	<p>Il server invia oggetti in risposta ad una richiesta. Può servire più richieste, provenienti anche da client diversi</p> <ol style="list-style-type: none"> 1) Accetta una connessione TCP da un client 2) Riceve il nome del file richiesto 3) Recupera il file dal disco 4) Invia il file al client 5) Rilascia la connessione



Connessioni HTTP

Connessioni Persistenti

- Modalità di default
- Più oggetti possono essere trasmessi su una **singola connessione TCP** tra client e server
- La connessione viene chiusa quando rimane inattiva per un lasso di tempo

Connessioni Non Persistenti

- Un solo oggetto viene trasmesso su una **connessione TCP** (file HTTP, immagine, ecc)
- Ciascuna coppia richiesta/risposta viene inviata su una **connessione TCP separata**

(timeout) configurabile

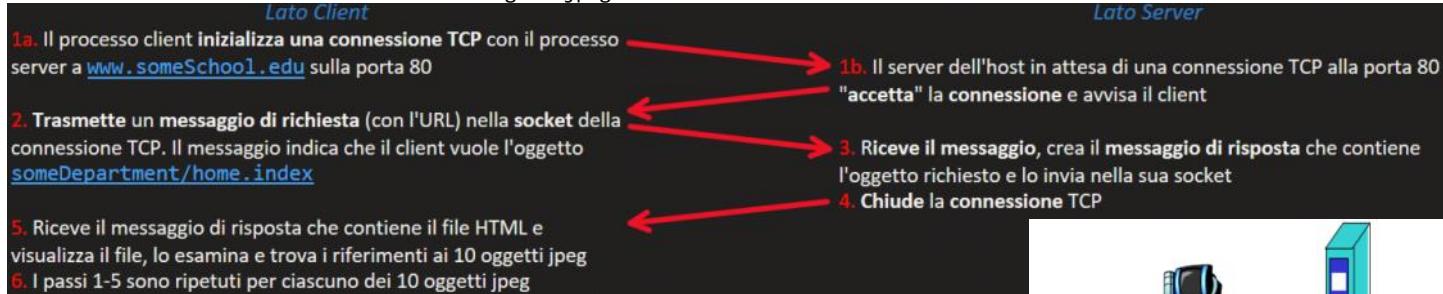
• Prima di inviare una richiesta al server, bisogna stabilire una connessione

Esempio Connessione Non Persistente

Supponiamo che l'utente immetta l'URL

www.someSchool.edu/someDepartment/home.index

che contiene testo e riferimenti a 10 immagini jpeg



Tempo di Risposta

RTT (Round Trip Time) è il tempo impiegato da un piccolo pacchetto per andare dal client al server e tornare al client. Include i ritardi di propagazione, accodamento e elaborazione.

Il tempo di risposta di trasmissione di un file si può definire come:

- Un RTT per inizializzare la connessione TCP
- Un RTT per la richiesta HTTP e i primi byte della risposta HTTP
- Tempo di trasmissione del file

$$\text{Tot} = 2 \cdot \text{RTT} + \text{tempo di trasmissione}$$



RTT Connessioni Persistenti

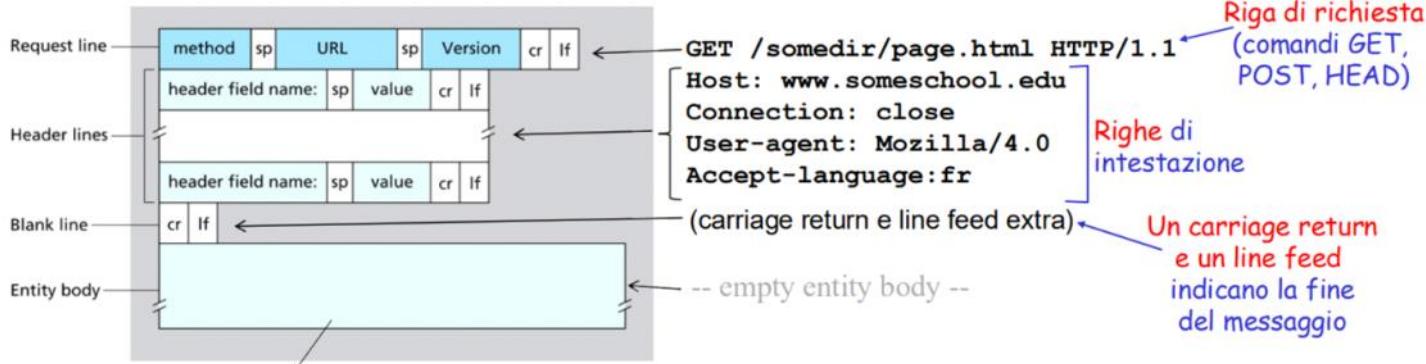
- Il server lascia la connessione TCP aperta dopo l'invio di una risposta
- I messaggi tra client/server sono trasmessi sulla connessione aperta
- Il client invia le richieste quando incontra un oggetto referenziato
- Un solo RTT di connessione per ogni oggetto (+ un RTT per ogni oggetto ricevuto dal server)

Svantaggi Connessioni Non Persistenti

- Richiede 2 RTT per oggetto
- overhead dell'OS per ogni connessione TCP
- I browser spesso aprono più connessioni TCP parallele per caricare gli oggetti referenziati

Richiesta HTTP

Differenza tra formato generale dei messaggi di richiesta HTTP ed un esempio di richiesta inviato dal client (in formato ASCII)



Campo vuoto per il GET, utilizzato per il POST

Upload dell'input di un form

Metodo POST:

Se la pagina contiene un form per l'input dell'utente, esso arriva al server nel corpo dell'entità.

Altrimenti si usa il metodo GET inserendo nel campo URL l'input richiesto

Intestazione	Descrizione
User-agent	Indica il programma client utilizzato
Accept	Indica il formato dei contenuti che il client è in grado di accettare
Accept-charset	Famiglia di caratteri che il client è in grado di gestire
Accept-encoding	Schema di codifica supportato dal client
Accept-language	Linguaggio preferito dal client
Authorization	Indica le credenziali possedute dal client
Host	Host e numero di porta del client
Date	Data e ora del messaggio
Upgrade	Specifica il protocollo di comunicazione preferito
Cookie	Comunica il cookie al server (verrà spiegato successivamente)
If-Modified-Since	Invia il documento solo se è più recente della data specificata

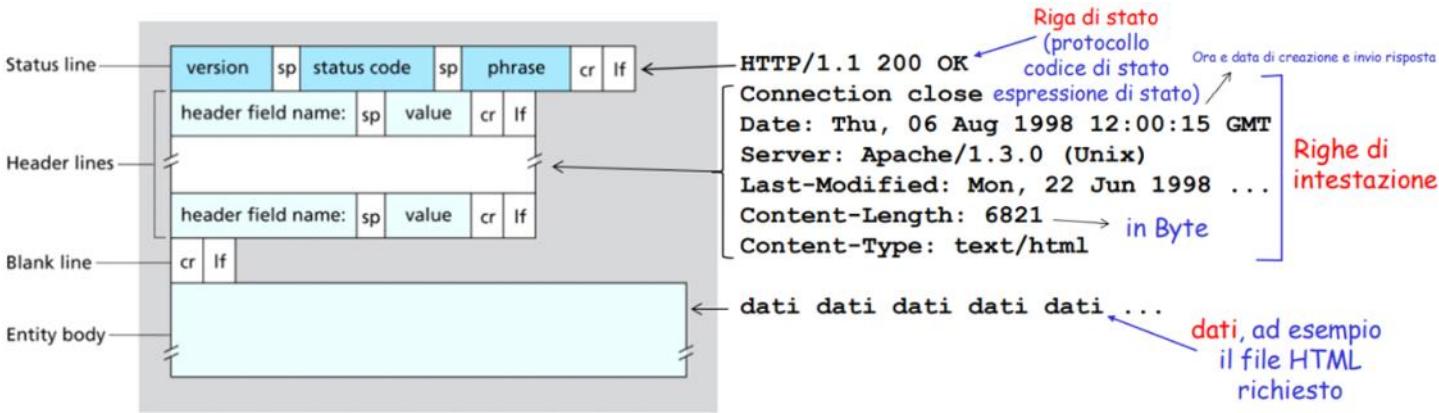
Intestazione della richiesta

GET	E' usato quando il client vuole scaricare un documento dal server. Il documento richiesto è specificato nell'URL. Il server normalmente risponde con il documento richiesto nel corpo del messaggio di risposta.
HEAD	E' usato quando il client non vuole scaricare il documento ma solo alcune informazioni sul documento (come ad esempio la data dell'ultima modifica). Nella risposta il server non inserisce il documento ma solo degli header informativi.
POST	E' usato per fornire input al server (contenuto dei campi di un form). L'input arriva al server nel corpo dell'entità.
PUT	E' utilizzato per memorizzare un documento nel server. Il documento viene fornito nel corpo del messaggio e la posizione di memorizzazione nell'URL.

Per inviare info al server
E' possibile anche usare GET
GET www.somesite.com/animalsearch?monkeys&banana

Tipi di Metodi

Risposta HTTP



Intestazione	Descrizione
Date	Data corrente
Upgrade	Specifica il protocollo preferito
Server	Indica il programma server utilizzato
Set-Cookie	Il server richiede al client di memorizzare un cookie
Content-Encoding	Specifica lo schema di codifica
Content-Language	Specifica la lingua del documento
Content-Length	Indica la lunghezza del documento
Content-Type	Specifica la tipologia di contenuto
Location	Chiede al client di inviare la richiesta a un altro sito
Last-modified	Fornisce data e ora di ultima modifica del documento

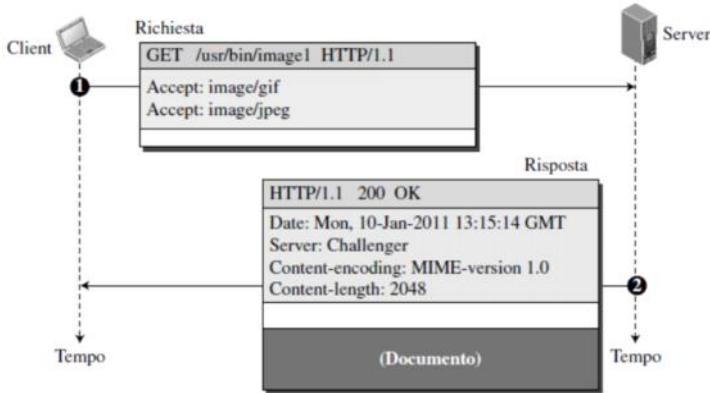
Intestazione della risposta

Code	Meaning	Examples
1xx	Information	100 = server agrees to handle client's request
2xx	Success	200 = request succeeded; 204 = no content present
3xx	Redirection	(301) = page moved; 304 = cached page still valid
4xx	Client error	403 = forbidden page; 404 = page not found
5xx	Server error	(500) = internal server error; 503 = try again later

Alcuni codici di stato della risposta

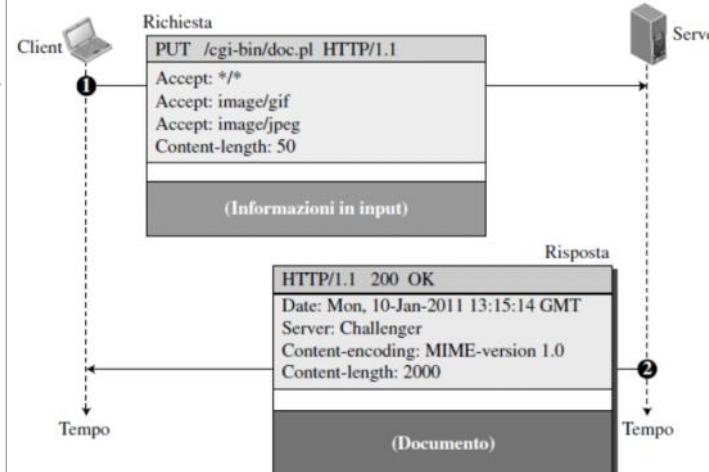
Esempio GET

- Client usa **GET** per **ottenere** un immagine nel percorso "/usr/bin/image1".
- La riga di richiesta contiene il **metodo** (GET), l'**URL** e la **versione** (1.1) del protocollo HTTP. L'intestazione contiene le specifiche delle immagini accettate dal client (GIF e JPEG). Il messaggio di richiesta non ha corpo
- Il messaggio di risposta contiene la riga di stato e quattro righe di intestazione che contengono la data, il server, il metodo di codifica del contenuto (MIME) e la lunghezza del contenuto
- Il corpo del messaggio segue l'intestazione



Esempio PUT

- Client usa **PUT** per pubblicare una pagina Web da pubblicare sul server
- La riga di richiesta contiene il **metodo** (PUT), l'**URL** e la **versione** (1.1) del protocollo HTTP. L'intestazione è costituita da quattro righe e il corpo contiene la pagina Web inviata
- Il messaggio di risposta contiene la riga di stato e quattro di intestazione
- Il documento creato è incluso nel corpo del messaggio di risposta



Cookie

HTTP è un **protocollo "senza stato"** (**stateless**). Il server una volta servito il client se ne dimentica (non mantiene informazioni sulle richieste fatte). Per tenere **traccia dell'utente**, invece di usare gli IP (es. poiché un utente può lavorare su pc condivisi), si usa il meccanismo **Cookie** (RFC 6265). Il meccanismo dei **Cookie** è un modo per **creare una sessione** di richieste/risposte HTTP **"con stato"** (**stateful**).

Ese: una sessione può essere utilizzata per creare uno "shopping cart" o un giornale online può presentare contenuti basati sulle letture precedenti

Sessione

Esistono diversi tipi di sessione in base al tipo di informazioni scambiate e in base al sito. Le caratteristiche generali sono:

1. Ogni sessione ha un **inizio** e una **fine**
2. Ogni sessione ha un **tempo di vita relativamente corto**
3. Sia il client che il server possono chiudere la sessione
4. La sessione è **implicita** nello scambio di informazioni di stato

Sessione vs Connessione: Per sessione **non** si intende una **connessione persistente** ma una **sessione logica** creata da richieste e risposte HTTP. Una sessione può essere creata su connessioni persistenti e non persistenti.

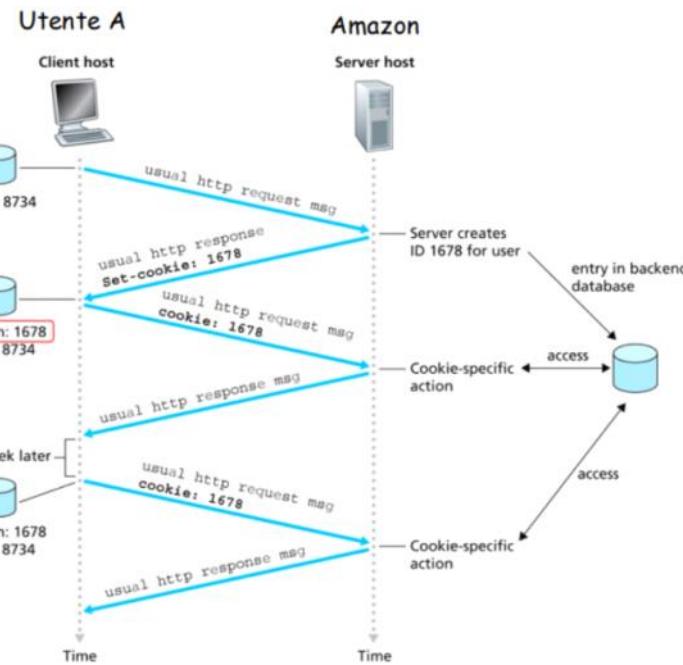
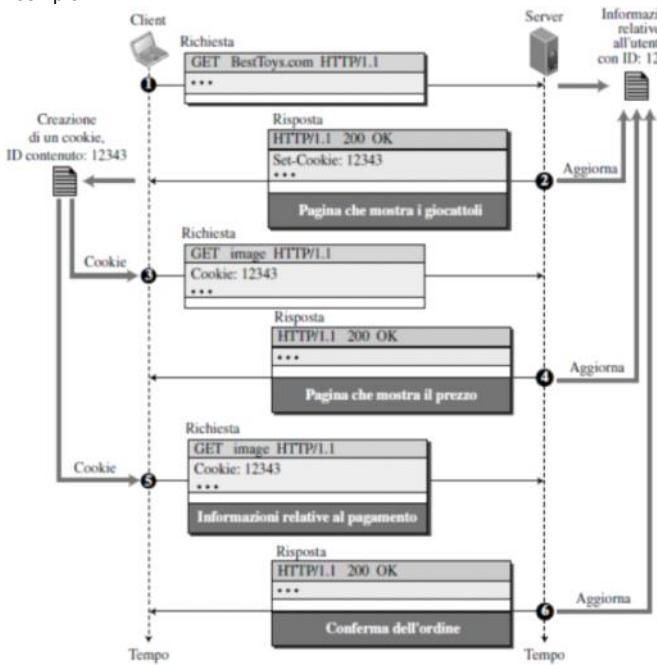
- Il server mantiene le informazioni riguardanti l'utente su un file e gli **assegna un identificatore di sessione (SID)** che viene fornito alla prima richiesta su quel sito (quindi se l'utente non ha già un cookie per quel sito) (es Server → User: Set-Cookie: SID=31d4d96e407aad42)
- Il client, ogni volta che manda una richiesta al server fornisce il suo identificatore (cookie) (es User → Server: SID=31d4d96e407aad42)
- Il server, tramite il cookie ricevuto dall'utente, accede al file relativo e fornisce risposte personalizzate

Vengono utilizzati 4 componenti:

1. Una riga di intestazione nel messaggio di risposta HTTP
2. Una riga di intestazione nel messaggio di richiesta HTTP
3. Un file cookie mantenuto sul client e gestito dal browser

4. Un database sul server per mantere l'ID per l'utente

Esempio



Nell'intestazione **Set-Cookie** si può impostare il tempo di vita di un cookie con **Max-Age=seconds**. Dopo i secondi impostati, il client dovrebbe rimuovere il cookie. Il server chiude una sessione inviando al client un intestazione Set-Cookie col messaggio Max-Age=0 (quindi deve essere rimosso subito)

I cookie possono mantenere:

- Autorizzazioni
- Carta per acquisti
- Preferenze dell'utente
- Stato della sessione dell'utente (email)

Web Caching

Il caching è una tecnica per **migliorare le prestazioni del server**, salvando le pagine richieste per riutilizzarle in seguito senza doverle richiedere al server. Il caching può essere eseguito da:

- Browser
- Proxy

Browser Caching

Il browser mantiene una cache delle pagine visitate, che è personalizzabile dall'utente.

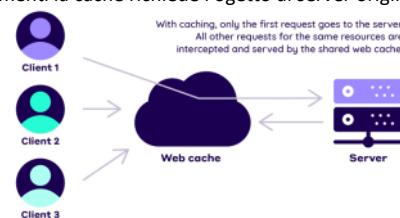
Esistono vari meccanismi per la gestione della cache locale:

- L'utente impone il num. di giorni, dopo i quali la cache viene svuotata
- La pagina può essere mantenuta in cache in base alla sua ultima modifica (es modificata un ora prima → mantenuta un ora o un giorno, un mese, ecc)
- Si possono utilizzare informazioni nei campi intestazioni dei messaggi per gestire la cache:
 - Es: campo **Expires** indica la scadenza dopo la quale la pagina è considerata obsoleta (non sempre rispettato dal browser)

Server Proxy

Viene soddisfata la richiesta del client senza coinvolgere il server d'origine

- Il server proxy ha una mem. per mantenere copie delle pagine visitate
- Il browser si può configurare per inviare richieste alla cache
- Il browser trasmette tutte le richieste HTTP alla cache
 - Se l'oggetto esiste nella cache: la cache fornisce l'oggetto
 - Altrimenti la cache richiede l'oggetto al server originale e lo inoltra



Efficienza Cache Web

La cache:

- Riduce i tempi di risposta alle richieste dei client
- Riduce il traffico sul collegamento di accesso a Internet
- Internet arricchita di cache consente ai provider meno efficienti di fornire dati con efficacia (tipicamente è installata da ISP (università, aziende, ecc))

Esempi:

- Dimensione media oggetto: 1Mb
- Frequenza media richieste dai browser ai server = 15 richieste al secondo
- Ritardo per recuperare un oggetto su Internet (**Internet Delay**) = 2 sec
- Il tempo totale di risposta (**total delay**) = LAN delay + access delay + Internet delay
- **Peso richieste (PR)** = $15\text{req/s} * 1\text{Mb/req} = 15\text{Mbps}$

Assenza di Cache 1

Uso LAN = PR/100Mbps = 0.15 (15%) (utilizzo basso → ritardo nell'ordine delle decine di ms)

Uso collegamento d'accesso = PR/15Mbps = 1 (100%) (canale saturato, si formano code → ritardo nell'ordine dei minuti)

Intensità → 1: ritardo si fa consistente

Total delay = 2sec + msec + min

Assenza di Cache 1

Soluzione possibile: Aumentare larghezza di banda del collegamento di accesso a 100Mbps

Uso LAN = PR/100Mbps = 15%

Uso collegamento d'accesso = PR/100Mbps = 15%

Total delay = 2sec + msec + msec (meglio)

Però non sempre attuabile ed è costoso aggiornare il collegamento

Assenza di Cache 1

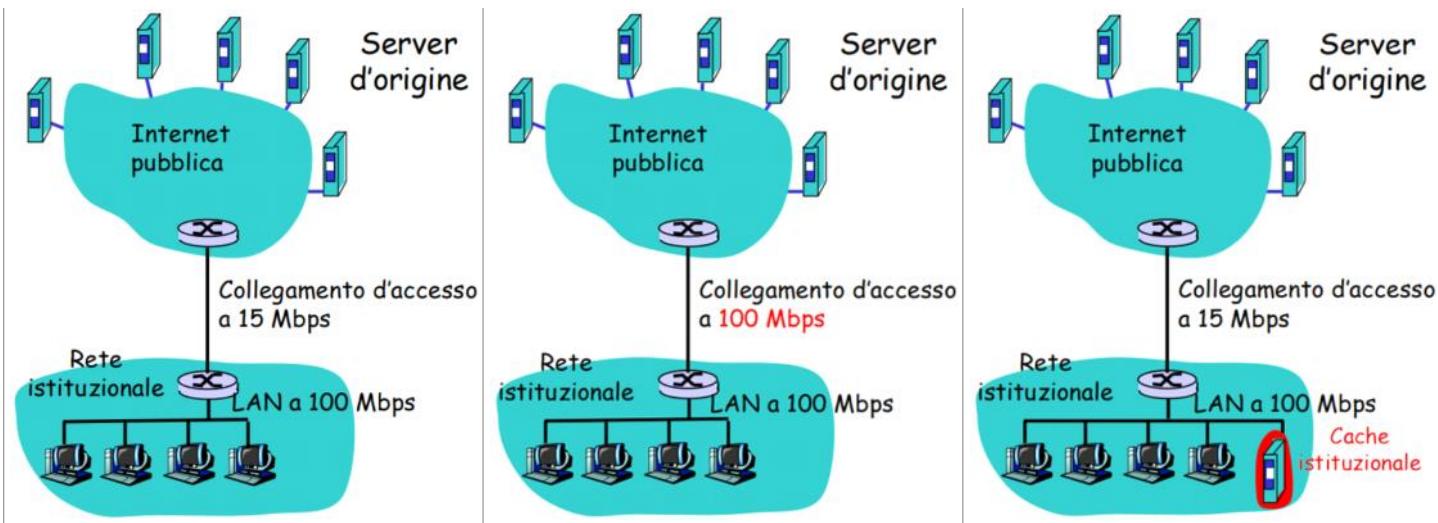
Soluzione possibile: Installare la cache con una **percentuale di successo (hit rate)** pari a 0.4 (40%)

40% delle richieste soddisfatte quasi subito (≈ 10ms)

60% delle richieste fatto al server d'origine

Uso del collegamento d'accesso ridotto al 60%, determinando ritardi trascurabili (≈ 10ms)

Total delay = $0.6 * 2.01\text{sec} + 0.4 * 0.01\text{sec} \approx 1.2\text{ sec}$



Inserimento Oggetto in Cache

- Client invia il messaggio di richiesta HTTP alla cache
GET /page/figure.gif
Host: www.sito.com
- La cache **non ha l'oggetto** → invia la richiesta HTTP al Server → Il server **invia risposta HTTP alla cache**
HTTP/1.1 200 OK
Date: ...
...
Last-Modified: Wed, 2 Jul 2008 09:23:24
(data data ...)
- La cache **memorizza la pagina** per richieste future mantenendo l'ultima data di modifica
- La cache invia la risposta al client

Validazione dell'oggetto

- Client invia il messaggio di richiesta HTTP alla cache
GET /page/figure.gif
Host: www.sito.com
- La cache **ha l'oggetto** → prima di inviarlo, **verifica** che non sia scaduto (**modificato sul server di origine**)
- La cache esegue una richiesta verso il Web server che mantiene l'oggetto, per verificare la validità mediante il metodo
GET condizionale
 - Usa metodo GET
 - Include una riga di intestazione **If-Modified-Since**

GET Condizionale

Obiettivo: non inviare l'oggetto se la cache ha una copia aggiornata dell'oggetto

Cache

Specifica la data della copia dell'oggetto nella richiesta HTTP
If-modified-since: <data>

Server

La risposta non contiene l'oggetto se la copia nella cache è **aggiornata**
HTTP/1.1 304 Not Modified

cache

server

*Caso 1:
oggetto
non
modificato*

cache

server

*Caso 2:
oggetto
modificato*

DNS

martedì 22 aprile 2025 17:03

Gli Host in Internet hanno nomi (hostname) (es: google.com, uniroma1.it, ecc) che sono facili da ricordare ma forniscono poche informazioni sulla collocazione degli host in Internet (w3.uniroma1.it ci dice che l'host si trova probabilmente in Italia ma non dove)

Indirizzi IP per gli host: Si usa un IP (32 bit) per indirizzare i datagrammi (più appropriato per le macchine) (es 121.34.230.94)

- Consiste di 4 byte: stringa in cui ogni punto separa un byte espresso con un num. decimale compreso tra 0 e 255 (0.0.0.0 → 255.255.255.255)
- Presenta una **struttura gerarchica**: rete di appartenenza e indirizzo nodo

DNS (Domain Name System) (RFC 1034, 1035)

Il DNS è un sistema per tradurre gli hostname in indirizzi IP (es: www.google.com → 192.168.1.1)

Il problema è che esistono 2^{32} indirizzi IP (\approx 4 miliardi) da memorizzare in coppia al loro hostname

- Memorizzazione → Database distribuito, implementato in una **gerarchia di server DNS**
- Accessibilità → Protocollo a livello applicazione che consente agli host di interrogare il database per risolvere gli hostname (traduzione IP/hostname)

Il DNS è usato dagli altri protocolli di liv. applicazione (HTTP, SMTP, FTP) per tradurre hostname in IP

Esso è un **protocollo** del liv. applicazione eseguito dai client, e usa il protocollo di **trasporto end-to-end UDP** e indirizza la porta 53

Esempio Interazione con HTTP

Un Client di un host utente richiede la URL www.someschool.edu

1. L'host esegue il lato client dell'app DNS
2. Il browser estrae il nome dell'host (www.someschool.edu) dall'URL e lo passa al lato client dell'app DNS
3. Il client DNS invia una query contenente l'hostname ad un server DNS
4. Il client DNS riceve una risposta che include l'indirizzo IP corrispondente all'hostname
5. Ottenuto l'indirizzo IP dal DNS, il browser può dare inizio alla connessione TCP verso il server HTTP localizzato a quell'IP

Aliasing

L'**host aliasing** permette di associare nomi più semplici da ricordare (alias) ad un host con un nome complesso

Esempio:

- www.relay1.west-coast.enterprise.com è un hostname **canonico**
- www.enterprise.com è l'**alias** (più facile da ricordare)

Il DNS si può invocare da un app per l'hostname canonico di un sinonimo così come l'IP

Mail server aliasin: spesso i mail server e il web server di una società hanno lo stesso alias, ma nomi canonici diversi

Distribuzione del Carico

DNS viene usato per **distribuire il carico** tra server replicati (es: web server)

I siti con molto traffico vengono **replicati su più server**, e ciascuno di questi gira su un sistema terminale diverso e presenta un IP differente

- Hostname canonico associato ad un insieme di indirizzi
- Il DNS contiene l'insieme degli IP
- Quando un client effettua una richiesta DNS per un nome mappato in un insieme di indirizzi, il server risponde con l'insieme di indirizzi ma variando l'ordinamento ad ogni risposta
- La rotazione DNS distribuisce il traffico sui server replicati

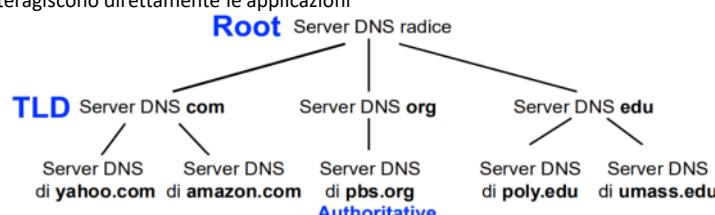
Gerarchia Server DNS

Poiché non si può utilizzare un singolo server DNS (poiché non scalabile) bisogna usare una gerarchia di server, però ogni server non può contenere il mapping per tutti gli host di Internet, quindi viene distribuito su svariati server DNS.

Si organizzano le informazioni in base al dominio degli host (.it, .com, .edu, ecc) ed esistono 3 classi di server DNS:

- Root
- Top-level domani (TLD)
- Authoritative

Ci sono poi i server DNS **locali** con cui interagiscono direttamente le applicazioni



Esempio: Il client vuole l'IP di www.amazon.com

- Interroga il **server radice (root)** per trovare il server DNS com
- Interroga il **server TLD com** per ottenere il server DNS amazon.com per ottenere l'IP di www.amazon.com

Root (Server DNS Radice)	TLD (Top-Level-Domain)	Authoritative (Server di Competenza)
1. Viene contattato da un server DNS locale 2. Contatta un Server DNS TLD se non conosce la mappatura 3. Ottiene la mappatura 4. Restituisce la mappatura al server DNS locale	Si occupa dei domini com, org, net, ecc e di domini locali di alto livello come it, uk, jp, ecc Es: <ul style="list-style-type: none">• Verisign gestisce il dominio com• Registro.it a Pisa gestisce il dominio it	Ogni organizzazione dotata di host Internet pubblicamente accessibili deve fornire i record DNS di pubblico dominio che mappano i nomi di tali host in IP. Possono essere mantenuti dall'organizzazione (università) o da un service provider

Il **Server DNS locale** non appartiene strettamente alla gerarchia dei server e ciascun ISP (università, ISP residenziale, ecc) ha un server DNS locale (detto "default name server").

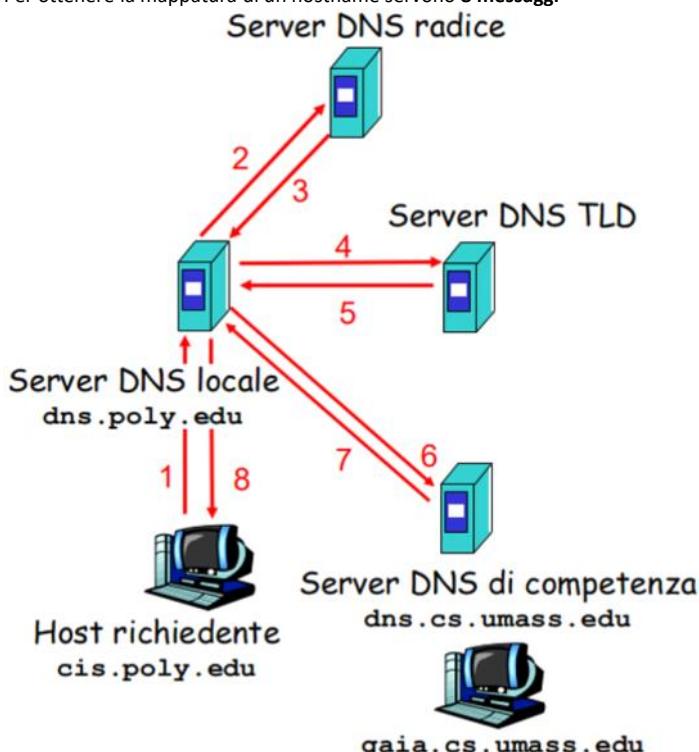
Quando un host effettua una richiesta DNS, la query viene inviata al suo server DNS locale, che opera da proxy e inoltra la query nella gerarchia di server DNS

Un host per recuperare l'IP di un host può usare una **query iterativa o ricorsiva**. Es: l'host www.cis.poly.edu vuole l'IP di www.gaia.cs.umass.edu

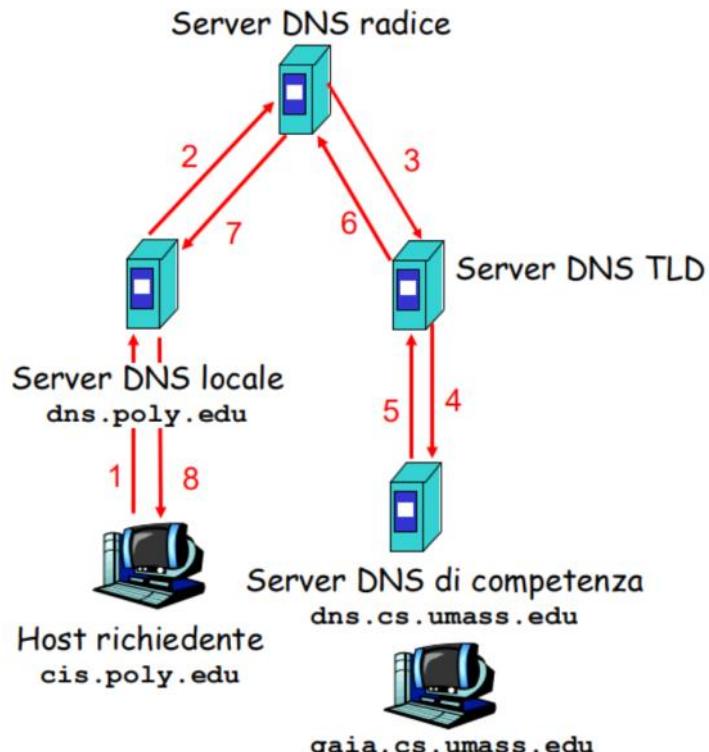
Query Iterativa

Query Ricorsiva

Il server contattato risponde col nome del server da contattare
Per ottenere la mappatura di un hostname servono **8 messaggi**



affida il compito di tradurre il nome al server DNS contattato



DNS Caching

Un Server DNS salva nella **cache** la **mappatura** per migliorare le prestazioni di ritardo, e dopo un certo periodo esse vengono eliminate. Tipicamente un DNS locale memorizza gli IP dei server TLD (ma anche quelli di competenza) (quindi i DNS root non vengono visitati spesso)

DNS Record e Messaggi

Record

Il mapping è mantenuto nei database come un **resource record (RR)** i quali mantengono un mapping (es. tra hostname e IP, alias e nome canonico, ecc) e vengono spediti tra server e host richiedenti all'interno dei **messaggi DNS**, il quale può contenere anche **più RR**

Formato RR: (Name, Value, Type, TTL)

Tempo residuo di vita

Type=A

Hostname → IP address

- ❖ **name** è il nome dell'host
- ❖ **value** è l'indirizzo IP

Es. (relay1.bar.foo.com, 45.37.93.126, A)

Type=MX

Alias → mail server canonical name

- ❖ **value** è il nome canonico del server di posta associato a **name**

Es. (foo.com, mail.bar.foo.com, MX)

Type=NS

Domain name → Name Server

- ❖ **name** è il dominio (ad esempio foo.com)
- ❖ **value** è il nome dell'host del server di competenza di questo dominio

Es. (foo.com, dns.foo.com, NS)

Type=CNAME

Alias → Canonical Name

- ❖ **name** è il nome alias di qualche nome "canonico" (nome vero)
- ❖ **value** è il nome canonico

Es. (foo.com, relay1.bar.foo.com, CNAME)

Tipo	Interpretazione del campo valore
A	Indirizzo IPv4 a 32 bit (si veda il Capitolo 4)
NS	Identifica i server autoritativi di una zona
CNAME	Indica che un nome di dominio è un alias (un nome alternativo) per il nome di dominio ufficiale (detto anche canonico)
SOA	Specifica una serie di informazioni autoritative riguardo una zona
MX	Indica il server di posta del dominio
AAAA	Indirizzo IPv6 (si veda il Capitolo 4)

Esempio:

- Server di competenza per un hostname:
 - Contiene record **tipo A** per l'hostname (es: (corsi.di.uniroma1.it, 131.111.45.68, A))
- Server non di competenza per un hostname:
 - Contiene record **tipo NS** per il dominio che include l'hostname
 - Contiene record **tipo A** con l'IP del server DNS nel campo **value** del record NS
 - Esempio: Un server TLD it non è competente per l'host "corsi.di.uniroma1.it" e contiene:
 - (uniroma1.it, dns.uniroma1.it, NS)
 - (dns.uniroma1.it, 128.119.40.111, A)

Messaggi

Protocollo DNS: domande (query) e messaggi di risposta, entrambi con lo stesso formato

- **Identificazione:** numero di 16 bit per la domanda; la risposta alla domanda usa lo stesso numero

- **Flag:**
 - ❖ domanda o risposta
 - ❖ richiesta di ricorsione
 - ❖ ricorsione disponibile
 - ❖ risposta di competenza (il server è competente per il nome richiesto)

- **Numero di:** numero di occorrenze delle quattro sezioni di tipo dati che seguono

Identificazione	Flag
Numero di domande	Numero di RR di risposta
Numero di RR autorevoli	Numero di RR addizionali
Domande (numero variabile di domande)	
Risposte (numero variabile di record di risorsa)	
Competenza (numero variabile di record di risorsa)	
Informazioni aggiuntive (numero variabile di record di risorsa)	

Campi per il nome richiesto e il tipo di domanda (A, MX)

RR nella risposta alla domanda
Più RR nel caso di server replicati

Record per i server di competenza

Informazioni extra che possono essere usate

Nel caso di una risposta MX, il campo di risposta contiene il record MX con il nome canonico del server di posta, mentre la sezione aggiuntiva contiene un record di tipo A con l'indirizzo IP relativo all'hostname canonico del server di posta

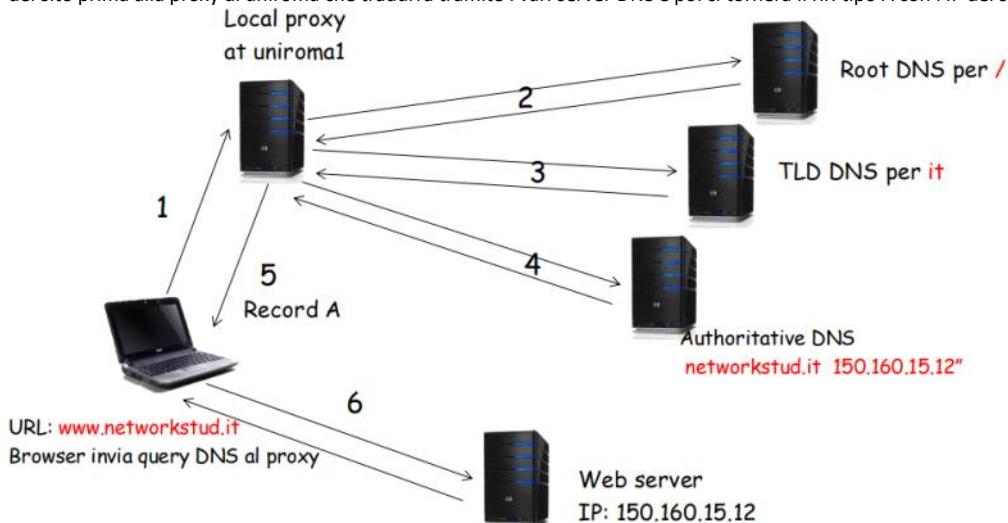
Inserire Record nel Database DNS

Esempio: vogliamo aggiungere un nuovo dominio della nostra società "Network Stud"

Si possono aggiungere nuovi domini al DNS contattando un **registrar** che verifica l'unicità del dominio richiesto e lo inserisce nel database

- **Registriamo il nome networkstud.it** presso **registrar** (www.registro.it)
 - Inseriamo nel server di competenza (es. nostro server DNS www.dns1.networkstud.it) un record **tipo A** per www.networkstud.it e un record **tipo MX** per networkstud.it per indicare il server di posta elettronica
 - **A record:** www.networkstud.it → 150.160.15.12 (indirizzo server web)
 - **MX record:** networkstud.it → mail.networkstud.it (posta elettronica)
 - **CNAME record:** mail.networkstud.it → mail.google.com (se usiamo Gmail)
 - Forniamo al registrar i nomi e gli IP dei server DNS di competenza (primario e secondario)
 - Registrar inserisce due RR nel server TLD it:
 - (networkstud.it, dns1.networkstud.it, NS)
 - (dns1.networkstud.it, 212.212.212.1, A)

Gli utenti otterranno l'IP del sito prima alla proxy di uniroma che tradurrà tramite i vari server DNS e poi ci tornerà il RR tipo A con l'IP del server



UDP viene usato rispetto a TCP per due motivi principali:

- **Less overhead**
 - Messaggi corti
 - Tempo instantaneo di setup per la connessione. Molto meglio rispetto al tempo di connessione di TCP
 - Viene scambiato un solo messaggio tra due server (se si usasse TCP ogni volta dovremmo mettere su la connessione)
- Se un messaggio non ha risposta entro un timeout viene semplicemente ri-inviato dal resolver

FTP e Posta Elettronica

domenica 27 aprile 2025 16:14

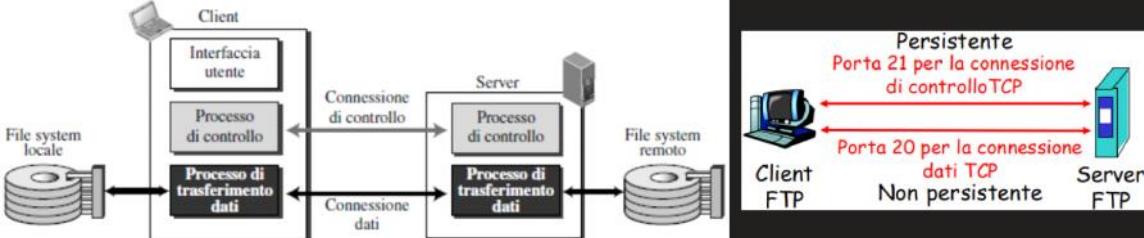
File Transfer Protocol (FTP)

L'FTP è un protocollo di **trasferimento file** da/a un host remoto. Esistono diversi comandi per eseguire più opzioni sull'host locale/remoto, ma i principali sono:

- **ftp HostName**: accedere ed essere autorizzato a scambiare informazioni con l'host remoto (vengono richiesti nome utente e password)
- **ftp> get file1.txt**: trasferimento di un file da un host remoto al locale
- **ftp> put file2.txt**: trasferimento di un file da locale all'host remoto

Nel modello client/server, il **client** è colui che **inizia il trasferimento** (da/a un host remoto) e il **server** e l'**host remoto**. La connessione è eseguita così:

- L'utente fornisce nome dell'host remoto → FTP client stabilisce una **connessione TCP** sulla **porta 21** con il **FTP server**
- Stabilita la connessione, il client fornisce nome e password che vengono inviate sulla connessione TCP come parte dei comandi
- Ottenuta l'autorizzazione dal server, il client può inviare i file dal file system locale a quello del server (o viceversa)



- **Connessione di controllo**: si occupa delle informazioni di controllo del trasferimento e, poiché usa regole molto semplici, lo scambio di informazioni è ridotto allo scambio di una riga di comando per ogni iterazione
- **Connessione dati**: si occupa del **trasferimento file**

Connessione di controllo

- Connessione sulla **porta 21** usata per inviare informazioni di controllo.
- Parte quando si richiede sul client **ftp HostName**
- Connessione di controllo è "fuori banda" (out of band), quindi invia le informazioni separatamente dai dati trasferiti
- Il Server FTP **mantiene lo "stato"** (es: directory corrente, autenticazione)
- Es. informazioni trasferite sulla connessione di controllo: ID utente, password, comandi cambio directory e per inviare(put)/ricevere(get) file

Connessione di dati

- Connessione dati TCP sulla **porta 20**
 - Parte quando il server riceve un comando per trasferire file (es: get, put)
 - Dopo il trasferimento di un file, il server chiude la connessione
- Quindi viene creata una connessione per ogni file trasferito nella sessione

Comandi e Risposte FTP

Ciascun comando del client è **seguito da una risposta** dal server (**codice di ritorno**)

Comandi comuni

Inviati come testo ASCII sul server remoto

- **USER username**
- **PASS password**
- **LIST**: elenca i file della **dir** corrente, la lista di file viene inviata dal server su una nuova connessione di dati
- **RETR filename**: **recupera (get)** un file dalla dir corrente remota
- **STOR filename**: **memorizza (put)** un file nella dir corrente remota

Comando	Argomenti	Descrizione
ABOR		Interruzione del comando precedente
CDUP		Sale di un livello nell'albero delle directory
CWD	Nome della directory	Cambia la directory corrente
DELE	Nome del file	Cancello il file
LIST	Nome della directory	Elenca il contenuto della directory
MKD	Nome della directory	Crea una nuova directory
PASS	Password utente	Password
PASV		Il server sceglie la porta
PORT	Numero di porta	Il client sceglie la porta
PWD		Mostra il nome della directory corrente
QUIT		Uscita dal sistema
RETR	Nome di uno o più file	Trasferisce uno o più file dal server al client
RMD	Nome della directory	Cancello la directory
RNTO	Nome (del nuovo) file	Cambia il nome del file
STOR	Nome di uno o più file	Trasferisce uno o più file dal client al server
USER	Identificativo	Identificazione dell'utente

Codici di ritorno comuni

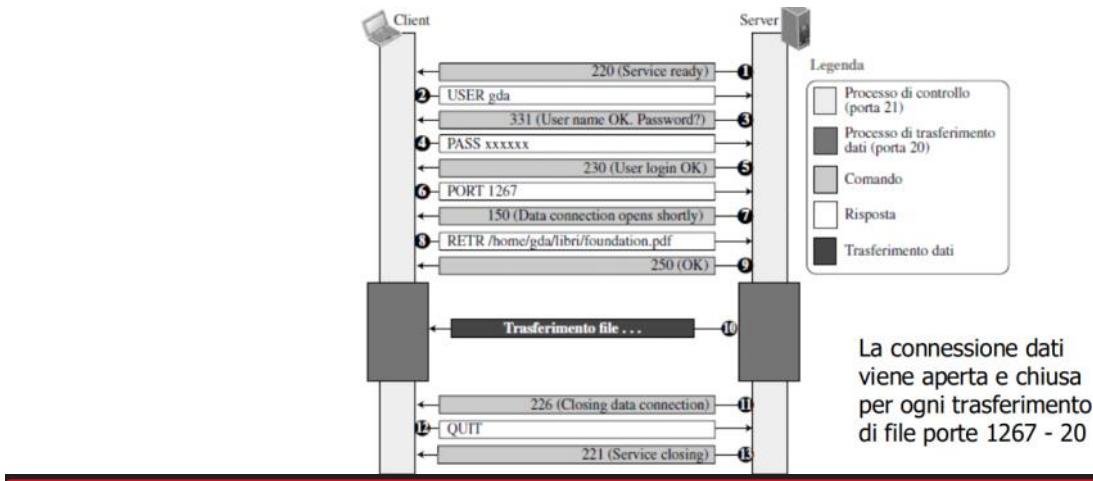
Codici di stato ed espressione (come in HTTP)

- **331 Username OK, password required**
- **125 data connection already open; transfer starting**
- **425 Can't open data connection**
- **452 Error writing file**

Le risposte sono composte da due parti:

- Num. di 3 cifre: indica il codice della risposta
- Un testo: contiene parametri necessari o informazioni supplementari

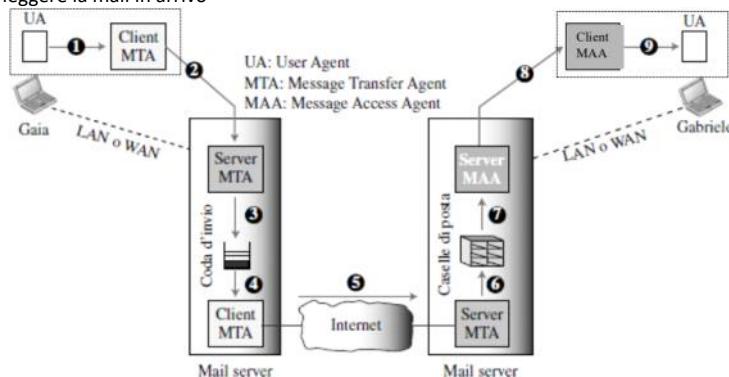
Codice	Descrizione	Codice	Descrizione
125	Connessione dati aperta	250	Azione sul file OK
150	Stato del file OK	331	Nome dell'utente OK; in attesa della password
200	Comando OK	425	Non è possibile aprire la connessione dati
220	Servizio pronto	450	Azione sul file non eseguita; file non disponibile
221	Servizio in chiusura	452	Azione interrotta; spazio insufficiente
225	Connessione dati aperta	500	Errore di sintassi; comando non riconosciuto
226	Connessione dati in chiusura	501	Errore di sintassi nei parametri o negli argomenti
230	Login dell'utente OK	530	Login dell'utente fallito



Posta Elettronica

Lo scenario classico è composto da tre componenti principali:

1. **User Agent**: usato per scrivere e inviare o leggere un messaggio
2. **Message Transfer Agent**: usato per trasferire il messaggio attraverso Internet
3. **Message Access Agent**: usato per leggere la mail in arrivo

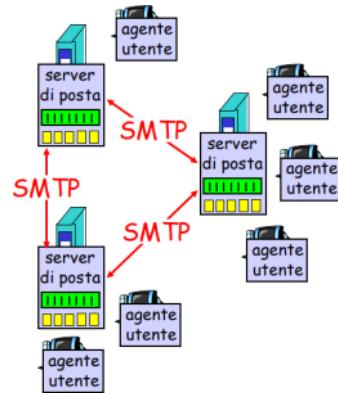


I messaggi in **uscita/arrivo** sono memorizzati sul server, mentre il messaggio **da inviare** viene inviato al **Message Transfer Agent**

Message Transfer Agent

Gli MTA comunicano tramite:

- **Server di posta**
 - Caselle di posta (mailbox) contenenti i messaggi in arrivo
 - Coda di messaggi da trasmettere
- **Protocollo SMTP (Simple Mail Transfer Protocol)**
 - Tra server di posta per inviare i messaggi di posta
 - Tra utente del mittente e il suo server di posta



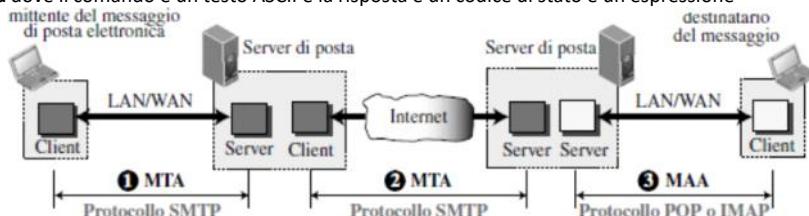
SMTP (Simple Mail Transfer Protocol) (RFC 5321)

È un protocollo che usa **TCP** per trasferire in modo affidabile i messaggi di posta elettronica dal client al server tramite la porta 25

Avviene un trasferimento **diretto**, dal server trasmittente al server ricevente, diviso in **tre fasi**:

- Handshaking
- Trasferimento messaggi
- Chiusura

L'interazione è a **comando/risposta** dove il comando è un testo ASCII e la risposta è un codice di stato e un'espressione



Il client SMTP (sull'host server di posta in invio) stabilisce una connessione TCP sulla **porta 25** verso il server SMTP (sull'host server di posta ricevente)

- Se il server è **attivo**, viene stabilita la connessione TCP
- Se il server è **inattivo**, il client riprova più tardi

Il client e il server effettuano una forma di handshaking (il client indica email del mittente e del destinatario) e il client invia il messaggio che viene ricevuto dal server

Se ci sono altri messaggi si usa la stessa connessione (**connessione persistente**) altrimenti il client richiede la chiusura della connessione

Esempio tra il client **crepes.fr** e il server **hamburger.edu**, subito dopo la connessione TCP

```

S: 220 hamburger.edu
C: HELO crepes.fr
S: 250 Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <rob@hamburger.edu>
S: 250 rob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection

```

Si ripete da qui per mail multiple

Messaggio di posta

Confronto con HTTP

- HTTP e SMTP sono usati per **trasferire file** da un host all'altro
- Connessione TCP:
 - HTTP **pull**: gli utenti scaricano i file e inizializzano le connessioni
 - SMTP **push**: il server di posta spedisce il file e inizializza la connessione
- Oggetti:
 - HTTP: ciascun oggetto è incapsulato nel messaggio di risposta
 - SMTP: più oggetti sono trasmessi in un solo messaggio

Note

- SMTP usa connessioni persistenti (ripete i passi da MAIL FROM:)
- SMTP richiede che il messaggio (intestazione e corpo) sia nel formato ASCII a 7 bit
- Il server SMTP usa **CRLF.CRLF** per determinare la fine del messaggio

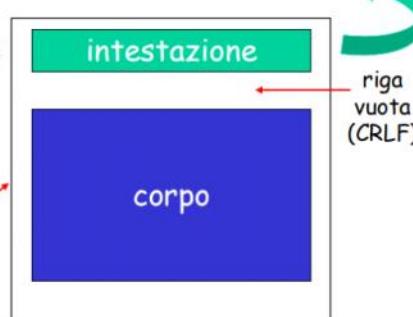
Formato dei messaggi di posta elettronica

To	indirizzo di uno o più destinatari.
From	indirizzo del mittente.
Cc	indirizzo di uno o più destinatari a cui si invia per conoscenza.
Bcc	blind Cc: gli altri destinatari non sanno che anche lui riceve il messaggio.
Subject	argomento del messaggio.
Sender	chi materialmente effettua l'invio (ad es. nome della segretaria).

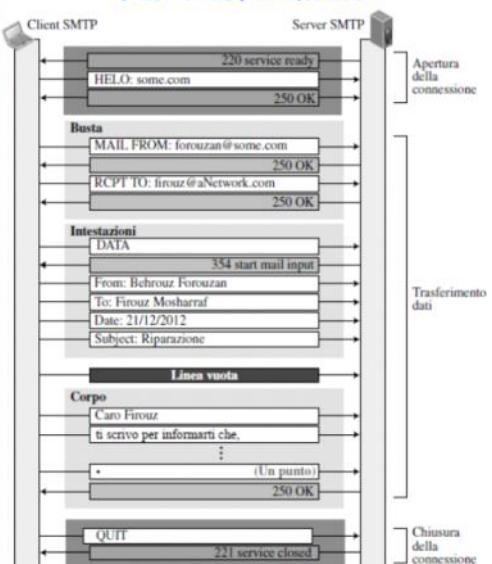
SMTP: protocollo per scambiare messaggi di posta elettronica

RFC 822: standard per il formato dei messaggi di testo:

- Righe di intestazione, per esempio
 - To:
 - From:
 - Subject:
 - differenti dai comandi SMTP!
- corpo
 - il "messaggio", soltanto caratteri ASCII



Fasi trasferimento



Formato del messaggio

Estensioni di messaggi multimediali

Per inviare contenuti diversi dal testo ASCII si usano intestazioni aggiuntive usando le estensioni **MIME**.

Alcune righe aggiuntive nell'intestazione indicano il tipo di contenuto MIME

Versione MIME
metodo usato per codificare i dati in ASCII
Tipo di dati multimediali, sottotipo, dichiarazione dei parametri
Dati codificati (corpo del messaggio)

```

From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg
base64 encoded data .....
.....base64 encoded data

```

Formato del messaggio ricevuto

Un'altra classe di righe di intestazione viene inserita dal server di ricezione SMTP. Es: il server di ricezione aggiunge **Received**, specificando il nome del server che ha inviato (**from**) il messaggio e che lo ha ricevuto (**by**) e l'orario di ricezione

```

Received: from crepes.fr by hamburger.edu; 12 Oct 98 15:27:39
GMT
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg
base64 encoded data .....
.....base64 encoded data

```

Protocolli di Accesso alla Posta

Siccome SMTP è un protocollo **push** che consegna/memorizza il messaggio sul server del destinatario, l'user agent destinatario non può usarlo per leggere il messaggio, perché deve eseguire un'**operazione pull**. Si usano quindi altri protocolli:

- **POP3** (Post Office Protocol): autorizzazione (agente <-> server) e download
- **IMAP** (Internet Mail Access Protocol): più funzioni e permette di manipolare i messaggi memorizzati sul server
- **HTTP**: es Gmail, Hotmail, ecc

POP3 (RFC 1939)

POP3 permette al client ricevente di aprire una connessione TCP verso il server di posta

S: +OK POP3 server ready

- **IMAP** (Internet Mail Access Protocol): più funzioni e permette di manipolare i messaggi memorizzati sul server
- **HTTP**: es Gmail, Hotmail, ecc

POP3 (RFC 1939)

POP3 permette al client ricevente di aprire una connessione TCP verso il server di posta sulla porta 110.

Esistono due modalità di visualizzazione dei messaggi:

- "scarica e cancella": elimina i messaggi dal server ma li mantiene sul client.
Quindi non si possono rileggere messaggi se si cambia client
- "scarica e mantieni": mantiene i messaggi sul server e sul client

Es "scarica e cancella": quando la connessione è stabilita si procede in 3 fasi:

- Autorizzazione**: L'user agent invia nome e password per essere identificato
- Transazione**: L'user agent recupera i messaggi
- Aggiornamento**: Dopo il **QUIT** inviato dal client, vengono cancellati i messaggi
- marcati per la rimozione

POP3 è un protocollo "stateless" e non fornisce all'utente nessuna procedura per creare cartelle remote e assegnare loro messaggi, l'utente può crearle solo localmente

IMAP (RFC 3501)

Con POP3, il server non può accedere alle cartelle locali, quindi le email scaricate e rimosse dal server non sono visibili su un altro pc

IMAP invece mantiene i messaggi sul server e permette all'utente di organizzare i messaggi in cartelle **salvando lo stato dell'utente** tra le varie sessioni. Quindi i nomi delle cartelle e i diversi identificatori dei messaggi sono salvati tra le diverse sessioni, quindi visibili anche se si accede da un altro pc.

Un server IMAP associa a una cartella (INBOX) ogni messaggio arrivato e permette all'utente di creare cartelle e spostare i messaggi tra le cartelle e effettuare ricerche in cartelle remote.

HTTP

Alcuni mail server forniscono l'accesso all'email direttamente via web col protocollo HTTP, quindi l'user agent diventa il browser stesso.

L'utente accede, comunica e riceve messaggi col suo mailbox tramite HTTP. SMTP rimane comunque il protocollo di comunicazione tra mail server

```

S: +OK POP3 server ready
C: user rob
S: +OK
C: pass hungry
S: +OK user successfully logged on
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off

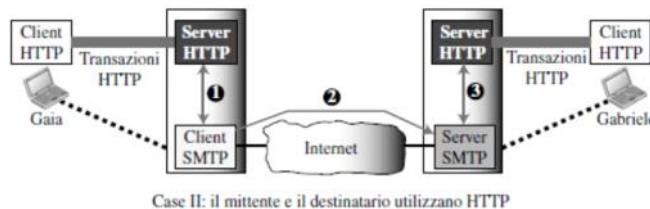
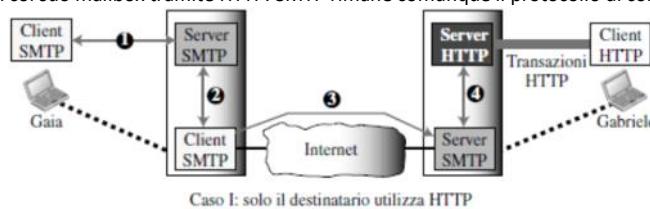
```

Fase di autorizzazione

- Comandi del client:
 - ♦ user: dichiara il nome dell'utente
 - ♦ pass: password
- Risposte del server
 - ♦ +OK
 - ♦ -ERR

Fase di transazione, client:

- list: elenca i numeri dei messaggi
- retr: ottiene i messaggi in base al numero
- dele: cancella
- quit



Livello Trasporto

lunedì 28 aprile 2025 12:08

I protocolli di trasporto vengono eseguiti nei **sistemi terminali** e forniscono la **comunicazione logica** tra processi applicativi di host differenti.

- Host trasmittente: **incapsula** il messaggio in **segmenti** e li passa al livello di rete
- Host ricevente: **decapsula** i segmenti in **messaggi** e li passa al livello di applicazione

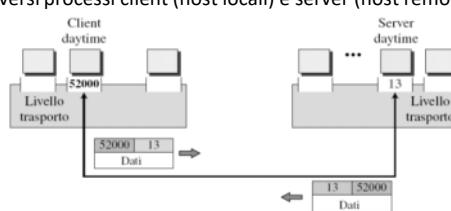
Il liv. di **rete** invia i segmenti tra gli host (**comunicazione tra host**) mentre il liv. di **trasporto** traduce i segmenti e li invia ai processi (**comunicazione tra processi**)

Poiché gli OS moderni sono **multiutente e multiprocesso** (diversi processi client (host locali) e server (host remoti) attivi), per poter stabilire una comunicazione tra due processi è necessario un metodo per individuarli:

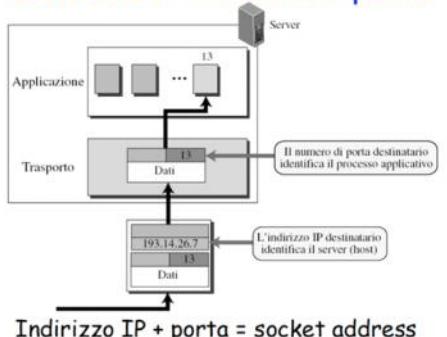
- Host **locale**
- Host **remoto**
- Processo **locale**
- Processo **remoto**

Host → IP

Processo → N° porta

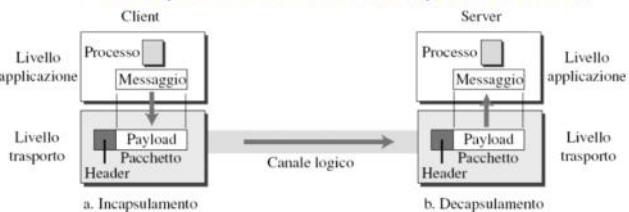


Indirizzi IP vs num. di porta



Indirizzo IP + porta = socket address

Incapsulamento/Decapsulamento



I pacchetti a livello di trasporto sono chiamati **segmenti** (TCP) o **datagrammi utente** (UDP)

Multiplexing/Demultiplexing

Il **multiplexing** raccoglie i dati da vari socket dei processi e li incapsula per poi inviarli al liv. di rete

Il **demultiplexing** al contrario raccoglie i segmenti ricevuti e li consegna alle socket appropriate.

Il multiplexer trasmittente incapsula nell'intestazione le informazioni per indicare al demultiplexer ricevente come e a quale socket indirizzare i dati ricevuti

Demultiplexing

L'host riceve i **datagrammi IP**

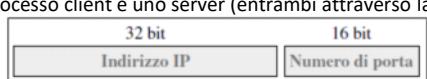
- Ogni **datagramma** ha un **IP di origine** e uno di **destinazione**
- Ogni **datagramma trasporta 1 segmento** a liv. di trasporto
- Ogni **segmento** ha un **n° di porta di origine e destinazione**

L'host usa gli IP e i num. di porta per inviare il segmento al **processo appropriato**.

Socket

Appare come un terminale o un file ma non è un entità fisica (astrazione)

È una struttura dati creata e usata dal liv. applicativo per permettere la comunicazione tra un processo client e uno server (entrambi attraverso la loro socket)



Socket Address

Campo n° porta: 16 bit con val da 0 a 65535

(da 1 a 1023 si chiamano "well known port number")

- 0: non usato
- 1-255: riservati per processi conosciuti
- 256-1023: riservati per altri processi
- 1024-65535: usati da app utente

well known port n° per server comuni: FTP 20, TELNET 23, SMTP 25, HTTP 80, POP3 110, ...

Per comunicare è necessaria una **coppia di indirizzi socket: locale** (mittente) e **remoto** (destinatario)

Individuare Socket Lato Client

Serve un **socket address locale** (client) e **remoto** (server) per comunicare

Socket address locale:

- Conosce l'IP del pc su cui il client è in esecuzione
- Il n° di porta è assegnato temporaneamente dall'OS

Socket address remoto:

- N° porta noto in base all'app (es: HTTP porta 80)
- IP fornito dal **DNS**
- Oppure porta e IP noti al programmatore se vuole verificare il funzionamento di un'app

Individuare Socket Lato Server

Serve un **socket address locale** (server) e **remoto** (client) per comunicare

Socket address locale:

- Conosce l'IP del pc su cui il client è in esecuzione
- Il n° di porta è noto al server perché assegnato dal progettista

Socket address remoto:

- È il socket address locale del client che si connette
- Poiché numerosi client possono connettersi, il server trova i socket address all'interno del pacchetto di richiesta

Il socket address remoto varia ad ogni interazione con client diversi

Servizi di Trasporto

TCP (affidabile)

UDP (non affidabile)

Orientato alla connessione: è richiesto un setup fra i processi client e server.
Mantiene l'ordine della sequenza dei bit consegnati

Offre:

- **Trasporto affidabile** fra processi di invio e di ricezione
- **Controllo di flusso**: il mittente non vuole sovraccaricare il destinatario
- **Controllo della congestione**: "strozza" il processo d'invio quando la rete è sovraccaricata

Non offre: temporizzazione, garanzie su ampiezza di banda minima, sicurezza (si può implementare, es SSL)

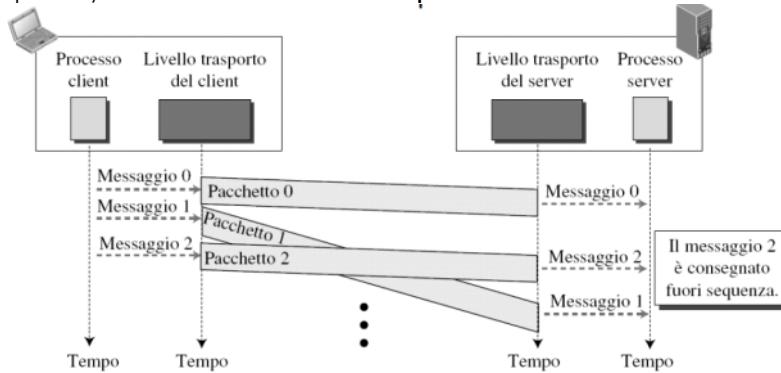
Senza connessione: non è richiesto un setup fra i processi client e server
Trasferimento dati inaffidabile. È possibile che due messaggi inviati arrivino in tempi diversi.

Non offre: setup della connessione, affidabilità, controllo di flusso e congestione, temporizzazione, ampiezza di banda minima e sicurezza

UDP (User Datagram Protocol)

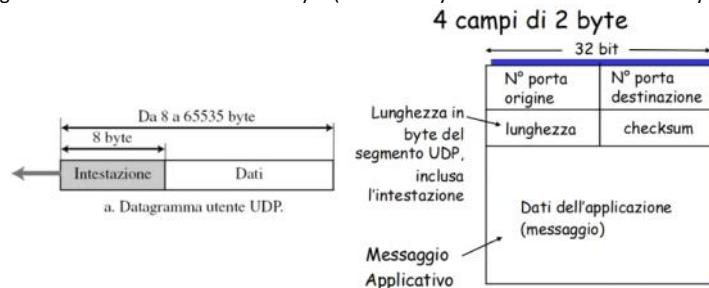
lunedì 28 aprile 2025 14:25

Il protocollo UDP è usato per trasmettere dati tra i socket client e server. È un protocollo inaffidabile e privo di connessione (setup iniziale) e non fornisce controllo di flusso, errori (tranne checksum) e congestione. Però è molto più veloce di TCP perché non richiede controlli aggiuntivi sulla connessione o sul flusso. Poiché i datagrammi inviati non sono numerati, il mittente invia pacchetti indipendenti uno dopo l'altro senza pensare al destinatario e al destinatario possono arrivare in ordine diverso da quello di partenza). Non c'è coordinazione tra i liv. trasporto del mittente e del destinatario.



Non vi è alcun flusso di dati, quindi il processo mittente non può inviare un flusso di dati e aspettarsi che UDP lo divida in datagrammi correlati.

Quindi i processi stessi devono dividere i dati in richieste con dimensione ridotta abbastanza per essere caricata ciascuna in un singolo **datagramma utente**. Un datagramma utente UDP usa messaggi di dimensioni inferiori a 65507 byte (65535 - 8 byte di intestazione UDP - 20 byte di intestazione IP)



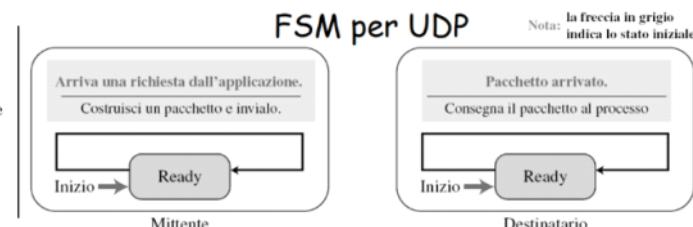
Grazie alla velocità e alla semplicità di UDP, esso è utilizzato spesso nelle applicazioni multimediali dove è tollerata la perdita limitata di pacchetti ed è sensibile alla frequenza. È utilizzato anche dal protocollo DNS poiché non richiede di stabilire una connessione e invia intestazioni di pacchetto corte

Application	Application-Layer Protocol	Underlying Transport Protocol
Electronic mail	SMTP	TCP
Remote terminal access	Telnet	TCP
Web	HTTP	TCP
File transfer	FTP	TCP
Remote file server	NFS	Typically UDP
Streaming multimedia	typically proprietary	UDP or TCP
Internet telephony	typically proprietary	UDP or TCP
Network management	SNMP	Typically UDP
Routing protocol	RIP	Typically UDP
Name translation	DNS	Typically UDP

Rappresentazione tramite FSM (Finite State Machine)

Il comportamento di un protocollo si può interpretare da un **automa a stati finiti**, il quale rimane in uno stato fin quando non **avviene un evento** che può modificare lo stato dell'automa (transizione di stato) e fargli compiere un'azione.

Protocollo molto semplice



Intro TCP (Transmission Control Protocol)

lunedì 28 aprile 2025 14:48

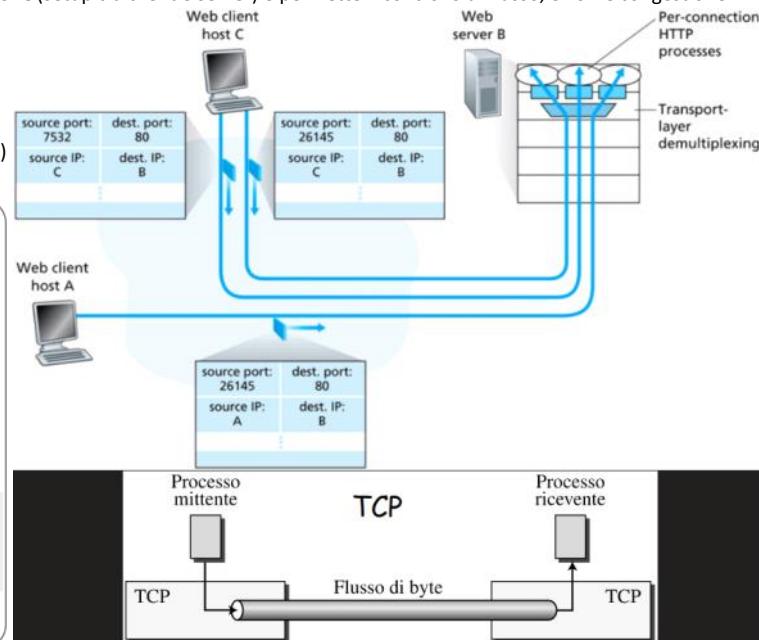
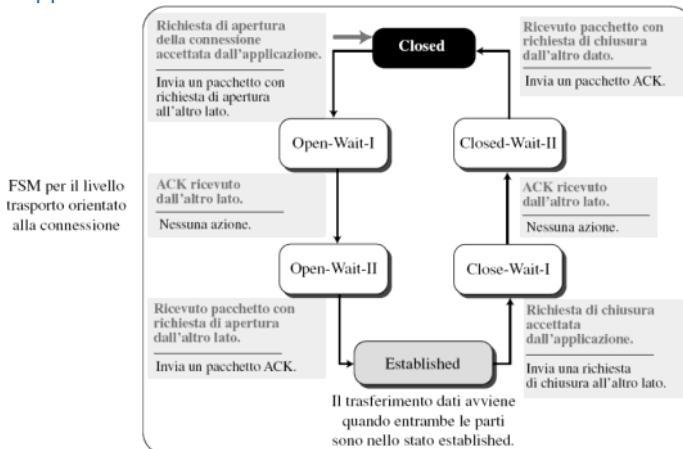
Rispetto all'UDP, il **TCP** è un protocollo più complesso **orientato alla connessione** (setup tra client e server) e permette il controllo di flusso, errori e congestione.

Per **identificare** la socket TCP si usano 4 entità principali:

- IP Host **locale** (origine)
- IP Host **remoto** (destinazione)
- N° porta process **locale** (origine)
- N° porta process **remoto** (destinazione)

Porta ≠ Socket. Una porta può avere **più socket** (es img: 3 socket su porta 80)

Rappresentazione tramite FSM



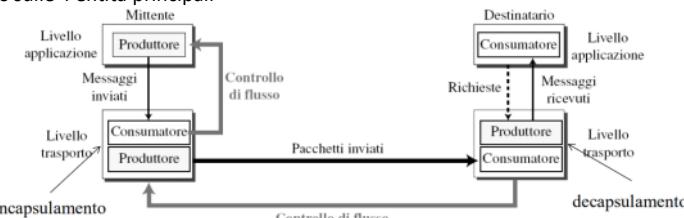
Controllo di Flusso

Quando un'entità produce dati che un'altra entità deve consumare, deve esistere un equilibrio tra la velocità di produzione e quella di consumo dei dati

- Se **velocità produzione > velocità consumo** → Il consumatore può essere sovraccaricato e costretto ad eliminare alcuni dati
- Se **velocità produzione < velocità consumo** → Il consumatore rimane in attesa riducendo l'efficienza del sistema

Il **controllo di flusso** si usa nel primo caso per evitare la perdita dei dati.

Vengono eseguiti 2 casi di controllo di flusso sulle 4 entità principali



Per realizzare il controllo vengono usati dei **buffer** (insieme di locazioni di mem. che possono contenere pacchetti) e il **consumatore invia segnali al produttore** in base alla **saturazione del buffer**, controllando quindi il flusso di dati inviati.

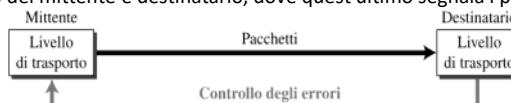
- Se il **buffer del liv. di trasporto del mittente** è **saturo**, **segnala al liv. applicazione** di sospendere l'invio di messaggi.
Quando si libera spazio nel buffer segnala al liv. applicazione di riprendere l'invio
- Se il **buffer del liv. di trasporto del destinatario** è **saturo**, **segnala al liv. trasporto del mittente** di sospendere l'invio di messaggi.
Quando si libera spazio nel buffer segnala al liv. trasporto del mittente di riprendere l'invio

Controllo degli Errori

Poiché il liv. di rete è **inaffidabile**, è necessario implementare l'affidabilità al liv. trasporto, implementando un **controllo degli errori** (sui pacchetti non bit)

- Rilevare e scartare pacchetti corrotti
- Tenere traccia dei pacchetti persi e gestirne il rinvio
- Riconoscere pacchetti duplicati e scartarli
- Bufferizzare i pacchetti fuori sequenza finché non arrivano i pacchetti mancanti

Il controllo degli errori coinvolge solo i liv. trasporto del mittente e destinatario, dove quest'ultimo segnala i problemi sui pacchetti al liv. trasporto del mittente



Per riconoscere quali pacchetti ritrasmettere e quali sono duplicati o fuori sequenza è necessario **numerare i pacchetti in sequenza** (campo nell'header). È necessario specificare la dim. massima per il num. di sequenza:

Se l'intestazione prevede **m bit** per il num. di sequenza, questi possono assumere val. da 0 a $2^m - 1$ (es m = 4, num. sequenza da 0 a 15)

Il num. di sequenza è utile al destinatario per capire:

- La sequenza di pacchetti in arrivo
- Pacchetti persi (da scartare)
- Pacchetti duplicati (da scartare)

Invece il mittente capisce se un pacchetto è andato perso tramite il **numero di riscontro** (acknowledgment, ack o conferma) che notifica la **corretta ricezione di un pacchetto**.

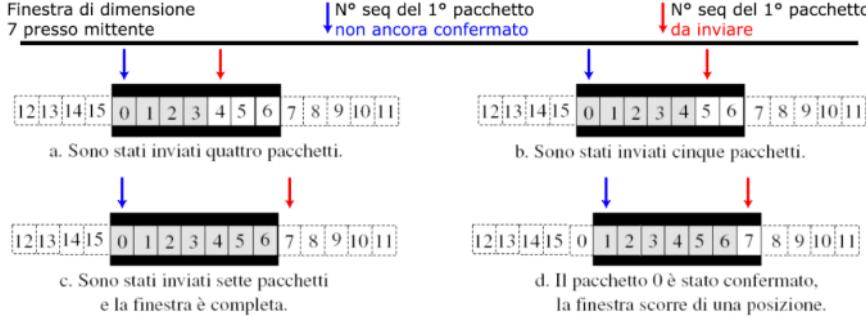
Integrazione Controllo degli Errori e di Flusso

Il controllo di **flusso** richiede **2 buffer** (mittente e destinatario) e il controllo degli **errori** richiede **num. di sequenza e ack**.

Si possono combinare usando un **buffer numerato** (su mittente e destinatario)

- **Mittente:**
 - Quando **prepara un nuovo pacchetto** usa come **num. di sequenza** il num. (x) della **prima locazione libera nel buffer**
 - Quando **invia il pacchetto** ne **memorizza una copia** della locazione x
 - Quando **riceve un ack** di un pacchetto, **libera la pos. di mem.** occupata da **quel pacchetto**
- **Destinatario:**
 - Quando **riceve un pacchetto** con num. di seq. y, lo **memorizza nella locazione y** fin quando il liv. applicazione è pronto a riceverlo
 - Quando passa il pacchetto y al liv. applicazione, **invia un ack** al mittente

Il buffer è rappresentato con un **insieme di settori**, chiamati **finestre scorrevoli o sliding windows**, che occupano una parte dei num. di sequenza



Controllo della Congestione

La **congestione** avviene quando il **carico della rete** (num. pacchetti inviati alla rete) è **superiore alla capacità della rete** (num. di pacchetti che può gestire).

Se router e switch non riescono a elaborare i pacchetti alla stessa velocità con cui arrivano, le code si sovraccaricano e avviene la congestione

Il **controllo della congestione** è un insieme di meccanismi e tecniche per mantenere il carico della rete al di sotto della sua capacità

Meccanismi Trasferimento Dati Affidabili

lunedì 28 aprile 2025 17:57

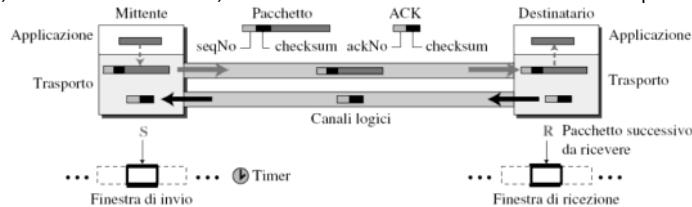
Stop-and-Wait

Protocollo **orientato alla connessione + controllo di flusso + errori**, dove il mittente e il destinatario usano una **finestra scorrevole** di dim. 1
Il mittente invia un pacchetto alla volta e ne attende l'ack prima di spedire il successivo.

Quando il pacchetto arriva al destinatario, si calcola il checksum:

- Pacchetto buono → ack al mittente
- Pacchetto corrotto → scartato senza informare il mittente

Per capire se un pacchetto è andato perso, il mittente usa un **timer**, la cui scadenza senza ricevere un ack indica un pacchetto perso, allora rinvia il pacchetto.



- Il mittente deve tenere una copia del pacchetto spedito finché non riceve riscontro
- Controllo errori (con num. sequenza + ack + timer)
- Controllo flusso (un solo pacchetto alla volta)

Num. di Sequenza nello Stop-and-Wait

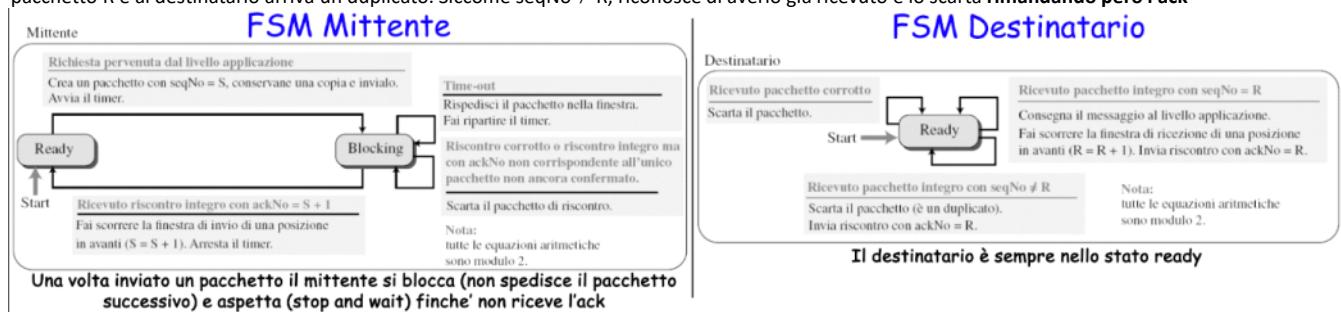
Siccome si utilizza il num. di sequenza, possiamo gestire facilmente i pacchetti duplicati. I num. 0 e 1 sono sufficienti per il protocollo stop-and-wait.

Convenzione: il num. di riscontro (ack) indica il num. di seq. del prossimo pacchetto atteso dal destinatario.

Se il destinatario ha ricevuto correttamente il pacchetto 0, invia un riscontro con val 1 (quindi il prossimo pacchetto atteso ha num. di seq. 1)

Si possono verificare 3 casi quando il mittente invia il pacchetto con num. di seq. x:

1. Il pacchetto **arriva correttamente** al destinatario (integro e seqNo = R), invia il messaggio al liv. applicazione, scorre la finestra R = R+1 mod 2 e invia un **ack R**. L'ack arriva al mittente che **invia il pacchetto successivo**
2. Il pacchetto **risulta corrotto o non arriva** al destinatario. Allo **scadere del timer**, il mittente **rinvia il pacchetto x**
3. Il pacchetto **arriva correttamente** al destinatario e scorre la finestra R = R+1 ma l'**ack viene perso o corrotto**. Allo scadere del timer, il mittente rinvia il pacchetto R e al destinatario arriva un duplicato. Siccome seqNo ≠ R, riconosce di averlo già ricevuto e lo scarta **rimandando però l'ack**



Esempio diagramma di flusso:

- Il pacchetto 0 viene inviato e riscontrato.
- Il pacchetto 1 viene perso e rispedito alla scadenza del timer e finalmente riscontrato
- Il pacchetto 0 viene spedito e confermato ma l'ack viene perso, quindi allo scadere del timer il pacchetto viene rinvito ma poiché è un duplicato, viene scartato ma viene inviato l'ack

Efficienza

Consideriamo **rate*ritardo**: misura num. di bit che il mittente può inviare prima di ricevere un ack, volume della pipe in bit)

Se il **rate (velocità trasmissione)** è elevato e il ritardo è consistente → stop-and-wait inefficiente

Esempio:

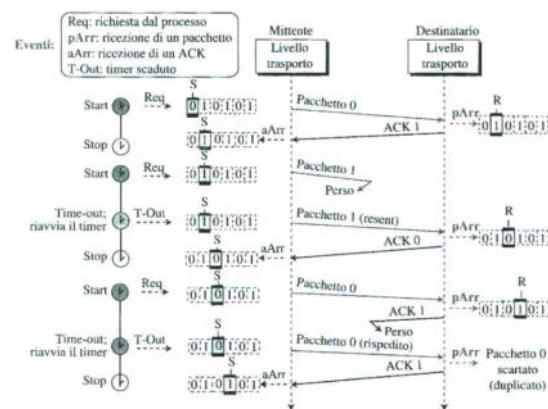
- Rate = 1Mbps
- Ritardo andata e ritorno di 1 bit = 20ms

Se i pacchetti hanno dimensione 1000 bit:

$$\text{Rate} * \text{ritardo} = (1 \times 10^6) \times (20 \times 10^{-3}) = 20000 \text{ bit}$$

Il mittente potrebbe inviare 20000 bit nel tempo necessario per andare dal mittente al ricevente e viceversa ma ne invia solo 1000

- Il coefficiente di utilizzo del canale è $1000/20000 = 5\%$ → molto inefficiente



Protocolli con Pipeline

Pipelining: il mittente ammette **più pacchetti** in transito, ancora da notificare. Ciò comporta:

- L'intervallo dei num. di sequenza deve essere incrementato
- Buffering dei pacchetti nel mittente e/o ricevente

Esistono due forme generiche di protocolli con pipeline: **Go-Back-N** e **ripetizione selettiva**

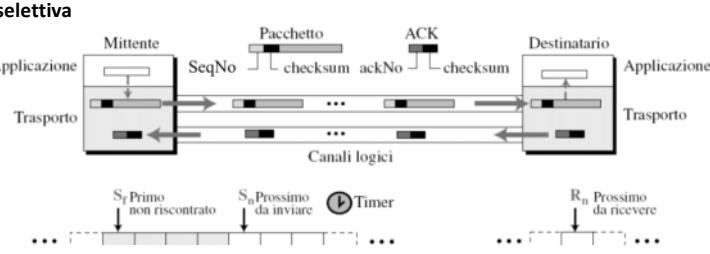
Go back N

I num. di seq. sono calcolati in modulo 2^m dove m è la dim. del campo "num di seq." in bit

Ack indica il **num. di seq. del prossimo pacchetto atteso**

Ack cumulativo (AckNo): tutti i pacchetti fino al num. di seq. indicato nell'ack sono stati ricevuti correttamente

Es: AckNo = 7 → i pacchetti fino al 6 sono stati ricevuti e si attende il 7



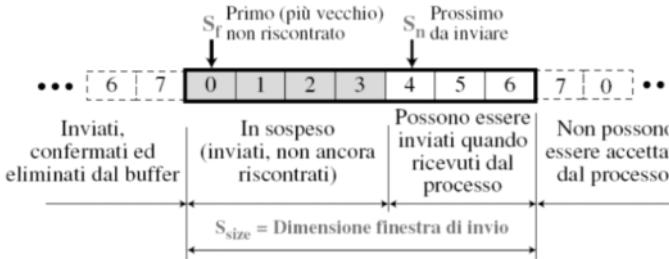
ACK CUMULATIVO (ACKNO): tutti i pacchetti fino al num. di seq. indicato nell'ACK sono stati ricevuti correttamente

Ese: AckNo = 7 → i pacchetti fino al 6 sono stati ricevuti e si attende il 7

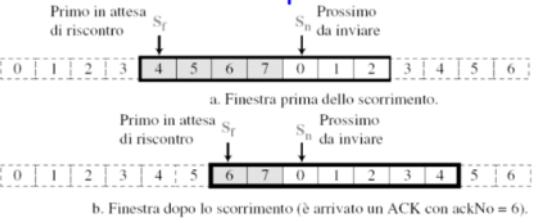
Finestra di Invio

La finestra di invio è un concetto astratto che definisce una porzione immaginaria di dim. max $2^m - 1$ con tre variabili S_f , S_n , S_{size} (se fosse 2^m non andrebbe bene)

La finestra di invio può scorrere uno o più pos. quando viene ricevuto un $\text{AckNo} \geq S_f$ e $\text{AckNo} < S_n$ in aritmetica modulare

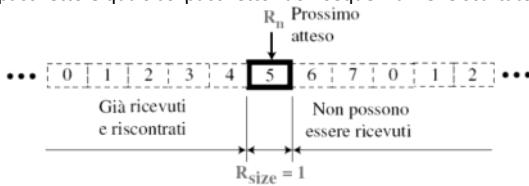


Esempio



Finestra di Ricezione

La finestra di ricezione ha dimensione 1. Il destinatario è sempre in attesa di uno specifico pacchetto e qualsiasi pacchetto fuori sequenza viene scartato



Timer e Rispedizione

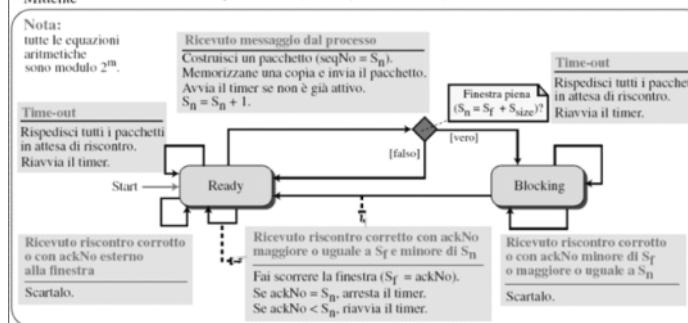
Il mittente mantiene un timer per il più vecchio pacchetto non riscontrato (S_f). Allo scadere del timer vengono rispediti tutti i pacchetti in attesa di riscontro (il destinatario ha finestra di ricezione pari a 1 e non può bufferizzare i pacchetti fuori sequenza)

Ese: $S_f = 3$ e il mittente ha inviato il pacchetto 6 ($S_n = 7$).

Scade il timer e i pacchetti 3, 4, 5, 6 (non riscontrati) vengono rinviiati

La finestra di ricezione può scorrere di una sola posizione: $R_n = (R_n + 1) \bmod 2^m$

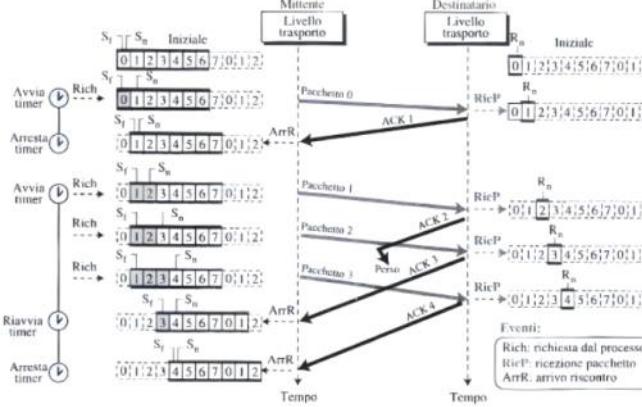
FSM Mittente



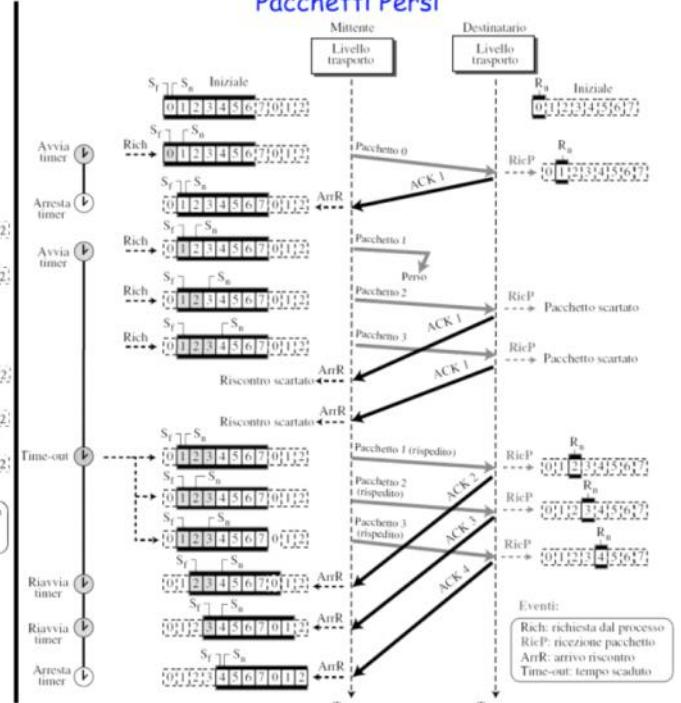
FSM Destinatario



Ack Cumulativo



Pacchetti Persi

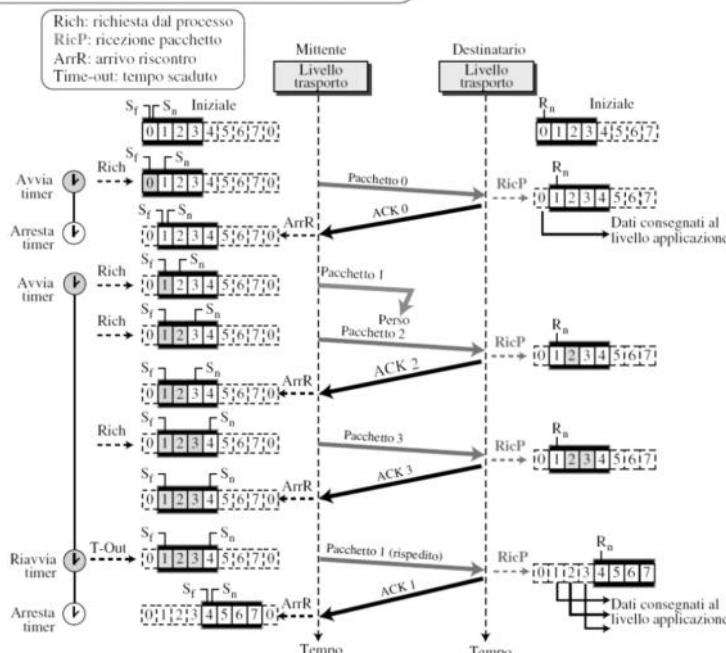
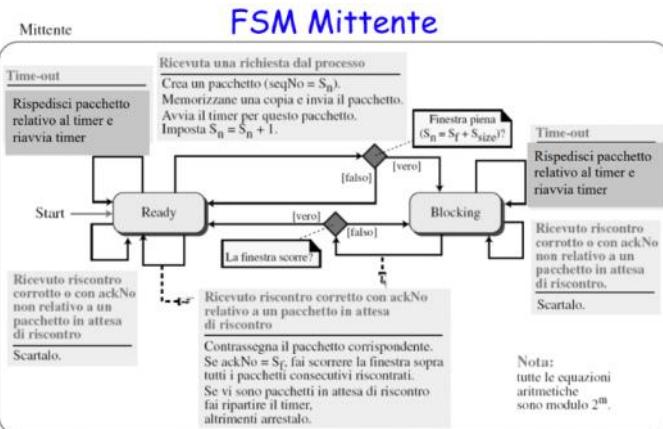
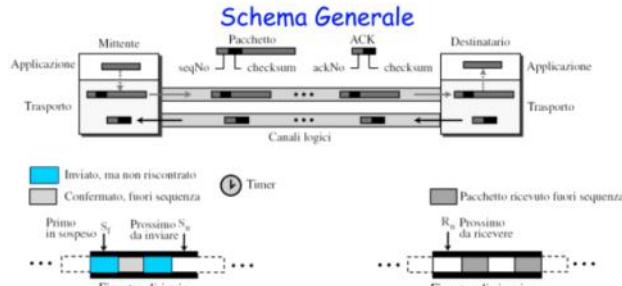


Ripetizione Selettiva

In Go-back-N per un solo pacchetto perso si **ritrasmettono tutti i successivi già inviati nel pipeline**, e nel caso di congestione di rete questo peggiora la situazione.

Nella **ripetizione selettiva**, vengono **rispediti soltanto** i pacchetti per i quali **non si ha ricevuto un ack** (timer del mittente per ogni pacchetto non riscontrato)

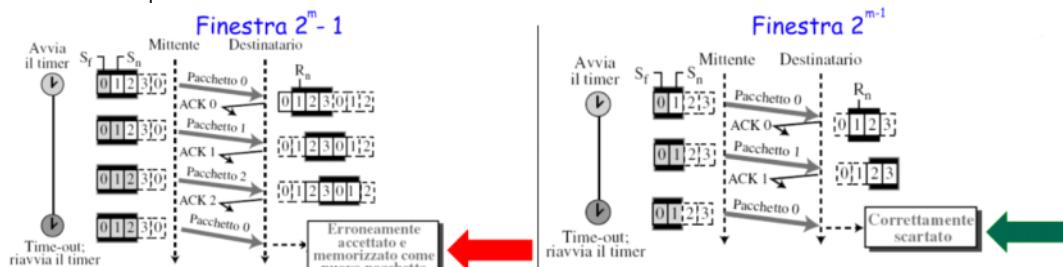
Il ricevente invia **riscontri specifici** per i pacchetti ricevuti correttamente (sia in ordine, sia fuori sequenza) (**buffer pacchetti**, se necessario, per consegnare in sequenza al liv. applicazione). Quindi l'ack indica il num. di seq. di un pacchetto ricevuto correttamente (non il prossimo atteso)



N.B.: i pacchetti possono essere consegnati al liv. applicazione se:

1. È stato ricevuto un insieme di pacchetti consecutivi
2. L'insieme deve partire dall'inizio della finestra

Rispetto al Go-back-N le finestre non possono essere di dim. $2^m - 1$ ma di dimensione **max** 2^{m-1}



Protocolli Bidirezionali: Piggybacking

I meccanismi precedenti sono **unidirezionali**: pacchetti dati in una direzione e ack nella direzione opposta.

Per migliorare l'efficienza dei protocolli bidirezionali si utilizza la tecnica del **piggybacking**: quando un pacchetto trasporta dati da A a B, può trasportare anche i riscontri relativi ai pacchetti ricevuti da B e viceversa

Riassunto meccanismi

Meccanismo	Uso
Checksum	Per gestire errori nel canale
Acknowledgment	Per gestire errori nel canale
Numero di sequenza	Ack con errori Perdita pacchetti
Timeout	Perdita pacchetti
Finestra scorrevole, pipeling	Maggior utilizzo della rete

Gestione inaffidabilità della rete

Miglioramento prestazioni

Esercizio 1

Usando num. di seq. a **5 bit**, qual è la dim. max delle finestre di invio e ricezione per ciascuno dei meccanismi visti?

- Stop-and-Wait:
 - Max Send Wsize = 1
 - Max Receive Wsize = 1
- Go-back-N:
 - Max Send Wsize = $2^5 - 1 = 31$
 - Max Receive Wsize = 1
- Ripetizione selettiva:
 - Max Send Wsize = $2^{5-1} = 16$
 - Max Receive Wsize = $2^{5-1} = 16$

Esercizio 2

In una rete con Go-back-N con $m=3$ e dimensione della finestra di invio = 7, i val. delle var sono $S_f = 0$, $S_n = 4$ e $R_n = 2$.

Si ipotizzi che la rete non duplichì e non alteri l'ordine dei pacchetti

1. Qual è il num. di seq. dei pacchetti in transito?
2. Qual è il num. di riscontro dei pacchetti ack in transito?

Siccome $S_n = 4$ allora abbiamo inviato 4 pacchetti (0, 1, 2, 3) e poichè S_f è ancora a 0 significa che il mittente non ha ricevuto un singolo ack per i pacchetti inviati, quindi questi pacchetti sono in transito.

Poi siccome $R_n = 2$ allora vuol dire che sono arrivati in sequenza correttamente i primi 2 pacchetti (0, 1) quindi invia ACK2 per ricevere il prossimo pacchetto 2, quindi questo ack è in transito

TCP

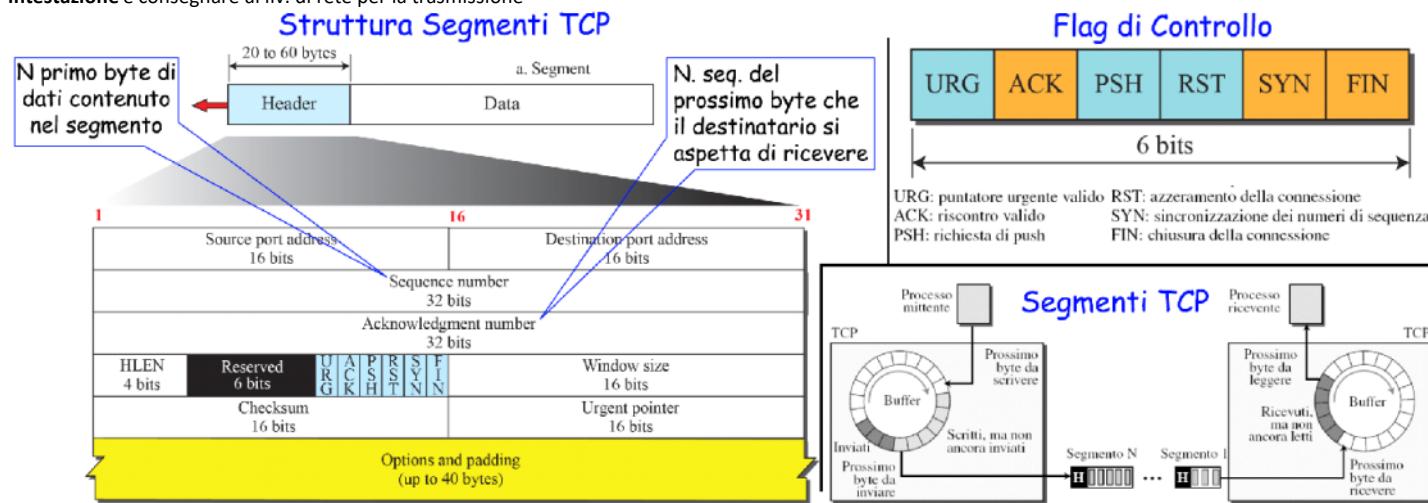
martedì 29 aprile 2025 10:47

- Protocollo con pipeline
- Bidirezionale (con piggybacking)
- Orientato al flusso di dati (stream-oriented)
- Orientato alla connessione
- Affidabile (controllo errori)
- Controllo del flusso
- Controllo della congestione

Segmenti TCP

Il TCP riceve uno stream di byte dal processo mittente

Utilizza il servizio di comunicazione tra host del liv. di rete che invia pacchetti. Deve quindi raggruppare un certo num. di byte in **segmenti**, aggiungere un **intestazione** e consegnare al liv. di rete per la trasmissione

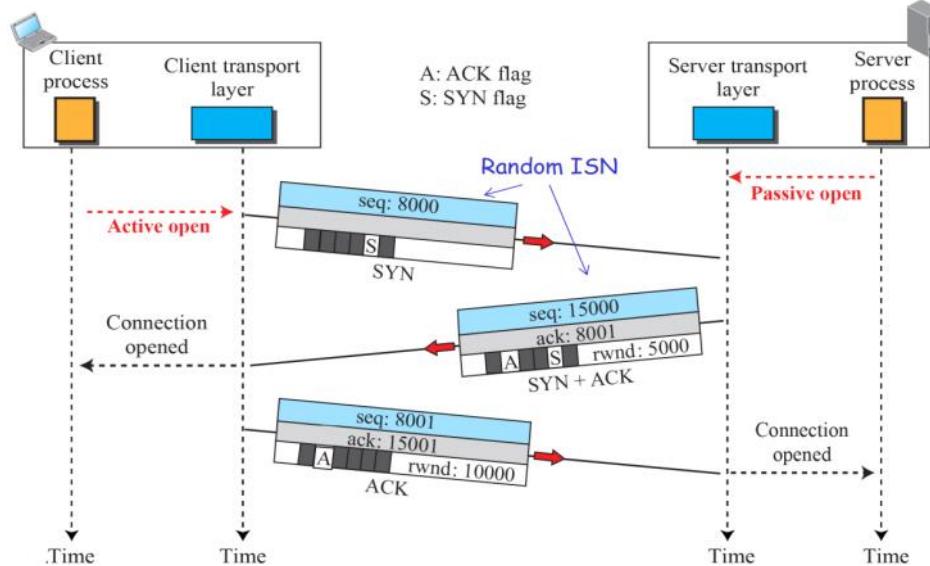


Connessione TCP

È un percorso virtuale tra il mittente e il destinatario, sopra IP che è privo di connessione. Si divide in 3 fasi:

1. Apertura della connessione
2. Trasferimento dei dati
3. Chiusura della connessione

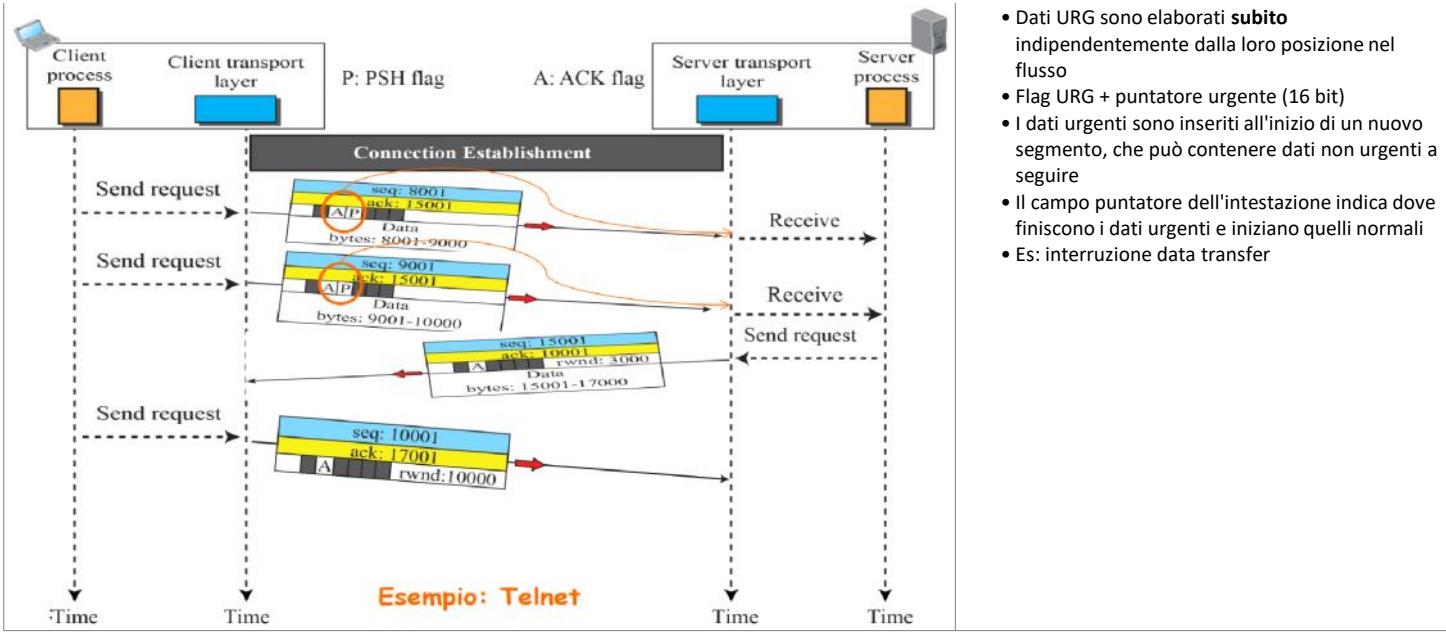
Apertura della connessione: 3 way handshake



Trasferimento Dati

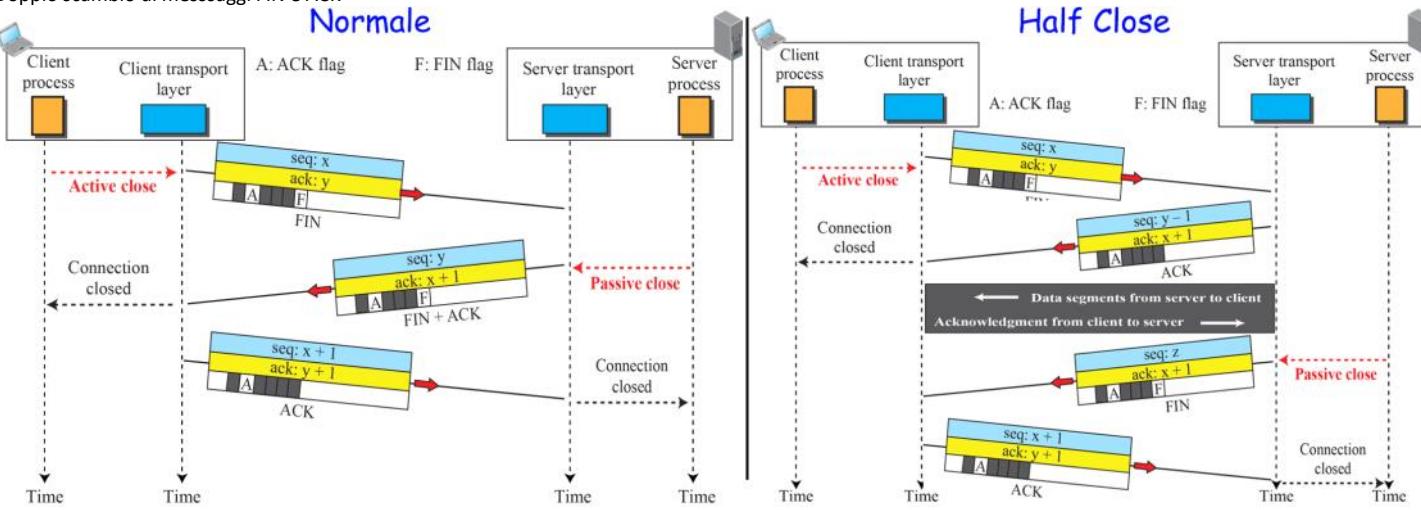
Push

Urgent



Chiusura della Connessione

Ciascuna delle due parti coinvolte nello scambio di dati può richiedere la chiusura della connessione, anche se solitamente richiesta dal client o dal timer nel server
Doppio scambio di messaggi FIN e ACK



TCP: Controllo Errori

giovedì 1 maggio 2025 16:26

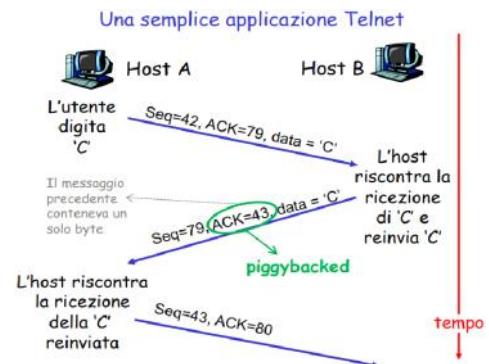
Numeri di sequenza: "num" primo byte di segmento nel flusso di byte

ACK:

- Num. di seq. del prossimo byte atteso dall'altro lato
- Ack cumulativo: tutti i pacchetti fino a quell'ultimo num. di seq. sono stati ricevuti

Il controllo degli errori in TCP è effettuato tramite:

- Checksum:** se un segmento arriva corrotto, viene scartato dal destinatario
- Riscontri + timer ritrasmissione (RTO)**
 - Ack cumulativi
 - Timer associato al più vecchio pacchetto non riscontrato
- Ritrasmissione:** ritrasmissione del segmento all'inizio della coda di spedizione



Generazione di ACK

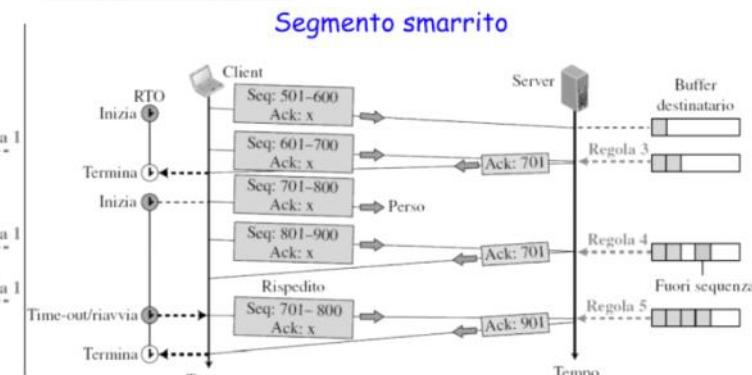
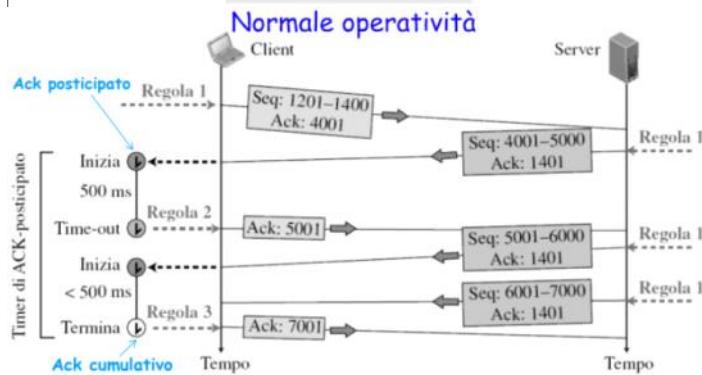
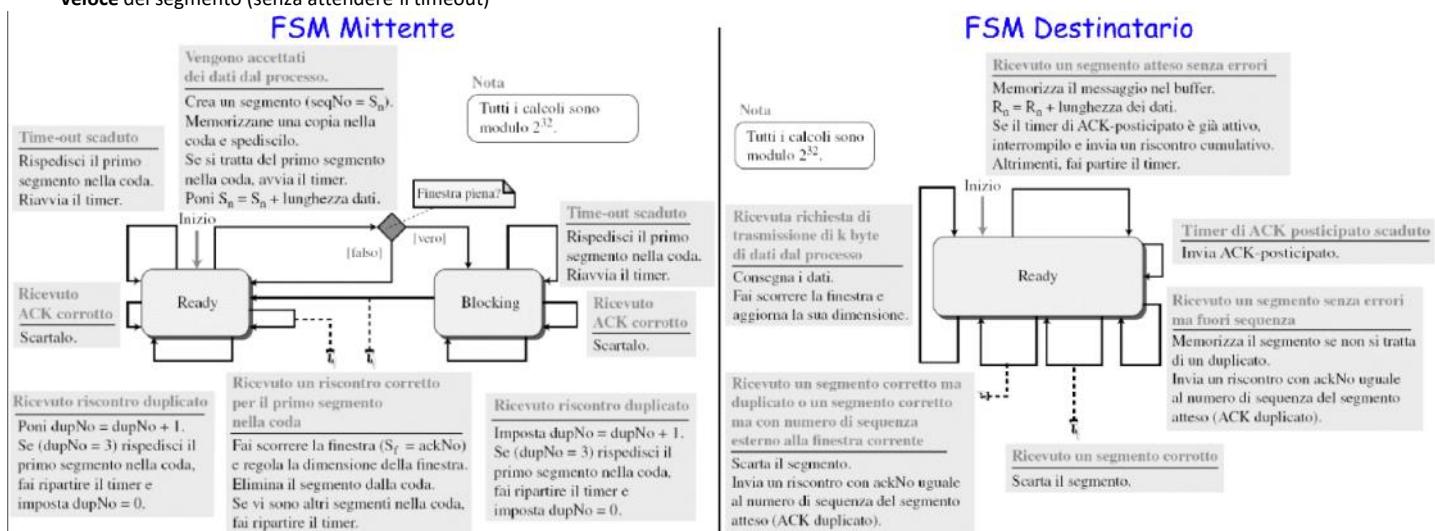
Rule	Evento	Azione
1	Arrivo di un segmento dati	Invia un ACK (piggybacking) col num. del prossimo segmento atteso
2	Arrivo ordinato di un segmento con num. seq. atteso. Tutti i dati fino al num. di seq. atteso sono già stati riscontrati	ACK delayed. Attende fino a 500ms l'arrivo del prossimo segmento per rispondere con un solo ACK cumulativo. Se il segmento non arriva, invia un ACK comunque per confermare quello ricevuto
3	Arrivo ordinato di un segmento con num. seq. atteso. Un altro segmento è in attesa di trasmissione dell'ACK (vedi precedente)	Invia immediatamente un singolo ACK cumulativo, riscontrando i segmenti ordinati
4	Arrivo non ordinato di un segmento con num. seq. superiore a quello atteso. Viene rilevato un buco	Invia immediatamente un ACK duplicato, indicando il num. seq. del prossimo byte atteso (ritrasmissione rapida)
5	Arrivo di un segmento mancante (uno o più dei successivi è stato ricevuto)	Invia immediatamente un ACK per aggiornare il mittente sul flusso in ordine
6	Arrivo di un segmento duplicato	Invia immediatamente un riscontro con num. seq. atteso

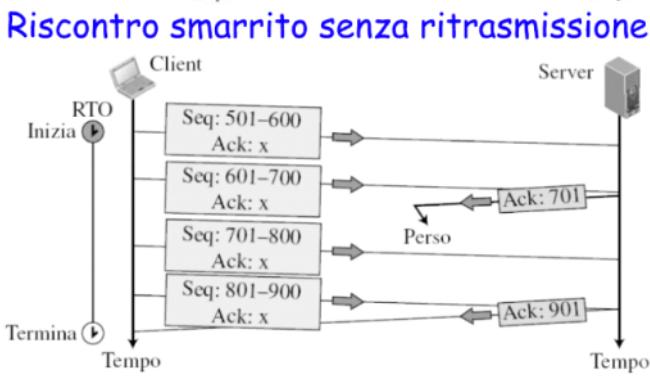
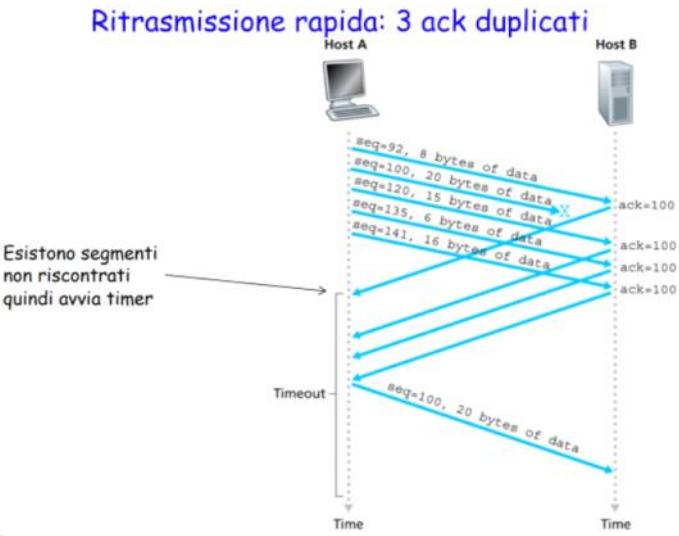
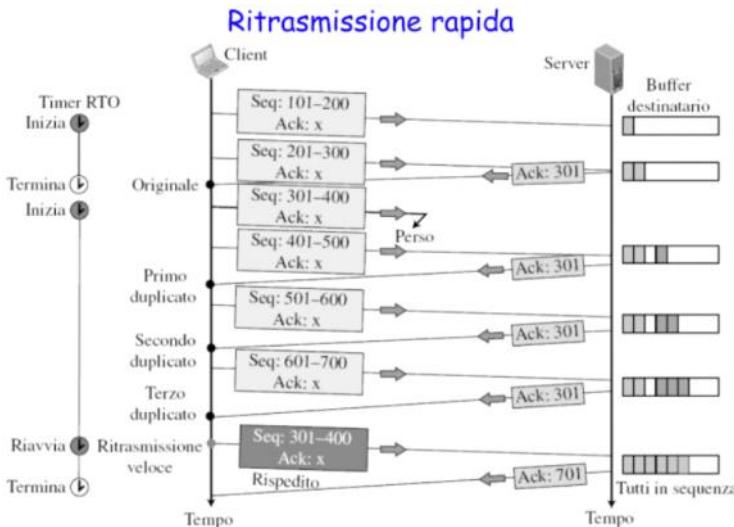
Ritrasmissione Segmenti

Quando un segmento viene inviato, una copia viene memorizzata in una coda in attesa di essere riscontrato (**finestra di invio**)

Se il segmento non viene riscontrato allora:

- Scade il timer** (è il primo segmento nella coda) → il segmento viene ritrasmesso e viene riavviato il timer
- Vengono ricevuti 3 ack duplicati** (indica un pacchetto perso, cioè il destinatario ha ricevuto almeno 3 segmenti dopo quello mancante) → **ritrasmissione veloce** del segmento (senza attendere il timeout)





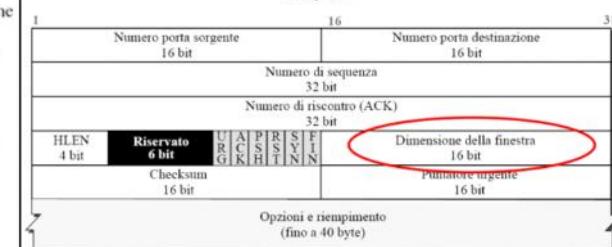
Riassunto Meccanismi Adottati da TCP

- **Pipeline** (approccio ibrido tra GBN e Ripetizione Selettiva): **non reinvia tutto** dopo un errore (come GBN) e **non gestisce ogni pacchetto separatamente** come la Ripetizione Selettiva pura. Invia più segmenti senza aspettare un ACK uno per uno (**pipeline**) ma può **ritrasmettere solo il pacchetto perso**
- **Num. seq**: indica il **primo byte** del segmento
- **ACK**:
 - **Cumulativo**: conferma tutti i byte ricevuti in ordine fino a un certo punto (es. ACK 501 indica che si hanno ricevuti i byte fino al 500)
 - **Ritardato**: dopo aver ricevuto un segmento in ordine può **aspettare fino a 500ms** per vedere se arrivano i prossimi, così da inviare un **ACK cumulativo**
- **Timeout basato su RTT**: usa un solo timer per la **ritrasmissione**, associato al **segmento non riscontrato più vecchio** (non tutti), se riceve un **ACK parziale**, **riavvia il timer** sul segmento non confermato
- **Ritrasmissione**:
 - **Singola**: ritrasmette **solo il pacchetto perso** (non tutti quelli successivi)
 - **Rapida**: se riceve **3 ACK duplicati**, ritrasmette **subito** il pacchetto mancante, **senza aspettare il timeout (ritrasmissione)**

TCP: Controllo del Flusso

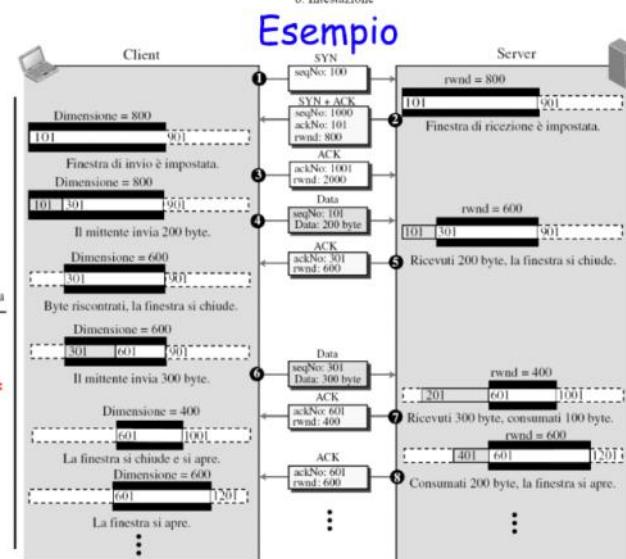
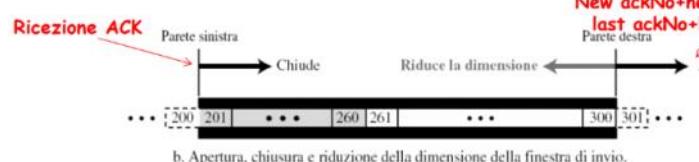
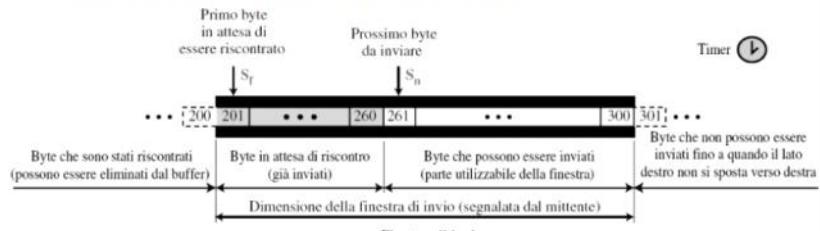
venerdì 2 maggio 2025 16:23

Il controllo del flusso serve per non sovraccaricare il buffer del destinatario con troppi dati. Si utilizza un **feedback esplicito del destinatario** che comunica al mittente lo spazio disponibile includendo il val. **RcvWindow (RWND)** nell'header



Finestra di Invio

L'apertura, chiusura e riduzione della finestra di invio sono controllate dal destinatario



TCP: Controllo della Congestione

venerdì 2 maggio 2025 17:07

Informalmente: "troppe sorgenti trasmettono troppi dati, troppo velocemente e la rete non è in grado di gestirli". Esso differisce dal controllo di flusso.

Sintomi:

- **Pacchetti smarriti** (overflow nei buffer dei router)
- **Lunghi ritardi** (accodamento nei buffer dei router)

Tra i dieci problemi più importanti del networking

Controllo Congestione vs Controllo Flusso

Nel controllo del flusso la **dim. della finestra** è controllata dal destinatario tramite il val. rwnd che viene indicato in ogni segmento trasmesso nella dir. opposta.

La finestra del ricevente **non viene mai sovraccaricata** con i dati ricevuti

I buffer intermedi (dei router) possono comunque congestionarsi, poiché esso **riceve dati da più mittenti**.

Congestione nei nodi intermedi → perdita di segmenti che vengono rinviati → **aumenta la congestione**

La congestione è un problema che riguarda IP ma viene gestito da TCP

Controllo Congestione

I due principali approcci sono i controlli:

End-to-End (metodo adottato da TCP)

- Nessun supporto esplicito dalla rete
- La congestione è dedotta osservando le perdite e i ritardi nei sistemi terminali

Assistito dalla Rete

- I router forniscono un **feedback ai sistemi terminali**
 - Un singolo bit per indicare la **congestione** (TCP/IP ECN)
 - Comunicare in modo esplicito la **frequenza trasmittiva**

Problematiche

1. Come può il mittente limitare la freq. di invio del traffico sulla propria connessione?

Finestra di Congestione

per controllare la congestione si usa la var. **CWND** (congestion window) che insieme a RWNW definisce la dim. della finestra di invio

CWND: relativa a congestione di rete

RWNW: relativa a congestione del ricevente (flusso)

Dim. finestra = min(RWNW, CWND)

2. Come può il mittente rilevare la congestione?

Rilevare la Congestione

Evento di perdita: ACK duplicati e timeout possono essere intesi come eventi di perdita (indicazioni sullo stato della rete)

Reazione in base allo stato di rete:

- Se ACK arrivano in sequenza e con buona freq. → si può incrementare la quantità di segmenti inviati
- Se ACK **duplicati** o **timeout** → bisogna ridurre la finestra dei pacchetti che si spediscono senza aver ricevuto riscontri

TCP è **auto-temporizzante**: reagisce in base ai riscontri che ottiene

3. Quale algoritmo si usa per limitare la freq. di invio in funzione della congestione end -to-end?

Controllo della Congestione

Idea base: incrementare il rate di trasmissione se non c'è congestione (ack) e diminuire se c'è congestione (segmenti persi)

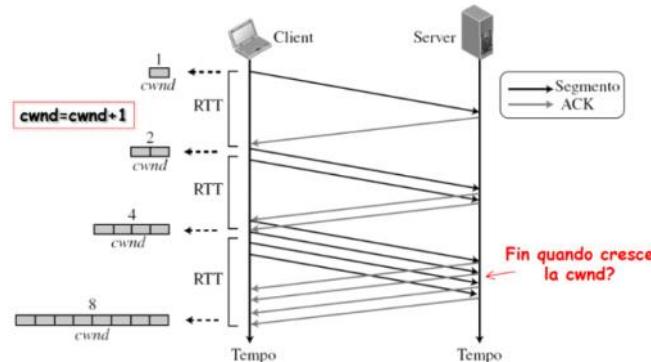
L'algoritmo di controllo della congestione si basa su tre componenti:

1. **Slow start**
2. **Congestion avoidance**
3. **Fast recovery**

Slow Start

CWND inizializzato a **1 MSS** (dimensione del segmento massimale) che **incrementa** di 1MSS per ogni segmento riscontrato fino ad una certa soglia (**ssthresh**)

Incremento Esponenziale



Se arriva un riscontro, $cwnd = cwnd + 1$

Inizio	$\rightarrow cwnd = 1 \rightarrow 2^0$
Dopo 1 RTT	$\rightarrow cwnd = cwnd + 1 = 1 + 1 = 2 \rightarrow 2^1$
Dopo 2 RTT	$\rightarrow cwnd = cwnd + 2 = 2 + 2 = 4 \rightarrow 2^2$
Dopo 3 RTT	$\rightarrow cwnd = cwnd + 4 = 4 + 4 = 8 \rightarrow 2^3$

La dimensione della finestra di congestione nell'algoritmo slow start viene aumentata esponenzialmente fino al raggiungimento di una soglia (ssthresh).

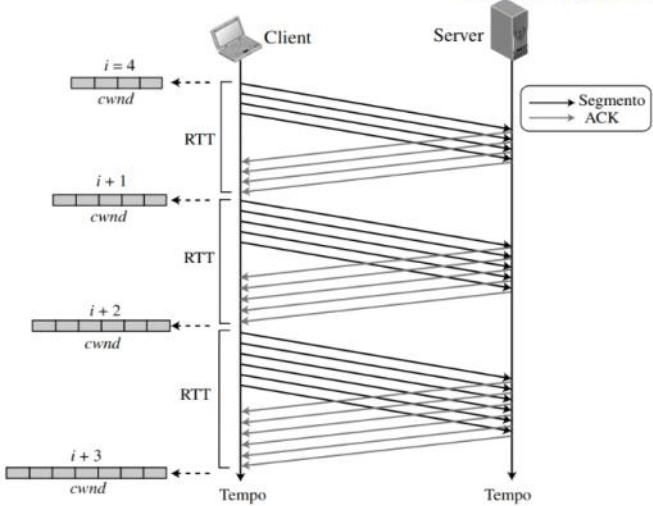
Congestion Avoidance

La CWND cresce **finché non viene perso un pacchetto** (in quel caso si pone $ssthresh = cwnd/2$) o **finché non raggiunge la ssthresh** (slow start threshold)

Si arresta slow start e inizia congestion avoidance:

- **Incremento lineare:** ogni volta che viene riscontrata l'intera finestra di segmenti, si incrementa di 1 la CWND
- Congestion avoidance **incrementa linearmente** finché non si rileva congestione (timeout o 3 ack duplicati)
- Al timeout $ssthresh = CWND/2$ e $CWND = 1$

Incremento Lineare



Se arriva un riscontro, $cwnd = cwnd + (1 / cwnd)$

Inizio	\rightarrow	$cwnd = i$
Dopo 1 RTT	\rightarrow	$cwnd = i + 1$
Dopo 2 RTT	\rightarrow	$cwnd = i + 2$
Dopo 3 RTT	\rightarrow	$cwnd = i + 3$

Con l'algoritmo congestion avoidance, la dimensione della finestra di congestione viene aumentata linearmente fino alla rilevazione della congestione.

Fast Recovery

Questo algoritmo è **opzionale** in TCP ed esiste solo nelle nuove versioni di TCP.

Inizia quando arrivano 3 riscontri duplicati che vengono interpretati come una leggera congestione di rete, cioè manca un pacchetto precedente.

Il fast recovery **incrementa esponenzialmente**, ma solo quando riceve un **riscontro duplicato** (dopo i 3 riscontri iniziali)

Se arriva un riscontro duplicato $\rightarrow \text{CWND} = \text{CWND} + 1$

Se arriva un **nuovo ACK maggiore** di quello del pacchetto perso, allora è un **ACK cumulativo** che indica che il pacchetto perso è arrivato, allora torniamo a **Congestion Avoidance** e impostiamo **CWND** a **ssthresh**

Se arriva un nuovo ACK \rightarrow Congestion Avoidance e $\text{CWND} = \text{ssthresh}$

Versioni TCP

sabato 3 maggio 2025 17:55

Esistono due versioni comuni di TCP con il controllo della congestione: **TCP Tahoe** e **TCP Reno** (estensione di Tahoe)

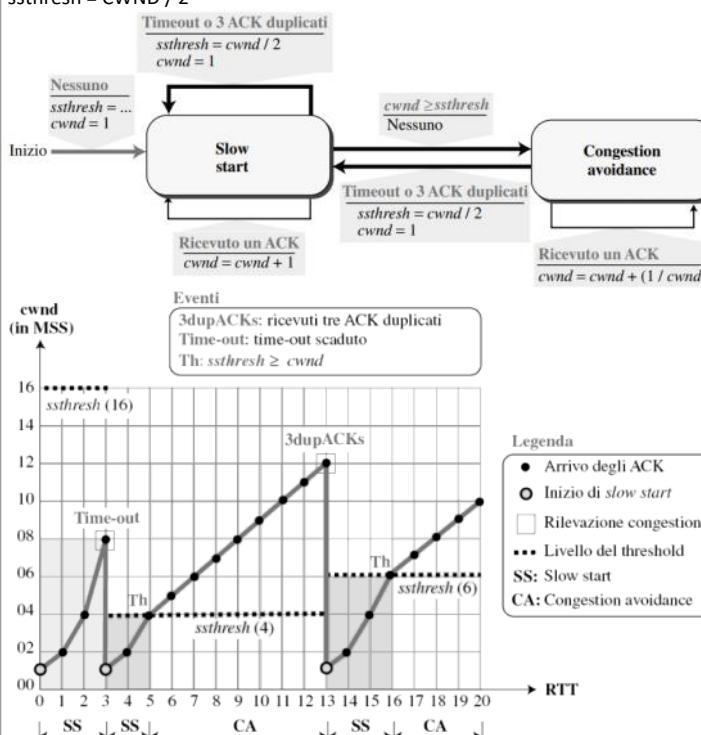
TCP Reno aggiunge il **Fast Recovery** che incrementa linearmente poiché indica congestione leggera.

Per distinguere il tipo di congestione si può indicare che:

- 3 ACK duplicati → capacità della rete di consegnare qualche segmento (3 pacchetti oltre a quello perso sono arrivati → congestione lieve)
- Timeout prima dei 3 ACK duplicati → "più allarmante" (non sono arrivati neanche i seguenti pacchetti) → congestione importante

TCP Tahoe

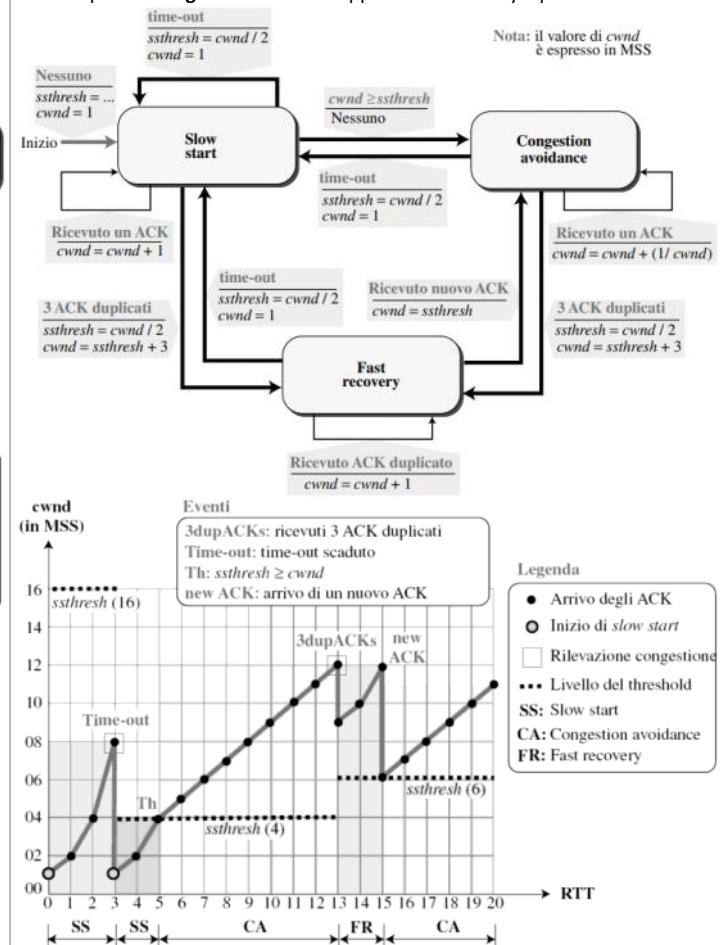
Considera timeout e 3 ack duplicati come congestione e riparte da 1 con $ssthresh = CWND / 2$



TCP Reno

Timeout: **congestione importante** → riparte da 1

3 ack duplicati: **congestione lieve** → applica fast recovery a partire da ssthresh+3



Il val. del timeout deve essere \geq RTT (però RTT varia)

Se il val. è troppo piccolo → timeout prematuro. Se il val. è troppo grande → reazione lenta alla perdita dei segmenti

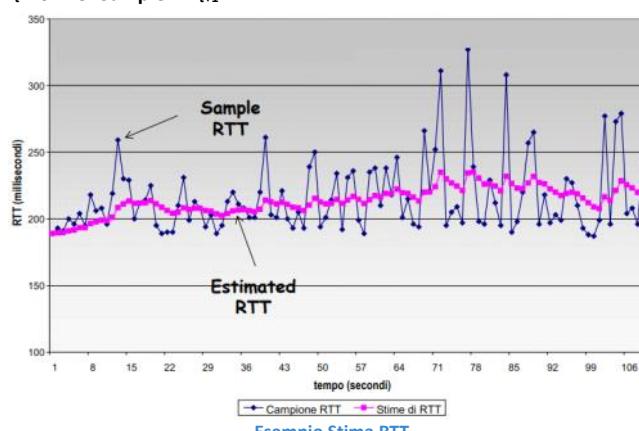
L'RTT viene stimato col **SampleRTT**, che è un tempo misurato dalla trasmissione del segmento fino alla ricezione di ACK. Esso ignora le ritrasmissioni e si usa un solo SampleRTT per più segmenti inviati insieme. Però SampleRTT varia a causa di congestione nei router e carico nei sistemi terminali, quindi serve una stima migliore. Si fa quindi una **media mobile esponenziale ponderata** influenzata dalle **misure passate** che la fanno decrescere esponenzialmente:

1° misurazione → $\text{EstimatedRTT}_0 = \text{SampleRTT}_0$

Prossime misurazioni → $\text{EstimatedRTT}_{t+1} = (1 - \alpha) * \text{EstimatedRTT}_t + \alpha * \text{SampleRTT}_{t+1}$

Valore tipico $\alpha = 0.125$ poiché si assegna minore peso alle misure recenti che a quelle più vecchie

$$\text{EstimatedRTT}_{t+1} = 0.875 * \text{EstimatedRTT}_t + 0.125 * \text{SampleRTT}_{t+1}$$



Impostazione del Timeout

Si utilizza la stima di quanto **si discostano tra loro** SampleRTT e EstimatedRTT come "**margine di sicurezza**":

1° misurazione → **DevRTT₀ = SampleRTT₀/2**

Prossime misurazioni → **DevRTT_{t+1} = (1-β)*DevRTT_t + β*|SampleRTT_{t+1}-EstimatedRTT_{t+1}|** (tipicamente β = 0.25)

Per impostare l'intervallo di timeout:

- Si imposta un val. iniziale pari a 1 sec
- Se avviene un timeout si **raddoppia**
- Appena viene ricevuto un segmento e aggiornato **EstimatedRTT**, si usa la formula:
TimeoutInterval = EstimatedRTT + 4*DevRTT

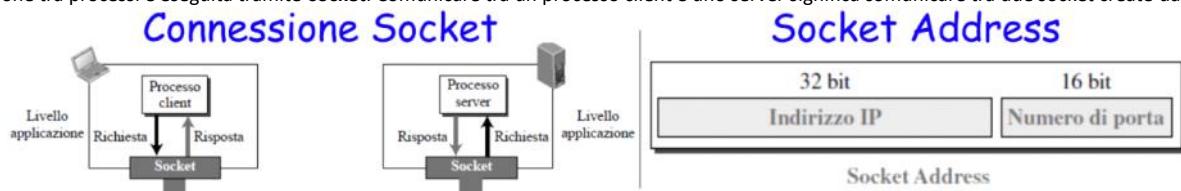
Livello Rete e Interfaccia Socket

sabato 3 maggio 2025 18:37

Nel paradigma client/server la **comunicazione** a liv. app avviene tra due programmi applicativi in esecuzione chiamati processi: un client e un server

- Un client è un programma in esecuzione che inizia la comunicazione inviando una richiesta
- Un server è un altro programma applicativo che attende la richiesta dal client

La comunicazione tra processi è eseguita tramite **socket**. Comunicare tra un processo client e uno server significa comunicare tra due socket create da entrambi

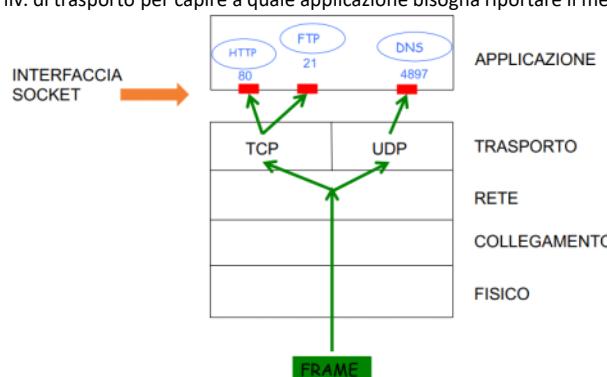


Indirizzamento dei Processi

Affinché un processo su un host invii un messaggio ad un altro host, il mittente deve identificare il processo destinatario.

Poiché l'IP dell'host non basta per conoscere il processo giusto, nell'identificatore si usa sia l'IP che i **numeri di porta** associati al processo destinatario.

N.B: il N° di porta è contenuto negli header del liv. di trasporto per capire a quale applicazione bisogna riportare il messaggio



Per comunicare è necessaria una **coppia di indirizzi socket**: **locale** (mittente) e **remoto** (destinatario)

Individuare Socket Lato Client

Serve un **socket address locale** (client) e **remoto** (server) per comunicare

Socket address locale:

- Conosce l'IP del pc su cui il client è in esecuzione
- Il n° di porta è assegnato temporaneamente dall'OS

Socket address remoto:

- N° porta noto in base all'app (es: HTTP porta 80)
- IP fornito dal **DNS**
- Oppure porta e IP noti al programmatore se vuole verificare il funzionamento di un'app

Individuare Socket Lato Server

Serve un **socket address locale** (server) e **remoto** (client) per comunicare

Socket address locale:

- Conosce l'IP del pc su cui il client è in esecuzione
- Il n° di porta è noto al server perché assegnato dal progettista

Socket address remoto:

- È il socket address locale del client che si connette
- Poiché numerosi client possono connettersi, il server trova i socket address all'interno del pacchetto di richiesta

Il socket address remoto varia ad ogni interazione con client diversi

La coppia di processi richiede un **servizio di trasporto** per la comunicazione. I due protocolli principali sono **TCP** e **UDP**

TCP (affidabile)

Orientato alla connessione: è richiesto un setup fra i processi client e server.

Mantiene l'ordine della sequenza dei bit consegnati

Offre:

- **Trasporto affidabile** fra processi di invio e di ricezione
- **Controllo di flusso**: il mittente non vuole sovraccaricare il destinatario
- **Controllo della congestione**: "strozza" il processo d'invio quando la rete è sovraccaricata

Non offre: temporizzazione, garanzie su ampiezza di banda minima, sicurezza (si può implementare, es SSL)

UDP (non affidabile)

Senza connessione: non è richiesto un setup fra i processi client e server

Trasferimento dati inaffidabile. È possibile che due messaggi inviati arrivino in tempi diversi.

Non offre: setup della connessione, affidabilità, controllo di flusso e congestione, temporizzazione, ampiezza di banda minima e sicurezza

Programmazione con Socket

Socket: interfaccia di un **host locale**, **creata dalle applicazioni**, **controllata dall'OS**, in cui il processo di un'applicazione può **inviare/ricevere** messaggi al/dal processo di un'altra applicazione

Si utilizza la **Socket API**:

- Esplicitamente creata, usata e distribuita dalle applicazioni
- Usa il paradigma client/server
- Due tipi di servizio di traporto tramite una Socket API:
 - Datagramma inaffidabile
 - Affidabile, orientata ai byte

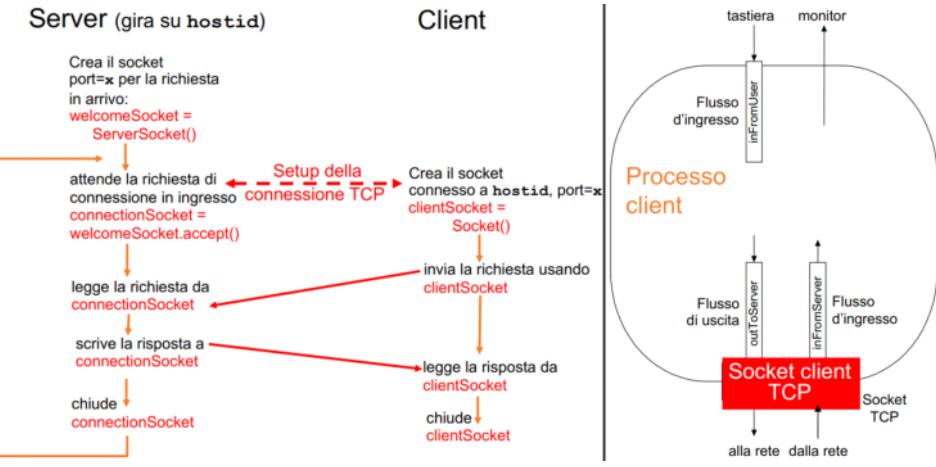
Programmazione Socket con TCP

Socket: un'ingresso tra il processo di un'applicazione e il protocollo di trasporto end-to-end (UDP o TCP)

Servizio TCP: trasferimento affidabile di **byte** da un processo all'altro

Connessione client/server:

- Il client crea un socket TCP per contattare il socket del server che è sempre in ascolto tramite un socket di benvenuto (per tutti i client diversi)
- Specifica l'IP e la porta del server e viene stabilita una connessione col server TCP
- Il server crea un nuovo socket per comunicare col client



Terminologia:

- **Flusso (stream):** sequenza di caratteri che fluisce verso/da un processo
- **Flusso d'ingresso (input stream):** collegato a un'origine di input per il processo (es: tastiera o socket)
- **Flusso di uscita (output stream):** collegato a un'uscita per il processo (es: monitor o socket)

Esempio client/server:

- 1) Client legge una riga dall'input standard (flusso `inFromUser`) e la invia al server tramite la socket (flusso `outToServer`)
- 2) Il server legge la riga dal socket
- 3) Il server converte la riga in lettere maiuscole e la invia al client
- 4) Il client legge nella sua socket la riga modificata e la visualizza (flusso `inFromServer`)

Package java.net

Il package `java.net` fornisce interfacce e classi per l'implementazione di applicazioni di rete. Fornisce le classi:

- **Socket e ServerSocket:** connessioni TCP
- **DatagramSocket:** connessioni UDP
- **URL:** connessioni HTTP
- **InetAddress:** rappresentare indirizzi Internet
- **URLConnection:** rappresentare connessioni a un URL

Esempio:

Java Client (TCP)

```
import java.io.*;
import java.net.*;
class TCPClient {
    public static void main(String argv[]) throws Exception {
        String sentence;
        String modifiedSentence;
        // Crea un flusso di ingresso
        BufferedReader inFromUser = new BufferedReader(
            new InputStreamReader(System.in));
        // Crea un socket client, connesso al server
        Socket clientSocket = new Socket("hostname", 6789);
        // Crea un flusso di dati in uscita collegato al socket
        DataOutputStream outToServer =
            new DataOutputStream(clientSocket.getOutputStream());
        // Crea un flusso di ingresso collegato al socket
        BufferedReader inFromServer =
            new BufferedReader(
                new InputStreamReader(clientSocket.getInputStream()));
        System.out.print("Inserisci una frase: ");
        sentence = inFromUser.readLine();
        // Invia la frase inserita dall'utente al server
        outToServer.writeBytes(sentence + '\n');
        // Legge la risposta dal server
        modifiedSentence = inFromServer.readLine();
        System.out.println("FROM SERVER: " + modifiedSentence);
        clientSocket.close(); // Chiude socket e connessione al server
    }
}
```

Java Server (TCP)

```
import java.io.*;
import java.net.*;
class TCPServer {
    public static void main(String argv[]) throws Exception {
        String clientSentence;
        String capitalizedSentence;
        // Crea un socket di benvenuto sulla porta 6789
        ServerSocket welcomeSocket = new ServerSocket(6789);
        while(true) {
            // Attende sul socket di benvenuto un contatto dal client
            Socket connectionSocket = welcomeSocket.accept();
            // Crea un flusso d'ingresso collegato al socket
            BufferedReader inFromClient =
                new BufferedReader(
                    new InputStreamReader(connectionSocket.getInputStream()));
            // Crea un flusso di uscita collegato al socket
            DataOutputStream outToClient =
                new DataOutputStream(connectionSocket.getOutputStream());
            // Legge la riga in input dal socket
            clientSentence = inFromClient.readLine();
            capitalizedSentence = clientSentence.toUpperCase() + '\n';
            // Scrive la riga in output sul socket
            outToClient.writeBytes(capitalizedSentence);
            // Fine del ciclo while, ricomincia il ciclo e
            // attende un'altra connessione col
        }
    }
}
```

Programmazione Socket con UDP

UDP: la connessione tra client e server è **inaffidabile** dei gruppi di byte ("datagrammi"), poiché i dati trasmessi possono perdere o arrivare in ordine sparso

Server (gira su hostid)



Client



Esempio:

Java Client (UDP)

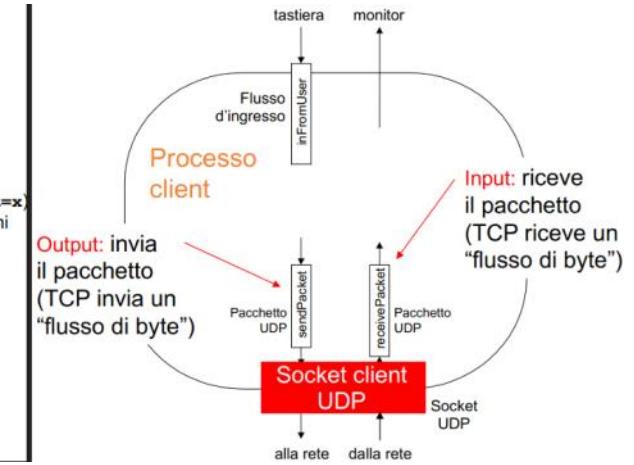
```

import java.io.*;
import java.net.*;
class UDPClient {
    public static void main(String args[]) throws Exception{
        // Crea un flusso d'ingresso
        BufferedReader inFromUser =
        new BufferedReader(new InputStreamReader(System.in));
        // Crea un socket client
        DatagramSocket clientSocket = new DatagramSocket();
        // Traduce l'hostname nell'IP tramite DNS
        InetAddress IPAddress = InetAddress.getByName("hostname");
        byte[] sendData = new byte[1024];
        byte[] receiveData = new byte[1024];
        String sentence = inFromUser.readLine();
        sendData = sentence.getBytes();
        // Crea il datagramma con i dati da trasmettere, lunghezza, IP, porta
        DatagramPacket sendPacket =
        new DatagramPacket(sendData, sendData.length, IPAddress, 9876);
        // Invia il datagramma al server
        clientSocket.send(sendPacket);
        DatagramPacket receivePacket =
        new DatagramPacket(receiveData, receiveData.length);
        // Legge il datagramma dal server
        // (client rimane inattivo finché non riceve un pacchetto)
        clientSocket.receive(receivePacket);
        String modifiedSentence = new String(receivePacket.getData());
        System.out.println("FROM SERVER:" + modifiedSentence);
        clientSocket.close();
    }
}
  
```

Java Server (UDP)

```

import java.io.*;
import java.net.*;
class UDPServer {
    public static void main(String args[]) throws Exception{
        // Crea un socket per datagrammi sulla porta 9876
        DatagramSocket serverSocket = new DatagramSocket(9876);
        byte[] receiveData = new byte[1024];
        byte[] sendData = new byte[1024];
        while(true){
            // Crea lo spazio per i datagrammi
            DatagramPacket receivePacket =
            new DatagramPacket(receiveData, receiveData.length);
            // Riceve i datagrammi in input dal client
            serverSocket.receive(receivePacket);
            String sentence = new String(receivePacket.getData());
            // Ottiene IP e porta del mittente
            InetAddress IPAddress = receivePacket.getAddress();
            int port = receivePacket.getPort();
            String capitalizedSentence = sentence.toUpperCase();
            sendData = capitalizedSentence.getBytes();
            // Crea il datagramma da inviare al client
            DatagramPacket sendPacket =
            new DatagramPacket(sendData, sendData.length, IPAddress,
            port);
            // Scrive il datagramma in output sul socket
            serverSocket.send(sendPacket);
        }
    }
}
  
```



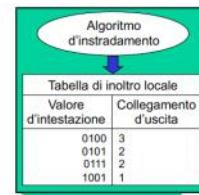
Instradamento

domenica 4 maggio 2025 16:33

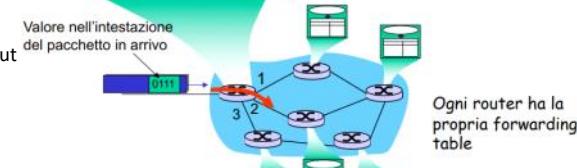
Quando un router riceve un pacchetto, come fa a sapere a chi inviarlo?

- **Instradamento (routing)**: determina il percorso seguito dai pacchetti dall'origine alla destinazione
 - Analogia: processo di pianificazione di un viaggio dall'origine alla destinazione
- **Inoltro (forwarding)**: trasferisce i pacchetti dall'input di un router all'output del router appropriato
 - Analogia: processo di attraversamento di un determinato svincolo

Gli algoritmi di routing creano le **tabelle di routing** che vengono usate per il forwarding



Routing algorithm: crea la forwarding table (determina valori inseriti nella tabella)

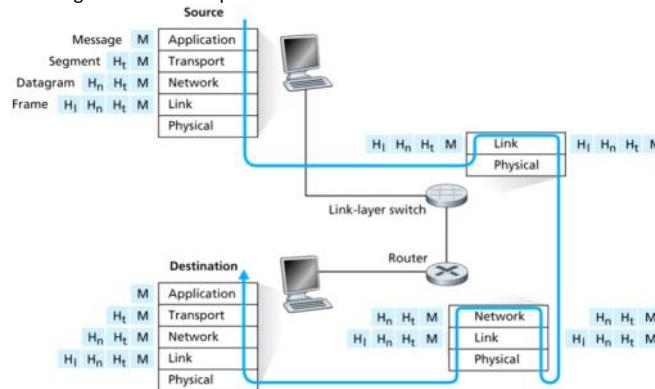


Forwarding table: specifica quale collegamento di uscita bisogna prendere per raggiungere la destinazione

Switch e Router

Packet switch (commutatore di pacchetto): dispositivo che si occupa di trasferire dall'interfaccia di input a quella di output, in base al val. del campo dell'intestazione del pacchetto

1. **Link-layer switch** (commutatore al liv. 2 di collegamento)
 - stabiliscono l'inoltro in relazione al val del campo nel liv. di collegamento (liv 2)
 - Usato per collegare singoli computer in una LAN
2. **Router** (commutatore al liv. 3 di rete)
 - stabiliscono l'inoltro in base al val del campo nel liv di rete (liv 3)
 - Riceve il pacchetto e in base alla routing table lo invia al prossimo salto verso la destinazione



Circuit Switching vs Packet Switching

Esistono due approcci per lo **switching**:

- Approccio a **circuito virtuale (circuit switched)** (orientato alle connessioni): prima che i datagrammi fluiscano, i due sistemi terminali e i router intermedi stabiliscono una connessione virtuale
- Approccio a **datagramma (packet switched)** (servizio senza connessione): ogni datagramma viaggia indipendente dagli altri

Rete a Circuito Virtuale

Il **circuito virtuale** consiste in:

1. Un percorso tra gli host
2. Num. VC, uno per ciascun collegamento
3. Righe nella tab. d'inoltro in ciascun router

Il pacchetto di un circuito virtuale ha un num. **VC** (etichetta di circuito) nella intestazione che rappresenta un'etichetta di flusso.

Il VC del pacchetto cambia su tutti i collegamenti lungo il percorso. Un nuovo VC viene rilevato dalla tabella d'inoltro (ogni router sostituisce il VC con un nuovo VC)

Un circuito virtuale può avere un num. VC diverso su ogni collegamento.

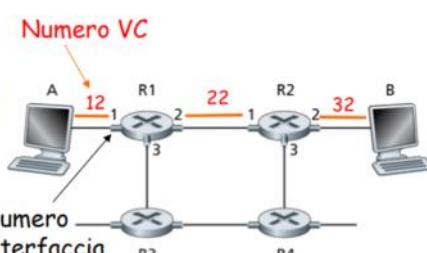
Tabella d'Inoltro

I router mantengono le informazioni sullo stato delle connessioni

Aggiungono alla tabella d'inoltro una nuova riga ogni volta che stabiliscono una nuova connessione (la cancellano quando viene rilasciata la connessione)

Tabella d'inoltro R1

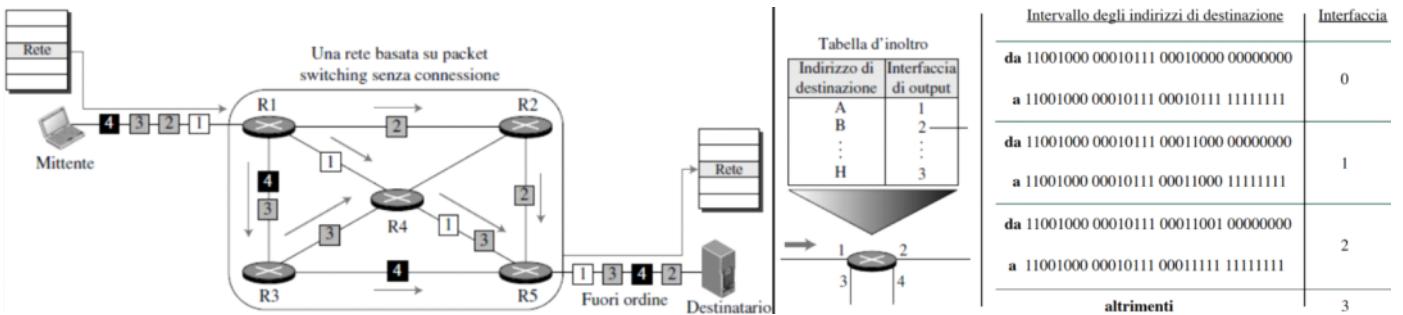
Interf. in ingresso	N° VC entrante	Interf. in uscita	N° VC uscente
1	12	2	22
2	63	1	18
3	7	2	17
1	97	3	87
...



Rete a Datagrammi

Internet è una rete a **datagramma** (packet switched)

- L'impostazione della chiamata non avviene a liv. di rete
- I router non conservano informazioni sullo stato dei circuiti virtuali (non c'è il concetto di "connessione" a liv di rete)
- I pacchetti sono inoltrati usando l'indirizzo dell'host destinatario. Passano attraverso una serie di router che usano gli indirizzi di destinazione per inviarli e possono intraprendere percorsi diversi



Confronto Prefisso

Quando si verificano corrispondenze multiple si prende quella col **prefisso più lungo**

Corrispondenza di un Prefisso	Interfaccia
11001000 00010111 00010	0
11001000 00010111 00011000	1
11001000 00010111 00011	2
Altrimenti	3

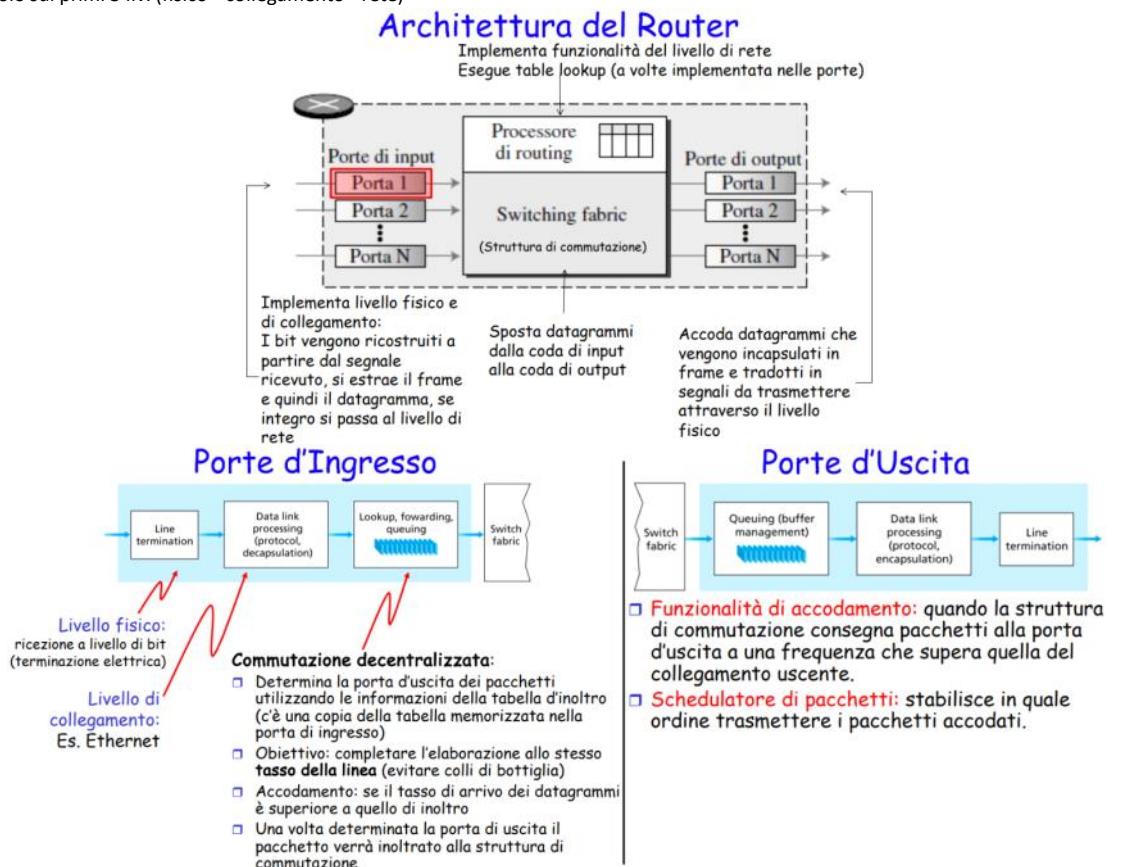
Esempi:

1. Con "11001000 00010111 00010110 10100001" corrisponde solo alla prima interfaccia (0)
2. Con "11001000 00010111 00011000 10101010" corrisponde per i primi 21 bit con le interfacce 1 e 2 però ha una corrispondenza col prefisso più lungo dell'interfaccia 1

Router

domenica 4 maggio 2025 19:52

I router lavorano solo sui primi 3 liv. (fisico - collegamento - rete)



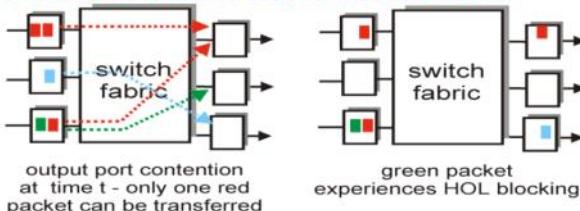
Accodamento

L'accodamento si verifica sia nelle porte d'ingresso che d'uscita

- **Velocità di commutazione:** freq. alla quale la struttura può trasferire pacchetti dalle porte d'ingresso a quelle d'uscita
- **Accodamento nelle porte d'ingresso:** quando vel. commutazione < rate porte d'ingresso. (per non avere accodamento la vel. di commutazione dovrebbe essere n*velocità della linea d'ingresso)
- **Accodamento nelle porte d'uscita:**
 - Quando vel. commutazione > rate porte d'uscita
 - Quando troppi pacchetti vanno sulla stessa porta d'uscita

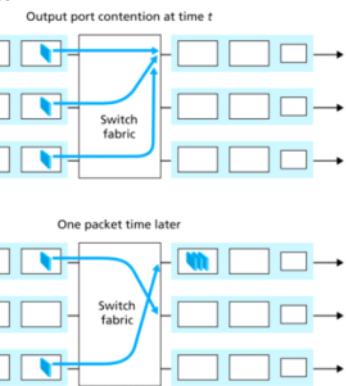
Accodamento sulle Porte d'Ingresso

- Oltre alla velocità inferiore della struttura di commutazione
- **Blocco in testa alla fila (HOL: head-of-the-line blocking):** un pacchetto nella coda d'ingresso deve attendere il trasferimento (anche se la propria destinazione è libera) in quanto risulta bloccato da un altro pacchetto in testa alla fila.
- **Se le code diventano troppo lunghe, i buffer si possono saturare e quindi causare una perdita di pacchetti!**



Accodamento sulle Porte d'Uscita

- Se la struttura di commutazione ha un rate superiore a quello della porta di uscita, si può verificare un accodamento.
- Se troppi pacchetti vanno sulla stessa uscita
- **Se le code diventano troppo lunghe, i buffer si possono saturare e quindi causare una perdita di pacchetti!**



Ricerca nella Tabella di Inoltro

La ricerca deve essere veloce per evitare accodamenti.

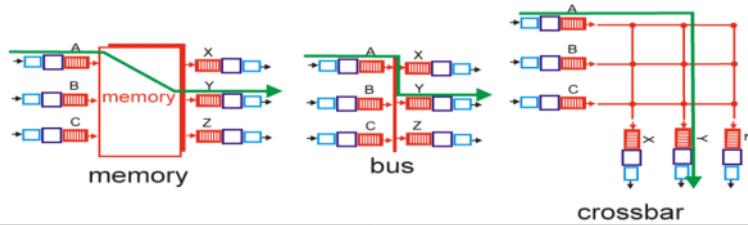
La tabella è implementata in una struttura ad albero:

- Ogni liv. dell'albero corrisponde a un bit dell'indir. di destinazione
- Per cercare un indir. si comincia dalla radice dell'albero
 - Se 0 allora il sottoalbero di **sinistra**
 - Se 1 allora il sottoalbero di **destra**
- Ricerca in **N passi** dove N è il num. di bit nell'indirizzo

Tecniche di Commutazione

Esistono 3 tecniche di commutazione:

- **In memoria:** 1° gen di router → tradizionali calcolatori con la commutazione effettuata tramite CPU. Il pacchetto viene copiato nella mem. del processore e veniva trasferito dalle porte d'ingresso a quella d'uscita
- **Tramite bus**
- **Attraverso rete d'interconnessione**



Commutazione tramite Bus

- Le porte input trasferiscono direttamente il pacchetto alle porte output tramite un **bus condiviso** (senza intervento processore d'instradamento)
- Si può trasferire **un solo pacchetto alla volta**
- I pacchetti che arrivano e trovano il bus occupato vengono accodati
- **Contesa per il bus:** la larghezza di banda è limitata da quella del bus
- Cisco 5600 usa un bus da 32 Gbps (sufficiente per reti aziendali/accesso)

Commutazione attraverso Rete d'Interconnessione

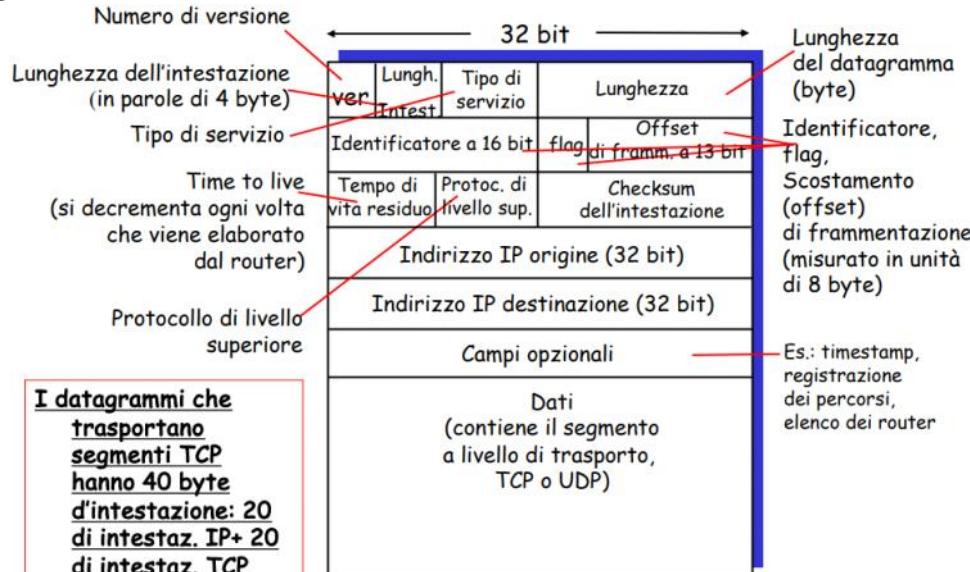
- Supera il limite di banda di un singolo bus condiviso
- Un **crossbar switch** è una **rete d'interconnessione** che consiste in **2n bus** che collegano n porte input a n porte output
- Tendenza attuale: frammentazione dei pacchetti IP a lunghezza variabile in celle di lunghezza fissa (riassemmati nelle porte output)
- Switch Cisco 12000 usa una crossbar da 60 Gbps

IP - Internet Protocol (IPv4)

domenica 4 maggio 2025 20:45

- Responsabile della **suddivisione in pacchetto**, dell'**inoltro** (forwarding) e della **consegna dei datagrammi** al liv. di rete (host to host)
- **Inaffidabile**, senza connessione, basato su datagrammi
- Offre un servizio di consegna best effort (non si garantisce la consegna dei dati o la loro integrità)

Formato dei Datagrammi



- **N° versione**: consente ai router la corretta interpretazione del datagramma (4: IPv4; 6: IPv6)
- **Lunghezza dell'intestazione**: siccome un IP può contenere un num. var di opzioni (incluse nell'intestazione), questi bit indicano dove inizia il campo dati
- **Tipo di servizio**: serve per distinguere diversi datagrammi con requisiti di qualità del servizio diverse (realtime)
- **Lunghezza del datagramma**: rappresenta la lunghezza tot. del datagramma IP inclusa l'intestazione (in byte). In genere non superiore a 1500 byte. Serve per capire se il pacchetto è arrivato completamente
- **Id, flag e offset di frammentazione**: servono per gestire la **frammentazione** dei pacchetti (IPv6 non prevede frammentazione)
- **Tempo di vita** (time to live o TTL): assicura che i datagrammi non restino in circolazione per sempre nella rete (es. instradamento ciclico). Il campo viene **decrementato** ad ogni hop e il datagramma viene **eliminato se TTL = 0**
- **Protocollo**: indica a quale protocollo a liv. trasporto **passare il datagramma**. Questo campo è usato solo quando il datagramma raggiunge la destinazione finale
 - 6: TCP
 - 17: UDP
 - 1: ICMP
 - 2: IGMP
 - 89: OSPF
- **C checksum dell'intestazione**: consente ai router di **rilevare errori** sui datagrammi ricevuti
- **IP di origine e destinazione**: inseriti dall'host che crea il datagramma (dopo aver effettuato una ricerca DNS)
- **Opzioni**: campi che consentono di **estendere l'intestazione IP** (usate per test o debug di rete)
- **Dati**: contiene il **segmento di trasporto da consegnare** alla destinazione (può contenere altri tipi di dati, es: messaggi ICMP, IGMP, ecc)

Frammentazione

Un datagramma IP può dover viaggiare attraverso varie reti, ognuna con caratteristiche diverse. Ogni router estraе il datagramma dal frame, lo elabora e lo incapsula in un nuovo frame

La **Maximum Transfer Unit (MTU)** è la max. quantità di dati che un frame a liv. di collegamento può trasportare. La MTU varia in base alla tecnologia. Quando viene inserito in rete un pacchetto di dim. > MTU, il pacchetto deve essere frammentato per essere trasmesso.

I datagrammi IP grandi vengono **suddivisi ("frammentati")** in datagrammi IP più piccoli. Arrivati a destinazione i frammenti vengono poi riassemblati.

La frammentazione però aumenta il num. di byte trasferiti (di poco in realtà) poiché per ogni nuovo datagramma creato da frammentazione bisogna dargli un header aggiuntivo di 20 byte.

I **bit dell'intestazione IP** sono usati per identificare e ordinare i frammenti.

Quando un host di destinazione riceve una serie di datagrammi dalla stessa origine, deve:

- Individuare i frammenti
- Determinare quando ha ricevuto l'ultimo
- Stabilire come debbano essere riassemblati

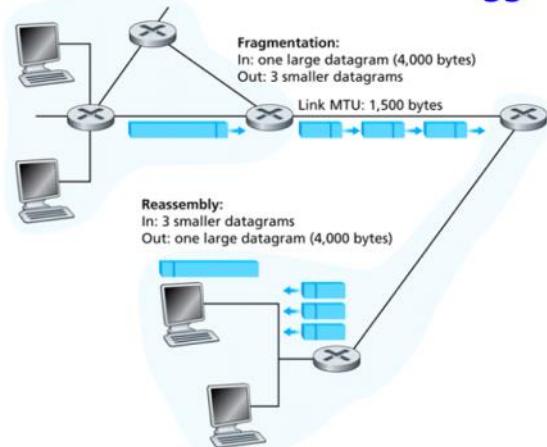
Identificazione (16 bit): ID associato a ogni datagramma durante la creazione (unico per tutti i frammenti). IP + ID identificano un datagramma

Flag (3 bit):

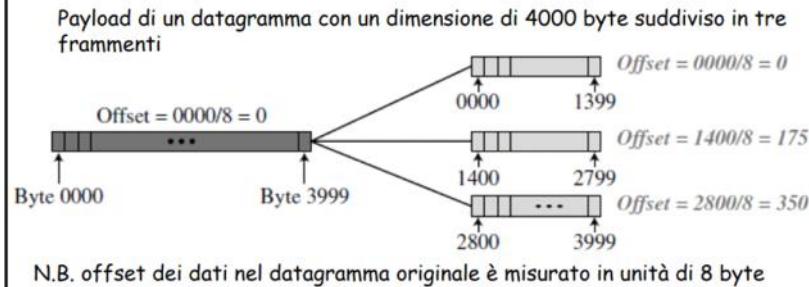
1. **Riservato**
2. **Do not fragment**: 1 non frammentare, 0 si può frammentare
3. **More fragments (M)**: 1 frammenti intermedi, 0 ultimo frammento

Offset (scostamento laterale): specifica l'ordine del frammento all'interno del datagramma originario

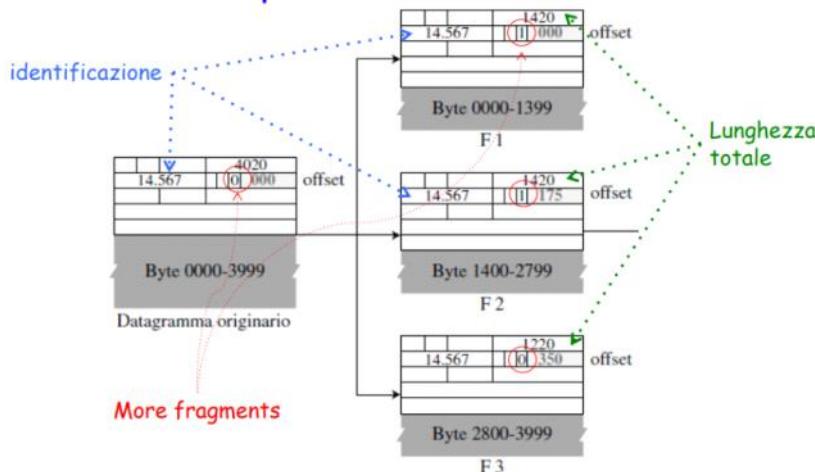
Frammentazione e Riassemblaggio



Offset (13 bit)

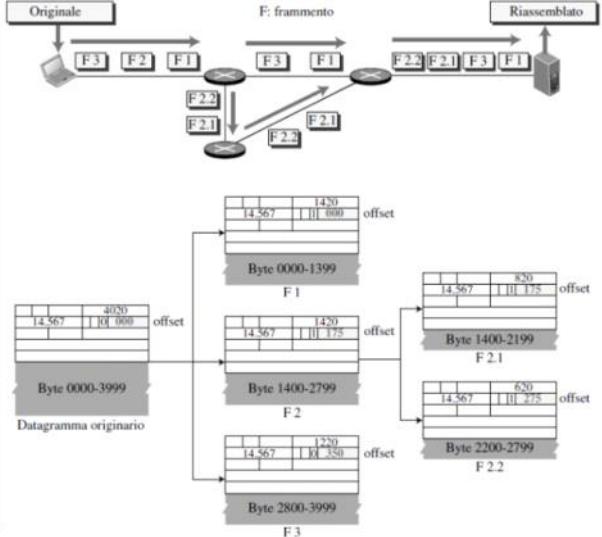


Esempio di Frammentazione



In totale vengo trasferiti 40 byte in più: ci sono due header aggiuntivi!!!

Frammentazione di un Frammento



Riassemblaggio a Destinazione

- Il primo frammento ha un val. del campo offset pari a 0.
- L'offset del secondo frammento si ottiene dividendo per 8 la lunghezza del primo frammento (esclusa intestazione di 20 byte)
 - Nell'immagine è $1420 - 20 \text{ byte (intestazione)} = 1400/8 = 175$ (offset secondo frammento)
- L'offset del terzo frammento si ottiene dividendo per 8 la somma della lunghezza del primo e secondo frammento (escluse intestazioni) (oppure sommare l'offset del secondo frammento con la lunghezza del secondo frammento divisa per 8)
 - Nell'immagine è $1420 - 20 + 1420 - 20 = 2800/8 = 350$ (offset terzo frammento) (oppure 175 (2° offset) + 1400 (2° frammento)/8 = $175 + 175 = 350$)
- ...
- L'ultimo frammento ha il bit M impostato a 0 (indica che è l'ultimo frammento appunto)

Indirizzamento IPv4

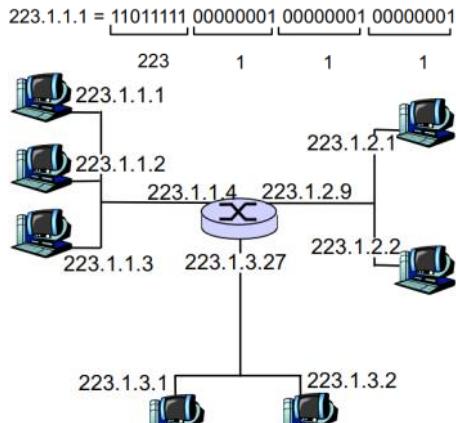
mercoledì 7 maggio 2025 13:00

Indirizzo IP: 32 bit (4 byte) in notazione decimale puntata (ciascun byte dell'indir è indicato come forma decimale)

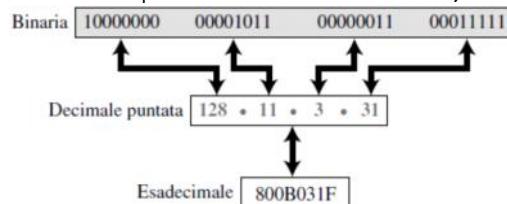
Interfaccia: confine tra host e collegamento fisico

- I router devono essere almeno connessi tra due collegamenti (due router, due host, un host e un router, ecc)
- Un host, in genere, ha un'interfaccia
- A ciascuna interfaccia è associato un indirizzo IP

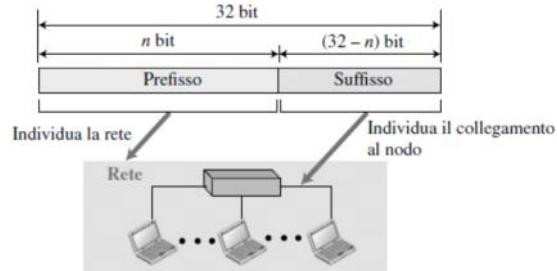
Esempio:



Si possono avere 2^{32} indirizzi totali (più di 4 miliardi) e un indirizzo si può descrivere in **notazione binaria, decimale puntata o esadecimale**



Gerarchia nell'Indirizzamento



In un indirizzo IP, il prefisso indica la rete o sottorete a cui appartiene l'indirizzo. Il suffisso invece indica l'host collegato a quella rete.

Il prefisso può avere lunghezza:

- **Fissa:** indirizzamento **con classi**
- **Variabile:** indirizzamento **senza classi**

Indirizzamento Con Classi

3 lunghezze fisse per il prefisso (8, 16, 24)

	8 bit	8 bit	8 bit	8 bit	Classe	Prefissi	Primo byte
Classe A	0	Prefisso	Suffisso		A	$n = 8$ bit	Da 0 a 127
Classe B	10	Prefisso	Suffisso		B	$n = 16$ bit	Da 128 a 191
Classe C	110	Prefisso	Suffisso		C	$n = 24$ bit	Da 192 a 223
Classe D	1110		Indirizzi multicast		D	Non applicabile	Da 224 a 239
Classe E	1111	Riservati per uso futuro			E	Non applicabile	Da 240 a 255

Pro: una volta trovato l'indirizzo, si può facilmente risalire alla classe e la lunghezza del prefisso

Problema: esaurimento degli indirizzi:

- Classe A: si può assegnare solo a 128 organizzazioni al mondo, ognuna con 16.777.216 nodi
 - La maggior parte degli indirizzi va sprecata
 - Poche organizzazioni (solo 128) possono usufruire della classe A
- Classe B: stessi problemi della A
- Classe C: pochi indirizzi (256) per rete

Indirizzamento Senza Classi

È necessaria una maggiore flessibilità nell'assegnare gli indirizzi.

Vengono usati blocchi di **lunghezza variabile** per il prefisso e il suffisso e la lunghezza del prefisso è variabile (da 0 a 32 bit) e viene aggiunta all'indirizzo separata da uno slash.

Notazione CIDR (Classless InterDomain Routing)

Struttura IP: l'indirizzo viene diviso in **due parti** → **a.b.c.d/n** dove:



vengono usati blocchi di lunghezza variabile per il preffiso e il suffisso e la lunghezza del preffiso è variabile (ad es 32 bit) e viene aggiunta al indirizzo separata da uno slash

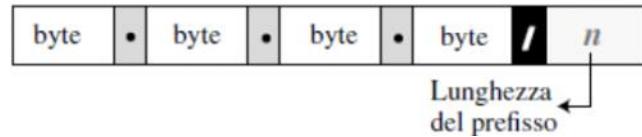
Notazione CIDR (Classless InterDomain Routing)

Struttura IP: l'indirizzo viene diviso in **due parti** → **a.b.c.d/n** dove:

- **a.b.c.d** è l'**indirizzo IP** normale composto da 4 num da 0 a 255 (quattro byte, 32 bit)
- **/n** indica il num. di bit dell'IP riservati alla parte di rete

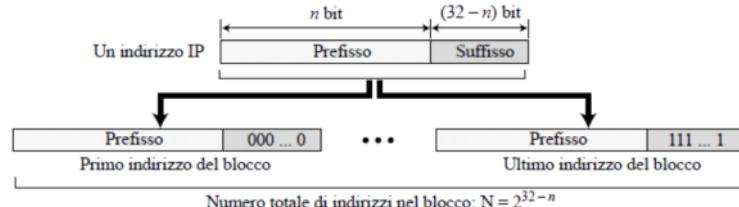
Esempio: 192.168.16.0/23 → Indirizzo IP 192.168.16.0 (32 bit), prefisso di rete = 23 bit, parte di host = 32-23 = 9 bit

Indirizzo in binario: 11000000.10101000.00001000.00000000 (23 bit rete, 9 bit host)



Il num. di indirizzi per gli host è dato da 2^{32-n} (es $2^9 = 512$ indirizzi)

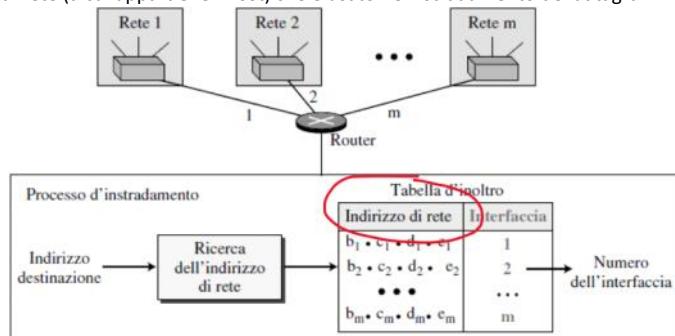
- Per trovare il primo indirizzo si impostano a 0 tutti i bit del suffisso
- Per trovare l'ultimo indirizzo si impostano a 1 tutti i bit del suffisso



Maschera e Indirizzo di Rete

Maschera dell'indirizzo (Subnet Mask): num. composto da 32 bit in cui i primi n bit a sinistra sono impostati a 1 e il resto (32-n) a 0 (es 255.255.255.0)

Mediante la maschera si ottiene l'indirizzo di rete (a cui appartiene l'host) che è usato nell'istradamento dei datagrammi verso la destinazione



La maschera può essere usata per calcolare in modo efficiente le informazioni di un blocco, usando solo tre operatori sui bit (Esempi con IP = 192.168.5.134 e Subnet mask = 255.255.255.0)

- Il num. degli indirizzi del blocco è $N = \text{NOT}(\text{maschera}) + 1$
 - Es: maschera/24 (255.255.255.0) indica che i primi 25 bit sono per la rete. Il NOT inverte tutti i bit nella maschera $\text{NOT}(255.255.255.0) = 0.0.0.255$. Con l'aggiunta di 1 si ottiene il num. di indirizzi possibili nella subnet: $255 + 1 = 256$ indirizzi
- Il primo indirizzo del blocco = (qualsiasi indir. del blocco) AND (maschera)
 - Es: 192.168.5.134 AND 255.255.255.0
IP: 11000000.10101000.00000101.10000110
Mask: 11111111.11111111.11111111.00000000
Result: 11000000.10101000.00000101.00000000 → 192.168.5.0
Primo indirizzo: 192.168.5.0
- L'ultimo indirizzo del blocco = (qualsiasi indir. del blocco) OR (NOT(maschera))
 - Es: 192.168.5.134 OR (NOT 255.255.255.0) = 192.168.5.134 OR 0.0.0.255
IP: 11000000.10101000.00000101.10000110
NOT: 00000000.00000000.00000000.11111111
Result: 11000000.10101000.00000101.11111111 → 192.168.5.255
Ultimo indirizzo: 192.168.5.255

Indirizzi IP Speciali

0 0	This host
0 0 ... 0 0	A host on this network
1 1	Broadcast on the local network
Network 1 1 1 1 ... 1 1 1 1	Broadcast on a distant network
127 (Anything)	Loopback

- **0.0.0.0** è usato dall'host al boot
- Gli IP con **tutti 0 come num. di rete** si riferiscono alla rete corrente
- 255.255.255.255 permette la trasmissione **broadcast** sulla rete locale (generalmente LAN)
- Gli IP con **tutti 1 nel campo host** permettono l'invio di pacchetti **broadcast** a LAN distanti
- Gli IP in forma **127.x.y.z** sono riservati al **loopback** (i pacchetti vengono elaborati localmente e trattati come pacchetti in arrivo)

DHCP

mercoledì 7 maggio 2025 13:00

Un ISP ottiene un blocco di indirizzi dall'**ICANN (Internet Corporation for Assigned Names and Numbers)** che gestisce l'allocazione dei blocchi di indirizzi, i server radice DNS e assegna e risolve dispute sui nomi di dominio

Un amministratore di rete deve contattare il proprio ISP per ottenere un blocco di indirizzi IP contigui con prefisso comune da usare in una sottorete
Otterrà indirizzi nella forma a.b.c.d/x, dove x bit indicano la sottorete e (32-x) bit indicano i singoli dispositivi dell'organizzazione (o una sottorete aggiuntiva)

Un dispositivo può ottenere un indirizzo assegnato persistente o temporaneo e manualmente o in automatico.

- Manualmente sulle impostazioni del dispositivo
- Automaticamente con **DHCP**

DHCP (Dynamic Host Configuration Protocol)

È un programma client/server di liv. applicazione che consente a un host di ottenere un IP **dinamicamente**.

- **Client:** host appena connesso desidera informazioni sulla configurazione di rete, non solo un IP
- **Server:**
 - Ogni sottorete in genere dispone di un server DHCP
 - Altrimenti il router fa l'agente di appoggio DHCP, conosce un server DHCP per quella rete

- Si può impostare un **IP temporaneo o persistente** (ogni volta che si entra in rete si assegna lo stesso IP)
- È possibile rinnovare la proprietà dell'indirizzo in uso e è possibile **riutilizzare gli indirizzi**.
- Molto utile quando gli host si aggiungono e rimuovo dalla rete con estrema frequenza

Quando un host vuole entrare in una rete, necessita di:

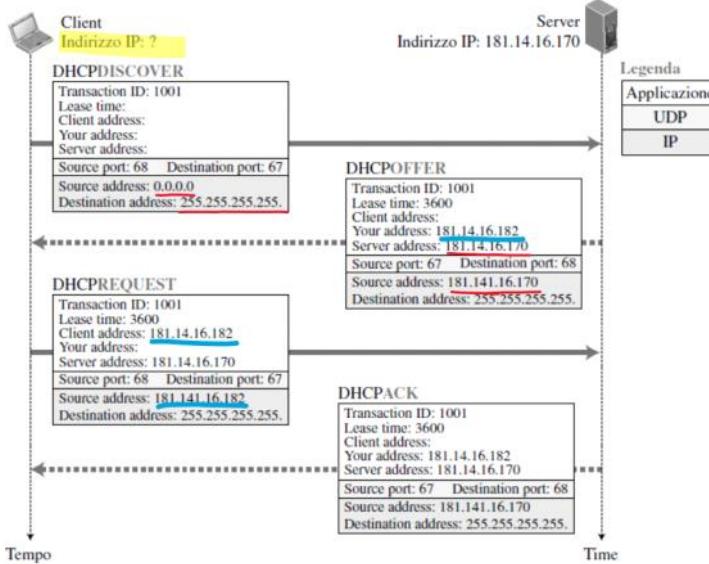
- Indirizzo IP
- Subnet Mask
- Indirizzo del router
- Indirizzo DNS

Panoramica di connessione DHCP:

1. L'host invia un messaggio broadcast "**DHCPDISCOVER**"
2. Il server DHCP risponde con "**DHCPOFFER**"
3. L'host richiede l'IP: "**DHCPRREQUEST**"
4. Il server DHCP invia l'indirizzo: "**DHCPACK**"

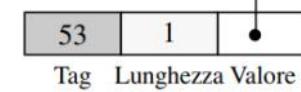
0	8	16	24	31	Campi:
Opcode	Htype	HLen	HCount		Opcode: codice operazione, richiesta (1) o risposta (2) Htype: tipologia dell'hardware (Ethernet, ...)
Transaction ID					HLen: lunghezza dell'indirizzo hardware
Time elapsed		Flags			HCount: numero massimo di hop che il pacchetto può compiere
					Transaction ID: un numero intero impostato dal client e ripetuto dal server
					Time elapsed: il numero di secondi da quando il client ha inviato il primo messaggio di richiesta
					Flags: il primo bit definisce l'unicast (0) o il multicast (1); gli altri 15 bit non vengono utilizzati
					Client IP address: l'indirizzo IP del client, impostato a 0 se il client non lo conosce
					Your IP address: l'indirizzo IP del client, inviato dal server
					Server IP address: l'indirizzo IP del server, impostato ad un indirizzo IP di broadcast
					se il client non lo conosce
					Gateway IP address: l'indirizzo del router di default
					Server name: il nome di dominio del server (64 byte)
					Boot file name: un nome di file (128 byte) usato per informazioni aggiuntive
					Options: un campo da 64 byte con un duplice scopo, come descritto nel testo

Formato messaggi DHCP



- Utilizza porte **well-known** (client: 68, server: 67) poiché la risposta dal server è **broadcast**
- Il server invia l'IP e nel **DHCPACK** inserisce il pathname per un file contenente le info mancanti (maschera, server DNS, router, ecc). Il client usa FTP per ottenere il file
- Nelle opzioni viene indicato un **magic cookie** pari a 99.130.83.99 che indica il **tipo di messaggio** (1: **DHCPDISCOVER**, 2: **DHCPOFFER**, ecc)

1	DHCPDISCOVER	5	DHCPACK
2	DHCPOFFER	6	DHCPNACK
3	DHCPRREQUEST	7	DHCPRRELEASE
4	DHCPDECLINE	8	DHCPINFORM



NAT

mercoledì 7 maggio 2025 13:00

Sottorete

La **sottorete** è una rete isolata i cui punti terminali sono collegati a un host o un router.

Indirizzo IP:

- Parte di sottorete (**prefisso**)
- Parte dell'host (**suffisso**)

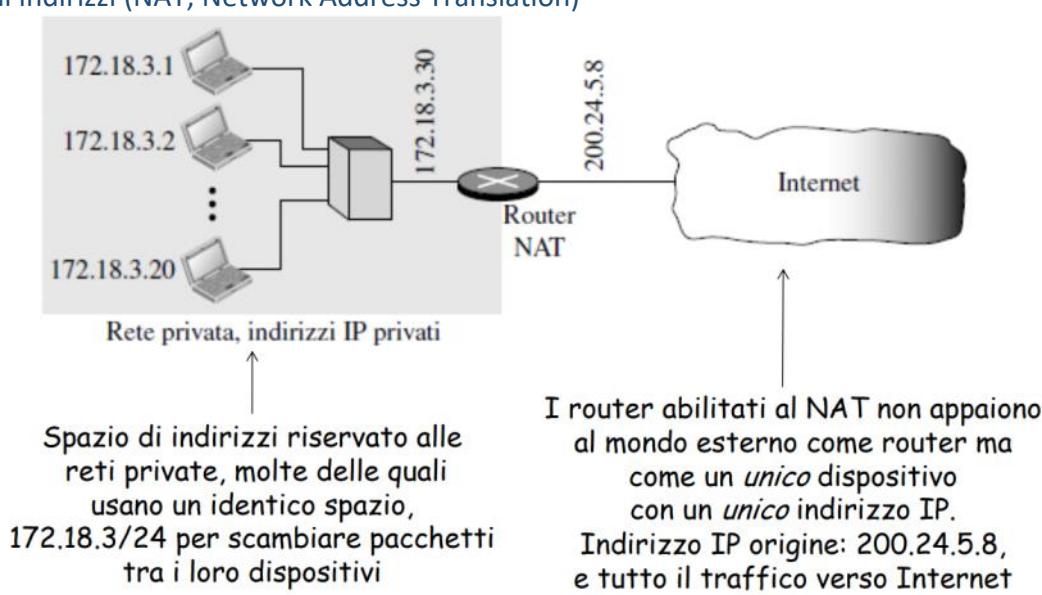
Se la subnet mask è /24 indica che i 24 bit più a sinistra dell'indirizzo definiscono l'indirizzo della sottorete.

Quindi ogni host connesso alla sottorete 223.1.1.0/24 deve avere un indirizzo nella forma 223.1.1.xxx

Se abbiamo un indirizzo IP nella forma a.b.c.d e la sua subnet mask, per conoscere a quale blocco appartiene dobbiamo controllare il prefisso.

- Se è **multiplo di 8 bit**, es a.b.c.d/24, allora controlliamo gli ultimi (32-24) = 8 bit (quindi d) → gli indirizzi andranno da a.b.c.0 a a.b.c.255
- Se non è **multiplo di 8 bit**, es a.b.c.d/26, allora controlli gli ultimi (32-26) = 6 bit (quindi gli ultimi 6 di d) → Se d = 10uvvxyz allora gli indirizzi andranno da 10000000 (128) a 10111111 (191)

Se l'amministratore di rete che ha ricevuto il blocco necessita di un num. maggiore di indirizzi la soluzione oppure l'ISP è impossibilitato ad allocare un nuovo intervallo di indirizzi per mancanza di indirizzi aggiuntivi nella sottorete, una soluzione è l'**uso di indirizzi privati** e si adotta la **traduzione degli indirizzi di rete (NAT)**

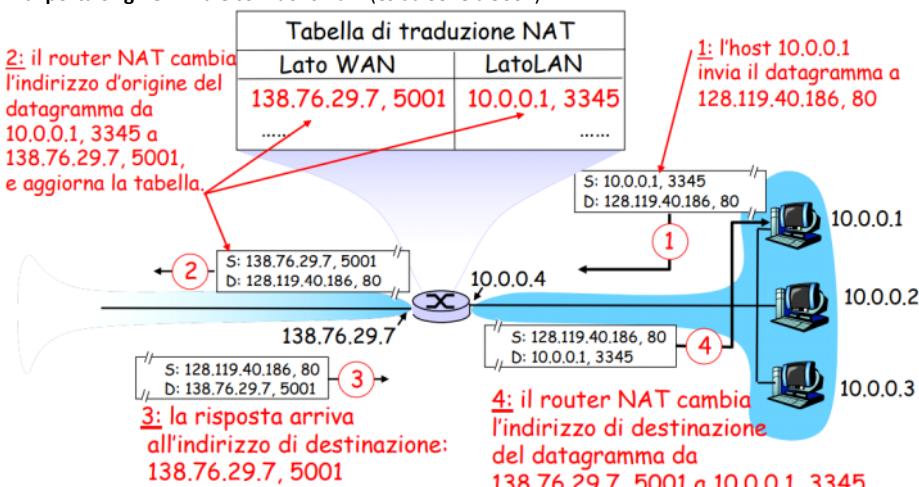


Il router abilitato a NAT nasconde i dettagli della rete domestica al mondo esterno

- Non è necessario allocare un intervallo di indirizzi da un ISP: **un unico IP è sufficiente per tutte le macchine nella rete locale**
- È possibile cambiare gli indirizzi nella rete privata senza doverlo comunicarlo all'Internet globale
- È possibile cambiare ISP senza modificare gli indir. delle macchine nella rete privata
- Dispositivi interni alla rete non esplicitamente indirizzabili e visibili dal mondo esterno (**maggior sicurezza**)

Implementazione:

Quando un router NAT riceve il datagramma, genera per se un **nuovo num. di porta d'origine** (es 5001), **sostituisce l'IP origine** col proprio IP sul lato WAN (es 200.24.5.8) e **sostituisce il num. di porta origine iniziale** col nuovo num (es da 3348 a 5001)



Poiché il num. di porta è di 16 bit, il NAT può supportare più di 60000 connessioni simultanee con un solo IP sul lato WAN

Contro della NAT:

- I router dovrebbero elaborare pacchetti solo fino al liv 3

- Il num. di porta è usato per identificare host e non processi
- Viola l'**argomento punto-punto** (gli host dovrebbero comunicare tra di loro direttamente, senza intromissione di nodi ne modifica di IP/porta)
- Interferenza con app P2P

Forwarding Datagrammi IP

giovedì 8 maggio 2025 11:33

Inoltrare significa **collocare il datagramma nel percorso giusto** (porta di uscita del router) che lo porterà verso la destinazione (**Inviare il datagramma al next hop**)
Ogni router nel percorso accede alla **tab. di routing** per trovare il next hop. L'inoltro richiede una riga nella tab. per ogni blocco di rete

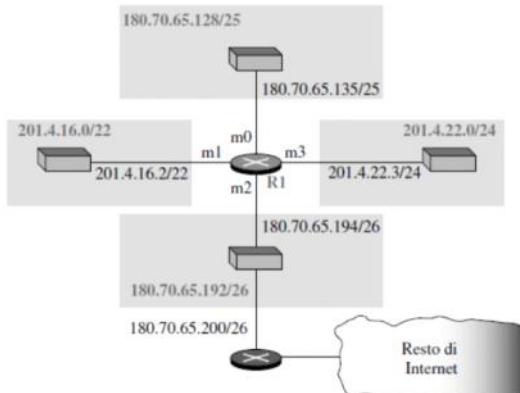


Tabella di inoltro per il router R1

Indirizzo di rete	Hop successivo	interfaccia
180.70.65.192/26	-	m2
180.70.65.128/25	-	m0
201.4.22.0/24	-	m3
201.4.16.0/22	-	m1
default	180.70.65.200	m2

Bit a sinistra nell'indirizzo di destinazione	Salto successivo	Interfaccia
10110100 01000110 01000001 11	-	m2
10110100 01000110 01000001 1	-	m0
11001001 00000100 00011100	-	m3
11001001 00000100 000100	-	m1
Default	180.70.65.200	m2

- Nella tab. di routing sono presenti i prefissi degli indir di rete (lunghezza < 32 bit)
- Un datagramma contiene l'IP intero (32 bit) e non indica la lunghezza del prefisso
- Quindi si controlla con quale elemento della tab. l'IP combacia di più.

Esempio: quando arriva un IP in cui i 26 bit a sx combaciano con i bit della prima riga, il pacchetto viene inviato nell'interfaccia m2. Analogamente negli altri casi

Esempio

Arriva un datagramma con indirizzo di destinazione 180.70.65.140 (10110100.01000110.01000001.10001100). Il router esegue i seguenti passaggi:

- Applica la prima maschera (/26) all'indir:

10110100.01000110.01000001.10001100 (indir destinazione)
10110100.01000110.01000001.11 (1° indir di rete)

Vediamo che **non combacia** con l'indir di rete corrispondente

- Applica la seconda maschera (/25) all'indir:

10110100.01000110.01000001.10001100 (indir destinazione)
10110100.01000110.01000001.1 (2° indir di rete)

Vediamo che **combacia** con l'indir di rete corrispondente. L'indir. del next hop e il num. di interfaccia m0 vengono usati per inoltrare il datagramma

Aggregazione degli Indirizzi

Inserire una riga per blocco può portare a tab. molto lunghe, aumentando il tempo di ricerca. La soluzione è **l'aggregazione degli indirizzi**

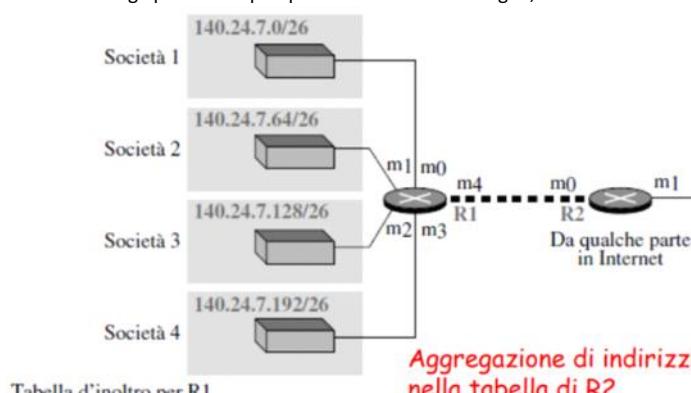


Tabella d'inoltro per R1

Indirizzo di rete/maschera	Indirizzo del salto successivo	Interfaccia
140.24.7.0/26	-----	m0
140.24.7.64/26	-----	m1
140.24.7.128/26	-----	m2
140.24.7.192/26	-----	m3
0.0.0.0/0	Indirizzo di R2	m4

Cosa accade se l'organizzazione 4 è connessa al router R2?

Tabella d'inoltro per R2

Indirizzo di rete/maschera	Indirizzo del salto successivo	Interfaccia
140.24.7.192/26	-----	m1
140.24.7.0/24	Indirizzo di R1	m0
0.0.0.0/0	Router di default	m2

Tabella d'inoltro per R1

Indirizzo di rete/maschera	Indirizzo del salto successivo	Interfaccia
140.24.7.0/26	-----	m0
140.24.7.64/26	-----	m1
140.24.7.128/26	-----	m2
0.0.0.0/0	Router di default	m3

Corrispondenza con la maschera più lunga!!!

ICMP (Internet Control Message Protocol)

giovedì 8 maggio 2025 12:00

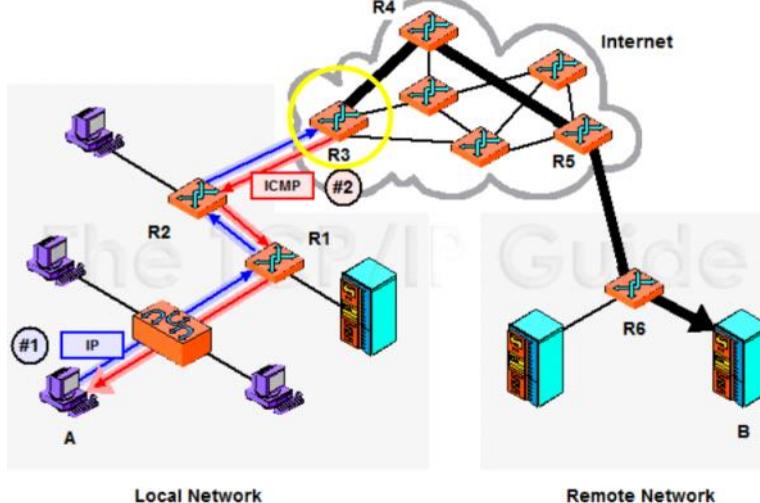
Il protocollo ICMP viene usato da host e router per scambiarsi informazioni a liv di rete e si occupa principalmente di:

- Notificare gli host di eventuali errori durante una connessione
- Segnalazioni del router
- Diagnostica di rete

Un uso tipico è di provvedere un meccanismo di feedback quando viene inviato un datagramma. Nell'esempio, A cerca di inviare un datagramma a B.

Però, incontra un problema al router R3 che causa al datagramma di essere perso.

R3 manda quindi un **messaggio ICMP ad A** per indicare che c'è stato un problema, così che A possa correggerlo, se possibile.



Diagnostica di Rete

Messaggi ICMP: hanno un campo **tipo** e un campo **codice** e contengono l'intestazione e i primi 8 byte del datagramma IP che ha provocato la generazione del messaggio

Tipo	Codice	Descrizione
0	0	Risposta eco (a ping)
3	0	rete destin. irraggiungibile
3	1	host destin. irraggiungibile
3	2	protocollo dest. irraggiungibile
3	3	porta destin. irraggiungibile
3	6	rete destin. sconosciuta
3	7	host destin. sconosciuto
4	0	riduzione (controllo di congestione)
8	0	richiesta eco
9	0	annuncio del router
10	0	scoperta del router
11	0	TTL scaduto
12	0	errata intestazione IP

È possibile usare ICMP per scambiare tra host e router dei report degli errori (host, rete, porta, protocollo irraggiungibile) o diagnosticare la rete con:

Ping

Verifica se un host è raggiungibile, misurando il ritardo (latenza) del collegamento (tempo per andare e tornare).

Funzionamento:

- Invia un **ICMP Echo Request** all'indir. di destinazione
- Se il destinatario riceve il pacchetto, risponde con **ICMP Echo Reply**
- Ping calcola il **round-trip time (RTT)** (tempo tra richiesta e risposta)

```
C:\>ping www.google.com

Pinging www.google.com [216.58.204.228] with 32 bytes of data:
Reply from 216.58.204.228: bytes=32 time=17ms TTL=118

Ping statistics for 216.58.204.228:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 17ms, Maximum = 17ms, Average = 17ms
```

Traceroute

Scopre il **percorso** per raggiungere una destinazione IP, mostrando tutti i **router intermedi** (hop) attraversati lungo la rete.

Come funziona:

- Il programma invia una serie di datagrammi IP con TTL crescente alla destinazione, ognuno con un segmento UDP con num. di porta inutilizzata
 - Il primo con TTL = 1, il secondo con TTL = 2, ecc S
 - Num. di porta improbabile (quindi quando arriverà tornerà un errore di porta non raggiungibile)
 - L'origine avvia un timer per ogni datagramma per l'RTT

- Siccome ogni router decremente TTL, quando esso arriva a 0 il router invia un messaggio **ICMP Time Exceeded** (tipo 11, cod 0) e Traceroute potrà capire quanto tempo ci mette quel messaggio a tornare e quale router ha scartato il pacchetto e in quale ordine
- Quando il messaggio ICMP arriva, l'origine può calcolare l'RTT totale

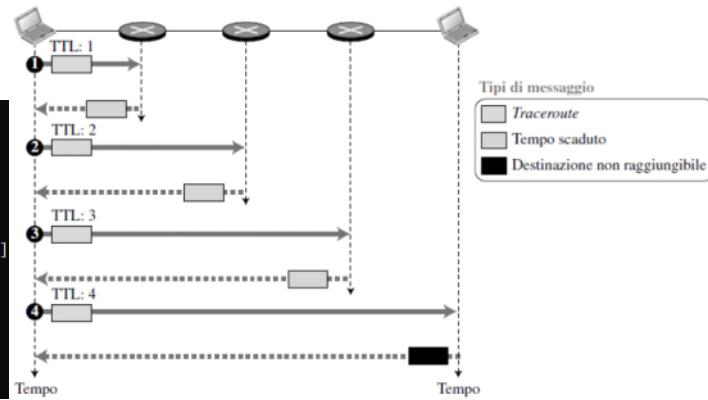
Criteri di arresto dell'invio:

- Il segmento UDP arriva a destinazione
- La destinazione restituisce un ICMP di porta non raggiungibile (tipo 3, cod 3)
- Quando l'origine riceve questo messaggio ICMP, si blocca

```
C:\>tracert www.google.com
```

```
Tracing route to www.google.com [216.58.204.228]
over a maximum of 30 hops:
```

```
1      2 ms      2 ms      2 ms  10.15.0.1
2     16 ms      4 ms      3 ms  192.168.1.5
3     10 ms      3 ms      2 ms  ru-uniroma3-r11-rm03.rm03.garr.net [193.206.142.57]
4     17 ms     11 ms     11 ms  r11-rm03-r11-rm02.rm02.garr.net [185.191.181.215]
5     *          *          * Request timed out.
6    41 ms   103 ms     10 ms  rs1-rm02-re1-mi02.mi02.garr.net [185.191.181.69]
7    17 ms     10 ms      9 ms  142.250.164.230
8    18 ms     11 ms     11 ms  192.178.99.215
9    17 ms     10 ms     14 ms  192.178.82.63
10   18 ms     10 ms     10 ms  mil07s18-in-f4.1e100.net [216.58.204.228]
```



Routing

lunedì 2 giugno 2025 19:19

Inoltrare = collocare il datagramma sul **giusto percorso (prossimo hop)**

L'host inoltra il datagramma al router locale, il router accede alla **tabella di routing** per trovare il next hop. L'inoltro richiede una riga nella tabella per ogni blocco di rete

Siccome esistono più percorsi, il routing si occupa trovare il **miglior percorso** e inserirlo nella tabella di routing (o tab. di forwarding)
Quindi il routing **costruisce le tabelle**, il forwarding le usa.

La rete si può immaginare come un **grafo pesato non diretto** $G = (N, E)$ con N = insieme di nodi (router) e E = insieme di archi (collegamenti)

Un **path** nel grafo G è una sequenza di nodi (x_1, x_2, \dots, x_n) t.c ognuna delle coppie $(x_1, x_2), (x_2, x_3), \dots, (x_{n-1}, x_n)$ sono archi in E

Il peso degli archi è il **costo c del collegamento** tra i nodi (es $c(w, z) = 5$) e il costo del **cammino** è la somma dei costi degli archi nel cammino

Il costo può rappresentare:

- Lunghezza fisica del collegamento
- Velocità del collegamento
- Costo monetario associato al collegamento

Un **algoritmo d'intradamento** determina il **cammino a costo minimo** nella rete

Algoritmo d'Intradamento con Vettore Distanza (Distance Vector - DV)

- **Distribuito**: ogni nodo riceve informazione dai vicini e opera su quelle
- **Asincrono**: non richiede che tutti i nodi operino al passo con gli altri

Si basa su:

1. **Equazione di Bellman-Ford**
2. **Concetto di vettore di distanza**

Equazione di Bellman-Ford

La formula di Bellman-Ford definisce:

$D_x(y) = \text{costo del percorso a costo minimo dal nodo } x \text{ al nodo } y$

Allora:

$$D_x(y) = \min_v \{c(x, v) + D_v(y)\}$$

dove \min_v riguarda tutti i nodi vicini di x

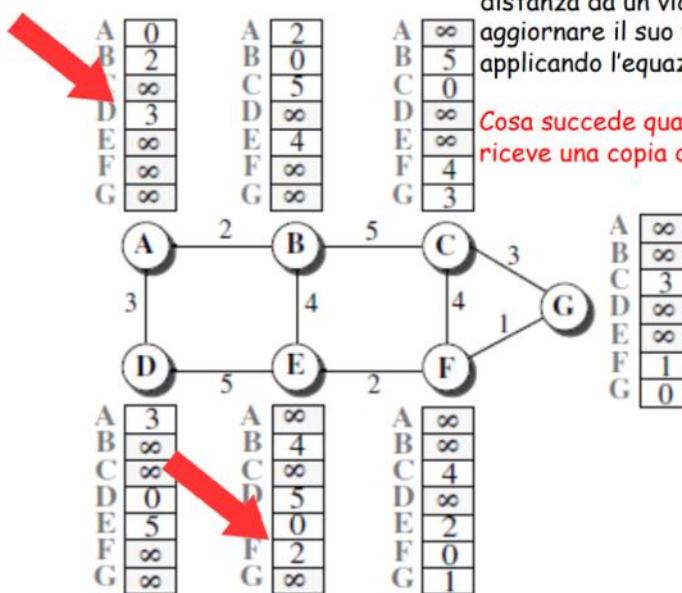
Vettore Distanza

Un **albero a costo minimo** è una combinazione di percorsi a costo minimo dalla radice dell'albero verso tutte le destinazioni.

Il **vettore di distanza** è un array monodimensionale che rappresenta l'albero, che non fornisce il percorso per giungere alla destinazione ma solo i costi minimi.

Creazione del vettore distanza:

- Ogni nodo della rete quando viene inizializzato crea un vettore distanza iniziale con le informazioni che il nodo riesce a **ottenere dai vicini** (nodi a cui è collegato)
- Invia messaggi di **hello** attraverso le sue interfacce (e lo stesso fanno i vicini) e scopre l'identità e le distanze dei vicini
- Il vettore iniziale rappresenta il vettore a costo minimo verso i vicini
- Dopo che ogni nodo crea il suo vettore, invia una copia ai vicini



Quando un nodo riceve un vettore distanza da un vicino provvede ad aggiornare il suo vettore distanza applicando l'equazione di Bellman-Ford

Nota:
X[]: l'intero vettore

Nuovo B	Vecchio B	A
A [2]	A [2]	A [0]
B [0]	B [0]	B [2]
C [5]	C [5]	C [∞]
D [5]	D [∞]	D [3]
E [4]	E [4]	E [∞]
F [∞]	F [∞]	F [∞]
G [∞]	G [∞]	G [∞]

$B[] = \min(B[], 2 + A[])$

a. Primo evento: B riceve una copia del vettore di A.
E se ora B riceve una copia di E?

Nuovo B	Vecchio B	E
A [2]	A [2]	A [∞]
B [0]	B [0]	B [4]
C [5]	C [5]	C [∞]
D [5]	D [5]	D [5]
E [4]	E [4]	E [0]
F [6]	F [∞]	F [2]
G [∞]	G [∞]	G [∞]

$B[] = \min(B[], 4 + E[])$

b. Secondo evento: B riceve una copia del vettore di E.

Ogni volta che un nodo x riceve un nuovo DV da un vicino, usa la formula B-F per aggiorna il proprio vettore come segue e poi manda il suo nuovo DV ai vicini:

b. Secondo evento: B riceve una copia del vettore di E.

Ogni volta che un nodo x riceve un nuovo DV da un vicino, usa la formula B-F per aggiornare il proprio vettore come segue e poi manda il suo nuovo DV ai vicini:

$$D_x(y) = \min_v \{c(x, v) + D_v(y)\} \quad \text{per ciascun nodo } y \text{ in } N$$

L'algoritmo con vettore distanza è:

- Iterativo, asincrono:** ogni iterazione locale è causata da:
 - Cambio del costo di uno dei collegamenti locali
 - Ricezione da un vicino del DV aggiornato
- Distribuito:** ogni nodo aggiorna i vicini solo quando il suo DV cambia
I vicini avvisano i vicini solo se necessario

Algoritmo**A ciascun nodo x:****Inizializzazione**

```
per tutte le destinazioni y in N:
  se y è un vicino
    Dx(y) = c(x, y)
  else
    Dx(y) = ∞
  per ciascun vicino w
    invia il vettore distanza Dx = [Dx(y) : y in N] a w
```

Attende (un messaggio del cambio del costo da parte del suo vicino)

Effettua il calcolo

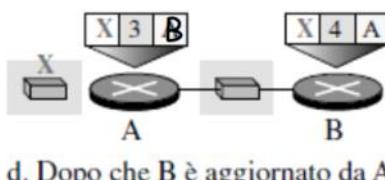
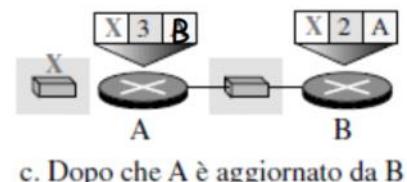
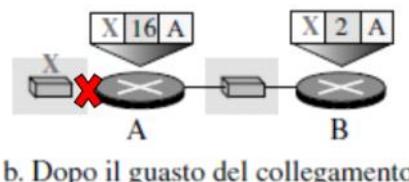
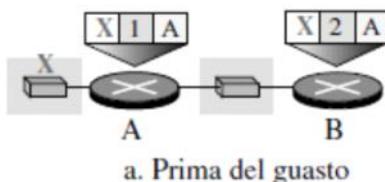
Se il DV cambia,
lo **notifica** ai suoi vicini.

Ciclo di ciascun nodo

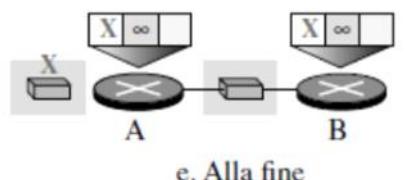
Algoritmo con Vettore Distanza: Modifica dei Costi e Problema**Modifica dei costi:**

- Un nodo rileva un cambiamento nel costo dei collegamenti
- Aggiorna il proprio DV
- Se si verifica un cambiamento nel costo, trasmette ai suoi vicini il nuovo DV

Potrebbe però verificarsi un problema di conteggio all'infinito

Esempio: si guasta il collegamento tra A e X

I pacchetti rimbalzano tra A e B creando un ciclo a due nodi. Il sistema diventa stabile dopo molti aggiornamenti

**Soluzioni****Split Horizon**

- Invece di inviare la tabella attraverso ogni interfaccia, ciascun nodo invia solo una parte della tabella tramite le interfacce
- Se il nodo B ritiene che il miglior percorso per arrivare a X passa per A, NON deve fornire questa informazione ad A (poiché già lo conosce)
- Nell'esempio B elimina la riga di X dalla tabella prima di inviarla ad A

Poisoned Reverse

- Si pone a ∞ il val del costo del percorso che passa attraverso il vicino a cui si sta inviando il vettore
- Nell'esempio B pone a ∞ il costo verso X quando invia il vettore ad A

RIP (Routing Information Protocol)È un protocollo a vettore distanza (DV) che come metrica di costo usa la **distanza misurata in hop** (max = 15 hop, 16 indica ∞). Ogni link ha lo stesso costo

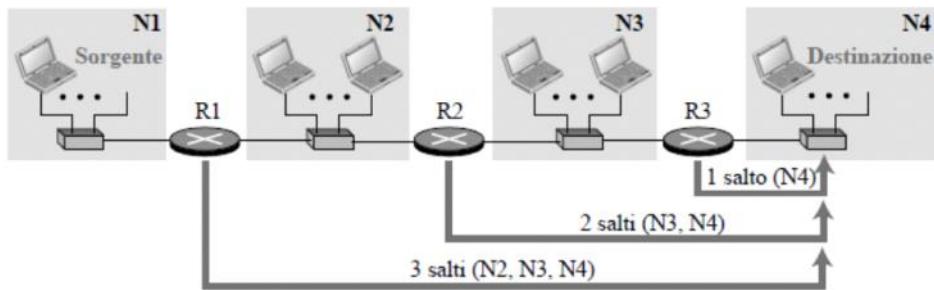


Tabella d'inoltro per R1

Rete di destinazione	Prossimo router	Costo (in hop)
N1	—	1
N2	—	1
N3	R2	2
N4	R2	3

Tabella d'inoltro per R2

Rete di destinazione	Prossimo router	Costo (in hop)
N1	R1	2
N2	—	1
N3	—	1
N4	R3	2

Tabella d'inoltro per R3

Rete di destinazione	Prossimo router	Costo (in hop)
N1	R2	3
N2	R2	2
N3	—	1
N4	—	1

L'informazione nella tabella di routing è sufficiente per raggiungere la destinazione

Con RIP i router inviano l'intera tabella di routing periodicamente agli altri router sulle reti e i dispositivi che raggiunge. Se il router R1 riceve un messaggio che dice che l'altro router R2 può raggiungere X a costo N, allora R1 sa che può raggiungere X con N+1 salti inviando il messaggio tramite R2

Messaggi RIP

RIP si basa su una coppia di processi client-server e sul loro scambio di messaggi

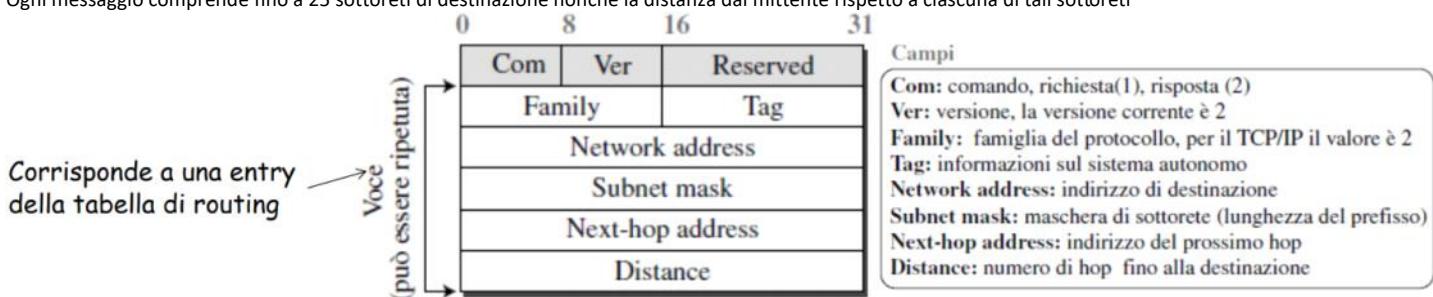
Rip Request:

- Quando un nuovo router viene inserito nella rete invia una RIP Request per ricevere informazioni di routing
- Fini diagnostici (richiedere una voce specifica)

Rip Response (o advertisements):

- In risposta a una Request (solicited response)
- Periodicamente ogni 30 secondi (unsolicited response)

Ogni messaggio comprende fino a 25 sottoreti di destinazione nonché la distanza dal mittente rispetto a ciascuna di tali sottoreti



Timer RIP

- Timer periodico: controlla invio messaggi di aggiornamento (ogni 25/35 secondi)
- Timer di scadenza: Regola la validità dei percorsi (180 secondi)
 Se entro lo scadere del timer non si riceve aggiornamento, il percorso viene considerato scaduto e il suo costo impostato a 16
- Timer per garbage collection: Elimina percorsi dalla tabella (120 secondi)
 Quando le informazioni non sono più valide, il router continua ad annunciare il percorso con costo pari a 16 (∞) e allo scadere del timer rimuove il percorso

RIP: Guasto sul Collegamento e Recupero

Se un router non riceve notizie dal suo vicino per 180 sec → il nodo adiacente viene considerato spento/guasto

- RIP modifica la tabella d'instradamento locale
- Propaga l'informazione ai router vicini
- I vicini inviano nuovi messaggi (se la loro tabella è cambiata)
- L'informazione che il collegamento è fallito si propaga sulla rete
- L'utilizzo del **poisoned reverse** evita i loop (16 hop = ∞)

Caratteristiche RIP

- Split horizon con Poisoned reverse:** Evita cicli mettendo a 16 il costo della rotta che passa dal vicino a cui si manda la risposta
- Triggered updates:** quando cambia un percorso si inviano immediatamente le info ai vicini senza aspettare il timeout, riducendo il tempo di convergenza
- Hold-down:** Alla info di un percorso non più valido, inizia un timer e ogni risposta che arriva da quella rotta entro il timeout vengono ignorati

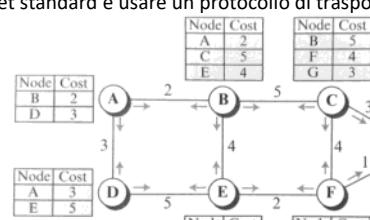
Implementazione

- Implementato come app sopra UDP con porta 520
- Un processo chiamato **routed** (route daemon) usa RIP (mantiene le info d'instradamento e scambia messaggi con i routers dei router vicini)
- Siccome è a liv. app, può inviare/ricevere messaggi su una socket standard e usare un protocollo di trasporto standard

Link State

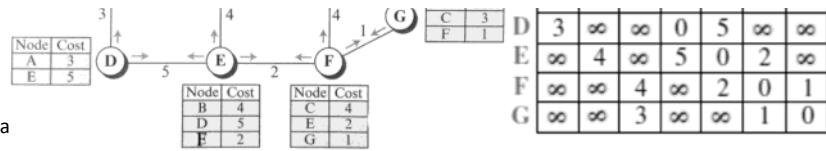
- Lo stato di un link indica il costo del link
- Se il costo è ∞ allora il link non esiste/è interrotto
- Ogni nodo deve conoscere i costi di tutti i link della rete
- Il **link state database** mantiene la mappa completa della rete

Link-State Database (LSDB)



A	B	C	D	E	F	G
0	2	∞	3	∞	∞	∞
2	0	5	∞	4	∞	∞
∞	5	0	∞	∞	4	3
3	∞	∞	0	5	∞	∞
∞	4	∞	5	0	2	∞
∞	∞	4	∞	2	0	1

- Ogni nodo deve conoscere i costi di tutti i link della rete
- Il link state database mantiene la mappa completa della rete



Link-State Database (LSDB)

L'LSDB è una matrice unica per rete e ogni nodo ne possiede una copia

Un router costruisce un LSDB:

- Conoscendo i suoi router vicini (con messaggi di hello) e il costo dei link verso di loro (dalla risposta degli hello)
- La lista (vicino, costo) si chiama **LS packet (LSP)**
- Ogni nodo esegue un **flooding dei LSP**: Invia ai vicini il suo LSP e quando riceve un nuovo LSP da un vicino, lo inoltra ai suoi vicini (tranne da cui lo ha ricevuto)

Algoritmo d'Instradamento a Link State

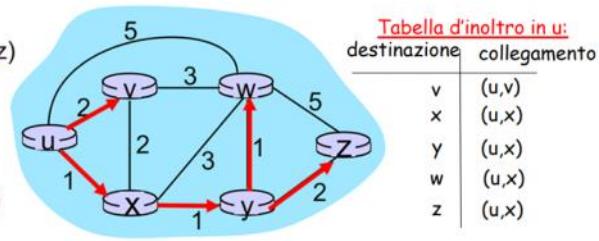
Per costruire la **tavella di routing** ogni nodo esegue l'**algoritmo di Dijkstra** usando la LSDB mettendo se stesso come radice dell'albero.

L'algoritmo di Dijkstra permette di trovare i cammini minimi per arrivare dalla radice ad un nodo dell'albero, creando una **tavella d'inoltro** per quel nodo

Definiamo la notazione:

- N : insieme dei nodi di rete
- $c(x, y)$: costo del link tra x e y
- $D(v)$: costo del cammino minimo dalla radice a v per l'iterazione corrente
- $p(v)$: predecessore di v nel cammino
- N' : sottoinsieme di nodi per cui il cammino a costo minimo è noto

Step	N'	V	W	X	Y	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					



8 Ciclo

- Inizializzazione:**
- $N' = \{r\}$ /* r è il nodo che esegue l'algoritmo (radice dell'albero) */
 - per tutti i nodi n
 - se n è adiacente a r
 - allora $D(n) = c(r,n)$
 - altrimenti $D(n) = \infty$
- determina un n non in N' tale che $D(n)$ sia minimo
- aggiungi n a N'
- per ogni nodo a adiacente a n e non in N' aggiorna $D(a)$:
- $D(a) = \min(D(a), D(n) + c(n,a))$
- /* il nuovo costo verso a è il vecchio costo verso a oppure il costo del cammino minimo noto verso a più il costo da n a a */
- Finché $N' = N$**

OSPF (Open Shortest Path First)

È un protocollo basato su **Link State** che usa il **flooding** di informazioni di link state e Dijkstra per determinare il costo minimo

Flooding:

- Con OSPF, ad ogni cambiamento **nello stato di un link**, il router manda info d'instradamento agli altri router
- Invia ogni 30 min messaggi OSPF all'intero sistema, usando il flooding

Messaggi OSPF

I messaggi OSPF vengono trasportati in datagrammi IP col protocollo 89 nel campo IP protocol

- Hello**: usato per annunciare la propria esistenza ai router vicini
- Database description**: risposta ad hello (il router nuovo ottiene il LSDB)
- Link-state request**: Usato per richiedere specifiche info su un link
- Link-state update**: usato per la costruzione del LSDB
- Link-state ack**: riscontro ai link-state update (per affidabilità)

Confronto tra LS e DV

Complessità messaggi:

- LS**: con n nodi, E collegamenti richiede invio di $O(n^2)$ messaggi (ogni nodo deve conoscere il costo degli E link)
- DV**: richiede scambi tra nodi adiacenti (costo convergenza varia)

Velocità convergenza:

- LS**: ha costo $O(n^2)$ (procede su $\frac{n(n+1)}{2}$ nodi)
- DV**: può convergere lentamente e presentare problemi di cicli/conteggio infinito

Robustezza: OSPF è **più robusto** di RIP. Cosa avviene se un router funziona male:

- LS**: un router può comunicare via broadcast un costo sbagliato per uno suo link connesso (non per altri) e i nodi calcolano solo le proprie tabelle
- DV**: un nodo può comunicare percorsi errati a tutte le destinazioni e la tabella di ogni nodo può essere usata dagli altri

Routing su Internet

Nell'Internet esistono 200 milioni di destinazioni quindi archiviare le info d'instradamento richiederebbe molta memoria, il traffico degli LS riempirebbe la banda e i DV non convergerebbero mai.

Ogni ISP è un **sistema autonomo (AS, autonomous system)** che esegue lo stesso protocollo di routing ai suoi router in base alle esigenze.

- Protocollo di routing interno al AS (**intra-AS**) o **intra-dominio**, o **interior gateway protocol (IGP)**
- Router di differenti AS possono eseguire protocolli d'instradamento intra-AS diversi

Quindi dobbiamo avere un solo protocollo all'esterno dell'AS (**inter-AS**) o **inter-dominio** o **exterior gateway protocol (EGP)** per il routing tra i vari AS

Router Gateway:

- Connettono gli AS tra loro
- Hanno il compito d'inoltrare pacchetti a destinazioni esterne

Ogni AS ha un **numero identificativo** di 16 bit (ASN) e gli AS sono classificati in base a come sono connessi con altri AS:

- AS stub**: una sola connessione ad un altro AS e genera/riceve traffico ma non transita da esso (es. grandi aziende)
- AS multihomed**: più connessioni ad altri AS ma non consente transito di traffico (azienda che usa più network provider ma non fornisce connettività ad altri AS)
- AS di transito**: più connessioni ad altri AS e consente traffico (network provider e dorsali)

Routing intra-dominio: RIP e OSPF

Routing inter-dominio: BGP

BGP (Border Gateway Protocol)

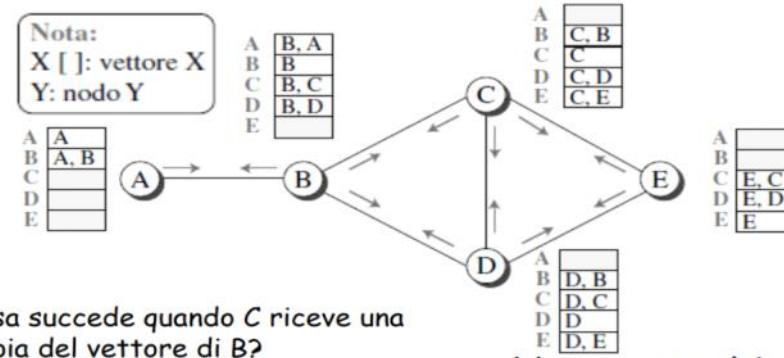
BGP viene usato per determinare percorsi per le coppie origine-destinazione che interessano più AS, permettendo a ciascuna sottorete di connettersi a vicenda

- Coppie di router (chiamati **peer BGP**) scambiano info di instradamento su TCP con la porta 179, e i messaggi BGP inviati sono detti **sessione BGP**
- È un protocollo **path vector (Distance Vector con percorsi)** e mette a disposizione per ogni AS un modo per:
 - Ottenere info per raggiungere sottoreti di AS confinanti
 - Propagare info di raggiungibilità ai router interni di un AS
 - Determinare percorsi "buoni" verso le sottoreti sulla base delle info di raggiungibilità e politiche dell'AS

Path Vector Routing

Ci sono casi dove il costo minimo non è l'obiettivo primario, tipo se un mittente non vuole che i suoi pacchetti passino per determinati router. Con **path-vector routing (routing a vettore di percorso)** la sorgente può controllare il percorso per evitare alcuni nodi e minimizzare il num. di hop.

In modo simile al DV, però inviando percorsi invece che destinazioni, ogni nodo invia un PV al vicino, aggiorna il suo PV con la politica di costo minimo



Cosa succede quando C riceve una Copia del vettore di B?

Nessun cambiamento

Nuovo C	Vecchio C	B
A [C, B, A]	A [B, A]	A [B, A]
B [C, B]	B [B]	B [B]
C [C]	C [C]	C [B, C]
D [C, D]	D [C, D]	D [B, D]
E [C, E]	E [C, E]	E [D, E]

C[] = migliore (C[], C + B[])

Nuovo C	Vecchio C	D
A [C, B, A]	A [C, B, A]	A [D, B]
B [C, B]	B [C, B]	B [D, C]
C [C]	C [C]	C [D]
D [C, D]	D [C, D]	D [D]
E [C, E]	E [C, E]	E [D, E]

C[] = migliore (C[], C + D[])

Evento 1: C riceve una copia del vettore di B

Evento 2: C riceve una copia del vettore di D

(ho saltato l'algoritmo scusate)

eBGP e iBGP

Per instradare correttamente i pacchetti, ogni **router di confine (border router)** dell'AS deve usare una variante del BGP, il **BGP esterno (external BGP o eBGP)**.

Tutti i **router** (anche border router) invece devono usare un'altra variante, il **BGP interno (internal BGP o iBGP)**

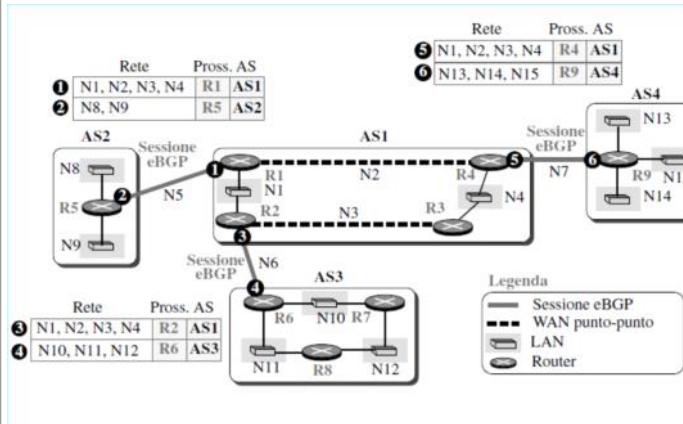
Quindi i border router dovranno eseguire tre protocolli di routing (intra-dominio, eBGP e iBGP), mentre gli altri solo intra-dominio e iBGP

eBGP

Due **border router** di due AS diversi formano una coppia di **peer BGP** e scambiano messaggi per indicare ad alcuni router come instradare i pacchetti destinati ad alcune reti, però non sono info complete:

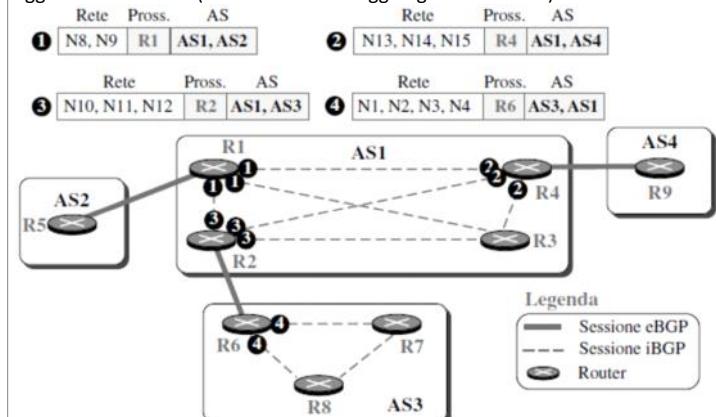
- I border router sanno instradare solo ad AS vicini
- Nessun border router sa come instradare a reti di altre AS

Soluzione: iBGP



iBGP

Crea una sessione tra ogni possibile coppia di router nell'AS, anche se i nodi non devono inviare messaggi (es R3, R7, R8 non sono collegati a router esterni) ma tutti ricevono. L'aggiornamento non termina dopo il primo scambio di messaggi: Es: R1 riceve l'aggiornamento di R2, combina le info di raggiungibilità di AS1 con quelle già conosciute relativamente a AS1 e invia un nuovo messaggio di aggiornamento a R5 (che ora sa come raggiungere AS1 e AS3)



Il processo di aggiornamento continua finché non ci sono più aggiornamenti e le info ottenute da eBGP e iBGP sono **combinate** per creare la **tavola dei percorsi**

Tabelle di Percorso

Le tabelle di percorso ottenute da BGP sono inserite nelle **tabelle di routing intra-dominio** (generate da RIP/OSPF).

- Nel caso di **AS stub**, l'unico router di confine aggiunge una regola di default alla fine della tab. di routing e definisce come prossimo router l'altro border router con cui si connette tramite eBGP
- Nel caso di **AS di transito**, il contenuto della tabella di percorso è inserito nella tab. di routing ma bisogna impostare il costo per raggiungere il primo AS

Rete	Pross.	Percorso
N8, N9	R5	AS1, AS2
N10, N11, N12	R2	AS1, AS3
N13, N14, N15	R4	AS1, AS4

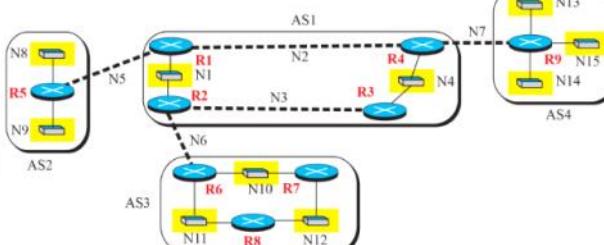
Tabella di percorso per R1

Rete	Pross.	Percorso
N1, N2, N3, N4	R2	AS3, AS1
N8, N9	R2	AS3, AS1, AS2
N13, N14, N15	R2	AS3, AS1, AS4

Tabella di percorso per R6

Rete	Pross.	Percorso
N8, N9	R1	AS1, AS2
N10, N11, N12	R6	AS1, AS3
N13, N14, N15	R1	AS1, AS4

Tabella di percorso per R2



Rete	Pross.	Percorso
N1, N2, N3, N4	R1	AS2, AS1
N10, N11, N12	R1	AS2, AS1, AS3
N13, N14, N15	R1	AS2, AS1, AS4

Tabella di percorso per R5

Rete	Pross.	Percorso
N8, N9	R2	AS1, AS2
N10, N11, N12	R2	AS1, AS3
N13, N14, N15	R4	AS1, AS4

Tabella di percorso per R3

Rete	Pross.	Percorso
N8, N9	R1	AS1, AS2
N10, N11, N12	R1	AS1, AS3
N13, N14, N15	R9	AS1, AS4

Tabella di percorso per R4

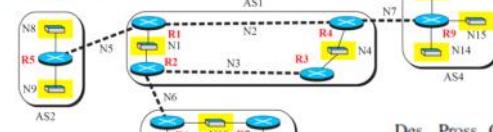
Tabella di Inoltro DOPO l'aggiunta delle Informazioni BGP

Tab. Inoltro AS Stub

Des.	Pross.	Costo
N8	—	1
N9	—	1
0	R1	1

Nel caso di stub, l'unico router di confine dell'area aggiunge una regola di default alla fine della sua tabella di routing e definisce come prossimo router quello che si trova dall'altro lato della connessione eBGP

Tabella per R5



Des.	Pross.	Costo
N13	—	1
N14	—	1
N15	—	1
0	R4	1

Tabella per R9

Des.	Pross.	Costo
N10	—	1
N11	—	1
N12	R7	2
0	R2	1

Tabella per R6

Des.	Pross.	Costo
N10	R6	2
N11	—	1
N12	—	1
0	R6	2

Tabella per R8

Des.	Pross.	Costo
N10	—	1
N11	R6	2
N12	—	1
0	R6	2

Tabella per R7

Tab. Inoltro AS di Transito

Des.	Pross.	Costo
N1	—	1
N4	R4	2
N8	R5	1
N9	R5	1
N10	R2	2
N11	R2	2
N12	R2	2
N13	R4	2
N14	R4	2
N15	R4	2

Des.	Pross.	Costo
N1	—	1
N4	R3	2
N8	R1	2
N9	R2	3
N10	R2	2
N11	R2	2
N12	R6	1
N13	R3	3
N14	R3	3
N15	R3	3

Tabella per R1

Des.	Pross.	Costo
N1	R2	2
N4	—	1
N8	R1	2
N9	R1	2
N10	R3	3
N11	R2	2
N12	R3	3
N13	R9	1
N14	R4	2
N15	R4	2

Des.	Pross.	Costo
N1	R1	2
N4	—	1
N8	R1	2
N9	R1	2
N10	R3	3
N11	R2	2
N12	R3	3
N13	R9	1
N14	R9	1
N15	R9	1

Tabella per R4

Attributi Percorso e Rotte BGP

Quando un router annuncia una **rotta per un prefisso** per una sessione BGP, include anche un certo num. di **attributi BGP** (prefisso + attributi = "rotta")

Due attributi importanti sono:

- **AS-PATH:** serve per selezionare i percorsi. Elenca gli ID degli AS attraverso i quali è passato l'annuncio del prefisso (quindi gli hop intermedi)
- **NEXT-HOP:** IP dell'interfaccia su cui viene inviato il pacchetto (un router ha più IP, uno per interfaccia)

Quando un router gateway riceve un **annuncio di rotta**, usa le proprie **politiche d'importazione** per decidere se accettare o filtrare la rotta

- L'AS può non voler inviare il traffico su un AS nel AS-PATH
- Router conosce già la rotta migliore

Selezione Percorsi BGP

Un router può ricavare **più di una rotta** verso una destinazione quindi deve sceglierne una. **Regole di eliminazione:**

- 1) Alle rotte si assegna come attributo **val. di preferenza locale** e si selezionano i valori più alti (riflette la politica scelta dall'amministratore)
- 2) Si seleziona la rotta con **AS-PATH più breve**
- 3) Si seleziona la rotta il cui router di **NEXT-HOP ha costo minore**: hot-potato routing
- 4) Se rimane ancora più di una rotta, si basa sugli **ID BGP**

Messaggi BGP

I messaggi BGP sono scambiati tramite TCP:

- **OPEN:** apre connessione TCP e autentica il mittente
- **UPDATE:** aggiorna il percorso
- **KEEPALIVE:** mantiene la connessione attiva in mancanza di UPDATE
- **NOTIFICATION:** riporta gli errori del precedente messaggio; usato anche per chiudere la connessione

Inter-AS vs Intra-AS

Politiche:

- **Inter-AS:** il controllo amministrativo controlla l'instradamento del traffico in uscita/arrivo
- **Intra-AS:** unico controllo amministrativo, quindi politiche meno importanti per l'instradamento interno

Prestazioni:

- **Inter-AS:** le politiche possono prevalere sulle prestazioni
- **Intra-AS:** orientato alle prestazioni

Routing Multicast

venerdì 13 giugno 2025 11:47

Broadcast

Rispetto all'**UNICAST** che permette la comunicazione da **UNA sorgente e UNA destinazione** (1 a 1), il **BROADCAST** permette di inviare un pacchetto da **UNA sorgente a TUTTI i nodi di rete** (1 a N). Si può eseguire un broadcast **non controllato o controllato** tramite controlli di visita sulla rete prima del broadcast

Broadcast: Uncontrolled flooding

Quando un nodo riceve un pacchetto broadcast lo invia a tutti i nodi vicini (tranne da chi lo ha ricevuto). Se ci sono cicli, copie del pacchetto cicleranno all'infinito

Broadcast: Controlled flooding

Sequence Number	RPF (Reverse Path Forwarding)	Spanning Tree
Tiene una lista di (IP, #seq) di pacchetti già ricevuti e se arriva uno già inoltrato lo scarta	Forwarda il pacchetto solo se arriva dal link che è sul suo shortest path verso la sorgente. NON elimina la trasmissione di pacchetti ridondanti	Per fare in modo che un nodo riceve un solo pacchetto (senza dovers scartare altri) si costruisce prima lo spanning tree : <ul style="list-style-type: none">• Si prende un nodo come centro• Ogni nodo invia un messaggio di join verso il centro finché arrivano: 1) a un nodo già nell'albero 2) alla radice I pacchetti si inviano solo ai link dell'albero

Multicast

Comunicazione tra **Una sorgente e un GRUPPO di destinazioni** (1 a k, con k≤N) (es: streaming ad un gruppo di utenti, aggiornamento SW su gruppo di macchine).

Rispetto all'**unicast multiplo** dove la sorgente invia k pacchetti che vengono instradati (inefficiente), i pacchetti multicast vengono direttamente **duplicati dal router**. Siccome l'IP destinazione è uno solo si usa un **unico indirizzo per tutto il gruppo (indirizzo multicast)**.

In IPv4 gli indirizzi riservati per il multicast sono:

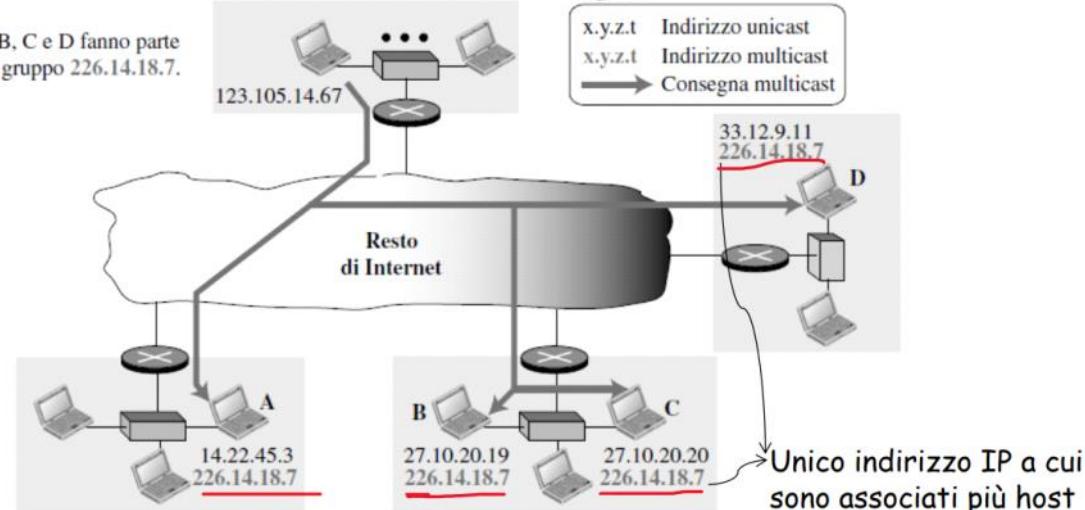
- 224.0.0.0/4
- 1110-ID del gruppo-- (da 224.0.0.0 a 239.255.255.255)
- Num. gruppi: 2^{28}

1110 | group identifier

First byte: 224 to 239

Legenda

x.y.z.t Indirizzo unicast
x.y.z.t Indirizzo multicast
→ Consegnare multicast



I router devono sapere quali host sono associati a un gruppo multicast !!!

4-15

Gruppi Multicast

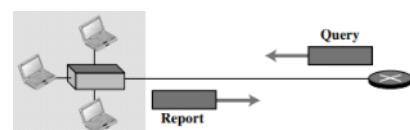
L'IP multicast è separato rispetto all'IP primario associato alla rete. Un router deve sapere quali gruppi sono tra le sue interfacce e inviare l'informazione a altri router. Servono quindi due protocolli: 1) per raccogliere info sui gruppi. 2) per diffondere le info di appartenenza

IGMP (Internet Group Management Protocol)

Lavora tra host e router e permette agli host di informare i router che vogliono aderire ad un gruppo multicast.

I messaggi sono incapsulati in datagrammi IP, con IP protocollo 2 mandati con TTL a 1:

- **Membership query**: router → host (**periodicamente**), determina a quali gruppi hanno aderito gli host su ogni interfaccia
- **Membership report**: host → router (**momento d'adesione**), informa il router su un'adesione
- **Leave group**: host → router (**opzionale**), quando si lascia un gruppo



Tiene una lista per ciascuna sottorete dei gruppi multicast (con almeno un elemento) con un **timer per membership** che deve essere **aggiornata da report o leave** prima che scada il timer.

Problema del Routing Multicast

Siccome solo i router collegati a host del gruppo multicast dovranno ricevere traffico multicast, serve un protocollo per coordinare i router multicast in Internet.

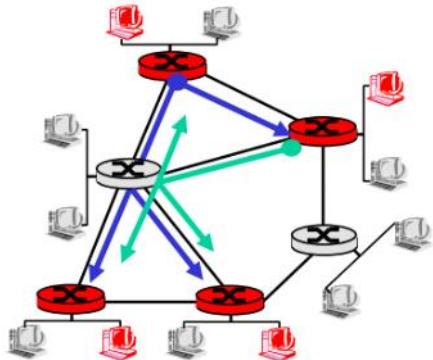
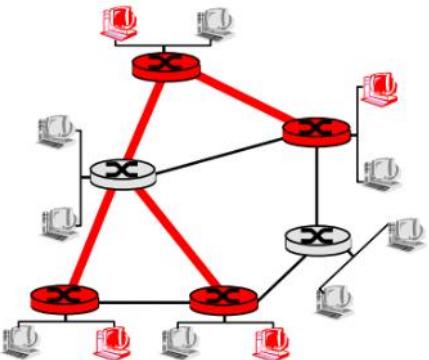
Obiettivo: trovare un albero che collega tutti i router connessi ad host che appartengono al gruppo multicast

Albero Condiviso dal Gruppo

Albero d'instradamento condiviso da tutto il gruppo multicast, dove un router agisce da **rappresentante del gruppo**. Il mittente invia il traffico in unicast al centro che provvederà a inviarlo al gruppo

Albero Basato sull'Origine

Si crea un albero per ogni origine nel gruppo multicast (tanti alberi quanti i mittenti). Per la costruzione si usa un algoritmo basato su reverse path forwarding con pruning (potatura)



Intradomäne Multicast in Internet

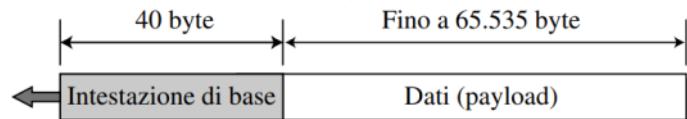
- **Intra-domäne multicast** (interno a AS)
 - DVMRP: distance-vector multicast routing protocol
 - MOSPF: multicast open shortest path first
 - PIM: protocol indipendent multicast
- **Inter-domäne multicast** (tra AS)
 - MBGP: multicast border gateway protocol

IPv6

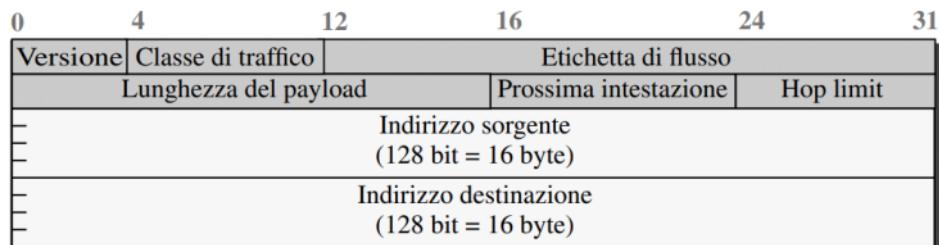
venerdì 13 giugno 2025 17:00

IPv6 è nato per aumentare lo spazio di indirizzi rispetto a ipv4, ridisegnare il formato dei datagrammi e rivedere protocolli ausiliari come ICMP

- Indirizzi IP lunghi 128 bit
- Nuovo formato header IP
- Nuove opzioni
- Possibilità di estensione
- Opzioni di sicurezza
- Maggiore efficienza: no frammentazione nodi intermedi, etichette di flusso per traffico audio/video



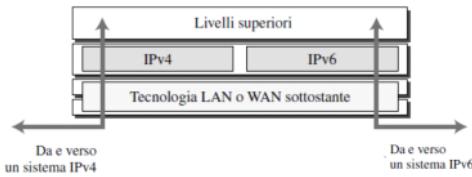
a. Datagramma IPv6



b. Intestazione di base

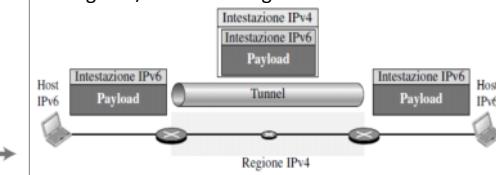
Dual Stack

Durante la transizione da IPv4 a IPv6 gli host devono avere una **doppia pila di protocolli**, e per determinare quale usare per inviare un pacchetto, l'host sorgente interroga il DNS e usa il protocollo relativo all'indirizzo ritornato



Tunneling

Se due host IPv6 devono comunicare attraverso una rete IPv4, si incapsula il datagramma IPv6 nel payload di un datagramma IPv4 e si inseriscono come IP sorgente/destinazione gli estremi del tunnel



Traduzione dell'intestazione

Quando un mittente IPv6 comunica con un destinatario IPv4, il datagramma viene tradotto prima che arrivi a destinazione



Livello Collegamento

sabato 14 giugno 2025 18:25

Le unità di dati scambiate a liv. di **collegamento** (o **link**) sono chiamate **frame**. Il liv. link è implementato in un **adattatore (NIC, network interface card)**.

- **Lato mittente:**
 - Incapsula un datagramma in un frame
 - Imposta il bit rilevazione degli errori, trasferimento dati affidabile, controllo di flusso, ecc
- **Lato ricevente:**
 - Estraie i datagrammi e li passa al nodo ricevente
 - Individua gli errori, trasferimento dati affidabile, controllo di flusso, ecc

I **protocolli a liv. di collegamento** si occupano del trasporto di **datagrammi** lungo un singolo canale di comunicazione, e si possono usare diversi protocolli per lo stesso datagramma su collegamenti diversi.

Servizi del Liv. di Collegamento

- **Framing:**
 - I protocolli incapsulano i datagrammi del liv. di rete nel frame a liv. di link per separare i vari messaggi durante la trasmissione
 - Per identificare origine e destinatario si usano indirizzi **MAC**
- **Consegna affidabile:**
 - Basata su **ACK** come nel trasporto
 - Spesso usata nei collegamenti soggetti a **elevati tassi di errori** (es: wireless)
 - Considerata non necessaria nei collegamenti con basso num. di errori sui bit (fibra ottica, coassiale e doppino intrecciato)
- **Controllo di flusso:** Evita che il nodo trasmittente **saturi quello ricevente**
- **Rilevazione degli errori:**
 - Gli errori sono causati dalle **interferenze**
 - Il nodo ricevente individua la presenza di errori tramite all'uso di **bit di controllo di errore** nel frame inseriti dal nodo trasmittente
- **Correzione degli errori:** Il nodo ricevente determina anche il punto in cui si è verificato l'errore e lo corregge

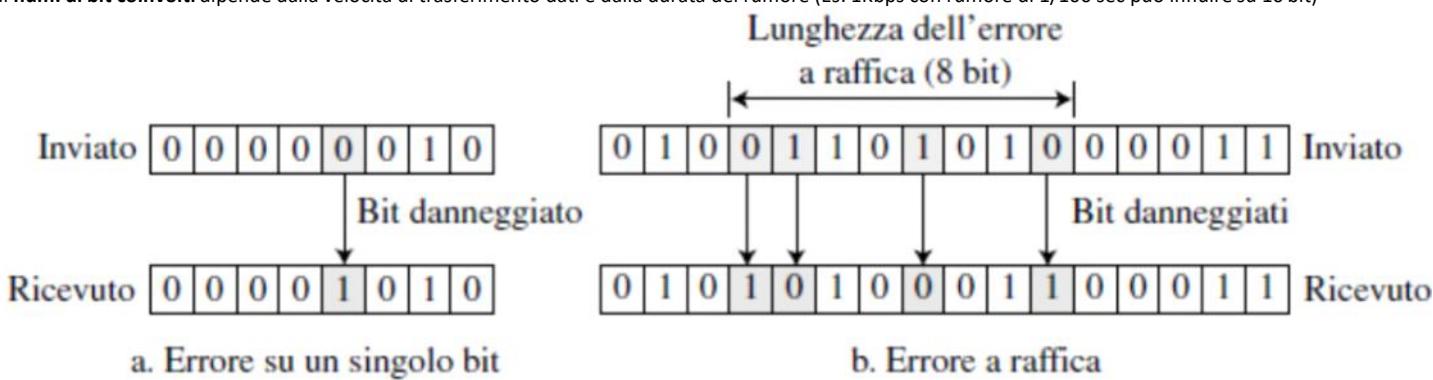
Errori

Errori su un Singolo Bit o a Burst

Gli errori sono dovuti a interferenze che possono cambiare la forma del segnale.

La probabilità che avvenga un errore di tipo **burst (raffica)** è **più elevata** rispetto a quella di un singolo bit, siccome la durata dell'interferenza (detta **rumore**) è più lunga rispetto a quella di un singolo bit.

Il **num. di bit coinvolti** dipende dalla velocità di trasferimento dati e dalla durata del rumore (Es: 1Kbps con rumore di 1/100 sec può influire su 10 bit)



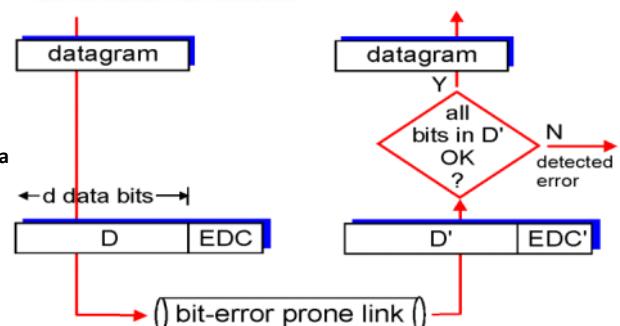
Tecniche di Rilevazione degli Errori

EDC = Error Detection and Correction

D = Dati che devono essere protetti da errori e ai quali vengono aggiunti dei bit **EDC**

La rilevazione non è attendibile al 100%

- Possono starci errori che non vengono rilevati
- Per ridurre le probabilità di questi eventi, le tecniche prevedono un'elevata **ridondanza**



Controllo di Parità

Il **bit aggiuntivo (di parità)** viene selezionato in modo da rendere **pari** il **num. tot. di 1** nella **codeword**

Unico bit di parità:

Si è verificato almeno un errore in un bit

← d data bits → parity bit

0111000110101011 | 0

Parità bidimensionale:

Individua e corregge il bit alterato

$d_{1,1}$...	$d_{1,j}$	row parity $d_{1, j+1}$	no errors
$d_{2,1}$...	$d_{2,j}$		
...
$d_{i,1}$...	$d_{i,j}$		
$d_{i+1,1}$...	$d_{i+1,j}$	$d_{i+1,j+1}$	

column parity ↓

101011
111100
011101
001010

no errors

101011
101100
011101
001010

parity error

correctable single bit error

Protocolli di Accesso Multiplo

giovedì 19 giugno 2025 18:44

Data-Link Control (DLC)	Media Access Control (MAC)
Si occupa di tutte le questioni comuni ai collegamenti punto-punto e broadcast (Framing, controllo flusso/errori, rilevamento/correzione errori) Si occupa delle procedure per la comunicazione tra due nodi adiacenti, indipendentemente dal fatto che il collegamento sia dedicato o broadcast	Si occupa degli aspetti specifici dei canali broadcast (controllo dell'accesso al mezzo condiviso)

I nodi di una rete sono fisicamente collegati tramite un **mezzo trasmissivo**, che è possibile utilizzare:

- **Interamente → collegamento punto-punto:** dedicato a due soli dispositivi (tra Ethernet e host, usato per connessioni telefoniche)
 - Viene usato il **Point-to-Point Protocol (PPP)** del DLC
- **Solo una parte → collegamento broadcast:** il collegamento è condiviso tra varie coppie di dispositivi (Ethernet tradizionale, Wireless LAN)
 - Necessità di un protocollo (**MAC**) per la gestione del canale condiviso

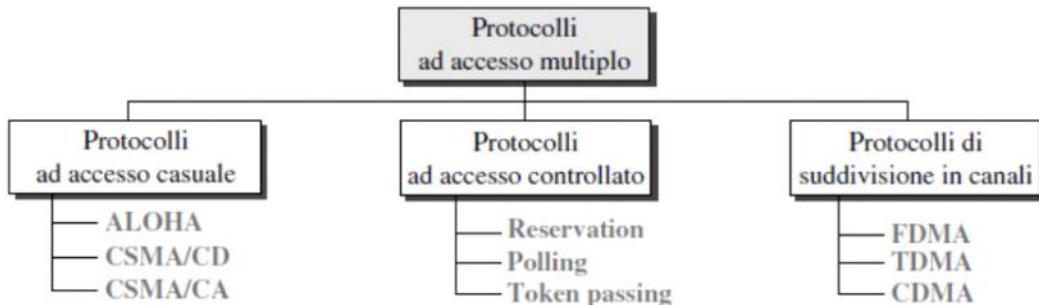
Siccome numerosi nodi possono comunicare su un canale broadcast, se più frame arrivano contemporaneamente allo stesso nodo si genera una **collisione**
I protocolli di accesso multiplo servono per realizzare una condivisione ordinata, fissando le modalità con cui i nodi **regolano le loro trasmissioni** sul canale condiviso

Canale broadcast con velocità di R bit al sec:

1. Quando un nodo deve inviare dati, questo dispone di un tasso trasmissivo pari a R bps. (1 Nodo → Tasso trasmissivo = R bps)
2. Quando M nodi devono inviare dati, questi dispongono di un tasso trasmissivo pari a R/M bps. (M nodi → Tasso trasmissivo = R/M bps)
3. Il protocollo è **decentralizzato: no nodi master, no sincronizzazione dei clock**

I **protocolli di accesso multiplo**, si possono dividere in tre categorie:

- **Protocolli a suddivisione del canale (channel partitioning)**
 - Suddivide un canale in "parti più piccole" (slot di tempo, frequenza, codice) e li alloca presso un nodo per uso esclusivo
 - **No collisioni**
 - **Efficiente con carichi elevati**
 - **Inefficiente con carichi non elevati**
- **Protocolli ad accesso casuale (random access)**
 - I canali **non vengono divisi** e si può verificare una collisione
 - I nodi coinvolti ritrasmettono ripetutamente i pacchetti
 - **Efficiente con carichi non elevati** (un solo nodo può usare interamente il canale)
 - **Eccesso di collisioni con carichi elevati**
- **Protocolli a rotazione ("taking-turn")**
 - Ciascun nodo ha il suo turno di trasmissione, ma i nodi che hanno molto da trasmettere possono avere turni più lunghi
 - Compromesso tra suddivisione del canale e accesso casuale



Come si può risolvere il problema della contesa di un canale condiviso?

- **Suddivisione del canale** per: tempo, frequenza, codice
 - TDM, FDM
- **Suddivisione casuale (dinamica)**
 - Rilevamento della portante (facile in tecnologie cablate, difficile in wireless)
 - ALOHA, S-ALOHA, CSMA, CSMA/CD (usato in Ethernet), CSMA/CA (usato in 802.11)
- **A rotazione**
 - Polling con un nodo principale; a passaggio di testimone
 - Token Ring

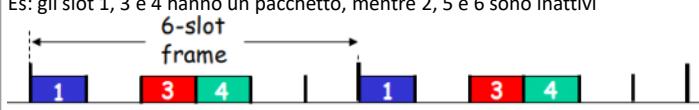
Protocolli a Suddivisione del Canale

TDMA (Accesso Multiplo a Divisione di Tempo)

Suddivide il canale in **intervalli di tempo** e ogni nodo ha un turno assegnato per accedere al canale (i turni non usati rimangono inattivi)

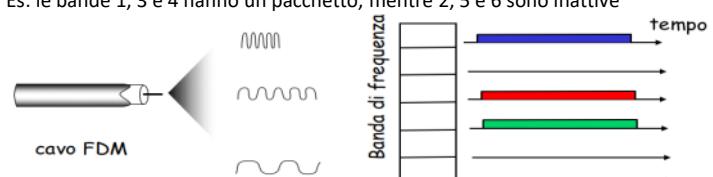
Tasso trasmissivo: R/N bps (R = velocità trasmissione, N = num. nodi). **Non flessibile** rispetto a variazione del num. di nodi

Es: gli slot 1, 3 e 4 hanno un pacchetto, mentre 2, 5 e 6 sono inattivi



FDMA (Accesso Multiplo a Divisione di Frequenza)

Suddivide il canale in **bande di frequenza** e ogni nodo ha una banda prefissata
Es: le bande 1, 3 e 4 hanno un pacchetto, mentre 2, 5 e 6 sono inattive



Protocolli ad Accesso Casuale

Ogni volta che un nodo vuole inviare dati, usa una procedura definita dal protocollo per decidere se spedire o meno. Nell' **accesso casuale**:

- Non c'è un tempo programmato nel quale la stazione deve trasmettere

- Nessuna regola specifica quale sarà la prossima stazione a trasmettere
- Invece le stazioni **competono** per accedere al mezzo trasmissivo (**contesa del canale**). Però se ci sono più nodi trasmittenti si causa una **collisione**, quindi il **protocollo ad accesso casuale** definisce:

- Come rilevare un'eventuale collisione
- Come ritrasmettere se si è verificata una collisione

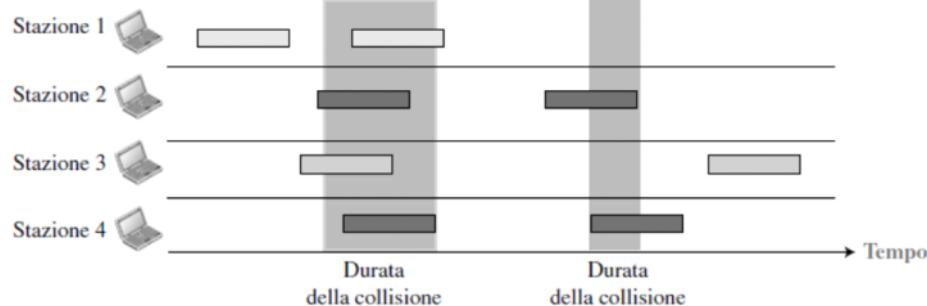
Esempi di protocolli ad accesso casuale: **ALOHA, Slotted ALOHA, CSMA, CSMA/CD, CSMA/CA**

ALOHA

Ogni stazione può inviare un frame quando vuole e il ricevente invia un **ACK** per notificare la ricezione del frame. Nel caso in cui non si riceve ACK entro un **timeout**, deve ritrasmettere.

Se due stazioni ritrasmettono contemporaneamente vi è una collisione, allora si attende un **tempo random (back-off)** prima di ritrasmettere per evitare altre collisioni.

Dopo un **num. max di tentativi K_{max}** , una stazione interrompe i tentativi e riprova più tardi.



Durata della collisione variabile, anche di un solo bit!

Timeout: Il periodo di timeout è uguale al **max ritardo di propagazione** di round-trip (andata e ritorno dell'ACK) tra le due stazioni più lontane ($2 \times T_p$)

- T_p (tempo di propagazione) = distanza mittente e destinatario / velocità di propagazione

Back-off esponenziale binario: Il tempo di back-off T_B è un val. scelto **casualmente** che dipende dal num. **K di trasmissioni fallite**

Backoff time = $R \cdot T_{fr}$ dove:

- $R \in [0, 2^k - 1]$
- $K = \# \text{tentativi}$
- $T_{fr} = \text{tempo x inviare un frame (lunghezza del pacchetto / velocità del canale)}$
- $K_{max} = 15$

Esempio:

Le stazioni in una rete wireless ALOHA sono a una distanza max di 600km. Supponendo che i segnali si propagano a 3×10^8 m/s, troviamo:

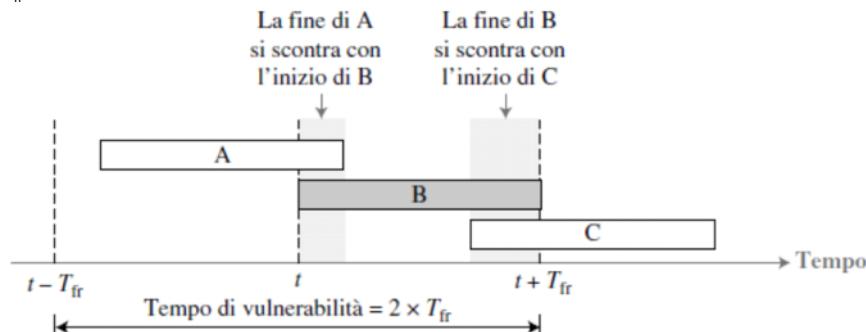
$$T_p = \frac{6 \times 10^5}{3 \times 10^8} = \frac{6}{3000} = 0.002 = 2ms$$

Per $K = 2$ l'intervallo R è $\{0, 1, 2, 3\}$. Quindi $T_B = R \cdot T_{fr}$ può essere 0, 2, 4 o 6 ms, sulla base del risultato della variabile casuale R (anche se in realtà T_p e T_{fr} sono due concetti diversi la prof li ha scelti con lo stesso val)

Nell'ALOHA puro c'è un **num. elevato di collisioni**.

Tempo di vulnerabilità: intervallo di tempo nel quale il frame è a rischio di collisione

- Il frame trasmesso al momento t si sovrappone con la trasmissione di un altro frame inviato in $[t-1, t+1]$ (dove 1 è la T_{fr})
- Tempo di vulnerabilità = $2T_{fr}$



Efficienza (o throughput)

L'**efficienza** è definita come la frazione di **slot vincenti** in presenza di un elevato num. N di nodi attivi, che hanno sempre un elevato num. di pacchetti da spedire

- Assumiamo che tutti i frame hanno la **stessa dimensione** e ogni nodo ha sempre un frame da trasmettere
- In ogni istante di tempo, p è la prob. che un nodo trasmetta un frame, $(1-p)$ che non trasmetta
- Supponendo che un nodo inizi a trasmettere al tempo t_0 , allora la trasmissione è andata a buon fine se nessun altro nodo ha iniziato una trasmissione nel tempo $[t_0-1, t_0]$. Tale probabilità è data da $(1-p)^{N-1}$
- Allo stesso modo nessun nodo deve iniziare a trasmettere nel tempo $[t_0, t_0+1]$ e la probabilità di questo evento è ancora $(1-p)^{N-1}$
- Quindi la prob. che un nodo trasmetta con successo è $p(1-p)^{2(N-1)}$ (poiché p è la prob. che trasmetta e $((1-p)^{N-1})^2$ è la prob che non ci siano collisioni)
- Studiando p che **massimizza la prob. di successo** per N che tende all'infinito si ottiene che l'efficienza max è $\frac{1}{2e} \approx 0.18 \rightarrow \text{molto bassa!}$

P(trasmissione con successo di un dato nodo) =

$$P(\text{il nodo trasmette}) * P(\text{nessun altro nodo trasmette in } [t_0-1, t_0]) * P(\text{nessun altro nodo trasmette in } [t_0, t_0+1]) =$$

$$p * (1-p)^{N-1} * (1-p)^{N-1} = p * (1-p)^{2(N-1)}$$

Scegliendo p migliore e $n \rightarrow \infty$

$$= \frac{1}{2e} \approx 0.18$$

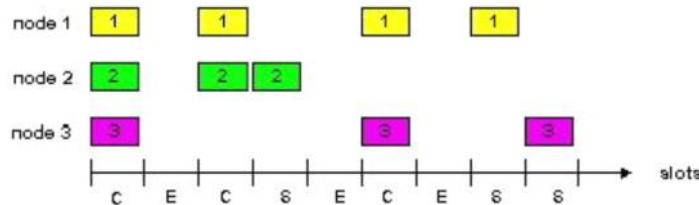
Il **throughput** è 0.18R bps

Slotted ALOHA

Lo **Slotted ALOHA** è una versione modificata dell'ALOHA dove il **tempo è diviso in intervalli discreti**, ciascun corrispondente a un frame time (T_{fr})

Sincronizzazione: i nodi devono essere d'accordo nel confine fra gli intervalli (si può fare emettendo un breve segnale all'inizio di ogni intervallo)

<p>Assumiamo che</p> <ul style="list-style-type: none"> Tutti i pacchetti hanno la stessa dimensione Il tempo è suddiviso in slot uguali al tempo di trasmissione di un pacchetto (T_{fr}) I nodi iniziano la trasmissione solo all'inizio degli slot I nodi sono sincronizzati Se in uno slot due o più pacchetti collidono, i nodi coinvolti rilevano l'evento prima del termine dello slot 	<p>Operazioni</p> <p>Quando a un nodo arriva un nuovo pacchetto da spedire, il nodo attuale attende l'inizio del prossimo slot</p> <ul style="list-style-type: none"> Se non si verifica una collisione: il nodo può trasmettere un nuovo pacchetto nello slot successivo Se si verifica una collisione: il nodo ritrasmette con probabilità p il suo pacchetto durante gli slot successivi
--	--



<p>Pro</p> <ul style="list-style-type: none"> Consente a un singolo nodo di trasmettere continuamente pacchetti alla massima velocità del canale Il tempo di vulnerabilità si riduce ad un solo slot (T_{fr}) 	<p>Contro</p> <ul style="list-style-type: none"> Una certa frazione degli slot presenterà collisioni e di conseguenza andrà sprecata Un'altra frazione rimane vuota, quindi inattiva
---	---

Efficienza

- Supponiamo N nodi con pacchetti da spedire, ognuno trasmette i pacchetti in uno slot con probabilità p
- La prob. di successo di un dato nodo è $p * (1 - p)^{N-1}$
- Poiché ci sono N nodi, la prob. che ogni nodo abbia successo è $N * p * (1 - p)^{N-1}$
- Per N che tende a ∞ , il limite $N * p * (1 - p)^{N-1}$ è uguale a 0.37

Quindi il throughput è 0.37R bps (Quindi nel caso migliore solo il 37% degli slot compie lavoro utile)

CSMA (Carrier Sense Multiple Access) (Accesso Multiplo a Rilevazione della Portante)

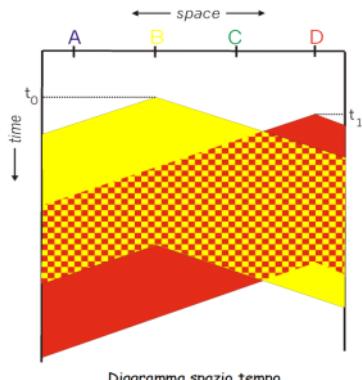
Un nodo **ascolta** prima di trasmettere:

- Se il canale è **libero**, trasmette l'intero pacchetto
- Se il canale **sta già trasmettendo**, il nodo aspetta un altro intervallo di tempo

Le collisioni **possono ancora verificarsi**: il ritardo di propagazione fa sì che due nodi non rilevino la reciproca trasmissione. Es dell'immagine: se nel mentre in cui il segnale di un nodo B si propaga, D non rileva ancora nulla (non gli è ancora arrivato il segnale) quindi può iniziare a trasmettere mentre il segnale di B è ancora in viaggio.

Tempo di vulnerabilità = tempo di propagazione

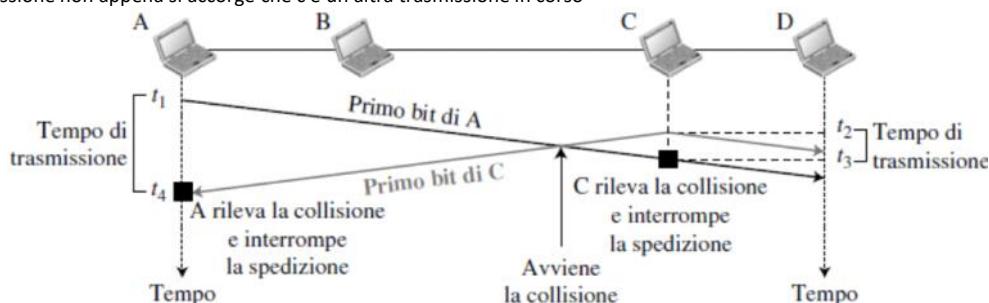
Nota: la distanza e il ritardo di propagazione giocano un ruolo importante nel determinare la prob. di collisione



CSMA/CD (Collision Detection) (Rilevazione di Collisione)

Un nodo ascolta il canale durante la trasmissione, così da poter:

- Rilevare la collisione in poco tempo (facile su LAN cablate, difficile su quelle wireless)
- Annullare la trasmissione non appena si accorge che c'è un'altra trasmissione in corso



- A ascolta il canale e inizia la trasmissione al tempo t_1
- C al tempo t_2 ascolta il canale (non rileva il primo bit di A) e inizia a trasmettere
- Al tempo t_3 C riceve il primo bit di A e interrompe la trasmissione perché c'è collisione
- Al tempo t_4 A riceve il primo bit di C e interrompe la trasmissione perché c'è collisione

Dimensione Minima del Frame

Una stazione una volta inviato un frame non tiene una copia del frame, né controlla il mezzo trasmisivo per rilevare collisioni

- Perche **Collision Detection** funziona, il mittente deve poter rilevare la trasmissione nel mentre in cui sta finendo di trasmettere, ovvero prima di inviare l'ultimo bit del frame
- Il tempo di trasmissione di un frame deve essere almeno due volte il tempo di propagazione T_p , quindi la prima stazione deve essere ancora in trasmissione dopo $2T_p$

E: Una rete che usa il CSMA/CD ha un rate di 10 Mbps. Se il tempo di propagazione max è 25.6 μ s, qual è la dim. minima del frame?

Il tempo di trasmissione minimo del frame è:

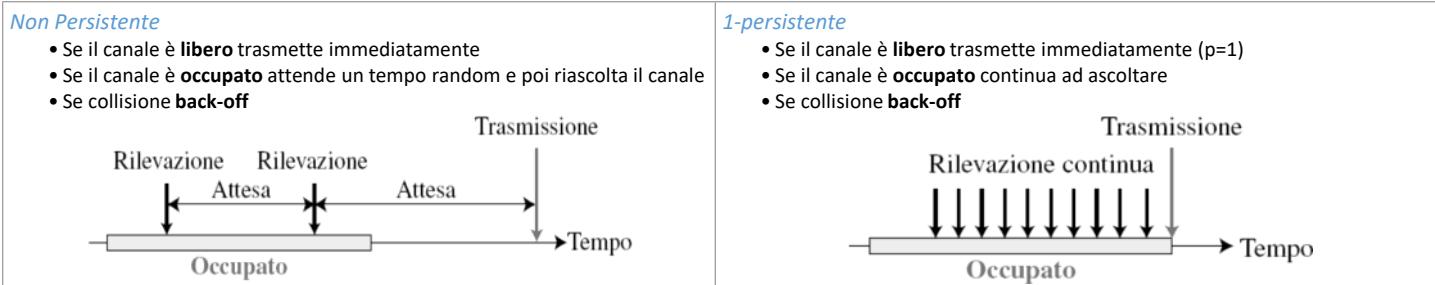
$$T_{fr} = 2 * T_p = 51.2 \mu\text{s}$$

Quindi nel peggior dei casi una stazione deve trasmettere per un periodo di 51.2 μ s per poter rilevare la collisione

Metodi di Persistenza

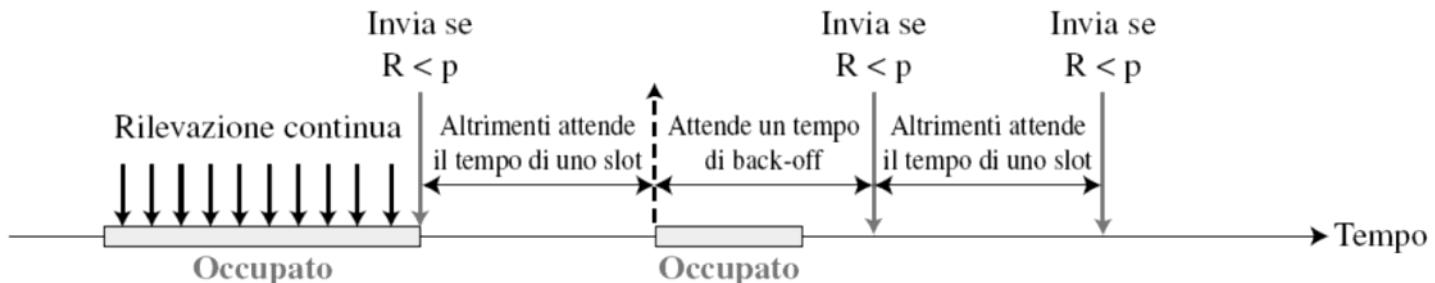
- Cosa fa un nodo se trova il **canale libero**?
 - Trasmette subito
 - Non persistente**
 - 1-persistent**
 - Trasmette con probabilità p
 - p-persistent**
- Cosa fa un nodo se trova il **canale occupato**?

- **Desiste:** riascolta dopo un tempo random
 - **Non persistente**
- **Persiste:** rimane in ascolto finché il canale non si è liberato
 - **1-persistent**
 - **p-persistent** (usato in presenza di time slot)



p-persistent

- Se il canale è **libero**:
 - Trasmette con prob p
 - Attende l'invio con prob $(1-p)$
- Se il canale è **occupato** usa la procedura di **back-off** (attesa di un tempo random e nuovo ascolto del canale)
- Se collisione **back-off**



(ho messo p-persistent separato perché l'immagine d'esempio è più lunga)

Efficienza

Un solo nodo può trasmettere al massimo rate se è il solo a trasmettere

Se più nodi trasmettono, il rate effettivo o throughput è **minore**

Il throughput del CSMA/CD è maggiore sia dell'ALOHA che dello Slotted ALOHA

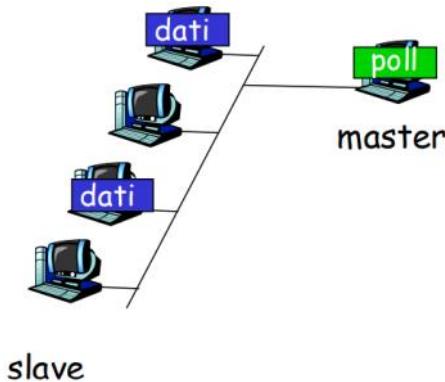
- Per il metodo 1-persistent il throughput max è del 50%

Protocolli a Rotazione

Protocollo Polling

Un nodo principale sonda "a turno" gli altri. In particolare:

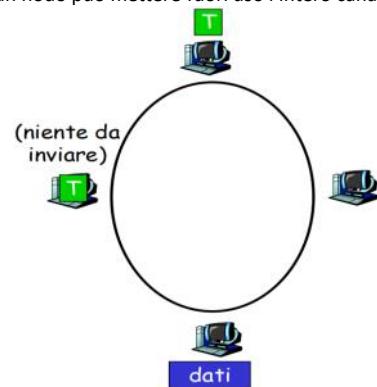
- Elimina le collisioni
- Elimina gli slot vuoti
- Ritardo di polling
- Se il nodo master si guasta, l'intero canale resta inattivo



Protocollo Token-Passing

Un **messaggio di controllo (token)** circola fra i nodi seguendo un ordine prefissato. In particolare:

- Decentralizzato
- Altamente efficiente
- Il guasto di un nodo può mettere fuori uso l'intero canale



Indirizzamento MAC

venerdì 20 giugno 2025 12:09

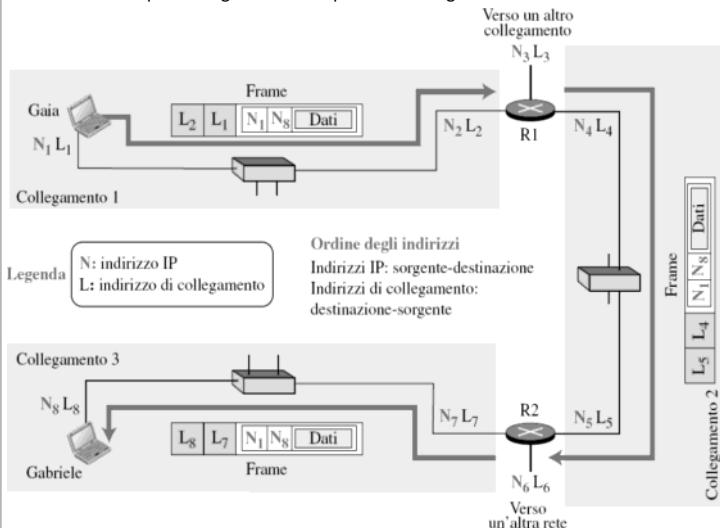
La IEEE sovrintende la gestione degli indirizzi MAC e quando una società vuole costruire adattatori, compra un blocco di spazio di indirizzi (unicità degli indirizzi)

Indirizzo orizzontale MAC → **portabilità**: è possibile spostare una scheda LAN da una LAN ad un'altra

Gli IP invece hanno una struttura gerarchica e devono essere aggiornati se spostati, poiché dipendono dalla sottorete il cui nodo è collegato

Indirizzi IP a 32 bit (Liv. Rete)

- Individuano i punti di Internet dove sono connessi gli host sorgente e destinazione
- IP sorgente e destinatario definiscono le estremità della rete ma non dicono attraverso quali collegamenti deve passare il datagramma

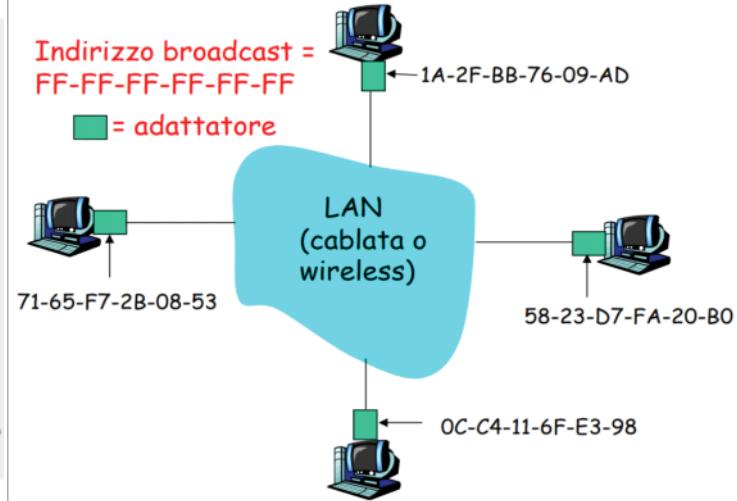


Indirizzi MAC (Liv. Collegamento)

- Indirizzo a 48 bit (6 byte, rappresentati in esadecimale)
- Quando un datagramma passa dal liv. di rete al liv. di collegamento, viene incapsulato in un frame con una intestazione che contiene gli **indirizzi di collegamento della sorgente e destinazione del frame** (non del datagramma)

Indirizzo broadcast = FF-FF-FF-FF-FF-FF

= adattatore



Protocollo per la Risoluzione degli Indirizzi (ARP)

Il protocollo ARP (Address Resolution Protocol) serve per determinare gli indir. di collegamento dalla sorgente alla destinazione

Ogni nodo IP (host, router) nella LAN ha una **tavella ARP**, che contiene la corrispondenza tra IP e MAC e ogni entry è formata così:

<IP, MAC, TTL>

TTL (tempo di vita): val. che indica quando bisognerà eliminare una entry (TTL normale è di 20 min)

Pacchetto ARP

I pacchetti ARP vengono incapsulati direttamente nel frame del liv. di collegamento

Protocollo del livello di collegamento (es. Ethernet)

0	8	16	31
Hardware Type	Protocol Type		
Hardware length	Protocol length	Operation	Richiesta:1, Risposta:2
Source hardware address			
Source protocol address			
Destination hardware address (vuoto nelle richieste)			
Destination protocol address			

Protocollo del livello di rete (es. IPv4)

Hardware: protocollo di collegamento della LAN o WAN

Protocol: protocollo del livello di rete

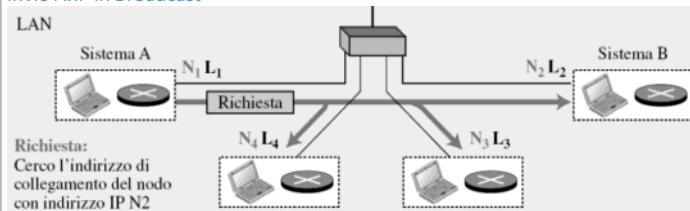
Protocollo ARP nella Stessa Sottorete

Esempio: nodi nella stessa sottorete

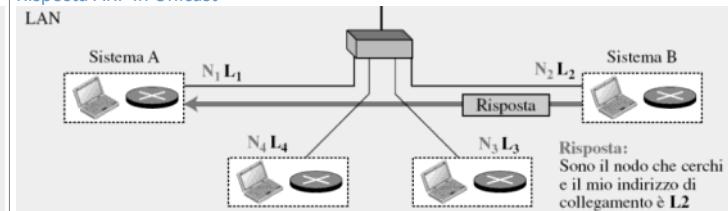
- A vuole inviare un datagramma a B, ma l'indir. MAC di B non è nella tabella ARP di A
- Quindi A trasmette in un pacchetto **broadcast** (FF-FF-FF-FF-FF-FF) il messaggio di richiesta ARP, contenente l'IP di B
- Ogni nodo riceve il pacchetto ma solo il nodo con l'IP specificato risponde (B) comunicando ad A il proprio indir. MAC (il frame viene inviato all'indir. MAC di A in unicast)

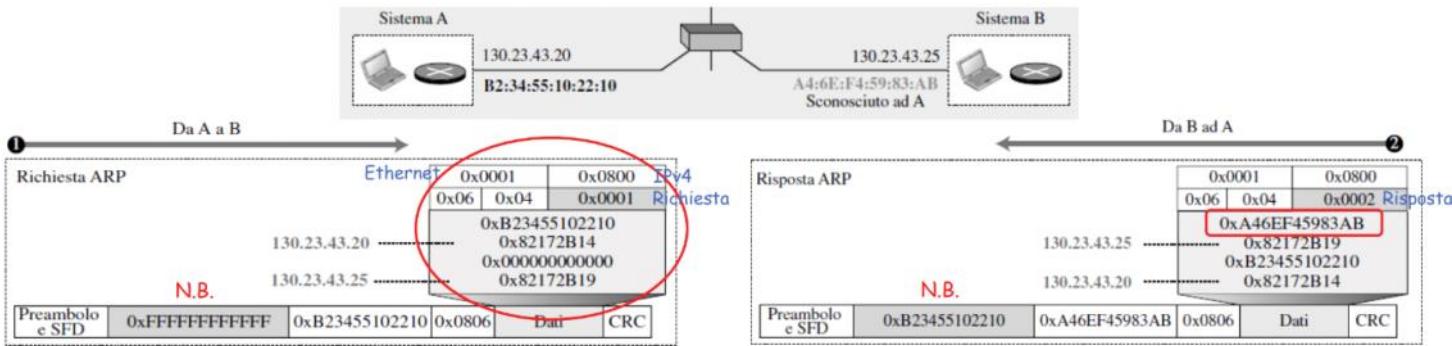
ARP è "plug-and-play": la tabella ARP si costruisce automaticamente e non deve essere configurata dall'amministratore di rete

Invio ARP in Broadcast



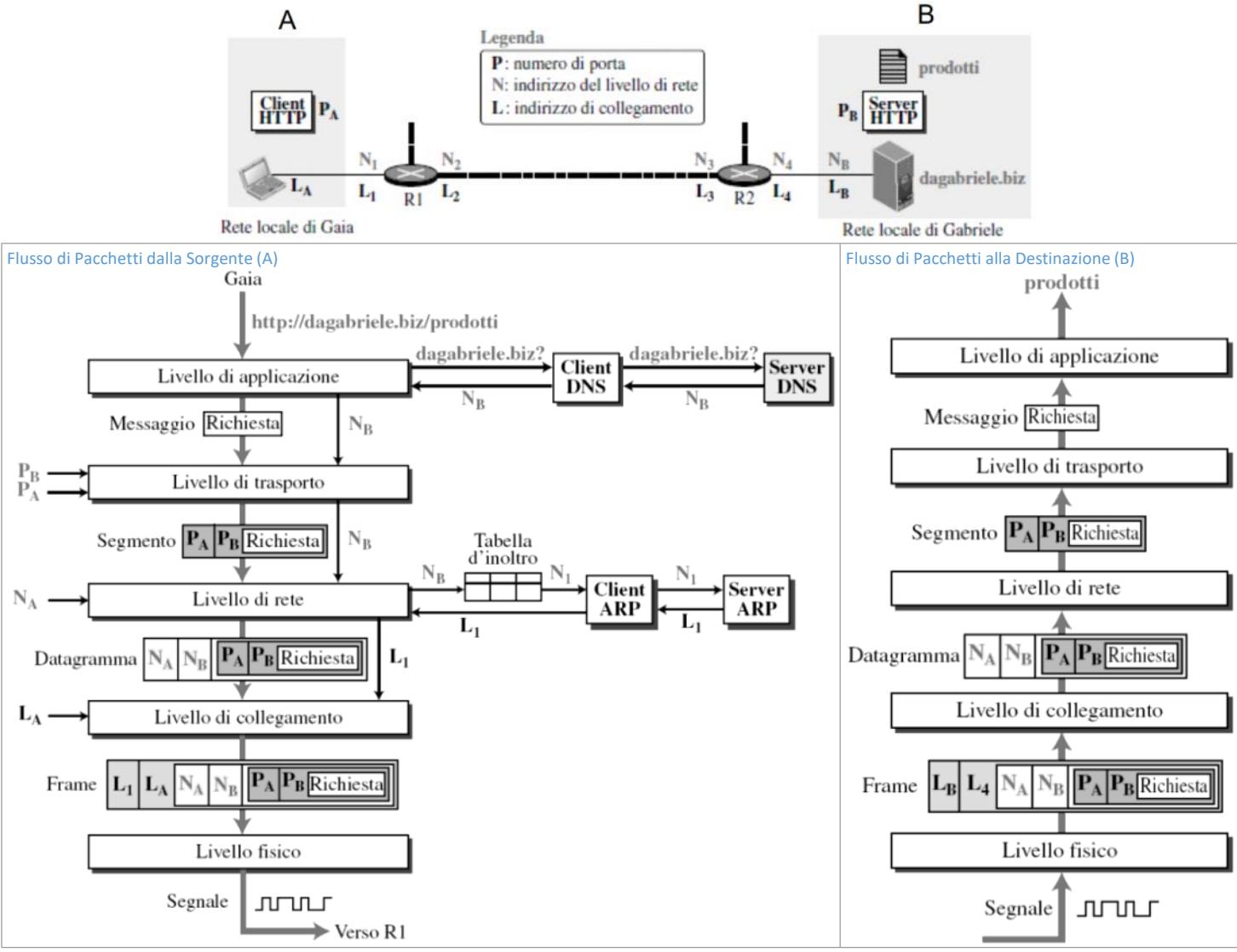
Risposta ARP in Unicast

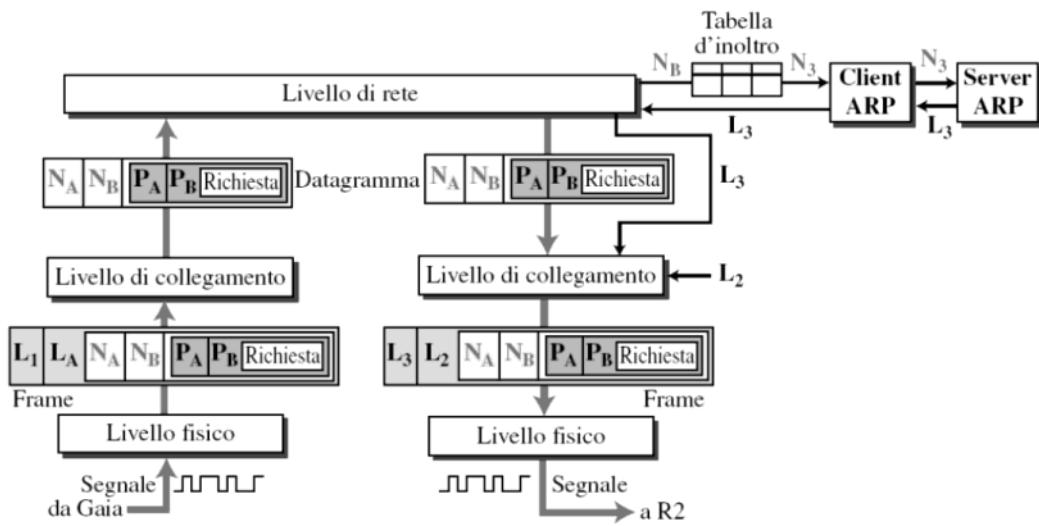




Invio verso un Nodo Esterno alla Sottorete

Richiesta HTTP da A verso B



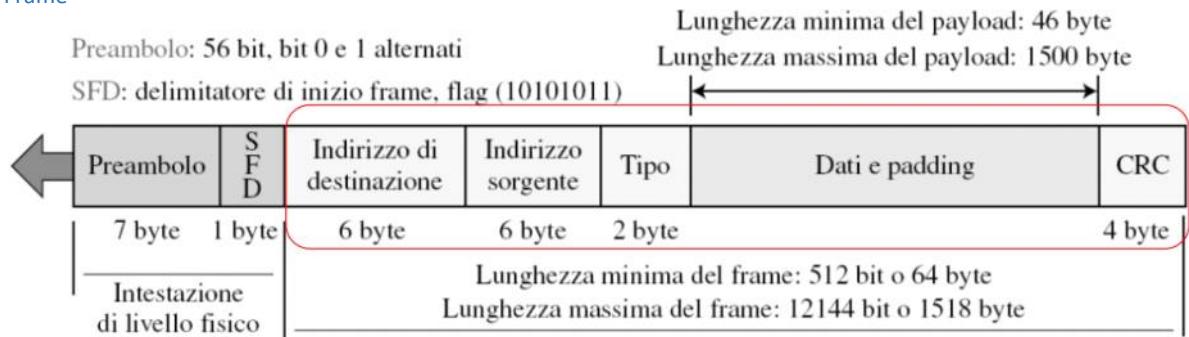


Ethernet Standard (10 Mbps)

Tutte le stazioni che fanno parte di una ethernet hanno una **Network Interface Card (NIC)** o scheda di rete che fornisce un indir. di rete di liv. di collegamento (MAC address)
Gli indir MAC vengono trasmessi da sinistra verso destra, byte per byte, ma per ciascun byte il bit meno significativo viene inviato per primo e quello più significativo per ultimo

- **Senza connessione:** non è prevista nessuna forma di handshake preventiva con il destinatario prima di inviare un pacchetto
- **Non affidabile (come IP e UDP):** la NIC ricevente non invia un riscontro

Formato dei Frame



- **Preamble:** 7 byte (ognuno con i bit i bit 10101010)
 - Serve per "attivare" la NIC dei riceventi e sincronizzare i loro orologi con quelli del transmettente
 - Fa parte dell'header del liv. fisico
- **SFD (Start Frame Delimiter):** 1 byte (10101011)
 - Flag che definisce l'inizio del frame
 - Gli ultimi due bit "11" indicano che inizia l'header MAC
- **Indirizzi sorgente e destinazione:** 6 byte
 - Quando una NIC riceve un pacchetto contenente il proprio indir. di destinazione o l'indir. broadcast, trasferisce il contenuto del campo dati del pacchetto al liv di rete
 - I pacchetti con altri indir. MAC vengono ignorati
- **Tipo:** 2 byte per multiplexing/demultiplexing - il protocollo superiore del pacchetto encapsulato nel frame (IP, ARP, OSPF, ecc)
- **Dati:** (da 46 a 1500 byte) contiene datagramma di rete.
 - Se il datagramma è inferiore alla dim. minima (46 byte) il campo viene **stuffed** (riempito) con degli zero fino a raggiungere quel valore
- **CRC:** consente alla NIC ricevente di rilevare la presenza di un errore nei bit sui campi indirizzo, tipo e dati
- **Lunghezza minima:** 64 byte (18 di intestazione e trailer e 46 di dati del liv. superiore) necessaria per il corretto funzionamento del CSMA/CD
- **Lunghezza massima:** 1518 byte (18 di intestazione e trailer e 1500 di dati) necessaria per evitare che una stazione possa monopolizzare il mezzo

Protocollo CSMA/CD di Ethernet

1. **Framing:** La NIC riceve un datagramma di rete dal nodo cui è collegato e prepara un frame Ethernet
 2. **Carrier Sense e trasmissione:** Se il canale è **inattivo** (misura il liv. di energia sul mezzo trasmissivo per un breve periodo di tempo, es 100μs), inizia la trasmissione.
Se il canale risulta **occupato**, resta in attesa fino a quando non rileva più il segnale, a quel punto trasmette
 3. **Collision detection:** Verifica, durante la trasmissione, la presenza di eventuali segnali di altre NIC. Se non rileva nulla, considera il pacchetto spedito
 4. **Jammer:** Se rileva segnali da altre NIC, interrompe la trasmissione e invia un **segnale di disturbo** (jam di 48 bit) per avvisare le altre NIC che trasmettono della collisione
 5. **Backoff esponenziale:** La NIC rimane in attesa. Quando riscontra l'n-esima collisione consecutiva, stabilisce un val k tra {0, 1, 2, ..., 2^m-1}, dove m è il **minimo tra n e 10**. La NIC aspetta un tempo pari a K volte 512 bit e ritorna al passo 2
- Obiettivo:** la NIC prova a stimare quanti sono gli adattatori coinvolti. Se sono numerosi il tempo di attesa potrebbe essere lungo
- **Prima collisione:** sceglie K tra {0, 1}; il tempo di attesa è pari a K volte 512 bit
 - **Dopo la seconda collisione:** sceglie K tra {0, 1, 2, 3}
 - **Dopo dieci collisioni:** sceglie K tra {0, 1, 2, 3, 4, ..., 1023}

(Scusate ho saltato il Fast e Gigabit Ethernet, switch e VLAN)

LAN Wireless (IEEE 802.11)

venerdì 20 giugno 2025 17:45

Nella LAN wireless abbiamo i **wireless hosts** (es: laptop, telefono) che sono collegati alle **stazioni base** detto **Access Point (AP)**, tipicamente collegate via cavo alla infrastruttura di rete, che sono responsabili dell'invio dei pacchetti dai wireless hosts nella sua area e la connessione cablata. Il **wireless link** è il link dorsale che connette gli host agli AP e dove la forza del segnale diminuisce all'aumentare della distanza dall'AP.

Caratteristiche del link wireless:

- **Propagazione multi-path:** poiché il segnale viaggia per onde radio, quando esso trova un ostacolo riflette con una perdita di potenza e può arrivare al destinatario attraverso percorsi multipli.
 - **Interferenza:** si può avere un interferenza del segnale causato:
 - Dalla sorgente stessa: se il segnale arriva più volte a causa del multi-path
 - Da altre sorgenti: altri trasmettitori usano la stessa banda di frequenza per comunicare
- Queste caratteristiche causano **errori**, e con il **Signal to Noise Ratio (SNR)** o rapporto segnale-rumore possiamo misurare il rapporto tra segnale buono e cattivo
- **Alto:** il segnale è più forte del rumore e possiamo recuperare i dati
 - **Basso:** il segnale è stato danneggiato dal rumore e i dati non sono recuperabili

Controllo dell'Accesso al Mezzo

Per controllare l'accesso al mezzo condiviso non possiamo usare CSMA/CD come con Ethernet, perché:

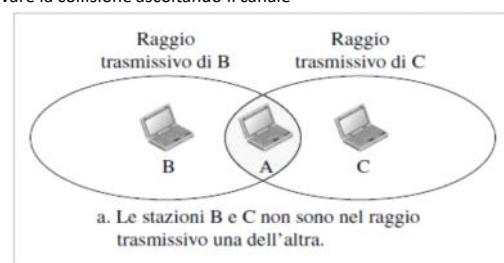
No collision Detection

Per rilevare una collisione un host deve trasmettere e ascoltare contemporaneamente sul canale

Poiche la potenza del segnale ricevuto è molto inferiore a quella trasmessa, sarebbe troppo costoso usare un adattatore che rileva le collisioni

Hidden Terminal Problem

Un host potrebbe non accorgersi che un altro sta trasmettendo e quindi non sarebbe in grado di rilevare la collisione ascoltando il canale



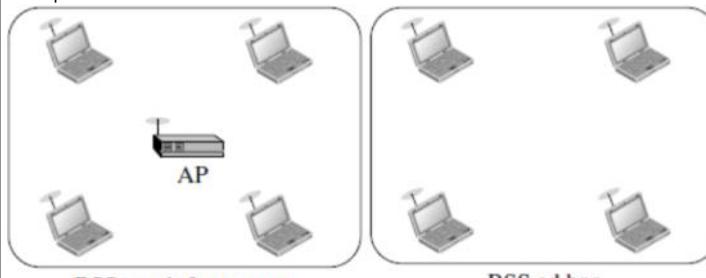
Problema di attenuazione del segnale

Architettura Wifi

BSS (Basic Service Set)

Costituita da uno o più host wireless e da un Access Point

Corrisponde alle **celle** delle reti cellulari



- L'AP è collegato a un router
- Architettura più diffusa

BSS ad hoc

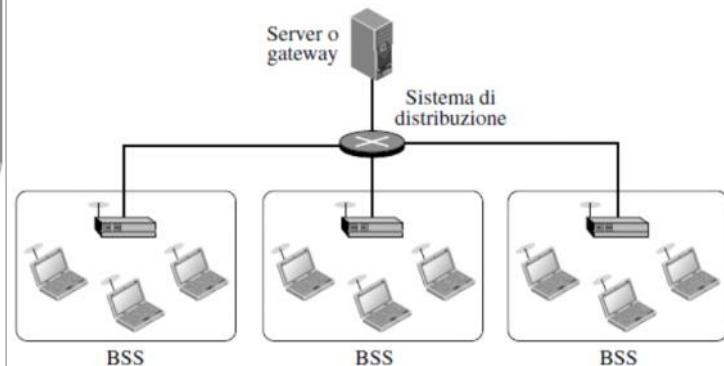
Rete standalone

ESS (Extended Service Set)

Costituito da due o più BSS con infrastruttura, collegati tramite un sistema di distribuzione che è una rete cablata o wireless

Quando i BSS sono collegati, le stazioni in visibilità comunicano direttamente mentre le altre comunicano tramite l'AP

Architettura molto comune nelle reti WiFi moderne



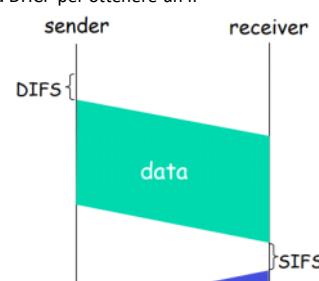
Canali e Associazione

Lo spettro 2.4GHz - 2.485GHz è diviso in **11 canali** parzialmente sovrapposti. L'admin dell'AP sceglie una frequenza però sono possibili interferenze con AP vicini nel caso in cui usano lo stesso canale

Il num. max di frequenze utilizzabili da diversi AP per evitare interferenze è 3, così i canali non interferiscono se separati da 4 o più canali

Associazione di una stazione a un AP:

- È necessario conoscere gli AP disponibili in un BSS
- È necessario un protocollo di associazione
 - AP invia **segnali periodici (beacon)** che includono l'ID dell'AP (**Service Set Identifier - SSID**) e il suo indirizzi MAC
 - La stazione wireless che vuole entrare in un BSS scandisce gli 11 canali trasmissivi alla ricerca di **frame beacon** (passive scanning)
 - La stazione sceglie poi l'AP da cui ha ricevuto il beacon con maggiore potenza di segnale e invia un **frame con la richiesta** di associazione
 - L'AP accetta la richiesta con un **frame di risposta associazione** che permette all'host entrante di inviare una richiesta DHCP per ottenere un IP
 - Può essere prevista un'autenticazione per eseguire l'associazione



Protocollo MAC 802.11

Per il controllo di accesso al mezzo per le stazioni, abbiamo 2 tecniche:

- **Distributed Coordination Function (DCF)** in cui i nodi si contendono l'accesso al canale
- **Point Coordination Function (PCF)** in cui non c'è contesa e l'AP coordina l'accesso dei nodi al canale

DCF: CSMA/CA (CSMA/Collision Avoidance)

DCF implementa CSMA/CA con un algoritmo di backoff esponenziale per gestire l'accesso

- **Evita le collisioni:** due o più nodi che trasmettono simultaneamente
- **Carrier sense:** ascoltare il canale prima di trasmettere
- **No collision detection** per 3 motivi:

DCF implementa CSMA/CA con un algoritmo di backoff esponenziale per gestire l'accesso

- **Evita le collisioni:** due o più nodi che trasmettono simultaneamente
- **Carrier sense:** ascoltare il canale prima di trasmettere
- **No collision detection** per 3 motivi:
 - Impossibilità di trasmettere e ricevere contemporaneamente
 - Hidden Terminal Problem
 - Raggio di trasmissione limitato (difficile ascoltare tutte le trasmissioni)

CSMA/ACK

Grazie ad un ACK di risposta possiamo avere un riscontro per capire se una trasmissione è andata a buon fine (no collisioni), però possono esserci collisioni anche su ACK.

Siccome il mittente non può aspettare all'infinito l'ACK, quindi imposta un **timer** (ACK timeout) e se scade senza aver ricevuto l'ACK, suppone che la trasmissione sia fallita e tenta una **ritrasmissione**

CSMA/CA: Spazio Interframe (IFS)

Gli IFS sono **tempi di attesa** che una stazione aspetta dopo aver rilevato che il **canale è libero** prima di trasmettere così da evitare che stazioni che hanno già iniziato a trasmettere collidano con la stazione che vuole trasmettere

- **SIFS:** Short IFS realizza alta priorità (usato per risposte rapide tipo ACK)
- **DIFS:** Distributed IFS realizza bassa priorità (usato per avviare una nuova trasmissione)

Mittente: ascolta il canale e se libero per **DIFS tempo** allora trasmette

Ricevente: se il frame è ricevuto correttamente invia un ACK dopo **SIFS tempo**

Se durante l'intervallo DIFS il canale diventa occupato:

- Il nodo **interrompe il conteggio del DIFS**
- **Aspetta** che il canale torni **completamente libero**
- Quando il canale torna libero, **riavvia da zero** il conteggio del DIFS completo (es 50μs)

CSMA/CA: Finestra di Contesa

Dopo aver atteso un tempo IFS, se il canale è **ancora inattivo**, la stazione attende un ulteriore tempo (base sul canale sia libero prima di trasmettere (nel caso in cui qualche altra stazione inizi a trasmettere) (il timer è:

- Sceglie R random in [0, CW]
- While $R > 0$:
 - Ascolta il canale per uno slot
 - Se il canale è libero per la durata dello slot → $R = R - 1$
 - Altrimenti attende, interrompe il timer e aspetta che il canale si liberi riavviando il timer

Fasi:

- 1) **Attesa dell'IFS (es: DIFS)**
 - Il nodo rileva il **canale libero**
 - Attende un tempo fisso: **DIFS** (es 34 μs)
 - Se il canale **rimane libero** per tutto il DIFS → passa alla **finestra di contesa**
- 2) **Finestra di contesa (Backoff)**
 - Il nodo **sceglie un num. casuale** tra [0, CW-1]
 - Ogni slot di tempo della finestra si chiama **Slot Time** (es 20 μs)
 - Il nodo **inizia a contare all'indietro** (backoff counter), solo se il canale rimane libero

Esempio:

- CW = 16 ⇒ intervallo casuale: [0, 15] ⇒ supponiamo scelga 7
- Il nodo attende $7 \times \text{SlotTime} = 7 \times 20\mu\text{s} = 140\mu\text{s}$ solo se il canale è libero
- Se durante questo tempo qualcun altro inizia a trasmettere, il nodo **pausa** il countdown
- Quando il canale torna libero, **riprende** da dove aveva interrotto

CSMA/CA: RTS/CTS e NAV

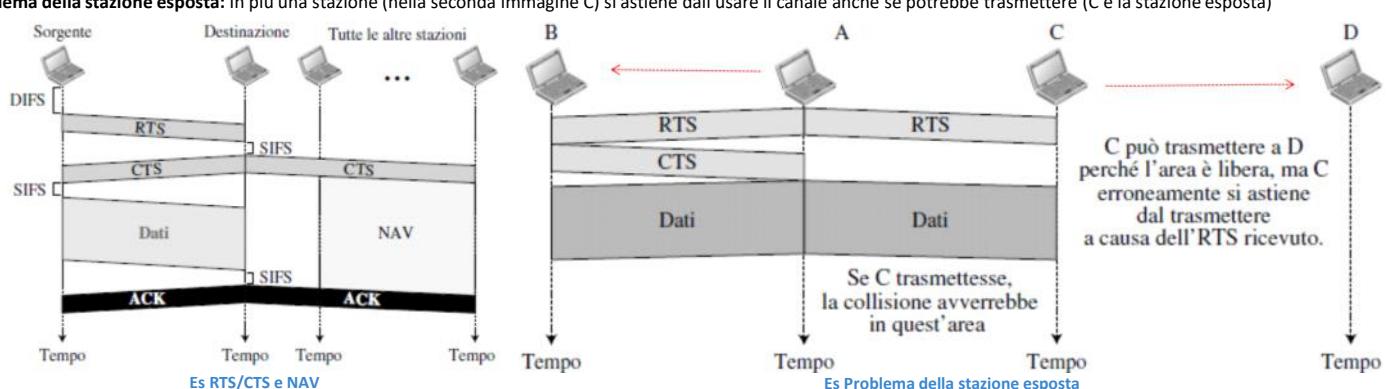
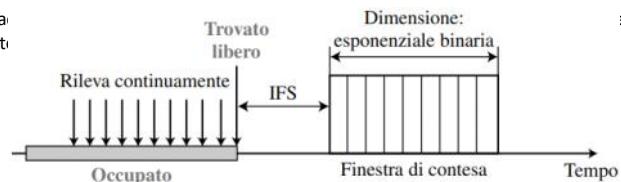
Il problema dell'hidden terminal viene risolto solo con un meccanismo di **prenotazione del canale**: Request-to-send (RTS) e Clear-to-send (CTS).

- La stazione sorgente invia un frame RTS al destinatario per poter prenotare il canale
- Il destinatario **conferma la prenotazione** inviando un frame CTS alla sorgente e a tutte le altre stazioni influenzate
- Ora il destinatario può iniziare a inviare i dati e le altre stazioni attendono che finisca l'invio dei dati
- Alla fine il destinatario invia alla sorgente e alle altre stazioni un ACK per confermare la fine della trasmissione così che le altre stazioni possano ricominciare a trasmettere

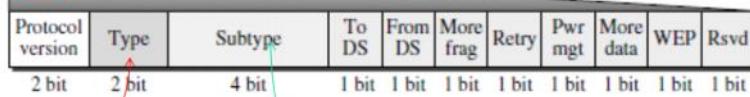
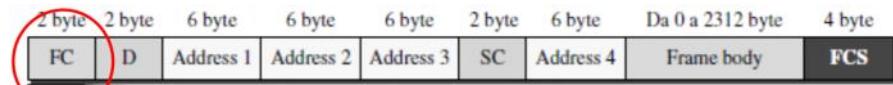
Quando una stazione invia un frame RTS include la durata di tempo in cui **occuperà il canale** per la trasmissione e l'ACK. Questo tempo viene incluso nel CTS e le altre stazioni influenzate dalla trasmissione avviano un **timer (NAV)** che indica **quanto tempo attendere** prima di ascoltare di nuovo il canale

Nota: Se il mittente **non riceve CTS** allora assume che c'è stata una collisione e riprova dopo un tempo di backoff

Problema della stazione esposta: In più una stazione (nella seconda immagine C) si astiene dall'usare il canale anche se potrebbe trasmettere (C è la stazione esposta)



Formato del Frame



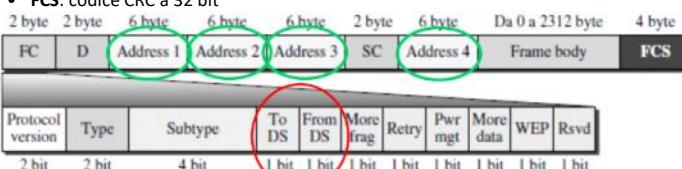
Una LAN wireless ha 3 categorie di frame: gestione, controllo e dati

00: Frame di gestione: usati per le comunicazioni iniziali tra stazioni e punti di accesso

01: Frame di controllo: si usano per accedere al canale e dare riscontro (1011: RTS, 1100: CTS, 1101: ACK)

10: Frame di dati: vengono usati per trasportare i dati

- **Frame Control (FC)**: tipo di frame e alcune info di controllo
- **D**: durata della trasmissione, usata per impostare il NAV (impostata sia per DATA che per RTS/CTS)
- **Indirizzi**: indirizzi MAC
- **SC**: info sui frammenti (#frammento e #sequenza). Il #sequenza distingue frame ritrasmessi come nel liv di trasporto (per ACK perduti)
- **Frame Body**: payload
- **FCS**: codice CRC a 32 bit

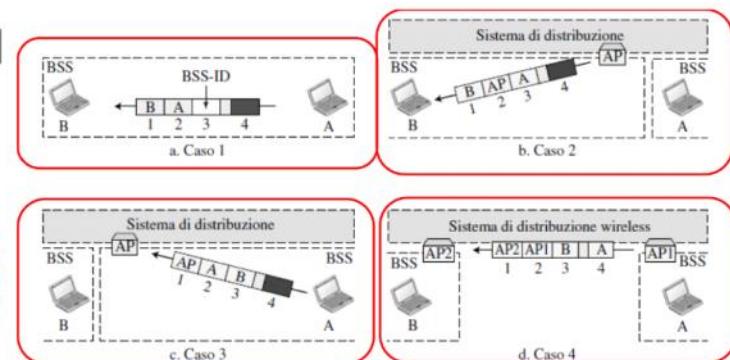


DS: sistema di distribuzione

To DS	From DS	Address 1	Address 2	Address 3	Address 4
10	0	Destinazione	Sorgente	BSS ID	N/A
10	1	Destinazione	AP mittente	Sorgente	N/A
11	0	AP ricevente	Sorgente	Destinazione	N/A
11	1	AP ricevente	AP mittente	Destinazione	Sorgente

Indirizzo del dispositivo successivo a cui viene trasmesso il frame

Indirizzo del dispositivo che il frame ha lasciato

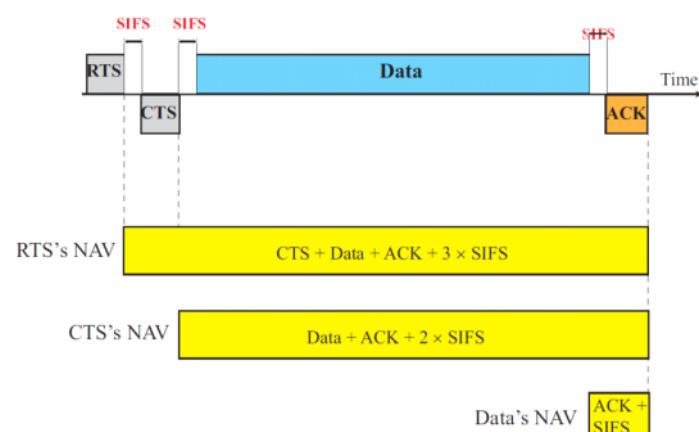


Esercizio

In una rete 802.11 la stazione A invia un frame dati alla stazione B. Qual è il valore del campo D (in microsecondi) che bisogna impostare per il periodo NAV in ognuno dei seguenti frame: RTS, CTS, dati e ACK? Si supponga che il tempo di trasmissione per RTS, CTS e ACK sia di 4 µs ciascuno, quello per il frame di dati sia di 40 µs e la durata del SIFS sia impostata a 1 µs. Ignorare il tempo di propagazione.

Da notare che ogni frame deve impostare la durata del NAV per il resto del tempo durante il quale il mezzo deve essere prenotato per completare la transazione.

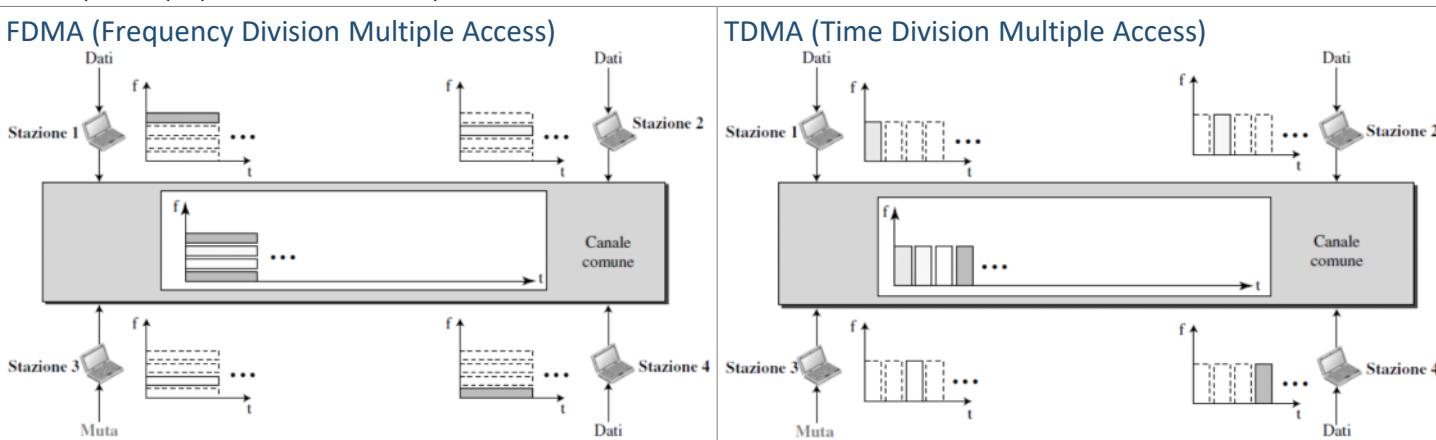
- RST = 4 + 40 + 3x1 = 51 microsecondi
- CST = 40 + 4 + 2x1 = 46 microsecondi
- Dati = 4 + 1 = 5 microsecondi
- ACK = 0 microsecondi (quando l'ACK viene trasmesso, non serve riservare il canale)



CDMA

mercoledì 25 giugno 2025 11:14

Esistono più modi per permettere l'accesso multiplo al mezzo mediante la **suddivisione del canale**:

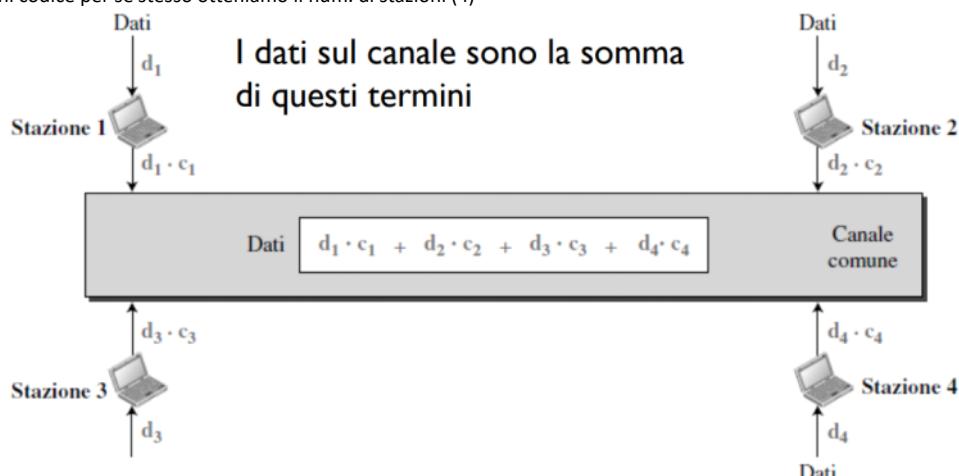


CDMA (Code Division Multiple Access)

Con CDMA una stazione può usare l'intera ampiezza di banda contemporaneamente insieme ad altre stazioni e la divisione della comunicazione è data dai codici diversi assegnati alle coppie di stazioni

Esempio: abbiamo 4 stazioni sullo stesso canale che spediscono i dati d_1, d_2, d_3, d_4 e i codici assegnati sono c_1, c_2, c_3, c_4 .

- Ogni stazione "moltiplica" i propri dati per il proprio codice e trasmette
- Se moltiplichiamo ogni codice per un altro otteniamo 0
- Se moltiplichiamo ogni codice per se stesso otteniamo il num. di stazioni (4)



- qualsiasi stazione voglia ricevere dati da una delle altre tre stazioni moltiplica i dati ricevuti per il codice del mittente e divide per il num. di stazioni

Esempio: stazione 2 vuole ricevere dalla stazione 1

$$DATI = \frac{[(d_1 \cdot c_1 + d_2 \cdot c_2 + d_3 \cdot c_3 + d_4 \cdot c_4) \cdot c_1]}{4} = \frac{[d_1 \cdot c_1 \cdot c_1 + d_2 \cdot c_2 \cdot c_1 + d_3 \cdot c_3 \cdot c_1 + d_4 \cdot c_4 \cdot c_1]}{4} = \frac{4 \cdot d_1 + 0 + 0 + 0}{4} = d_1$$

Sequenze di Chip

Ad ogni stazione viene assegnato un codice che è una **sequenza ortogonale di numeri (chip)**. Proprietà delle sequenze ortogonali

C_1

C_2

C_3

C_4

$[+1 \quad +1 \quad +1 \quad +1]$

$[+1 \quad -1 \quad +1 \quad -1]$

$[+1 \quad +1 \quad -1 \quad -1]$

$[+1 \quad -1 \quad -1 \quad +1]$

1. Ogni seq. è composta da N elem. (N = stazioni), N deve essere una potenza di 2

2. Se moltiplichiamo una seq. per un num., ogni elemento della seq. viene moltiplicato per quel num.

$$2 * [+1 \quad +1 \quad -1 \quad -1] = [+2 \quad +2 \quad -2 \quad -2]$$

3. Se moltiplichiamo due seq. uguali e sommiamo i risultati otteniamo N

$$[+1 \quad +1 \quad -1 \quad -1] * [+1 \quad +1 \quad -1 \quad -1] = 1 + 1 + 1 + 1 = 4$$

4. Se moltiplichiamo due seq. diverse e sommiamo i risultati otteniamo 0

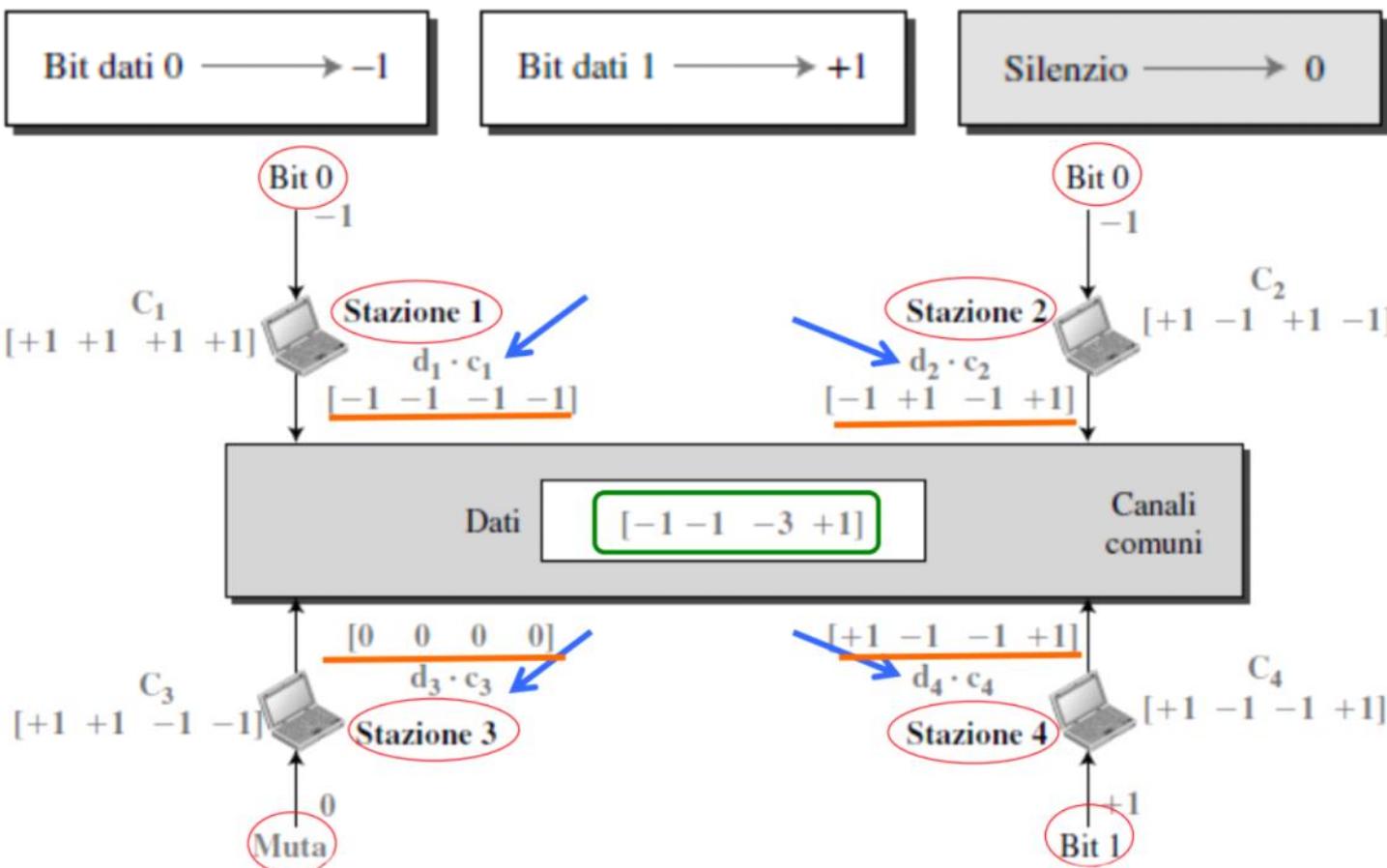
$$[+1 \quad +1 \quad -1 \quad -1] * [+1 \quad +1 \quad +1 \quad +1] = 1 + 1 - 1 - 1 = 0$$

5. Sommare due seq. significa sommare gli elem. corrispondenti

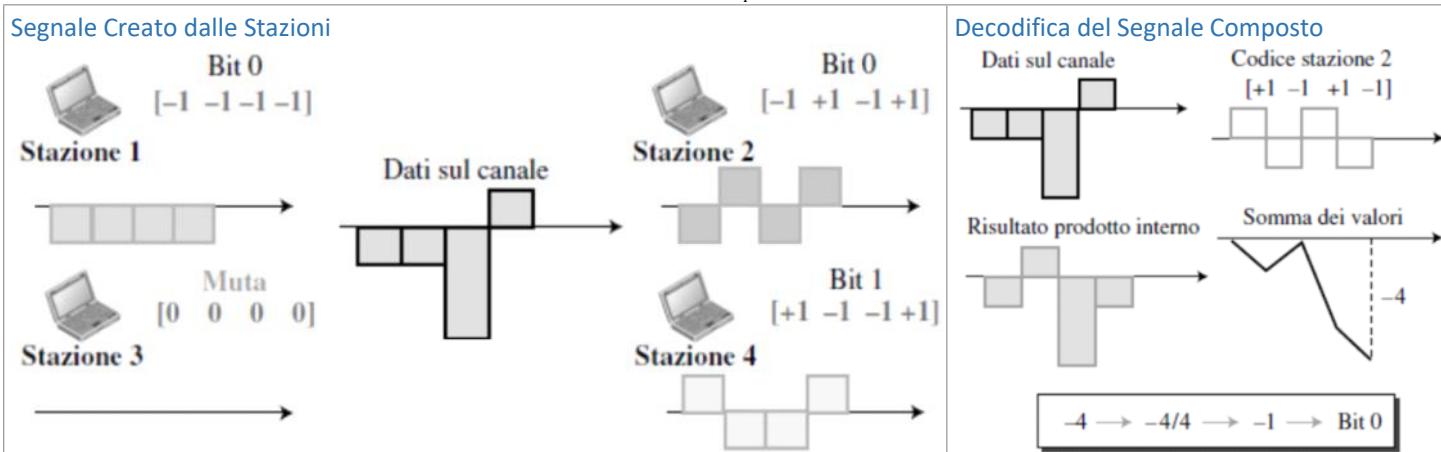
$$[+1 \quad +1 \quad -1 \quad -1] * [+1 \quad +1 \quad +1 \quad +1] = [+2 \quad +2 \quad 0 \quad 0]$$

Rappresentazione dei Dati

Regole per la codifica:



- La seq. sul canale è la somma delle quattro seq. inviate dalle stazioni
- La stazione 3 ascolta la 2 $\rightarrow [-1 -1 -3 +1] * [+1 -1 +1 -1] = -\frac{4}{4} = -1 \rightarrow \text{bit } 0$



Generazione Sequenze di Chip

Per generare sequenze di chip usiamo una **tavella di Walsh** (matrice quadrata)

Nella tab. di Walsh ogni riga è una seq. di chip

- W_1 indica una seq. con un chip solo (con una riga e una colonna) e può assumere val. +1 o -1 (a scelta)
- Conoscendo W_N possiamo creare W_{2N} nel seguente modo

$$W_1 = \begin{bmatrix} +1 \end{bmatrix} \quad W_{2N} = \begin{bmatrix} W_N & W_N \\ W_N & \overline{W_N} \end{bmatrix}$$

Complemento

a. Due regole base

Esempio: Le righe di W_2 e W_4 sono seq. di chip per reti con 2 e 4 stazioni

$$W_2 = \begin{bmatrix} +1 & +1 \\ +1 & -1 \end{bmatrix} \quad W_4 = \begin{bmatrix} +1 & +1 & +1 & +1 \\ +1 & -1 & +1 & -1 \\ +1 & +1 & -1 & -1 \\ +1 & -1 & -1 & +1 \end{bmatrix}$$

b. Generazione di W_1 , W_2 e W_4

Argomenti

sabato 28 giugno 2025 12:55

Fatto con ChatGPT basandomi sui vari esami/esoneri

1. Lista dettagliata degli argomenti da studiare

Argomenti ricorrenti (più frequenti e importanti)

Argomento	Tipo di esercizio / domanda
Calcolo della trasmissione TCP	Calcolo di tempo, RTT, sequenze di pacchetti (Go-Back-N, Stop&Wait, Sliding Window, ecc.)
Protocollo ARP	Analisi di tabelle ARP, funzionamento del protocollo
CSMA / CSMA/CD / CSMA/CA	Descrizione e confronto, identificazione di collisioni, tempo di vulnerabilità
Calcolo del tempo di trasmissione	Con bitrate e lunghezza pacchetto, spesso accoppiato a propagazione
Indirizzamento IP (IPv4)	Calcolo di indirizzi di rete, subnetting, numero di host disponibili, maschere
Routing (statico, dinamico)	Tabelle di routing, forward IP, analisi next-hop
Commutazione di circuito vs pacchetto	Confronti quantitativi e concettuali
Funzionamento protocolli (Ethernet, IP, TCP, UDP, HTTP, DNS, DHCP)	Comprensione a livello applicativo o di processo

Argomenti meno frequenti ma presenti

Argomento	Tipo di esercizio
Protocolli di livello fisico (es. NRZ, Manchester)	Comparazioni o spiegazioni teoriche
Controllo di flusso e congestione	Differenze, meccanismi come AIMD, slow start
Formati di frame	Identificare campi in frame Ethernet, trame IP
Encapsulamento	Sequenza di encapsulamento tra livelli OSI / TCP-IP
ICMP	Tipi di messaggi, ruolo
Segmentazione e riassemblaggio	Descrizione o esempi
Sliding window	Calcoli di throughput, finestra massima, pacchetti simultanei

2. Riassunto dettagliato degli argomenti chiave

TCP e Trasmissione affidabile

- **Stop-and-Wait:**
 - Ogni pacchetto deve essere ACKato prima del successivo.
 - Throughput = pacchetto / (RTT + tempo trasmissione).
- **Go-Back-N:**
 - Si possono inviare N pacchetti senza attendere ACK.
 - In caso di perdita: si ritrasmette tutto da quello perso.
- **Sliding Window:**
 - Finestra scorrevole con ACK cumulativi.
 - Throughput \approx (dimensione finestra * pacchetto) / RTT

CSMA / CSMA/CD / CSMA/CA

- **CSMA:** il nodo ascolta il canale, trasmette se è libero.
- **CSMA/CD** (Ethernet): collisioni rilevate durante trasmissione.
- **CSMA/CA** (Wi-Fi): si attende un IFS, RTS/CTS opzionali.
- **Collisioni:** possono avvenire a causa del tempo di propagazione.
- **Tempo di vulnerabilità:** pari al tempo di propagazione (due nodi remoti potrebbero non sentire la trasmissione dell'altro).

ARP

- Mappa indirizzi IP in indirizzi MAC.
- Funziona con broadcast su LAN.
- Tabelle ARP sono dinamiche e possono essere consultate o aggiornate.

Subnetting

- Maschere di sottorete: determinano la rete e gli host disponibili.
- Esercizi tipici:
 - Calcolo indirizzo di rete, broadcast, numero host.
 - Es. IP: 192.168.1.130/26 → Rete: 192.168.1.128, Broadcast: 192.168.1.191

Routing IP

- **Statico:** configurato manualmente.
- **Dinamico:** via RIP, OSPF, ecc.
- **Tipico esercizio:**
 - Dato un IP di destinazione e una tabella, determinare il next hop.
 - Importante: usare il match più specifico (longest prefix match).

Commutazione

- **Circuito:** stabilisce un percorso fisso con risorse riservate.
- **Pacchetto:** invio dinamico di pacchetti con indirizzamento.
- Esercizi: confronto tra ritardi totali, throughput, efficienza.

Livelli OSI / TCP-IP

- Comprendere i livelli e i protocolli associati.
- Applicazione (HTTP, DNS), Trasporto (TCP/UDP), Rete (IP), Collegamento (Ethernet), Fisico (bit su cavo).

ICMP

- Diagnostica (es. ping usa echo request/reply).
- Tipi: destination unreachable, time exceeded, ecc.

Encapsulamento

- Ogni livello aggiunge un'intestazione (header) al pacchetto.
- Es. HTTP data → TCP segment → IP packet → Ethernet frame

Controllo Congestione TCP

- Slow Start: inizia con finestra bassa, cresce esponenzialmente.
- Congestion Avoidance: crescita lineare.
- Fast Retransmit/Recovery: ACK duplicati innescano ritrasmissione.

Extra: Formule utili

- **Tempo trasmissione:** L / R (L : lunghezza in bit, R : bitrate in bps)
- **Tempo propagazione:** distanza / velocità
- **Throughput:** quantità di dati trasmessi / tempo totale
- **Utilizzo Stop-and-Wait:** $U = L / (L + 2 * RTT * R)$
- **Max host per subnet:** $2^{(32 - CIDR)} - 2$

Esame 8/1/18 M-Z

lunedì 24 febbraio 2025 11:12

Esami presi da <https://github.com/sapienzastudentsnetwork/reti-di-elaboratori>

Scelta Multipla

1. Il ritardo di propagazione
 - o a. E' il tempo che si impiega a immettere tutti i bit di un pacchetto sul link
 - o b. E' il tempo che si impiega per far transitare un bit da un dispositivo a un altro
 - o c. Si calcola come (Lunghezza del pacchetto)/rate
 - o d. Viene usato per calcolare la lunghezza minima di un frame Ethernet
2. Il controllo della congestione
 - o a. E' un problema gestito a livello di trasporto
 - o b. E' un problema gestito a livello di collegamento
 - o c. Serve per regolare la velocità di spedizione in una coppia mittente-destinatario
 - o d. Serve per regolare il carico di traffico nella rete
 - o e. Utilizza un campo nell'intestazione dei segmenti TCP
3. Un protocollo di routing:
 - o a. Definisce come interagiscono i router per trovare le rotte tra i nodi della rete
 - o b. Trasferisce i pacchetti dall'input di un router all'output del router appropriato
 - o c. Gestisce l'accesso a un canale condiviso
 - o d. E' implementato sui sistemi terminali (host)
4. In una LAN wireless:
 - o a. Si può verificare il problema dell'hidden terminal (terminale nascosto)
 - o b. Si può eseguire carrier sense
 - o c. Si può eseguire collision detection
 - o d. E' necessario l'invio di un ACK per confermare la corretta ricezione di un frame

Es 5

Quali dei seguenti resource record non sono corretti? Motivare la risposta

- a) [it, a.dns.it., NS]
- b) [a.dns.it., 138.154.165.1, A]
- c) [prodotti.it, a.dns.it., A]
- d) [course.example.com, ex.example.it., CNAME]
- e) [mygamesnetwork.com, 127.187.66.77, NS]

Es 6

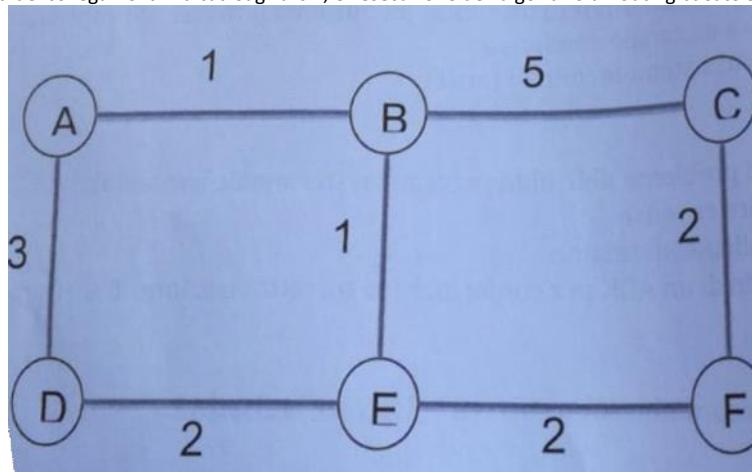
Un server TPC ha ricevuto e riscontrato all'interno di una connessione i byte fino al 5000.

Dire quale azione esegue il server TCP in seguito ai seguenti eventi:

- 1. Il server riceve un segmento di 1000 byte con numero di sequenza pari a 7001.
- 2. In seguito all'evento 1, il server riceve un segmento di 1000 byte con numero di sequenza pari a 5001
- 3. In seguito all'evento 2, il server riceve un segmento di 1000 byte con numero di sequenza pari a 6001
- 4. In seguito all'evento 3, il server riceve un segmento con numero di sequenza 8001

Es 7

Si consideri la seguente rete, con i costi dei collegamenti indicati sugli archi, e l'esecuzione dell'algoritmo di routing basato su vettore di distanza



- a) Scrivere il vettore di distanza iniziale dei nodi C, D, E, F (riportare i vettori sul grafo nell'immagine precedente)
- b) Mostrare il vettore di distanza del nodo C dopo che esso riceve il vettore di distanza iniziale del nodo F.
- c) Quale equazione viene applicata per aggiornare il vettore di distanza? Scrivere e spiegare brevemente l'equazione, fornendo un esempio di applicazione nel caso b)
- d) Qual protocollo implementa l'algoritmo di routing basato su vettore di distanza?
Descriverne brevemente il funzionamento

Es 8

Il protocollo di accesso al mezzo CSMA/CD richiede che ogni stazione rilevi che il canale sia libero prima di iniziare a trasmettere.

- a) Questo metodo elimina del tutto le collisioni?
- b) Motivare la risposta al punto a).
- c) Spiegare cosa si intende per Collision Detection (CD)

Vero/Falso

Domande vero/falso. Ogni risposta corretta vale +1, ogni risposta sbagliata vale -1. La non risposta vale 0.

1. Il ritardo di propagazione tra due nodi dipende dalla loro distanza. [V/F]
2. Nel protocollo FTP è sufficiente una sola connessione dati per inviare e ricevere più file. [V/F]
3. Due sistemi terminali appartenenti a reti LAN diverse possono avere lo stesso indirizzo IP. [V/F]
4. Il protocollo CSMA/CD è implementato nelle LAN Wi-Fi. [V/F]
5. Il protocollo CDMA è un protocollo di routing. [V/F]
6. Il protocollo Aloha puro è più efficiente del protocollo slotted Aloha. [V/F]
7. Il valore del campo indirizzo sorgente nell'intestazione di un frame di collegamento non cambia nel passaggio attraverso un router. [V/F]
8. La crittografia a chiave asimmetrica prevede l'uso di una coppia di chiavi. [V/F]

Es 1

Si consideri un router A che trasmette pacchetti, ognuno di lunghezza L bits, su un canale di trasmissione con Rate R Mbps verso un router B all'altro estremo del link.

Si supponga L=4000 e R=10Mbps.

Si supponga inoltre il ritardo di propagazione pari a 0,2 millisecondi.

- 1) Quanto impiega il router A a trasmettere un pacchetto?
- 2) Qual è il tempo di trasmissione di 1 bit?
- 3) Qual è il massimo numero di pacchetti (L=4000) al secondo che possono essere trasmessi sul link?
- 4) Supponendo che il router A invii i pacchetti uno dopo l'altro senza introdurre ritardi tra la trasmissione di un pacchetto e il successivo, quanto tempo impiega il router B a ricevere 4 pacchetti?
- 5) Qual è il massimo numero di bit che possono essere presenti sul canale?

Es 2

Un server TCP ha ricevuto e riscontrato (invia ACK) all'interno di una connessione i byte fino al 5000.

Dire quale azione esegue il server TCP in seguito ai seguenti eventi:

1. Il server riceve un segmento di 500 byte con un numero di sequenza pari a 6001.
2. In seguito all'evento 1, il server riceve un segmento di 500 byte con numero di sequenza pari a 5001.
3. In seguito all'evento 2, il server riceve un segmento di 500 byte con numero di sequenza pari a 6501.
4. In seguito all'evento 3, il server riceve un segmento di 500 byte con numero di sequenza 5501.

Es 3

a) Disegnare una rete di massimo 6 nodi, non tutti direttamente connessi, e indicare i costi (scelti a piacere) sugli archi.

Scrivere:

- i vettori di distanza iniziali dei nodi (considerando un algoritmo distance vector)
- il link state database (considerando un algoritmo link state)

b) Nel caso dell'algoritmo distance vector mostrare un esempio di aggiornamento del vettore di distanza iniziale di un nodo, quando riceve il vettore di distanza di un nodo vicino.

Spiegare brevemente la formula usata per eseguire l'aggiornamento

Es 4

In quale livello della gerarchia DNS dovrebbe essere memorizzato un resource record come il seguente?

<edu, b.edu-servers.net, NS>

Motivare la risposta, e spiegare quale altro tipo di resource record è necessario insieme a quello indicato.

Es 5

Si supponga che il RTT stimato per un sender TCP sia pari a 320ms, e che il successivo RTT misurato sia pari a 370ms (senza ritrasmissioni).

Si scriva la formula per calcolare il nuovo RTT stimato, considerando che $\alpha=0.5$.

Esonero 15/4/25 M-Z

venerdì 27 giugno 2025 17:58

Vero/Falso

1. Gli Host (sistemi terminali) implementano solo i livelli di trasporto e applicazione [V] [F]
2. Il protocollo FTP usa il protocollo di trasporto TCP [V] [F]
3. Il DNS è un protocollo decentralizzato e gerarchico [V] [F]
4. Il multiplexing consente a più applicazioni di usare contemporaneamente la rete [V] [F]
5. Il ritardo di propagazione dipende dalla lunghezza del pacchetto [V] [F]
6. Il throughput può essere maggiore del bitrate [V] [F]
7. UDP usa un handshake a tre vie per stabilire una connessione [V] [F]
8. Il protocollo TCP fornisce la funzionalità controllo della congestione [V] [F]
9. Il ritardo di accodamento è il tempo che serve a un bit per attraversare il mezzo di trasmissione [V] [F]
10. Il protocollo Go-Back-N usa l'ack cumulativo [V] [F]

Es 1

Esercizio 1 (5 punti)

Dato il seguente esempio di messaggio, spiegare brevemente il ruolo ed effetto di ogni riga del messaggio (scrivere la descrizione direttamente accanto ad ogni riga)

```
GET /usr/users/doc HTTP/1.1
Host: www.esamedireti.it
Connection: close
User-agent: Mozilla/4.0
Content-type: gif/jpeg
content-encoding: MIME-version 1.0
If-Modified-Since: Wed, 25 Jan 2021 09:23:24
```

Es 2

esercizio 2 (3 punti)

Dare un esempio di coppia di resource record memorizzati in un server DNS di tipo radice.

Es 3

Esercizio 3 (3 punti)

Si consideri un host A che trasmette un pacchetto di 5000 bit, su un canale di trasmissione con Rate 10 Mbps verso un host B all'estremo del link. Si supponga inoltre il ritardo di propagazione pari a 0,1 millisecondi.

1. Quanto impiega l'host A a trasmettere un pacchetto?
2. Qual è il massimo numero di bit che possono essere presenti sul canale?
3. Supponendo che l'host A invii i pacchetti uno dopo l'altro senza introdurre ritardi tra la trasmissione di un pacchetto e il successivo, quanto tempo impiega il router B a ricevere 4 pacchetti?

Es 4

esercizio 4. (4 punti)

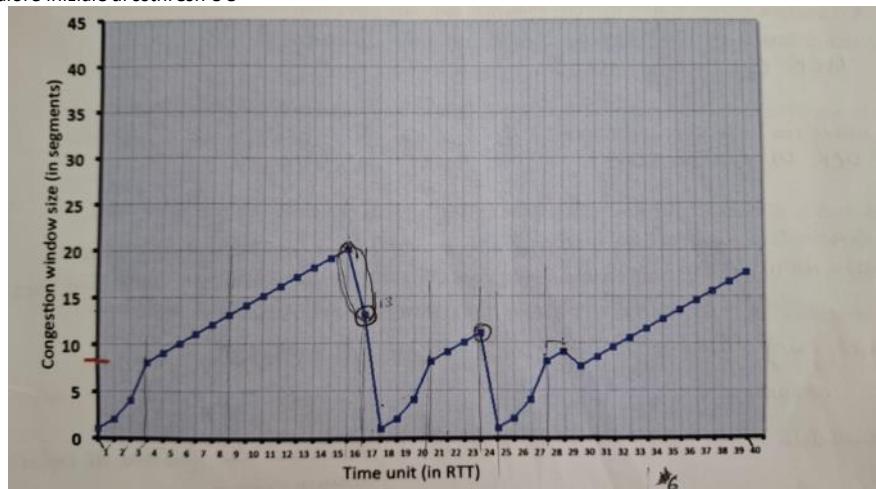
Un host TCP sta ricevendo un byte stream da B. Ha già ricevuto e riscontrato i byte fino al 4000 (numero di sequenza iniziale). Spiegate quali azioni esegue e che tipo di messaggi invia l'host TCP A in seguito ai seguenti eventi:

1. A riceve un segmento di 1000 byte con numero di sequenza pari a 5001
2. In seguito all'evento 1., A riceve un segmento di 1000 byte con numero di sequenza pari a 7001
3. In seguito all'evento 2., A riceve un segmento di 1000 byte con numero di sequenza pari a 4001
4. In seguito all'evento 3., A riceve un segmento di 1000 byte con numero di sequenza 6001

Es 5

Esercizio 5. (7 punti)

Si consideri la figura seguente, che traccia l'evoluzione della finestra di congestione del TCP all'inizio di ogni unità di tempo (dove l'unità di tempo è uguale all'RTT). Il valore iniziale di cwnd è 1 e il valore iniziale di ssthresh è 8



1. Indicare le unità di tempo in cui il TCP è in slow start
2. Indicare le unità di tempo in cui il TCP è in congestion avoidance
3. Indicare le unità di tempo in cui il TCP è in fast recovery
4. Indicare quando avviene un timeout
5. Indicare quando si ricevono 3 ack duplicati
6. Indicare le unità di tempo in cui la ssthresh cambia
7. Quanti segmenti sono stati inviati dall'inizio fino all'unità di tempo 8?

Esonero 29/5/25 A M-Z

venerdì 27 giugno 2025 18:06

Vero/Falso

1. Il protocollo BGP usa il trasporto UDP
2. Il NAT consente a più dispositivi con indirizzi IP privati di accedere a Internet utilizzando un solo indirizzo IP pubblico
3. L'indirizzo IP di destinazione di un datagramma viene aggiornato da ogni router attraversato
4. La prima parte dell'indirizzo MAC identifica la rete di appartenenza
5. Il protocollo CSMA/CD è implementato nelle LAN cablate
6. Il meccanismo RTS/CTS serve a mitigare il problema del terminale nascosto
7. Il tempo di propagazione influenza il tempo di vulnerabilità del protocollo CSMA/CD
8. Il protocollo CSMA/CA rileva le collisioni mentre trasmette

Es 1

Nel protocollo Ethernet Standard, la lunghezza minima di un frame è di 64byte. Spiegare perché Ethernet impone una lunghezza minima del frame. Cosa potrebbe accadere se un frame più corto venisse trasmesso?

Es 2

Disegnare una rete di minimo 3 nodi e massimo 6 nodi, non tutti direttamente connessi, e indicare i costi (scelti a piacere) sugli archi.

1. Scrivere:
 - i vettori di distanza iniziali dei nodi (considerando un algoritmo distance vector)
 - i link state database (considerando un algoritmo link state)
2. Spiegare la differenza tra le informazioni contenute in un vettore di distanza iniziale e il link state database
3. Nel caso dell'algoritmo distance vector, mostrare un esempio di aggiornamento del vettore di distanza iniziale di un nodo quando riceve il vettore di distanza di un nodo vicino. Spiegare brevemente la formula usata per eseguire l'aggiornamento.

Es 3

Si considerino i seguenti tempi di arrivo di pacchetti da spedire ai nodi di una rete broadcast:

i. $t \leq 0,5, 1,9, 2,1, 3,1, 4,3, 4,6$

Ogni trasmissione richiede esattamente un'unità di tempo.

1. Si supponga che tutti i nodi stiano utilizzando il protocollo Aloha puro. Per ciascun pacchetto, indicare l'istante in cui inizia la trasmissione. Quali pacchetti vengono trasmessi con successo?
2. Si supponga che tutti i nodi stiano utilizzando il protocollo Slotted Aloha. Per ciascun pacchetto, indicare l'istante in cui inizia la trasmissione. Quali pacchetti vengono trasmessi con successo?
3. Supponendo che tutti i nodi stiano utilizzando il protocollo CSMA (Carrier Sense Multiple Access), ma senza rilevamento delle collisioni, e che il tempo che intercorre tra l'inizio della trasmissione di un messaggio e l'inizio della sua ricezione da parte degli altri nodi sia di 0,4 unità di tempo (ciò significa che, se un nodo inizia a trasmettere un messaggio a $t = 2,0$ e lo trasmette fino a $t = 3,0$, allora qualsiasi nodo che effettui il carrier sensing nell'intervallo [2,4, 3,4] (estremi inclusi) percepisce il canale come occupato.). Per ciascun messaggio, indica l'istante in cui inizia la trasmissione, oppure indica che la trasmissione non avviene perché il nodo rileva il canale occupato al momento dell'arrivo del messaggio. Quali messaggi vengono trasmessi con successo?
4. Si supponga che:
 - tutti i nodi stiano usando il protocollo Carrier Sense Multiple Access con rilevamento della collisione (CSMA/CD)
 - il tempo tra l'inizio della trasmissione di un messaggio e il momento in cui viene percepito dagli altri nodi sia di 0,4 unità di tempo
 - un nodo possa interrompere istantaneamente la trasmissione non appena viene rilevata una collisione.
 - Per ciascun messaggio, indicare l'istante in cui inizia la trasmissione, oppure indicare che la trasmissione non inizia perché il canale viene percepito come occupato all'arrivo del messaggio. Quali pacchetti vengono trasmessi con successo?

Esonero 29/5/25 B M-Z

venerdì 27 giugno 2025 18:07

Vero/Falso

1. Il protocollo BGP usa connessioni TCP
2. Lo scopo del NAT è tradurre indirizzi IP privati in indirizzi IP pubblici
3. L'indirizzo MAC di destinazione di un frame cambia nel passaggio attraverso un router
4. Gli indirizzi IP hanno una struttura gerarchica
5. Il protocollo CSMA/CA è implementato nelle LAN cablate
6. Il protocollo CSMA/CD risolve il problema del terminale nascosto
7. Il tempo di vulnerabilità del protocollo CSMA/CD dipende dal tempo di trasmissione
8. Il protocollo CSMA/CD interrompe la trasmissione di un frame se un segnale di jam è ricevuto durante la trasmissione

Es 1

Es 2

Disegnare una rete di minimo 3 nodi e massimo 6 nodi, non tutti direttamente connessi, e indicare i costi (scelti a piacere) sugli archi.

1. Scrivere:
 - i vettori di distanza iniziali dei nodi (considerando un algoritmo distance vector)
 - i link state database (considerando un algoritmo link state)
2. Spiegare la differenza tra le informazioni contenute in un vettore di distanza iniziale e il link state database
3. Nel caso dell'algoritmo distance vector, mostrare un esempio di aggiornamento del vettore di distanza iniziale di un nodo quando riceve il vettore di distanza di un nodo vicino. Spiegare brevemente la formula usata per eseguire l'aggiornamento.

Es 3

Si considerino i seguenti tempi di arrivo di pacchetti da spedire ai nodi di una rete broadcast:

$$t = \{0.6, 1.2, 1.8, 2.4, 3.8, 3.9, 4.9\}$$

Ogni trasmissione richiede esattamente un'unità di tempo.

1. Si supponga che tutti i nodi stiano utilizzando il protocollo Aloha puro. Per ciascun pacchetto, indicare l'istante in cui inizia la trasmissione. Quali pacchetti vengono trasmessi con successo?
2. Si supponga che tutti i nodi stiano utilizzando il protocollo Slotted Aloha. Per ciascun pacchetto, indicare l'istante in cui inizia la trasmissione. Quali pacchetti vengono trasmessi con successo?
3. Supponendo che tutti i nodi stiano utilizzando il protocollo CSMA (Carrier Sense Multiple Access), ma senza rilevamento delle collisioni, e che il tempo che intercorre tra l'inizio della trasmissione di un messaggio e l'inizio della sua ricezione da parte degli altri nodi sia di 0,4 unità di tempo (ciò significa che, se un nodo inizia a trasmettere un messaggio a $t = 2.0$ e lo trasmette fino a $t = 3.0$, allora qualsiasi nodo che effettui il carrier sensing nell'intervallo [2.4, 3.4] (estremi inclusi) percepisce il canale come occupato.). Per ciascun messaggio, indica l'istante in cui inizia la trasmissione, oppure indica che la trasmissione non avviene perché il nodo rileva il canale occupato al momento dell'arrivo del messaggio. Quali messaggi vengono trasmessi con successo?
4. Si supponga che:
 - tutti i nodi stiano usando il protocollo Carrier Sense Multiple Access con rilevamento della collisione (CSMA/CD)
 - il tempo tra l'inizio della trasmissione di un messaggio e il momento in cui viene percepito dagli altri nodi sia di 0,4 unità di tempo
 - un nodo possa interrompere istantaneamente la trasmissione non appena viene rilevata

una collisione.

- Per ciascun messaggio, indicare l'istante in cui inizia la trasmissione, oppure indicare che la trasmissione non inizia perché il canale viene percepito come occupato all'arrivo del messaggio. Quali pacchetti vengono trasmessi con successo?

Esame 31/3/23 A-L

sabato 28 giugno 2025 10:57

Vero/Falso

Domande vero/falso. Ogni risposta corretta vale +1, ogni risposta sbagliata vale -1. La non risposta vale 0.

1. L'algoritmo di Slow Start è usato da TCP Reno [V/F]
2. Il throughput può essere maggiore del bitrate [V/F]
3. Il protocollo UDP usa l'ACK per riscontrare i datagrammi [V/F]
4. La funzionalità di flow control adatta la finestra di invio del mittente per non sovraccaricare il destinatario [V/F]
5. Il protocollo TCP fornisce la funzionalità controllo della congestione [V/F]
6. Il ritardo di accodamento è il tempo che serve a un bit per attraversare il mezzo di trasmissione [V/F]
7. Il valore del tempo di timeout dopo il quale considerare un pacchetto perso dipende dalla velocità di trasmissione [V/F]
8. UDP implementa un handshake a due vie invece che tre [V/F]
9. Il protocollo FTP usa il protocollo di trasporto TCP [V/F]
10. Il protocollo Go-Back-N usa l'ACK cumulativo [V/F]

Es 1

Dato il seguente esempio di messaggio:

```
GET /usr/users/doc HTTP/1.1 Host: www.esamedireti.it
Connection: close
User-agent: Mozilla/4.0
Content-type: gif/jpeg
Content-encoding: MIME-version 1.0
If-Modified-Since: Wed, 25 Jan 2021 09:23:24
```

Rispondere alle seguenti domande:

1. Spiegare brevemente il ruolo ed effetto di ogni riga del messaggio (scrivere la descrizione direttamente accanto al messaggio sopra).
2. Che tipo di messaggio è? A quale protocollo si riferisce?
3. Spiegare brevemente il funzionamento del protocollo e quali altri protocolli dello stack sono coinvolti

Es 2

Quale dei seguenti resource record non sono corretti? Motivare la risposta

- 1) <it nameserver.cnr.it NS>
- 2) <it nameserver.cnr.it A>
- 3) <it nameserver.cnr.it CNAME>
- 4) <it 151.100.27.38 NS>
- 5) <nameserver.cnr.it 151.100.27.38 A>

Es 3

Si consideri un host A che trasmette pacchetti, ognuno di lunghezza L bit, su un canale di trasmissione con Rate R Mbps verso un host B all'altro estremo del link. Si supponga L=4000 e R=10Mbps. Si supponga inoltre il ritardo di propagazione pari a 0,2 millisecondi.

- 1) Quanto impiega l'host A a trasmettere un pacchetto?
- 2) Qual è il massimo numero (L=4000) al secondo che possono essere trasmessi sul link?
- 3) Supponendo che l'host A invii i pacchetti uno dopo l'altro senza introdurre ritardi tra la trasmissione di un pacchetto e il successivo, quanto tempo impiega il router B a ricevere 4 pacchetti?
- 4) Qual è il massimo numero di bit che possono essere presenti sul canale?
- 5) Quando l'host A ha terminato di trasmettere un pacchetto, l'host B ha già ricevuto parte di esso?

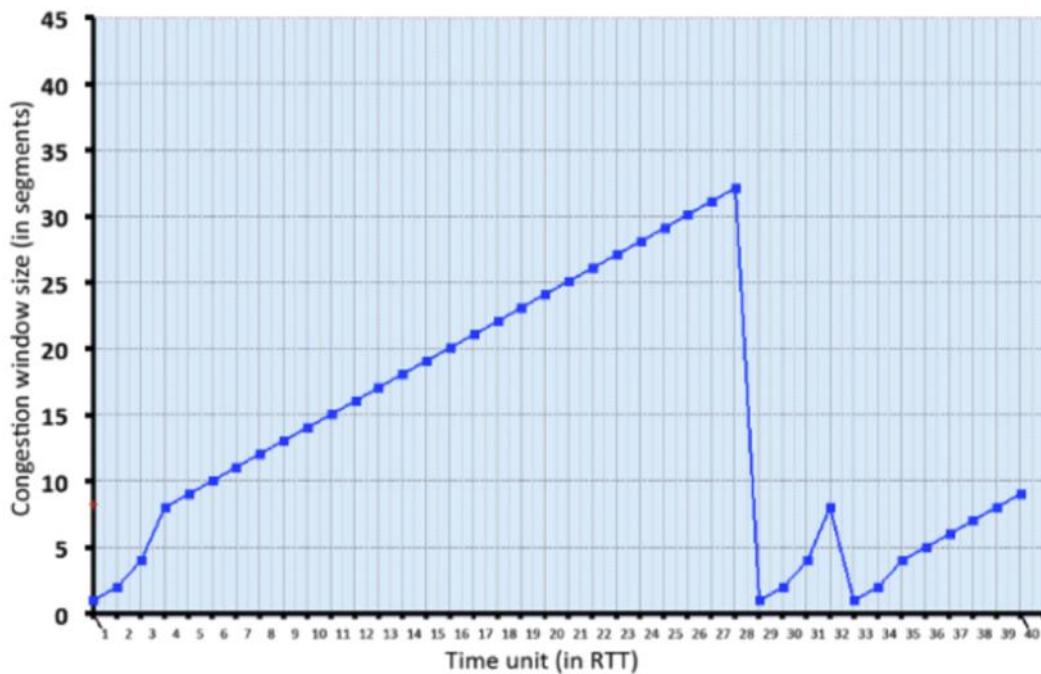
Es 4

Un host TCP A sta ricevendo un byte stream da B. Ha già ricevuto e riscontrato i byte fino al 4000 (numero di sequenza iniziale 1). Spiegare quali azioni esegue e che tipo di messaggi invia l'host TCP A in seguito ai seguenti eventi:

1. A riceve un segmento di 1000 byte con numero di sequenza pari a 3001.
2. In seguito all'evento 1. A riceve un segmento di 1000 byte con numero di sequenza pari a 6001.
3. In seguito all'evento 2, A riceve un segmento di 1000 byte con numero di sequenza pari a 5001.
4. In seguito all'evento 3, A riceve un segmento di 1000 byte con numero di sequenza 4001.

Es 5

Si consideri la figura seguente, che traccia l'evoluzione della finestra di congestione del TCP all'inizio di ogni unità di tempo (dove l'unità di tempo è uguale all'RTT). Il val. iniziale di cwnd è 1 e il val. iniziale di ssthresh è 8



- 1) Indicare le unità di tempo in cui il TCP è in slow start (es. 1, 5, 7)
- 2) Indicare le unità di tempo in cui il TCP è in congestion avoidance
- 3) Indicare le unità di tempo in cui la sstresht cambia
- 4) Quanti segmenti sono stati inviati in tutto all'unità di tempo 6?

Vero/Falso

Domande V/F. Ogni risposta corretta vale +1, ogni risposta sbagliata vale -1. La non risposta vale 0

- 1) Il protocollo UDP fornisce la funzionalità di flow control [V/F]
- 2) L'algoritmo di Slow Start incrementa la finestra di invio in modo lineare quando non ci sono perdite. [V/F]
- 3) Il throughput può essere maggiore del bitrate [V/F]
- 4) Il ritardo di processamento è il tempo che serve a un bit per attraversare il mezzo di trasmissione [V/F]
- 5) Il protocollo HTTP1.0 usa il protocollo di trasporto UDP [V/F]
- 6) L'algoritmo di Fast Recovery è usato da TCP Reno [V/F]
- 7) La switch table fornisce la mappatura tra IP address e MAC address [V/F]
- 8) Il protocollo CSMA/CA è implementato nelle LAN Wi-Fi [V/F]
- 9) Il problema della stazione esposta è più rilevante quando RTS/CTS è abilitato [V/F]
- 10) La firma digitale si basa sulla cifratura di un digest con la chiave privata del mittente [V/F]

Es 1

Si consideri un router A che trasmette pacchetti, ognuno di lunghezza L bit, su un canale di trasmissione con Rate R Mbps verso un router B all'altro estremo del link di lunghezza 50km. Si supponga L=2000 e R=10Mbps. Si supponga inoltre che la velocità della luce nel mezzo trasmisivo è di $2,5 \cdot 10^8$ m/s.

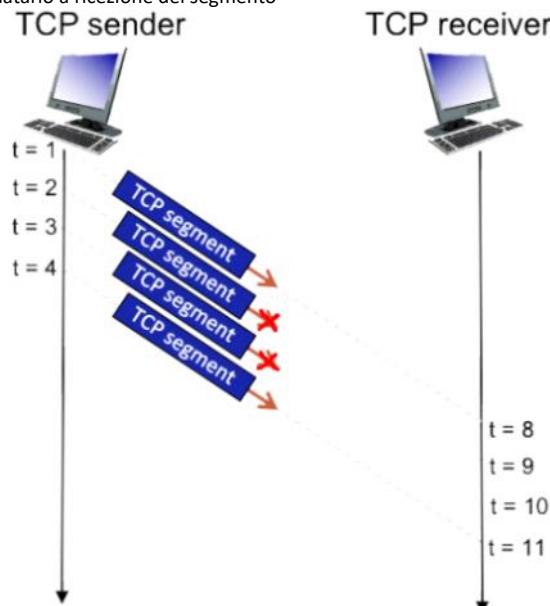
1. Quanto impiega il router A a completare la trasmissione di un pacchetto nel link?
2. Qual è il tempo impiegato dal router A a completare la trasmissione di 1 bit?
3. Supponendo che il router A invii i pacchetti uno dopo l'altro senza introdurre ritardi tra la trasmissione di un pacchetto e il successivo, quanto tempo impiega il router B a ricevere completamente 5 pacchetti (a partire dal tempo in cui il router A comincia a trasmettere)?
4. Qual è il massimo numero di pacchetti al secondo che possono essere trasmessi sul link?
5. Qual è il massimo numero di bit che possono essere presenti sul canale nello stesso istante (supponendo che non ci siano ritardi tra la trasmissione di un pacchetto e il successivo)?

Es 2

Si consideri la figura accanto in cui un mittente e un destinatario TCP comunicano su una connessione in cui i segmenti possono essere persi

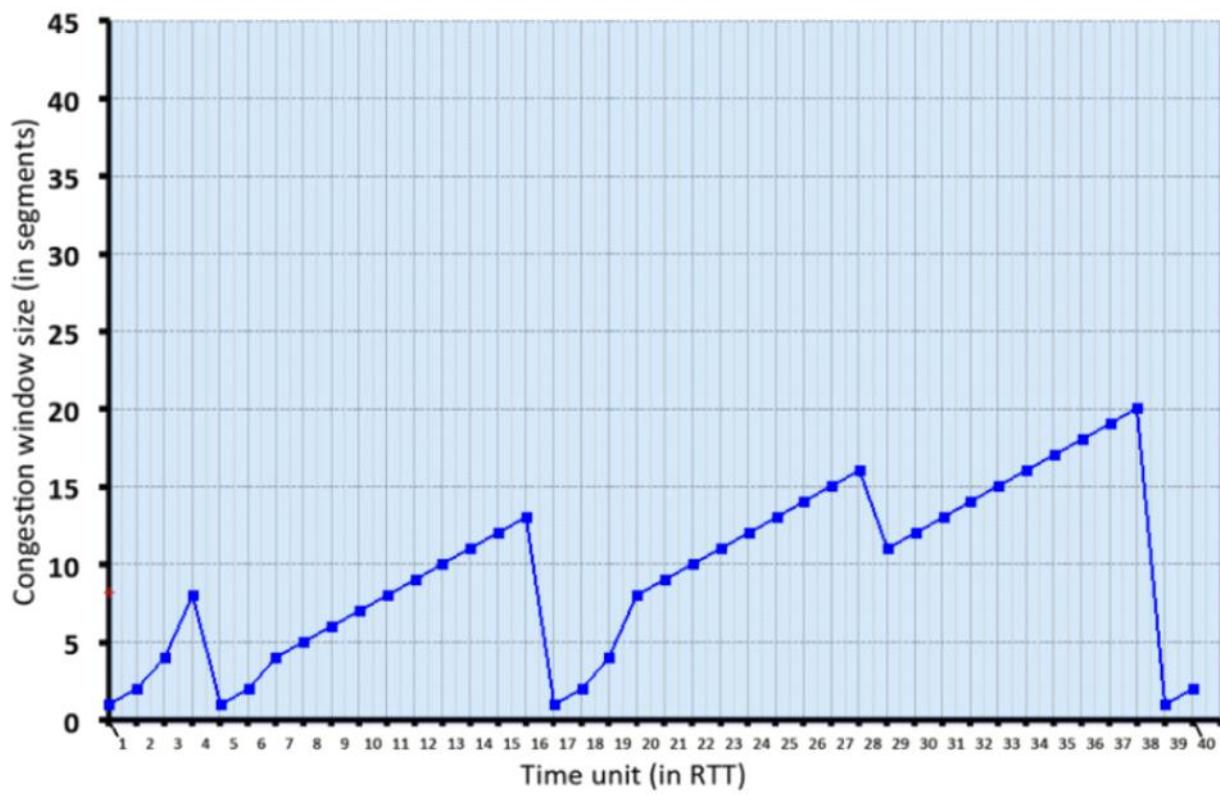
Supponiamo che il val. iniziale del numero di sequenza sia 23 e che i segmenti contengano ciascuno 235 byte. Il ritardo tra mittente e destinatario è di 7 unità di tempo, quindi il primo segmento arriva al destinatario a $t = 8$. Come mostrato in figura (con una X), 2 dei 4 segmenti sono persi tra mittente e destinatario

1. Per il primo segmento, indicare (a) il numero di sequenza del segmento inviato dal mittente e (b) l'eventuale ACK number del riscontro inviato dal destinatario a ricezione del segmento
2. Per il secondo segmento, indicare (a) il numero di sequenza del segmento inviato dal mittente e (b) l'eventuale ACK number del riscontro inviato dal destinatario a ricezione del segmento
3. Per il terzo segmento, indicare (a) il numero di sequenza del segmento inviato dal mittente e (b) l'eventuale ACK number del riscontro inviato dal destinatario a ricezione del segmento
4. Per il quarto segmento, indicare (a) il numero di sequenza del segmento inviato dal mittente e (b) l'eventuale ACK number del riscontro inviato dal destinatario a ricezione del segmento



Es 3

Si consideri la figura sotto che mostra l'evoluzione della finestra di congestione TCP (ogni unità di tempo corrisponde a un RTT, a partire da 1 a fino a 40). Il val. iniziale della finestra di congestione è 1 e il val. iniziale della ssthresh è 8. Notare che l'unità di misura è un RTT, quindi la finestra di congestione può cambiare più di una volta durante questa unità temporale



Sbarramento

Es 1

Nella codifica Pulse Code Modulation

- a. Il segnale vocale è campionato 8000 volte al secondo e ciascun campione è rappresentato con 8 bit
- b. Il segnale vocale è campionato 4000 volte al secondo e ciascun campione è rappresentato con 16 bit
- c. Il segnale vocale è campionato 54000 volte al secondo e ciascun campione è rappresentato con 8 bit

Es 2

Quali delle seguenti tecniche possono essere utilizzate per ridurre l'impatto degli errori di trasmissione su uno stream multimediale?

- a. Forward Error Correction
- b. Protocolli di ritrasmissione automatica (Automatic Repeat Request)
- c. Playout buffer
- d. Interleaving
- e. Ripetizione del pacchetto o interpolazione

Es 3

Quali delle seguenti affermazioni è vera riguardo allo sviluppo di protocolli di medium access control per una wireless LAN?

- a. Si può fare carrier sense
- b. Si può implementare collision detection
- c. Bisogna affrontare il problema dell'hidden terminal

Domande Aperte

Es 1

Si descrivano le varie classi di protocolli MAC, spiegando quali criteri desiderabili per un protocollo di medium access control sono soddisfatti.
Si descrivano e mettano a confronto i diversi protocolli MAC ad accesso casuale visti a lezione

Es 2

Si descriva il funzionamento del protocollo ARP (a cosa serve, dettagli del protocollo e logica di funzionamento):

Es 3

Si descriva a cosa servono, come sono organizzate e come funzionano le content distribution networks, discutendo le diverse possibili loro implementazioni nel dettaglio.

Esame 7/6/16 A-L

sabato 28 giugno 2025 11:03

Es 1

Nel protocollo MAC usato da Ethernet cosa succede quando si verifica una collisione?

- A) si aspetta un tempo scelto casualmente in un intervallo fisso e si ritrasmette.
- B) si sceglie casualmente un tempo in un intervallo con dimensione dipendente dal numero di ritrasmissione della trama e si ritrasmette.
- C) si lancia una moneta e con probabilità P si ritrasmette
- D) le collisioni non possono avvenire

Es 2

Quali tra i seguenti protocolli cerca di risolvere il problema dell'esaurimento degli indirizzi IP?

- A) IPv4
- B) NAT
- C) ICMP
- D) tutti e tre i protocolli

Es 3

Quale tra i seguenti protocolli di MEDIUM ACCESS CONTROL raggiunge una migliore utilizzazione del canale?

- A) ALOHA
- B) CSMA
- C) SLOTTED ALOHA
- D) Raggiungono tutte la stessa utilizzazione

Es 4

Per instradare un pacchetto tra Host X dell'Autonomous System C e l'Host Y dell'Autonomous System B, quale/quali protocollo di routing sono coinvolti?

- A) RIP o OSPF
- B) RIP o OSPF o IGRP all'interno di ciascuno dei due AS, OSPF sul "backbone" che interconnette gli Autonomous System.
- C) RIP o OSPF o IGRP all'interno di ciascuno dei due AS, BGP sulla tratta che interconnette gli Autonomous System.
- D) Nessuna delle tre

Es 6

Si descriva come funziona il protocollo di routing OSPF (algoritmo usato, organizzazione della rete, metriche usate per la scelta dei percorsi, caratteristiche e dettagli del protocollo).

Es 7

Discutere il problema dell'esaurimento degli indirizzi IP, descrivendo tutte le varie soluzioni che sono state proposte in questi anni per mitigare e/o risolvere questo problema.

Es 8

Si discutano le diverse architetture ed i protocolli per lo streaming di traffico multimediale.

Esonero 11/4/16 A-L

sabato 28 giugno 2025 11:03

Sbarramento

Es 1

Quali livelli dello stack protocollare TCP/IP sono implementati in un host

- a. Applicazione
- b. Rete, trasporto, applicazione
- c. Fisico, collegamento, rete, trasporto, applicazione

Es 2

Il controllo del flusso

- a. Serve per regolare la velocità di spedizione in una coppia mittente-destinatario
- b. Serve per regolare la congestione della rete
- c. Utilizza un campo nell'intestazione dei segmenti TCP

Es 3

In una rete, la dimensione della finestra di trasmissione è di 4 pacchetti e quella di ricezione è un pacchetto.

Quale dei seguenti meccanismi si sta utilizzando?

- a. Selective Repeat
- b. Stop and Wait
- c. Go back N

Es 4

BGP è un protocollo di routing di tipo:

- a. Inter-dominio
- b. Intra-dominio
- c. Distance-vector
- d. Path-vector
- e. Link-state

Domande Aperte

Es 1

Si descriva il funzionamento del protocollo DNS:

- a. In quale strato di rete opera
- b. Quale è il suo compito e da quali protocolli è usato
- c. Quali sono gli elementi di rete coinvolti nello scambio di messaggi e quale è il loro ruolo
- d. Come funziona in dettaglio il protocollo
- e. Quale è il formato dei messaggi scambiati
- f. Quali ottimizzazioni sono possibili per migliorare l'efficacia del protocollo

Es 2

Si descriva il funzionamento del protocollo RIP:

- a. In quale strato di rete opera e che funzionalità offre
- b. Quali sono gli algoritmi alla base del protocollo
- c. Quali sono gli elementi di rete coinvolti nello scambio di messaggi e quale è il loro ruolo
- d. Come funziona in dettaglio il protocollo
- e. Quale è il formato dei messaggi scambiati
- f. Quali sono i problemi che devono essere affrontati per la sua applicazione in sistemi reali e quali sono le soluzioni proposte/adottate per risolvere questi problemi

Es 3

Si descriva nel dettaglio il meccanismo di trasmissione affidabile in TCP.

- a. Come sono calcolati i sequence number dei segmenti;
- b. Si dettagli il protocollo di ritrasmissione automatica dell'informazione
- c. Si chiarisca come è calcolato il retransmission time-out

Vero/Falso

Domande vero/falso. Ogni risposta corretta vale +1, ogni risposta sbagliata vale -1. La non risposta vale 0

1. Il protocollo BGP usa l'algoritmo di Dijkstra per l'instradamento tra AS [V/F]
2. Il protocollo CSMA/CA è implementato nelle LAN Wi-Fi. [V/F]
3. Il protocollo CSMA/CD usa pacchetti di riscontro (ACK) [V/F]
4. Il protocollo CSMA/CA interrompe la trasmissione di un frame se un segnale di jam è ricevuto durante la trasmissione [V/F]
5. Se una switch table è vuota, lo switch effettuerà il flooding del prossimo pacchetto ricevuto su tutte le porte tranne quella dalla quale è stato ricevuto il pacchetto [V/F]
6. Il protocollo CDMA permette la comunicazione senza collisioni [V/F]
7. La tecnologia Bluetooth usa TDMA [V/F]
8. Il certificato digitale usa il sistema di crittografia a chiave simmetrica [V/F]
9. Lo switching fabric di un router usa il protocollo ARP [V/F]
10. Lo switch di rete ha un indirizzo MAC per porta [V/F]

Es 1

Descrivere le soluzioni adottate, in termini di tecnologie e protocolli, per ovviare alla scarsità di indirizzi IPv4, includendo una discussione delle soluzioni usate per gestire la transizione verso nuove tecnologie di indirizzamento mantentendo la compatibilità con parti della rete non aggiornate.

Es 2

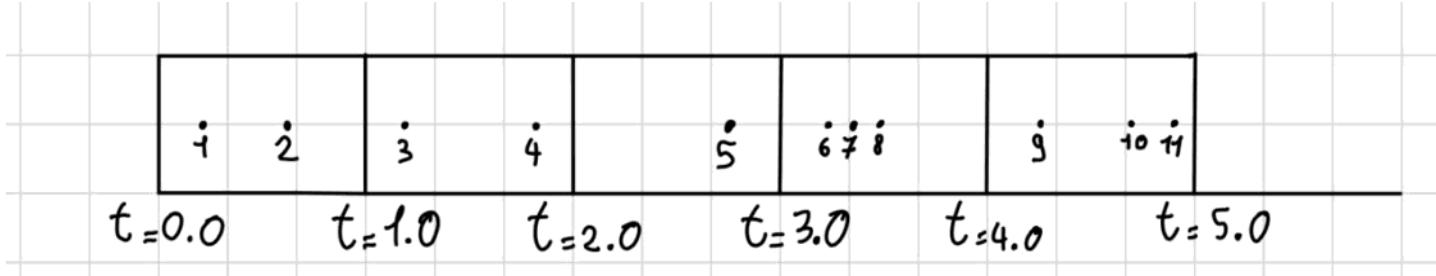
- 1) Disegnare una rete di 5 nodi e 5 archi e indicare i costi (scelti a piacere) sugli archi. Scrivere
 - o I vettori di distanza iniziali dei nodi (considerando un algoritmo distance vector)
 - o Il link state database (considerando un algoritmo link state)
- 2) Si supponga ora
 - o che tutti i nodi stiano implementando il protocollo CSMA/CD
 - o il tempo trascorso dall'inizio della trasmissione di un pacchetto fino all'inizio della ricezione presso altri nodi sia di 0,4 unità di tempo (pertanto, se un nodo inizia a trasmettere un pacchetto a $t=2.0$ e lo trasmette fino a $t=3.0$, allora ogni nodo che effettua il carrier sense nell'intervallo [2.4, 3.4] percepisce il canale come occupato),
 - o un nodo possa interrompere la trasmissione istantaneamente quando viene rilevata una collisione,
 - o per semplicità, si ignori ogni possibile ritrasmissione di pacchetti: se un pacchetto non viene inviato perché il canale viene rilevato occupato, non verrà trasmesso affatto.

Per ogni pacchetto, indicare se la trasmissione ha successo, e in tal caso indicare anche il tempo in cui inizia tale trasmissione.

Es 3

Si consideri la figura che mostra i tempi di arrivo (dal livello superiore) di pacchetti da spedire ai nodi di una rete broadcast (canale condiviso):
 $t = <0.2, 0.6, 1.2, 1.8, 2.8, 3.2, 3.3, 3.4, 4.3, 4.7, 4.9>$ (il separatore decimale è il punto in questa notazione)

Ogni trasmissione richiede esattamente un'unità di tempo.



- 1) Si supponga che tutti i nodi stiano implementando il protocollo Slotted Aloha.
 Per semplicità, si ignori ogni possibile ritrasmissione ($p=0$)
 - o Per ogni pacchetto, indicare il momento in cui inizia ogni trasmissione.
 - o Quali pacchetti vengono trasmessi con successo?

Es 4

Si supponga una rete wireless che implementa il protocollo CSMA/CA, in cui il nodo A vuole spedire un frame di lunghezza L al nodo B.

Mostrare (anche mediante rappresentazione grafica) lo scambio di pacchetti necessari affinché A possa spedire con successo il frame a B, in assenza di trasmissioni da altri nodi.

Specificare i diversi spazi di interframe.

Vero/Falso

Domande V/F parte I. Ogni risposta corretta vale +1, ogni risposta sbagliata vale -1. La non risposta vale 0

1. Il ritardo di trasmissione dipende dalla velocità della luce
2. UDP implementa un handshake a due vie invece che tre
3. Il valore del tempo di timeout dopo il quale considerare un pacchetto perso in TCP dipende dalla variabilità del round trip time
4. Il ritardo di propagazione è il tempo che impiega un dispositivo per far transitare un pacchetto dalla propria porta di input alla propria porta di output
5. La funzionalità di flow control adatta la finestra di invio del mittente per non sovraccaricare il destinatario

Domande V/F parte II (sia appello che esonero). Ogni risposta corretta vale +1, ogni risposta sbagliata vale -1.

La non risposta vale 0

1. NETCONF e YANG vengono usati per stabilire una connessione cifrata tra due entità
2. Il valore del campo Time To Live nell'intestazione di un datagramma IP cambia al passaggio attraverso un router
3. Il protocollo CSMA/CD prevede la possibilità di invio dei pacchetti RTS/CTS
4. Gli indirizzi MAC vengono assegnati dai router di rete
5. La firma digitale si basa sulla cifratura di un digest con la chiave privata del mittente

Es 1

Si consideri un router A che trasmette pacchetti, ognuno di lunghezza L bits, su un canale di trasmissione con Rate R Mbps verso un router B all'altro estremo del link di lunghezza 100km.

Si supponga $L = 4000$ e $R = 10\text{Mbps}$.

Si supponga inoltre che la velocità della luce nel mezzo trasmisivo è di $2,5 \times 10^8 \text{ m/s}$.

1. Quanto impiega il router A a completare la trasmissione di un pacchetto nel link?
2. Qual è il tempo impiegato dal router A a completare la trasmissione di 1 bit?
3. Supponendo che il router A invii i pacchetti uno dopo l'altro senza introdurre ritardi tra la trasmissione di un pacchetto e il successivo, quanto tempo impiega il router B a ricevere 4 pacchetti (a partire dal tempo in cui il router A comincia a trasmettere)?
4. Qual è il massimo numero di pacchetti al secondo che possono essere trasmessi sul link?
5. Qual è il massimo numero di bit che possono essere presenti sul canale?

Es 2

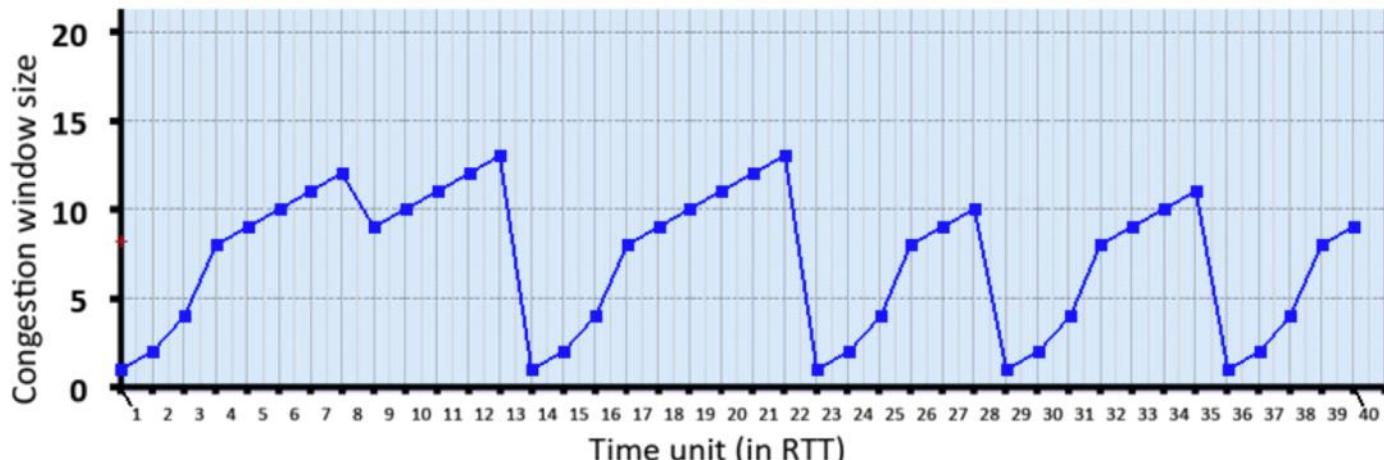
Un server TCP ha ricevuto e riscontrato (invia ACK) all'interno di una connessione i byte fino al 7000 (numero di sequenza iniziale 1).

Indicare quale azione esegue il server TCP in seguito ai seguenti eventi:

1. Il server riceve un segmento di 500 byte con numero di sequenza pari a 8001.
2. In seguito all'evento 1. Il server riceve un segmento di 500 byte con numero di sequenza pari a 7001.
3. In seguito all'evento 2, il server riceve un segmento di 500 byte con numero di sequenza pari a 8501.
4. In seguito all'evento 3 il server riceve un segmento di 500 byte con numero di sequenza 7501.

Es 4

Si consideri la figura sotto che mostra l'evoluzione della finestra di congestione TCP (ogni unità di tempo corrisponde a un RTT, a partire da 1 fino a 40). Il valore iniziale della finestra di congestione è 1 e il valore iniziale della ssthresh è 8.



1. In quali intervalli temporali il TCP è in slow start?
2. In quali intervalli temporali il TCP è in congestion avoidance?
3. In quali intervalli temporali il TCP è in fast recovery?
4. In quali unità di tempo avviene un timeout?
5. In quali unità di tempo si riceve il terzo ack duplicato?
6. In quali unità di tempo cambia la ssthresh?

Es 5 (sia appello che esonero)

1. Disegnare una rete di 5 nodi, non tutti direttamente connessi, e indicare i costi (scelti a piacere) sugli archi.
Scrivere:
 - o i vettori di distanza iniziali dei nodi (considerando un algoritmo distance vector)
 - o il link state database (considerando un algoritmo link state)
2. Spiegare la differenza tra le informazioni contenute in un vettore di distanza e nel link state database.
3. Nel caso dell'algoritmo distance vector, mostrare un esempio di aggiornamento del vettore di distanza iniziale di un nodo, quando riceve il vettore di distanza di un nodo vicino (usando la stessa rete del punto 1).
Spiegare brevemente la formula usata per eseguire l'aggiornamento.
4. Spiegare brevemente i concetti di split horizon e poison reverse.

Es 6 (sia appello che esonero)

Spiegare vantaggi e svantaggi di token ring

Vero/Falso

Domande V/F parte II (sia appello cheesonero). Ogni risposta corretta vale +1, ogni risposta sbagliata vale -1.

La non risposta vale 0

1. NETCONF e YANG vengono usati per stabilire una connessione cifrata tra due entità
2. Il valore del campo Time To Live nell'intestazione di un datagramma IP cambia al passaggio attraverso un router
3. Il protocollo CSMA/CD prevede la possibilità di invio dei pacchetti RTS/CTS
4. Gli indirizzi MAC vengono assegnati dai router di rete
5. La firma digitale si basa sulla cifratura di un digest con la chiave privata del mittente

Es 1 (sia appello cheesonero)

1. Disegnare una rete di 5 nodi, non tutti direttamente connessi, e indicare i costi (scelti a piacere) sugli archi.

Scrivere:

- o i vettori di distanza iniziali dei nodi (considerando un algoritmo distance vector)
- o il link state database (considerando un algoritmo link state)

2. Spiegare la differenza tra le informazioni contenute in un vettore di distanza e nel link state database.
3. Nel caso dell'algoritmo distance vector, mostrare un esempio di aggiornamento del vettore di distanza iniziale di un nodo, quando riceve il vettore di distanza di un nodo vicino (usando la stessa rete del punto 1). Spiegare brevemente la formula usata per eseguire l'aggiornamento.
4. Spiegare brevemente i concetti di split horizon e poison reverse.

Es 2 (sia appello cheesonero)

Spiegare vantaggi e svantaggi di token ring

Es 3

Spiegare come funziona traceroute, e come si possa evitare il tracciamento all'interno di una rete amministrata.

Es 4

Si consideri la figura che mostra i tempi di arrivo (dal livello superiore) di pacchetti da spedire ai nodi di una rete broadcast (canale condiviso):

$$t = \langle 0.6, 0.8, 0.9, 1.3, 1.4, 2.9, 3.5, 4.8 \rangle$$

(il separatore decimale è il punto in questa notazione)

Ogni trasmissione richiede esattamente un'unità di tempo. Ogni pacchetto viene inviato da un nodo diverso.

Si supponga che tutti i nodi stiano implementando il protocollo Pure Aloha.

Per semplicità, si ignori ogni possibile ritrasmissione di pacchetti ($p = 0$).

1. Per ogni pacchetto, indicare il momento in cui inizia ogni trasmissione.
2. Quali pacchetti vengono trasmessi con successo? Spiegare brevemente il motivo.

Es 5

In un sistema slotted Aloha, assumere che ci siano 2 nodi attivi, che intendano trasmettere un numero illimitato di frame di dimensione L bit.

Indicare l'efficienza del sistema se entrambi i nodi trasmettono con probabilità (parametro Aloha) $p = 0.33$, e spiegare brevemente come l'efficienza cambi al crescere del numero di nodi attivi (mantenendo p invariato).

Es 6

Tre host A,B,C vengono connessi per la prima volta a un router tramite uno switch (ogni host è connesso soltanto allo switch).

Si assume che gli indirizzi IP di tutti gli host sono noti a tutti i dispositivi della rete.

Descrivere come le switch table e ARP table (se presenti) dei dispositivi vengono riempite quando l'host A

contatta l'host B.

Prestazione 2025

venerdì 27 giugno 2025 19:48

Esercizio 1

Un file contiene 2 milioni di byte.

1. Quanto si impiega a trasmettere il file usando un canale a 56 kbps?
2. E usando un canale a 1Mbps?

Soluzione

2 milioni di byte = $2000000 * 8 = 16000000$ bits

- 1) 56 kbps = 56000 bps
Ritardo trasmissione = $\frac{16000000 \text{ bits}}{56000 \text{ bps}} = 289\text{s}$
- 2) 1 Mbps = 1000000 bps
Ritardo trasmissione = $\frac{16000000}{1000000} = 16\text{s}$

Esercizio 2

Si consideri un Host A che vuole inviare un file molto grande a un Host B. Il percorso tra A e B ha 3 link, con rate R1=500kbps, R2=2Mbps, R3=1Mbps.

- a) Assumendo l'assenza di ulteriore traffico nella rete, qual è il throughput per il file transfer?
- b) Si supponga che il file sia grande 4 milioni di byte. Dividendo la grandezza del file per il throughput, quanto impiegherebbe all'incirca trasferire il file all'host B?
- c) Ripetere le domande a) e b) con R2=100kbps

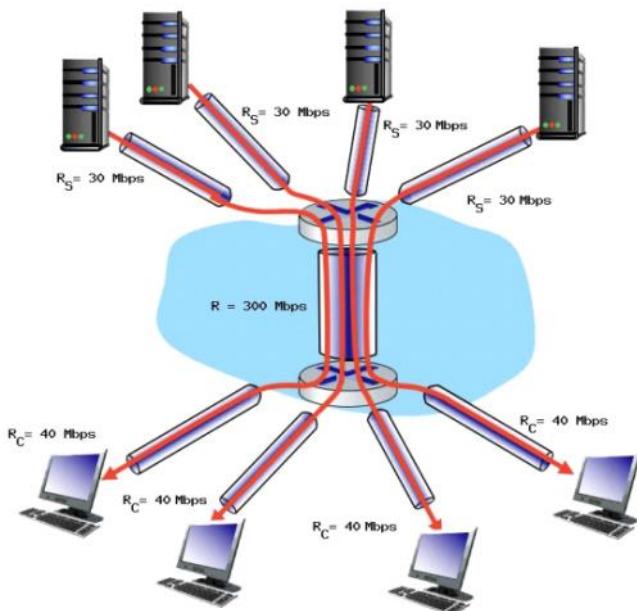
Soluzione

R1 = $5 * 10^5$, R2 = $2 * 10^6$ bps, R3 = 10^6 bps

- a) Il throughput è il minimo tra i link quindi R1
- b) 4 milioni di byte = $4000000 * 8 = 32000000$ bits, diviso per R1 = $\frac{32 * 10^6 \text{ b}}{5 * 10^5 \text{ bps}} = \frac{320b}{5\text{bps}} = 64\text{s}$
- c) R2 = 100kbps = 100000bps = 10^5 bps
 - a) Il throughput minimo ora è R2
 - b) $\frac{32 * 10^6 \text{ b}}{10^5 \text{ bps}} = 320\text{s}$

Esercizio 3

- 1) Quale è il throughput massimo end-to-end (in Mbps) per ciascuna delle quattro coppie client-server, assumendo che il collegamento centrale sia condiviso equamente (divide il rate in parti uguali)?
- 2) Quale collegamento è il collo di bottiglia? (Rs , Rs o R)
- 3) Supponendo che i server stiano inviando alla massima velocità possibile, qual è l'utilizzo dei link sui collegamenti dei server (RS)?
- 4) Supponendo che i server stiano inviando alla massima velocità possibile, qual è l'utilizzo dei link sui collegamenti dei client (RC)?
- 5) Supponendo che i server stiano inviando alla massima velocità possibile, qual è l'utilizzo del collegamento condiviso (R)?



Soluzione

Il throughput di R è diviso tra il num. di link a cui i router sono collegati quindi $R = \frac{300}{4} = 75$

- 1) Il throughput end-to-end è dato dal link con capacità minore quindi 30Mbps
- 2) Il collo di bottiglia è dovuto dal link minimo, quindi Rs (quello tra router e server)
- 3) L'utilizzo è $Rs/Rs = 30/30 = 1$
- 4) L'utilizzo è $R_{throughput}/Rc = 30/40 = 0.75$
- 5) L'utilizzo è $R_{throughput}/R = 30/75 = 0.4$

Esercizio 4

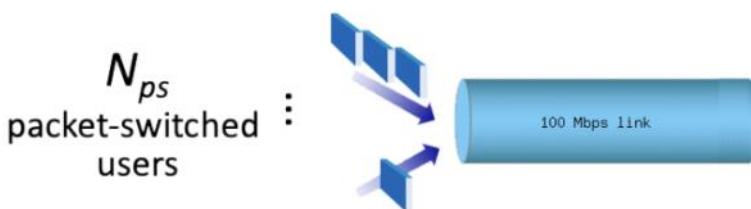
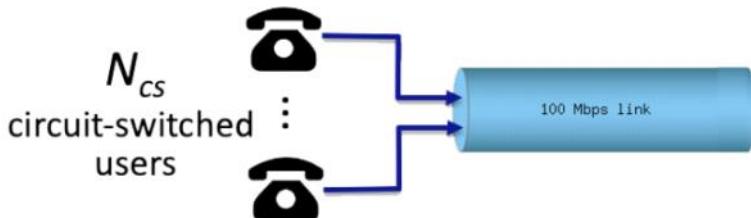
Si vuole inviare un file di 160000 bits dall'host A all'host B su una rete a commutazione di circuito. I link hanno rate pari a 1536 kbps e usano il TDM con 48 slot/sec. Il tempo per stabilire il circuito tra A e B è 500 ms.

Quanto impiega l'host A a trasmettere il file?

[Soluzione](#)

Es 5

- Comutazione di circuito con:
 - N_{CS} utenti
 - Ognuno necessita di $R = 25$ Mbps
 - Capacità collegamento 100 Mbps
 - Comutazione di pacchetto con:
 - N_{PS} utenti
 - Capacità collegamento 100Mbps
 - Ogni utente richiede ancora 25 Mbps durante la trasmissione, ma deve trasmettere solo il 30% del tempo
- 1) Quando viene usata la commutazione di circuito qual è il massimo numero di utenti che possono essere supportati?
 - 2) In caso di commutazione di pacchetto, qual è la probabilità che uno specifico utente stia trasmettendo e gli altri no?
 - 3) In caso di commutazione di pacchetto, qual è la probabilità che un utente (qualsiasi) stia trasmettendo e gli altri no?



Applicazione 2025

venerdì 27 giugno 2025 19:49

Es 1

Si vuole aggiungere un nuovo protocollo nel liv. applicazione: quali modifiche è necessario apportare agli altri livelli?

Es 2

Quando si dice che il liv. di trasporto effettua il multiplexing e il demultiplexing dei messaggi a liv. applicazione, si intende che il protocollo di liv. applicazione può combinare più messaggi del liv. applicazione in un pacchetto? Spiegare

Es 3

Spiegare il motivo per cui, nel contesto del paradigma client/server, il server debba essere permanentemente in esecuzione mentre il client possa essere eseguito solo quando necessario

Es 4

Un client FTP deve prelevare due file dal server e depositarvi un altro file: quante connessioni di controllo e quante connessioni di trasferimento dati sono necessarie?

Es 5

E' possibile per un server FTP ottenere l'elenco dei file o directory dal client?

Es 6

Quali tipi di resource record sono memorizzati in un server DNS radice? Dare un esempio

Es 7

Si consideri il seguente messaggio:

```
GET /kurose_ross_sandbox/interactive/quotation7.htm HTTP/1.1
Host: gaia.cs.umass.edu
Accept: text/plain, text/html, image/png, image/gif, audio/mp4, audio/mpeg, video/mp4, video/wmv,
Accept-Language: en-us, en-gb;q=0.8, en;q=0.2, fr, fr-ch, da, fi, ar, cs
If-Modified-Since: Wed, 10 Apr 2024 01:14:34 -0700
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.11 (KHTML, like Gecko) Chrome/17.0.963.56 Safari/535.11
1) A quale protocollo appartiene?
2) Che tipo di messaggio è?
3) Qual è il file richiesto?
4) Il client accetta immagini Jpeg?
5) Il client ha già una copia del file in cache?
```

Trasporto 2025

venerdì 27 giugno 2025 22:41

Es 1

In una rete con un valore fisso $m > 1$, è possibile utilizzare entrambi i meccanismi Go-Back-N e Selective-Repeat.

- 1) Indicare i vantaggi e gli svantaggi dell'impiego di ciascuno di essi.
- 2) Quali altre considerazioni si devono fare per decidere quale meccanismo utilizzare?

Es 2

Un server TCP ha ricevuto e riscontrato all'interno di una connessione i byte fino al 4000. Dire quale azione esegue il server dopo i seguenti eventi:

- 1) Il server riceve un segmento di 1000 byte con numero di sequenza pari a 3001
- 2) In seguito all'evento 1 il server riceve un segmento di 1000 byte con numero di sequenza pari a 6001
- 3) In seguito all'evento 2 il server riceve un segmento di 1000 byte con numero di sequenza pari a 5001
- 4) In seguito all'evento 3 il server riceve un segmento di 1000 byte con numero di sequenza pari a 4001

Es 3

Un server TCP ha ricevuto e riscontrato all'interno di una connessione i byte fino al 4000. Dire quale azione esegue il server dopo i seguenti eventi:

- 1) Il server riceve un segmento di 1000 byte con numero di sequenza pari a 5001 2
- 2) In seguito all'evento 1 il server riceve un segmento di 1000 byte con numero di sequenza pari a 4001
- 3) In seguito all'evento 2 il server riceve un segmento di 1000 byte con numero di sequenza pari a 6001
- 4) In seguito all'evento 3 il server riceve un segmento di 1000 byte con numero di sequenza pari a 7001

Es 4

Nel protocollo TCP la finestra di invio può essere più piccola, più grande o della stessa dimensione della finestra di ricezione?

Es 5

L'utente A utilizza il proprio browser per aprire due connessioni con il server HTTP in esecuzione sull'host B. Come può il protocollo TCP distinguere queste due connessioni?

Es 6

In una rete basata su Go-Back-N con $m=3$ e dimensione della finestra di invio uguale a 7, i valori delle variabili sono: $S_f = 62$, $S_n = 66$ e $R_n = 64$. Si ipotizzi che la rete non duplichi e non alteri l'ordine dei pacchetti.

- 1) Qual è il numero di sequenza dei pacchetti in transito?
- 2) Qual è il numero di riscontro dei pacchetti ack in transito?

Es 7

Si può definire il prodotto rate-ritardo come il numero di pacchetti che possono essere in transito nella rete durante un tempo pari a RTT. Calcolare il prodotto rate-ritardo nel caso in cui:

- Rate = 1Mbps
- RTT = 20ms
- Dimensione dei pacchetti = 1000 bit

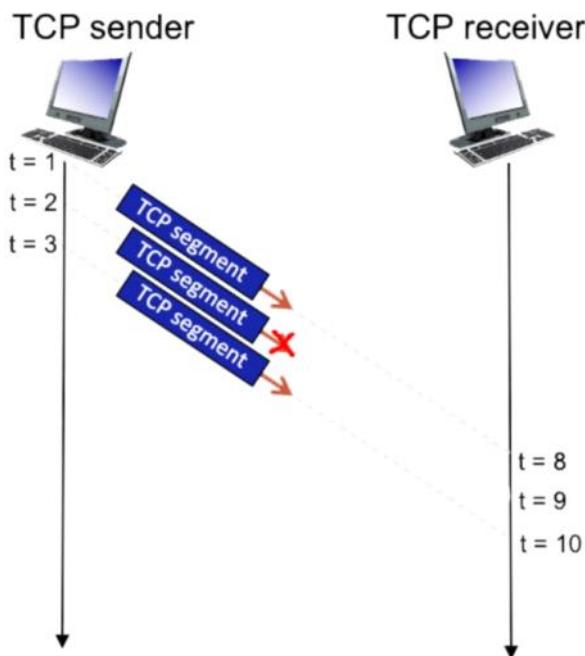
Es 8

- 1) Si deve progettare un protocollo a finestra scorrevole con meccanismo Selective-Repeat per una rete nella quale il rate=1Gbps e la distanza media fra mittente e destinatario è 5000km. La dimensione media dei pacchetti è di 50000 bit e la velocità di propagazione nel mezzo trasmissivo è 2×10^8 m/s.
- 2) Calcolare la dimensione massima delle finestre di invio e di ricezione, il numero di bit (m) nel campo numero di sequenza e un valore appropriato per il timeout del timer.

Es 9

Si consideri la figura, in cui un mittente e un destinatario TCP comunicano su una connessione in cui i segmenti inviati dal mittente al destinatario possono andare persi. Il mittente TCP invia una finestra iniziale di 3 segmenti. Si supponga che il valore iniziale del numero di sequenza dal mittente al destinatario sia 123 e che ciascuno dei primi 3 segmenti contenga 897 byte. Il ritardo tra il mittente e il destinatario è di 7 unità di tempo, quindi il primo segmento arriva al destinatario al tempo $t = 8$. Come mostrato nella, 1 dei 3 segmenti viene perso tra il mittente e il destinatario.

- 1) Indica i numeri di sequenza associati a ciascuno dei 3 segmenti inviati dal mittente.
- 2) Indica i numeri di ACK che il destinatario invia in risposta a ciascuno dei segmenti.



Esempio 10

Si supponga che i valori attuali stimati da TCP siano:

- EstimatedRTT = 230 millisecondi
- DevRTT = 40 millisecondi

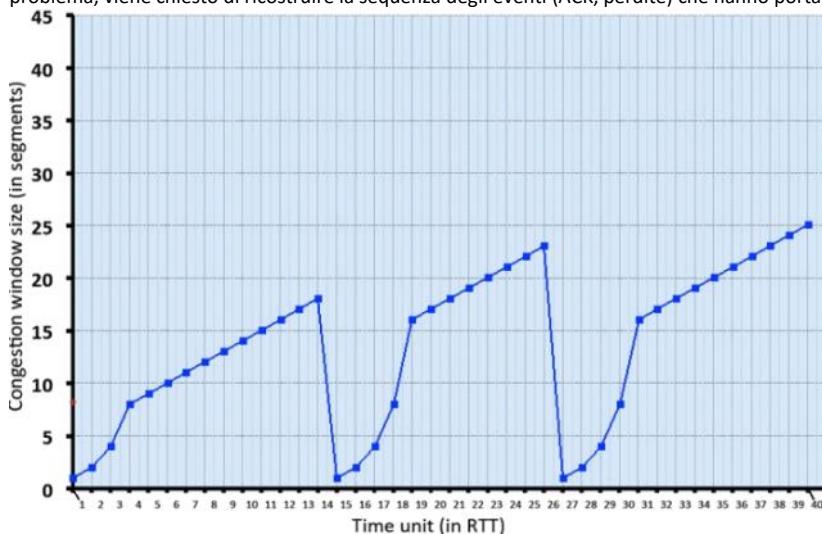
I successivi tre valori misurati del round-trip time (RTT) sono:

1. 220 ms
2. 220 ms
3. 390 ms

Per ciascuno dei tre RTT misurati, calcolare: Il nuovo valore di EstimatedRTT Il nuovo Timeout TCP considerando $\alpha=0.125$ e $\beta=0.25$

Esempio 11

Si consideri la figura che rappresenta l'evoluzione della finestra di congestione di TCP all'inizio di ogni unità di tempo (dove l'unità di tempo è uguale al RTT). TCP invia un «gruppo» di pacchetti di dimensioni cwnd all'inizio di ogni unità di tempo. Il risultato dell'invio di quel gruppo di pacchetti è che o tutti i pacchetti vengono confermati alla fine dell'unità di tempo, oppure c'è un timeout per il primo pacchetto, oppure c'è un triplo ACK duplicato per il primo pacchetto. In questo problema, viene chiesto di ricostruire la sequenza degli eventi (ACK, perdite) che hanno portato all'evoluzione di cwnd di TCP mostrata di seguito.



Rete 2025

venerdì 27 giugno 2025 22:50

Es 1

Spiegare la differenza tra routing e forwarding

Es 2

Ognuno dei seguenti indirizzi appartiene a un blocco. Trovare il primo e l'ultimo indirizzo di ogni blocco.

14.12.72.8/24

200.107.16.17/18

70.110.19.17/16

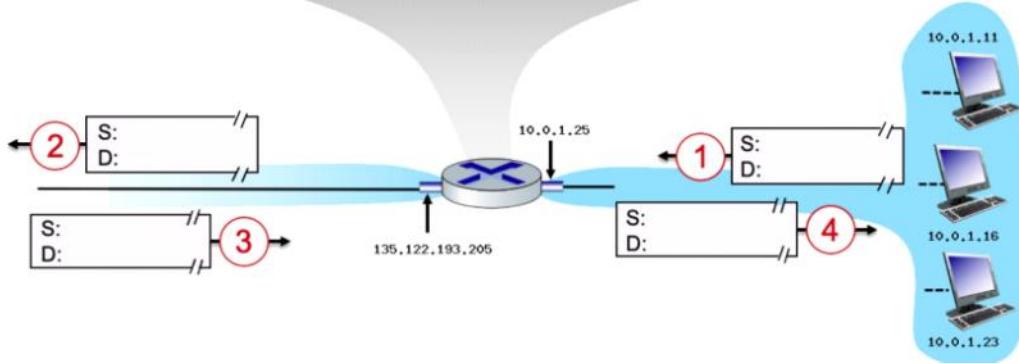
Es 3

Considerare lo scenario seguente in cui tre host, con indirizzi IP privati 10.0.1.11, 10.0.1.16, 10.0.1.23, si trovano in una rete locale dietro un router NAT che si trova tra questi tre host e la rete Internet più ampia. L'interfaccia del router sul lato LAN ha indirizzo IP 10.0.1.25, mentre l'indirizzo del router sul lato Internet è 135.122.193.205.

Si supponga che l'host con indirizzo IP 10.0.1.16 invii un datagramma IP destinato all'host 128.119.160.188. La porta sorgente è 3362 e la porta di destinazione è 80.

- 1) Considera il datagramma al passo 1, dopo che è stato inviato dall'host ma prima che abbia raggiunto il router. Qual è l'indirizzo IP sorgente per questo datagramma?
- 2) Al passo 1, qual è l'indirizzo IP di destinazione?
- 3) Ora considera il datagramma al passo 2, dopo che è stato trasmesso dal router. Qual è l'indirizzo IP sorgente per questo datagramma?
- 4) Al passo 2, qual è l'indirizzo IP di destinazione per questo datagramma?
- 5) La porta sorgente sarà cambiata? Sì o No.
- 6) Ora considera il datagramma al passo 3, appena prima che venga ricevuto dal router. Qual è l'indirizzo IP sorgente per questo datagramma?
- 7) Al passo 3, qual è l'indirizzo IP di destinazione per questo datagramma?
- 8) Infine, considera il datagramma al passo 4, dopo che è stato trasmesso dal router ma prima che sia stato ricevuto dall'host. Qual è l'indirizzo IP sorgente per questo datagramma?
- 9) Al passo 4, qual è l'indirizzo IP di destinazione per questo datagramma?

NAT translation table	
WAN side addr	LAN side addr
135.122.193.205	10.0.1.16



Es 4

Si router A invia due messaggi RIP ai due router vicini (uno a B e uno a C). I due datagrammi che incapsulano i messaggi RIP hanno lo stesso indirizzo IP sorgente? Hanno lo stesso indirizzo IP destinazione?

Es 5

Si assuma che la distanza minima tra i nodi a,b,c,d e il nodo y e i costi tra x e a,b,c,d siano: Day=5 Dby=6 Dcy=4 Ddy=3

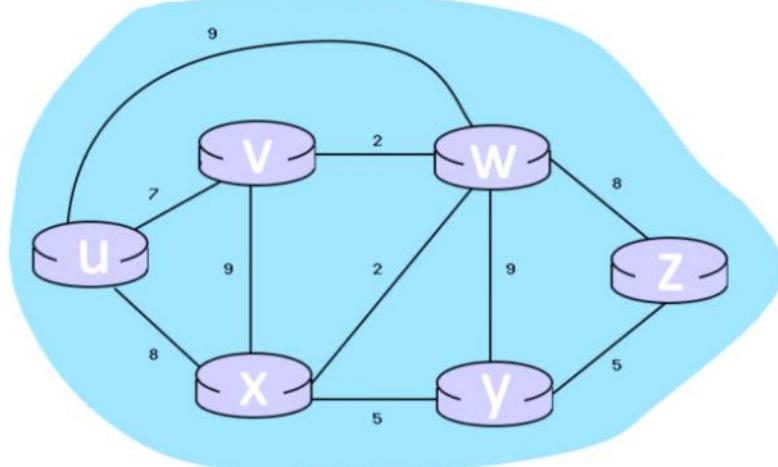
Cxa=2 Cxb=1 Cxc=3 Cxd=1

Qual è la distanza più breve tra il nodo x e il nodo y (Dxy) secondo l'equazione di Bellman-Ford?

Es 6

Si consideri la seguente rete e si applichi l'algoritmo di Dijkstra, trovare i percorsi a costo minimo a partire dal nodo U

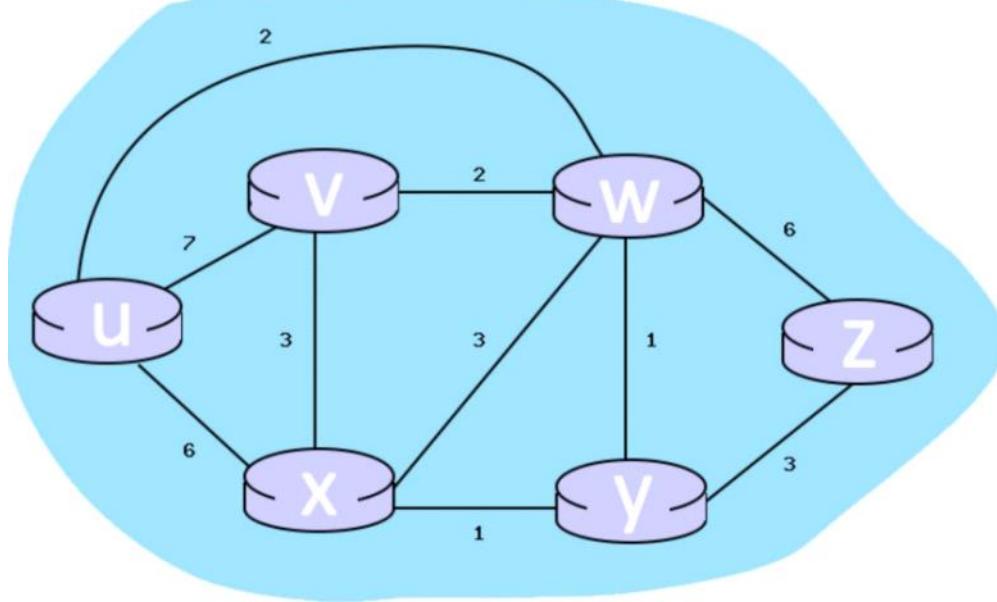
- 1) Qual è la distanza più breve verso il nodo v e quale nodo è il predecessore?
- 2) Qual è la distanza più breve verso il nodo x e quale nodo è il predecessore?
- 3) Qual è la distanza più breve verso il nodo u e quale nodo è il predecessore?



Es 7

Si consideri la seguente rete e si applichi l'algoritmo di Dijkstra, trovare i percorsi a costo minimo a partire dal nodo U

- 1) Qual è la distanza più breve verso il nodo w e quale nodo è il predecessore?
- 2) Qual è la distanza più breve verso il nodo x e quale nodo è il predecessore?
- 3) Qual è la distanza più breve verso il nodo z e quale nodo è il predecessore?



Link 1 2025

venerdì 27 giugno 2025 18:01

Es 1

Le stazioni di una rete Aloha puro inviano frame da 1000 bit su un canale con rate pari a 1Mbps
A quanto equivale il tempo di vulnerabilità per tale rete?

Es 2

Quale tra uno switch e un router ha più overhead? spiegare

Es 3

Considerando la formula di backoff esponenziale, trovare la probabilità che una stazione possa trasmettere immediatamente nei seguenti casi:

- 1) Dopo un fallimento
- 2) Dopo tre fallimenti

Es 4

Due host in due reti diverse possono avere lo stesso indir. di liv. collegamento?

Es 5

Quattro stazioni sono collegate a un hub in una rete Ethernet. Le distanze tra l'hub e le stazioni sono rispettivamente di 300m, 400m, 500m e 700m. Qual' è la lunghezza di questa rete quando dobbiamo calcolare T_p ?

Link 2 2025

venerdì 27 giugno 2025 18:10

Esercizio 1

Qual è l'effetto massimo (in numero di bit) di un rumore di 2ms sui dati trasmessi alle seguenti velocità:

- a) 1500bps
- b) 12 Kbps
- c) 100 Kbps
- d) 100Mbps

Esercizio 2

Ci sono solo 3 stazioni attive in una rete Slotted ALOHA: A, B, C. Dato uno slot di tempo ogni stazione genera un frame rispettivamente con probabilità: $PA=0.2$, $PB=0.3$, $PC=0.4$

- a) Qual è il throughput di ogni stazione?
- b) Qual è il throughput della rete?

Esercizio 3

In una rete a bus CSMA/CD, con una velocità di trasmissione di 10Mbps, avviene una collisione 20 μs dopo che il primo bit del frame la lasciato la stazione mittente. Quale dovrebbe essere la lunghezza del frame in modo che il mittente sia in grado di rilevare la collisione?

Esercizio 4

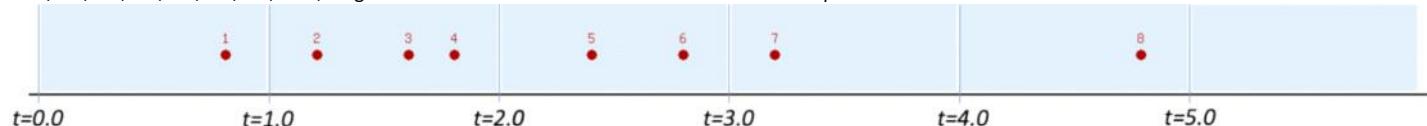
Si consideri una rete Aloha puro con rate R pari a 10Mbps e dimensione del frame di 1000 bit. Qual è il numero di frame al secondo che questa rete può trasportare con successo?

Aloha e CSMA

sabato 28 giugno 2025 10:50

Es 1

Considera la figura sottostante, che mostra l'arrivo di 8 pacchetti per la trasmissione presso diversi nodi wireless a accesso multiplo, ai seguenti istanti di tempo: $t = \langle 0.8, 1.2, 1.6, 1.8, 2.4, 2.8, 3.2, 4.8 \rangle$, e ogni trasmissione richiede esattamente un'unità di tempo.



Si supponga che tutti i nodi stiano utilizzando il protocollo Aloha

- 1) Per ciascun pacchetto, indicare l'istante in cui inizia la trasmissione.
- 2) Quale pacchetto viene trasmesso con successo?

Si supponga che tutti i nodi stiano utilizzando il protocollo Slotted Aloha

- 1) Per ciascun pacchetto, indicare l'istante in cui inizia la trasmissione
- 2) Quali pacchetti vengono trasmessi con successo?

Supponi che tutti i nodi stiano utilizzando il protocollo CSMA (Carrier Sense Multiple Access), ma senza rilevamento delle collisioni. Si supponga inoltre che il tempo che intercorre tra l'inizio della trasmissione di un messaggio e l'inizio della sua ricezione da parte degli altri nodi sia di 0,4 unità di tempo.

(Ciò significa che, se un nodo inizia a trasmettere un messaggio a $t = 2.0$ e lo trasmette fino a $t = 3.0$, allora qualsiasi nodo che effettui il carrier sensing nell'intervallo $[2.4, 3.4]$ percepisce il canale come occupato.)

- 1) Per ciascun messaggio, indica l'istante in cui inizia la trasmissione, oppure indica che la trasmissione non avviene perché il nodo rileva il canale occupato al momento dell'arrivo del messaggio.
- 2) Quali messaggi vengono trasmessi con successo?

Si supponga che tutti i nodi stiano utilizzando il protocollo Carrier Sense Multiple Access con rilevamento della collisione (CSMA/CD). Si supponga che il tempo tra l'inizio della trasmissione di un messaggio e il momento in cui viene percepito dagli altri nodi sia di 0,4 unità di tempo, e che un nodo possa interrompere instantaneamente la trasmissione non appena viene rilevata una collisione.

(Ciò significa che, se un nodo inizia a trasmettere un messaggio a $t = 2.0$ e lo trasmette fino a $t = 3.0$, qualsiasi nodo che effettua il carrier sensing nell'intervallo $[2.4, 3.4]$ rileverà il canale come occupato.)

- 1) Per ciascun messaggio, indicare l'istante in cui inizia la trasmissione, oppure indicare che la trasmissione non inizia perché il canale viene percepito come occupato all'arrivo del messaggio. Se il canale è occupato, scrivere "s" al posto dell'orario.
- 2) Quali messaggi sono stati trasmessi con successo?
- 3) A che orario ogni messaggio ha interrotto la trasmissione a causa di una collisione?

Es 2

Ripetere l'esercizio su tutti e 4 i casi: Aloha, Slotted Aloha, CSMA, CSMA/CD con i seguenti tempi di arrivo dei pacchetti: $t = \langle 0.1, 0.9, 1.1, 1.8, 1.9, 2.4, 2.6, 2.8, 3.4, 4.1, 4.6 \rangle$