

# Come Possiamo DEFINIRE Internet?

1 modo  
2 modo

È una rete di commutatori che interconnette miliardi di dispositivi di calcolo. Tutti i nostri dispositivi sono detti: "host" o "sistemi periferici" e sono connessi tra loro. Tramite COMMUNICATION LINK PACKET SWITCH (commutatori) e i principali:  
 ↓  
 Route2 e link-layer switch  
 e usati nelle zeti usati nelle zeti  
 nucleo della rete di accesso

**Velocità di Trasmissione:** velocità di trasmissione dati misurate in bps (bit/secondo)

I sistemi periferici accedono a Internet tramite **Internet service provider (ISP)** che comprendono compagnie telefoniche, reti aziendali, accesso alla rete, rendono disponibile l'accesso a Internet.  
**Provider:** insieme di commutatori di pacchetti e di collegamenti.

Ciascuna rete ISP, sia di alto livello (come quelle connesse con fibra ottica, che di basso livello, sono gestite in modo indipendente, fanno uso del protocollo IP e forniscono ai sistemi periferici servizi). Tipi di accesso alla rete: Tra cui quello residenziale a larga banda come la DSL, quello locale o wireless.

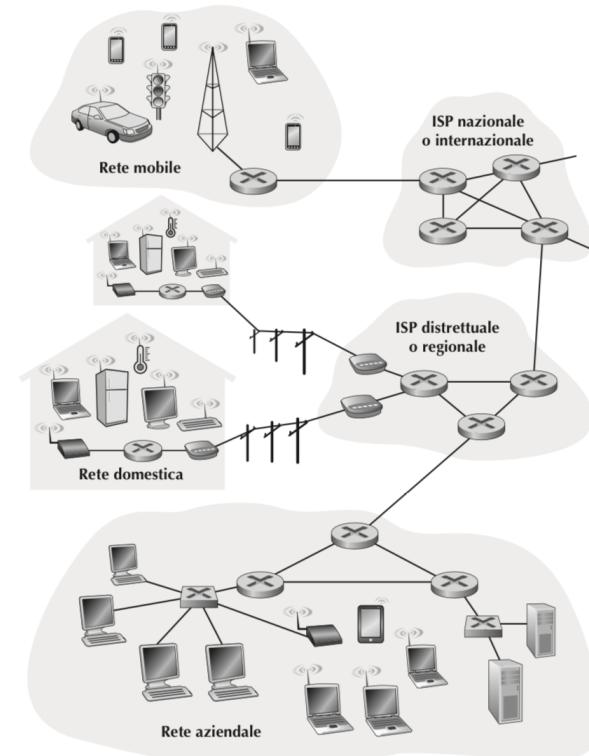
Sistemi periferici commutatori di pacchetto e altro fanno uso di uno stack protocolare che controlla l'invio e la ricezione di informazioni all'interno della rete.

Essi sono raggruppati in un protocollo chiamato TCP/IP. Che si compone dei due principali protocolli di Internet cioè TCP e IP (si occupa di specificare il formato dei pacchetti scambiati tra router e sistemi periferici).

Infrastruttura che fornisce servizi alle applicazioni delle distribuite perché coinvolgono più sistemi periferici che si scambiano dati.

I sistemi periferici collegati a Internet forniscono un'interfaccia UN INSERIMENTO DI REGOLE DI che specifica come un programma chiede a Internet di recapitare dati a un programma su un altro sistema periferico.

**RFC:** request for comment definiscono i vari protocolli. Ne esistono 7000+  
WWW, HTTP, TCP, IP, SMTP



Un sistema di comunicazione è composto da due parti:

① Un mezzo fisico; → MI SERVE PER SCAMBIRE i SEGNALI

② Una struttura logica (software). → ASSOCIA AD UN SEGNALE UN SIGNIFICATO

de informazioni vengono veicolate attraverso un PROTOCOLLO: il modo di seguire un'azione rispettando certi vincoli, una serie di operazioni per soddisfare una richiesta.

### ✓ Protocollo umano di comunicazione

Permette a componenti hardware e software di scambiarsi messaggi

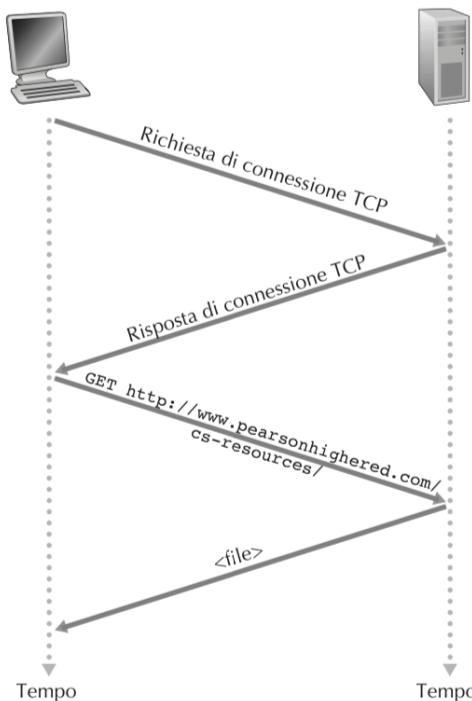
Qualsiasi attività su Internet è gestita da un protocollo



i protocolli cablati nelle schede di rete di due calcolatori fisicamente connessi controllano il flusso di bit sul "cavo" tra le due schede.

I protocolli nei router determinano un percorso per il pacchetto dalla sorgente alla destinazione

Cosa succede quando si invia una richiesta ad un web server?



Viene usato il protocollo HTTP e un protocollo senza stato cioè una volta conclusa l'ultima operazione e avviata un'altra richiesta avrà una sessione (?) indipendente da quella precedente.

→ Il calcolatore invia un messaggio di richiesta al server web attendendo una risposta

→ Il server web risponde

→ Il nostro calcolatore invia una richiesta Get con la pagina desiderata

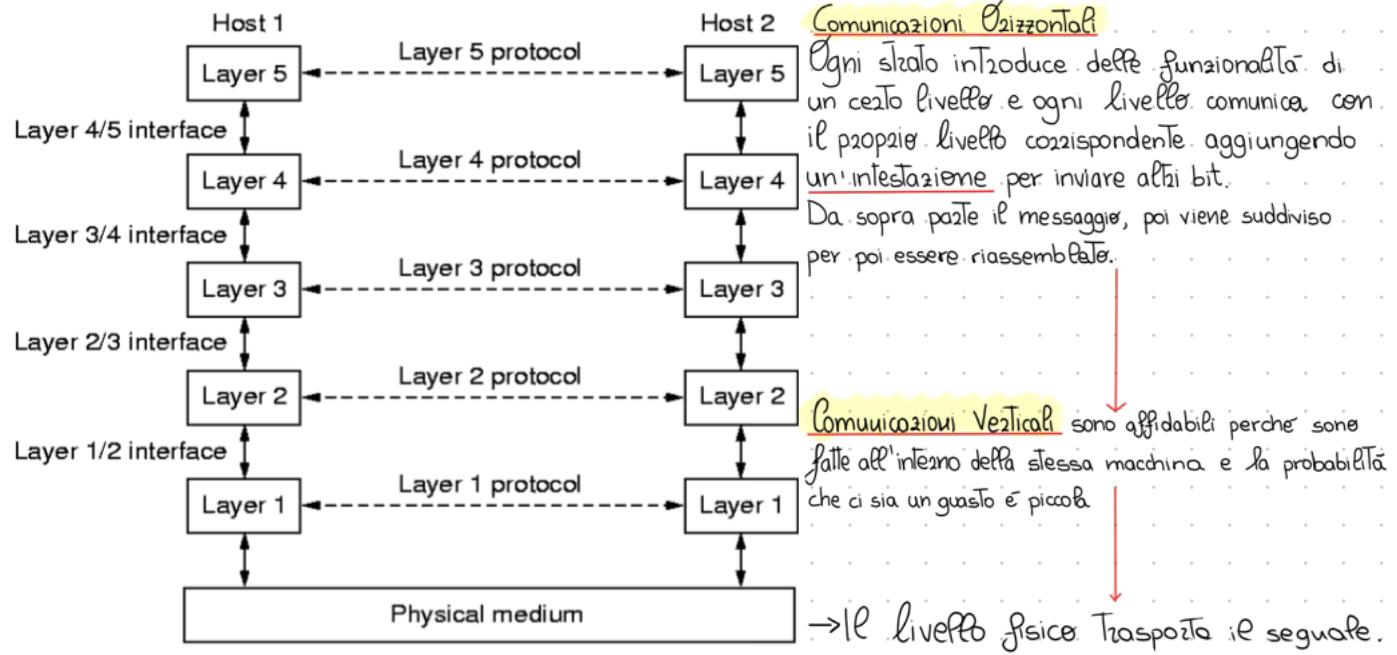
→ Il server web restituisce la pagina web

- Se si verificasse un'immediata chiusura della connessione TCP significa che non abbiamo un protocollo

• Un protocollo non è infallibile, per verificarne la correttezza si usano i fuzzer

• Il protocollo TCP deve garantire una distribuzione uniforme della banda, seguendo una proprietà fairness che evita il problema della congestione TCP (eccesso di domanda rispetto alle capacità massime della rete).

TAHOE ↘ RENO ↗

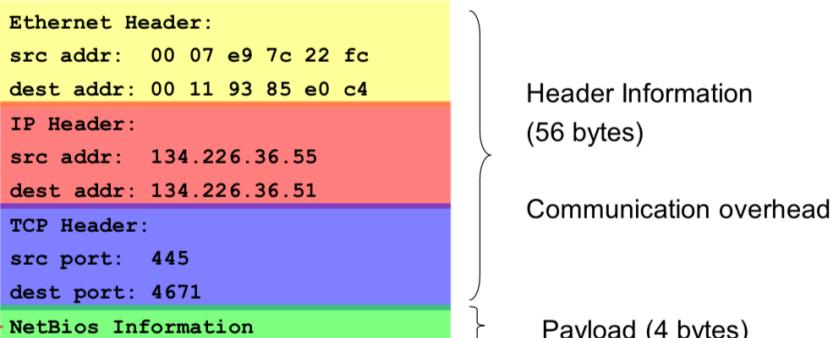


Università  
Catania

## Protocol Headers

Intestazione  
riguarda le informazioni  
tra mittente e destinatario

0000	00 07 e9 7c 22 fc 00 11 93 85 e0 c4	08 00 45 00	...   ".....E.
0010	00 2c db 26 40 00 3f 06 0e 77 86 e2 20 37 86 e2		.., &@.?..w... 7...
0020	24 33 01 bd 12 3f 3d fa 0f b6 a8 6f 87 c0 50 18		\$3...?=....o..P.
0030	bc 40 8a 7c 00 00 85 00 00 00 00 00 00 00 00 00		.@.   .....



Se il Payload è  
molto piccolo ho uno spreco  
di spazio

Payload è il messaggio

- Gli utenti (le applicazioni) vogliono un canale di comunicazione affidabile e privo di errori
- Il canale virtuale è implementato utilizzando canali fisici

SIMPLEX

Telecomande  
in quanto da solo non  
ha alcun valore perché  
abbiamo bisogno di un  
feedback.

HALF-DUPLEX

Un dispositivo parla e  
un dispositivo ascolta.  
Tramite il protocollo  
indichiamo quando il  
uno ha finito e l'altro  
ha iniziato.

FULL-DUPLEX

entrambe le  
direzioni  
ESEMPIO (?)

- I messaggi a basso livello, a livello applicativo non possono essere di lunghezza arbitraria, quindi non prevediamo una dimensione massima, ma visto che non posso inserire +∞ inserisco un numero molto grande e man mano che scendo devo frammentare e prevedere un numero m di bit. di spazio per la frammentazione è un costo per l'utente.
- Un Trasmettitore veloce non deve sommergere un ricevitore lento, serve un protocollo che garantisca la ricezione dei dati così che il sistema operativo scriva i vari frame sul disco senza perderne alcuno, può aumentare la latenza.
- Determinare il percorso migliore per arrivare a destinazione
- L'ordine di arrivo dei messaggi deve essere uguale a quello di spedizione (possiamo aggiungere un identificativo)

de comunicazioni sono di due tipi

→ **Connectionless** (comunicazione senza connessione)

Spedisco la lettera ma mi serve avere un feedback.

È un metodo di trasmissione dati in cui ogni pacchetto trasporta informazioni nell'intestazione e due l'indirizzo di destinazione è sufficiente per permettere una spedizione indipendente dae pacchetto.

Vantaggio: basso sovraccarico di lavoro. Permette operazioni di broadcast e multicast

salvano più di una risorsa in rete quando c'è bisogno di

trasmettere gli stessi dati a diversi destinatari

UDP

→ **Affidabile** se messaggio arriva sicuramente a destinazione.

→ **Non affidabile**, i canali non sono affidabili per destinazione, ma l'utente li vuole affidabili

→ **Connection-oriented**. Tramite quale i dispositivi terminali usano un protocollo di comunicazione per stabilire una connessione logica end-to-end tra gli agenti della comunicazione prima della trasmissione di qualsiasi tipo di dati.

In questo caso di Apertura, comunicazione e chiusura

La chiusura non ha ancora una soluzione esatta: Certo numero di invii di ACK.

TCP

	Service	Example
Connection-oriented	Reliable message stream	Sequence of pages
	Reliable byte stream	Remote login
Connection-less	Unreliable connection	Digitized voice
	Unreliable datagram	Electronic junk mail
	Acknowledged datagram	Registered mail
	Request-reply	Database query

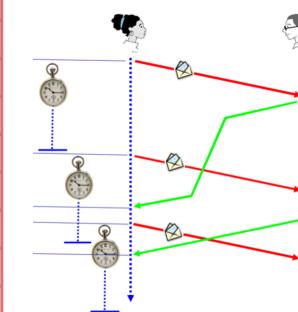
### Utilizzo del Timing

Troppo basso

Se non ricevo nulla alla fine del timing siamo in comunicazione o rispondo al messaggio.

Affidabilità

Posso usare un ACK. Un segnale di riconoscimento emesso in risposta in ricezione di un'informazione completa.



## Esempi:

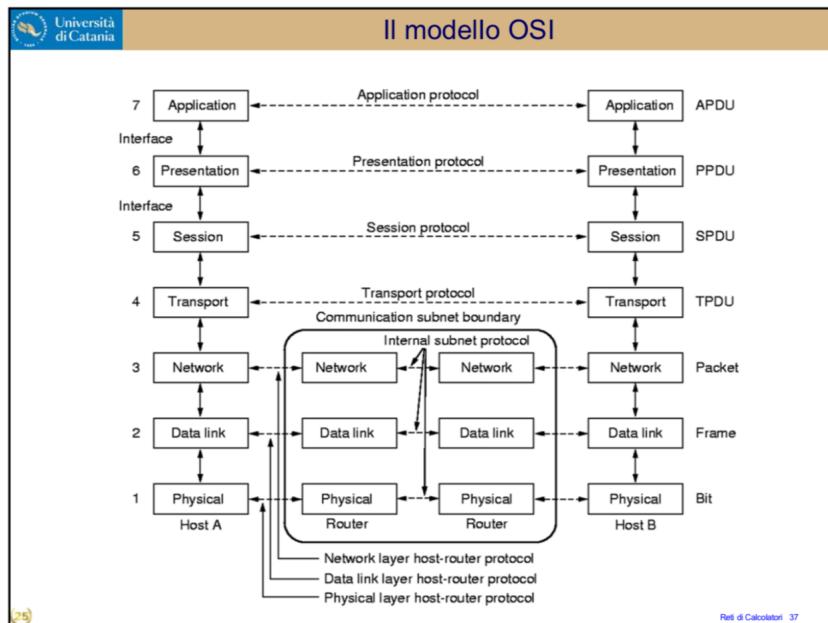
- trasferire un messaggio con connessione affidabile e sicura.
- transmissione di video non affidabile e connectionless.
- Database connection-less in quanto effettua operazioni atomiche



tcp / ip sono stack - protocoli

## IL MODELLO OSI

↓  
È SOLO  
TEORICO



È organizzato in sette livelli:

- 1) Applicazione
- 2) Presentazione cioè fornire servizi che consentono ad applicazioni che vogliono comunicare di interpretare il significato dei dati scambiati.  
Questi servizi comprendono:  
Per esempio la compressione e la cifratura dei dati
- 3) Sessione: fornisce la delimitazione e la sincronizzazione dello scambio di dati, compresi i mezzi per costruire uno schema di controllo e di recupero degli stessi.
- 4) Trasporto
- 5) Rete
- 6) Collegamento
- 7) Fisico



{ Parlare dei protocolli Internet o parlare del modello  
tcp / ip sono le stesse cose}

I protocolli dei vari livelli sono detti "PILA DI PROTOCOLLI" o "MODELLO TCP\IP" consiste in **cinque livelli**



1) È la sede delle applicazioni di rete e dei relativi protocolli quali

HTTP

consente la richiesta e il trasferimento dei documenti web.

SMTP

consente il trasferimento dei messaggi di posta elettronica.

FTP

che consente il trasferimento di file tra due sistemi remoti.

Si occupa anche di tradurre in rete nomi host (www.ciaer.it) in indirizzi IP a 32 bit tramite DNS (Domain name system) → **Protocollo a livello applicativo**

2) Trasferisce i messaggi del livello di applicazione tra punti periferici gestiti dalle applicazioni. I due protocolli di Trasporto sono:

TCP

TCP fornisce alle applicazioni un servizio orientato alla connessione, che include la consegna garantita dei messaggi a livello di applicazione alla destinazione e il controllo di flusso. Inoltre fraziona i messaggi lunghi e fornisce un meccanismo di controllo della congestione così che una sorgente regola la propria velocità trasmissiva quando la rete è congestionata.

UDP

fornisce alle proprie applicazioni un servizio non orientato alla connessione che è davvero un servizio senza fronzoli, senza affidabilità, né controllo di flusso e della congestione

3) **livello di rete**

si occupa di trasferire i pacchetti a livello di rete, detti datagrammi da un host all'altro. Comprende il protocollo IP e vari protocolli di ins tradamento che determinano i percorsi che i datagrammi devono seguire tra la sorgente e la destinazione

4) **livello di collegamento**

serve per trasferire un pacchetto da un nodo (host o router) a quello successivo sul percorso. Per esempio Ethernet e WiFi  
I pacchetti a livello di collegamento sono chiamati frame. Sposta interi frame da un elemento della rete a quello adiacente

5) **livello fisico**

Trasferisce i singoli bit del frame da un nodo a quello successivo (doppino o fibra ottica)

Ricorda che a ciascun livello il pacchetto ha 2 campi: Intestazione (?) e payload (il campo utile trasportato proveniente dal livello superiore).

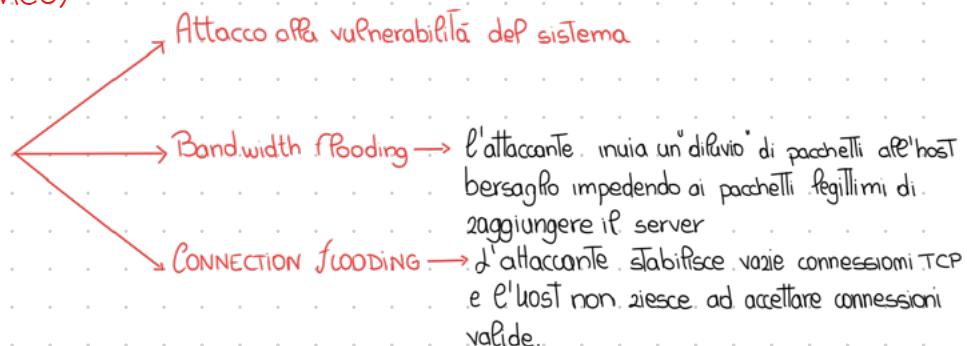
PER NOME E LUOGO DESTINATARIO

FRAMMENTAZIONE → un messaggio grande può essere diviso in più segmenti a livello di Trasporto, al momento della ricezione, tale segmento va ricostruito a partire dai datagrammi.

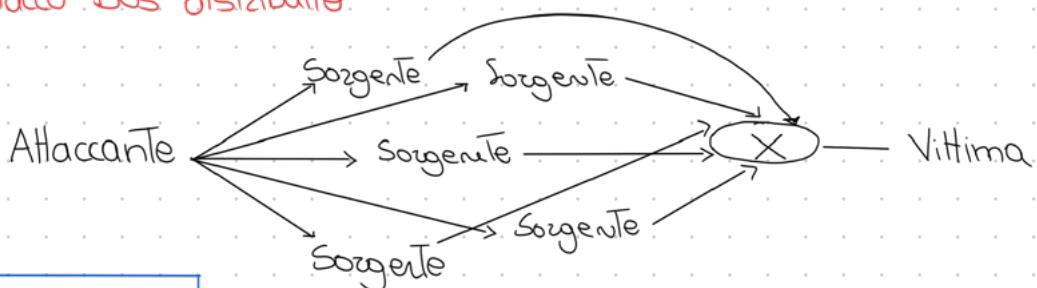
Reti sotto attacco attraverso malware che si diviso in virus (richiedono una certa interazione con l'utente) o worm (esecuzione applicazione vulnerabile)

### Attacchi DOS (denial of service)

un attacco Dos rende inutilizzabile dagli utenti legittimi una rete, un host o un'altra parte di infrastruttura



### Esempio Attacco Dos distribuito



Un ricevitore passivo detto packet sniffer in prossimità di un trasmettitore wireless può ottenere una copia di ogni pacchetto trasmesso.

La capacità di immettere pacchetti in Internet con un indirizzo sorgente falso è nota come IP spoofing

Soluzione?

### AUTENTICAZIONE

Raffica di Richiesta ad una un host

I ricorda proxy-server!

STORIA → libro Pagine 58 - 62

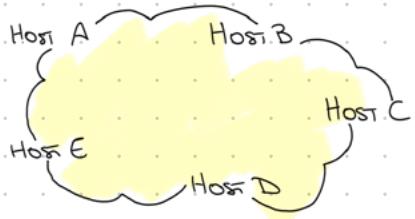
## TIPI DI TRASMISSIONE: COLPOGGIE DI RETE

ISOLAMENTO  
DIFF TRA HUB E SWITC  
SWITC  
C'è INGEGNERIA IN  
PARE A GLOBALE

### • Punto - Punto Host A — Host B

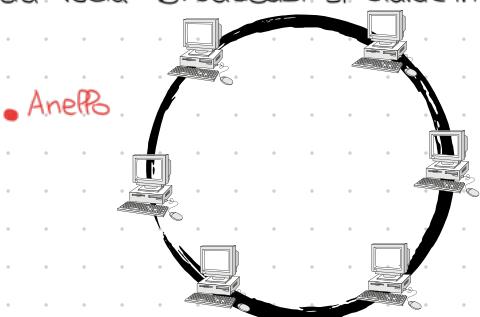
Due macchine sono collegate ad un solo cavo

### • Broadcast



Tutti possono ascoltare e partire contemporaneamente. Al fine di evitare le collisioni di segnali, fornisce regole di comunicazione. Ogni volta che parla deve decidere il destinatario.

A sua volta Broadcast si divide in:



### • Anello

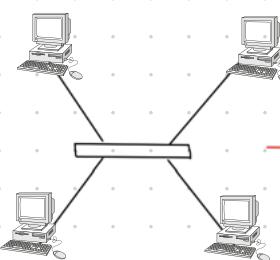
d'informazione viene immessa sul circuito e saetta da una macchina all'altra fino a quando non si trova a quella mittente.

### • Bus Condiviso



Tutti possono leggere e tutti possono scrivere d'informazione si propaga dalla macchina generata a tutte le altre.

### • A stella



Un hub centrale è collegato punto a punto con altre macchine. L'hub non converte i segnali ma li sìtrasmette. Se riceve due segnali contemporaneamente ne sìtrasmette la collisione.

Trasmissione store-and-forward:  
il commutatore deve ricevere l'intero pacchetto prima di poterne cominciare a trasmettere sul collegamento in uscita il primo bit

Dati N pacchetti a L bit ciascuno, con velocità di trasmissione R

$$d_{\text{end-to-end}} = N * \frac{L}{R}$$

Ritardo della trasmissione.

RITARDO DI FIABORAZIONE  
ossia il tempo che i bit impiegano a percorrere il collegamento, esaminare l'intestazione del pacchetto, controllo di eventuali errori (Velocità della luce)

RITARDO DI ACCODAMENTO Dipende dai pacchetti in coda  
Se la cosa è vuota il ritardo nullo

RITARDO DI TRASMISSIONE

RITARDO DI PROPAGAZIONE dato dal rapporto fra la distanza fra mittente e destinatario e la velocità di propagazione

RITARDO END-TO-END Somma ritardi complessivi di ogni singolo nodo.

## COMMUTAZIONE DI CIRCUITO

TDH  
ADH

In queste reti le risorse richieste lungo il percorso (buffer e velocità di trasmissione(banda)) per consentire la comunicazione, sono riservate per l'intera durata della sessione di comunicazione. Viene riservato anche il buffer di output nel router

Un esempio sono le linee telefoniche. Quando due host desiderano comunicare la rete stabilisce una connessione end-to-end(punto a punto) dedicata a loro con una certa banda garantita.

Ex. Abbiamo 4 commutatori connessi tramite 4 collegamenti con una velocità di trasmissione massima pari a 1Mbps. Questo significa che ogni collegamento avrà una banda pari a 250Kbps garantita. Se due di questi commutatori vogliono comunicare allora, nonostante gli altri collegamenti non siano usati, potranno al più scambiarsi dati a 250Kbps.

Dal punto di vista delle risorse la commutazione di circuito è molto più dispendiosa in quanto non è detto che tutti i 4 collegamenti vengano usati (non è detto che tutti gli utenti accedono ad internet) e quindi risulta inutile riservare banda a connessioni inutilizzate.

## COMMUTAZIONE DI PACCHETTO

Reti con circuiti misurati  
Rete datagram

La differenza con la commutazione di circuito sta nel fatto che in questo modello, le risorse riservate alla comunicazione vengono stanziate solo nel momento in cui si vuole stabilire una comunicazione. Non c'è alcuna garanzia infatti a differenza della commutazione di circuito, il pacchetto potrebbe trovare il buffer di output del router pieno e in questo caso uno dei pacchetti verrebbe perso e viene scelto o quello che sta in coda oppure l'ultimo arrivato

Nell'esempio precedente, se due commutatori vogliono comunicare e gli altri collegamenti sono liberi, allora la connessione avrebbe un'ampiezza di banda pari a 1Mbps: ovvero viene sfruttata tutta la banda disponibile.

**JITTER** → Quantità di tempo che trascorre tra la trasmissione del pacchetto m e quella di m+1

Ma come fa il router a determinare su quale collegamento il pacchetto dovrebbe essere inoltrato?

Quando un pacchetto giunge a un router nella rete, quest'ultimo esamina una parte dell'indirizzo di destinazione e lo inoltra a un router adiacente. Ogni router ha una tabella di inoltro (forwarding table) che mette in relazione gli indirizzi di destinazione con i collegamenti in uscita. Quando un pacchetto giunge a un router, questo esamina l'indirizzo e consulta la propria tabella per determinare il collegamento uscente appropriato. Il router quindi dirige il pacchetto verso quel collegamento di uscita.

Lui sa ogni porta a quale indirizzo posta e invia il pacchetto in base a questa informazione.

## CIRCUITO VIRTUALE:

Le routing nella commutazione di un circuito viene fatta a sola volta, usando 2 metodi.

- ① Scrivere non la destinazione ma la strada da percorrere. (SCONSIGLIATA!)
- ② Uso i **CIRCUITI VIRTUALI**, ogni route, tranne il primo viene considerato una scheda di rete e possono decidere quale strada intraprendere i pacchetti. In quanto tutti i routes garantiscono il flusso di dati. Si chiama virtuale perché viene creato un circuito di comunicazione ma alla base abbiamo la commutazione di pacchetti.  
Questo sistema garantisce che i pacchetti arriveranno sempre in ordine.

## LEVELLO APPLICATIVO

**Processi** → un programma in esecuzione su un sistema che possono comunicare tra loro scambiandosi messaggi.

Per ciascuna coppia di processi comunicanti ne etichettiamo uno come server e uno come client.

Ci sono due principali architetture:

1- **Architettura client-server** vi è un host sempre attivo (server) che risponde alle richieste di servizio del client.

Nel contesto di una sessione di comunicazione tra una coppia di processi quello che avvia la comunicazione si chiama client mentre quello che attende di essere contattato si chiama server.

I client non comunicano tra di loro. Il server, inoltre dispone di un indirizzo fisso, detto indirizzo IP. Il client non ha bisogno di essere sempre attivo, potrebbe inviare una richiesta e poi spegnersi. Ma spesso, un singolo host non è in grado di rispondere a tutte le richieste dei client, per tale motivo esistono datacenter che ospitano diversi server.

**ESEMPIO** → Nel web, e. browser rappresenta il client e il web è un processo server, trasferimento file con FTP, definita e posta elettronica.

## • Architettura Peer-To-Peer

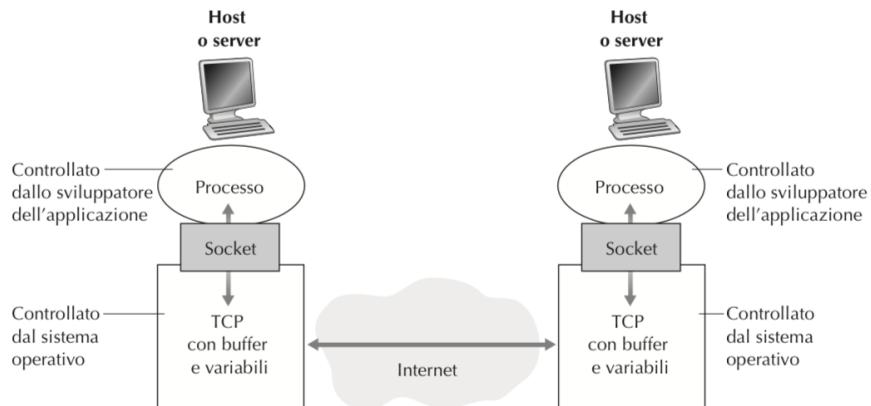
non sussiste un'architettura datacenter. I peer sono computer fissi e portatili, controllati dagli utenti. Ogni host comunica direttamente con un altro suo pari senza che i messaggi passino attraverso un server.

### BitTorrent, telefonia e videoconferenze

• Alcune applicazioni di messaggistica istantanea usano entrambe le architetture, i server tengono traccia degli indirizzi IP degli utenti, ma i messaggi tra utenti sono inviati in P2P.

P2P è scalabile → CHE SIGNIFICA? — HA TUTTE LA SUA EFFICIENZA INDEPENDENTMENTE IL NUMERO DEGLI UTENTI.  
P2P è economicamente conveniente ma ha problemi di sicurezza, prestazione e affidabilità.

Socket → Interfaccia software attraverso cui un processo invia e riceve messaggi dalla rete. Essa permette di collegare il livello applicativo con il livello di Trasporto. Esso rappresenta una porta mediante la quale il processo applicativo spedisce i pacchetti a livello di Trasporto.



Indirizzamento → Per spedire un pacchetto da un host ad un altro è necessario conoscere 2 dati fondamentali

Indirizzo univoco che identifica l'host, un numero di 32 bit che conosco attraverso un processo APACHE ("apach")

- o d'indirizzo IP dell'host destinataria
- o Numero di porta di destinazione che specifica la socket perché sulle host potrebbero essere in esecuzione molte applicazioni di rete.

NUMERO DI PORTE DI DESTINAZIONE STANDARD		
WELL KNOWN PORTS (0 - 1023)	REGISTERED PORTS 1024 - 49151	DYNAMIC AND OR PRIVATE PORTS (49152 - 65535)
Web Server → 80	→ 80 sono i servizi	
SMTP → 25	utenti	Porte dinamiche

### Proprietà dei protocolli di trasporto

Trasferimento dati affidabili  
i dati inviati sono corretti e compatti, abbiamo la certezza che quei dati arriveranno.

LOSS-TO-TOLERANT: applicazione che tollerano perdite. Alcuni utilizzano algoritmi di interpolazione.

Throughput

Velocità di trasmissione in bps (bit/s). È difficile garantire una velocità costante in rete.

Tempo di ritardo

o timing delay. La certezza che i dati arriveranno entro un tempo T dipende dal traffico in rete.

Sicurezza  
fornire la sicurezza dei dati attraverso la crittografia da parte del processo mittente e la decodifica da parte del processo destinatario.

redito a pag 224 di uvello di Tresporta.

**Telnet:** è un protocollo di rete utilizzato tramite interfaccia a riga di comando per fornire sessioni di login remoto.  
Esso utilizza TCP perché vuole una connessione affidabile.  
da differenza tra SSH e Telnet è che SSH è sicuro, la comunicazione è cifrata in SSH.

## HTTP

(Hypertext transfer protocol)

è un protocollo a livello di applicazione del Web. Definisce il modo in cui i client web chiedono le pagine ai web server e come questi ultimi le trasferiscono ai client.  
Il client web è rappresentato dal browser che l'utente usa mentre il web server rappresenta un host che ospita oggetti web indirizzabili tramite URL.

Una pagina web è costituita da oggetti, cioè file indirizzabili tramite URL. Ogni URL ha 2 componenti

Nome dell'host del server che ospita l'oggetto  
Percorso dell'oggetto

Tra i web server ricordiamo Apache e Microsoft Internet Information Server  
è sempre attivo, ha un indirizzo IP fisso.

HTTP utilizza TCP ed ha numero di porta 80.

Il client HTTP per prima cosa inizia una connessione TCP con il server e una volta stabilita i processi client e server accedono a TCP attraverso le proprie socket (porta tra un processo e la sua connessione TCP)

È un protocollo senza stato perché il server invia i file richiesti al client senza memorizzare alcuna informazione di stato a proposito del client. Quindi il server crea una connessione ogni volta che riceve una richiesta dal client anche se fosse la stessa medesima richiesta nel giro di pochi millisecondi.

Connessione persistente

il server web lascia la connessione aperta dopo aver inviato la risposta al client. In tal modo il server può inviare più oggetti alla volta durante la stessa connessione. Deve allocare buffer e mantenere variabili TCP sia su client che su server (HTTP 1.1).

Il server chiude la connessione dopo aver inviato l'oggetto richiesto.

Se la pagina ha 10 oggetti affora saranno necessarie 10 connessione TCP.

Ma cosa accade precisamente? Ecco i 5 passi in una connessione non persistente

Connessione non persistente

- 1 - Il processo client HTTP inizia una connessione TCP con il processo server.
- 2 - Dopo aver instaurato una connessione, il client tramite la propria socket invia un messaggio HTTP che include l'URL.
- 3 - Il processo server HTTP riceve il messaggio, lo incapsula l'oggetto in un pacchetto e lo spedisce al client.
- 4 - Il server comunica a TCP di interrompere la connessione (solo quando il client avrà ricevuto il messaggio).
- 5 - Il client riceve il messaggio e la connessione termina.
- 6 - Vengono ripetuti i primi 4 passi per ciascuno degli oggetti.

**RTT (round-Trip-time)** rappresenta il tempo impiegato da un pacchetto per viaggiare dal client al server e poi tornare al client, include anche i vari ritardi.

## MESSAGGI HTTP

Tra i vari metodi dei messaggi di richiesta:

- GET prendo una pagina dal server web, manda richieste per ottenere delle pagine web
- HEAD verifica la correttezza del codice prodotto
- PUT è stato disabilitato per problemi di sicurezza, invia un oggetto ad un percorso specifico
- POST ~~la ricerca è visibile nel pacchetto non nella URL~~
- DELETE permette la cancellazione di un oggetto sul server
- TRACE
- CONNECT
- OPTIONS

## MESSAGGI DI RICHIESTA HTTP

Il messaggio è scritto in Testo ASCII per ragioni di regolarità, le intestazioni dei pacchetti sono variabili ma app. sempre gli stessi campi.

GET /somedir/page.html HTTP/1.1

Host: www.someschool.edu

Connection: close

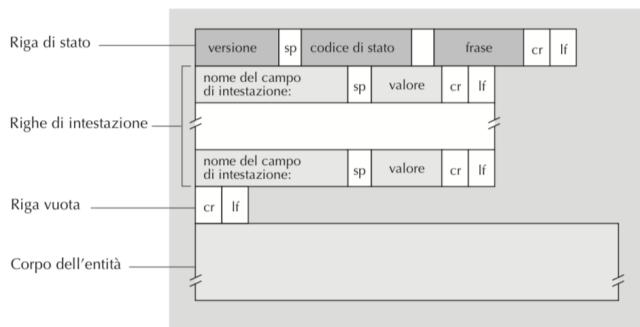
User-agent: Mozilla/5.0 *specifico tipo di browser che sta effettuando la richiesta.*

Accept-language: fr

Riga di richiesta  
Riga Host specifica l'host su cui si trova l'oggetto  
Riga che indica se la connessione è persistente o meno.  
l'utente preferirebbe la lingua francese ma se non fosse disponibile invia la versione di default

da riga di richiesta ha 3 campi: Metodo, URL e il campo di versione HTTP

## FORMATO GENERALE MESSAGGI HTTP



→ Chiedere a Ricezione!

→ È vuoto nel caso del metodo Get, nel caso del metodo "Post" il corpo contiene ciò che l'utente ha immesso nei campi del form.

## MESSAGGI DI RISPOSTA HTTP

HTTP/1.1 200 OK Riga di stato Iniziale, include la versione del protocollo, un codice di stato e un messaggio di stato

Connection: close

Date: Thu, 18 Aug 2015 15:44:04 GMT indica l'ora e la data di creazione e invio da parte del server, è momento in cui il server recupera l'oggetto del system.

Server: Apache/2.2.3 (CentOS) → messaggio generato da Apache

Last-Modified: Tue, 18 Aug 2015 15:11:03 GMT → importante per la gestione dell'oggetto della cache

Content-Length: 6821 numero byte oggetto inviato

Content-Type: text/html l'oggetto nel corpo è Testo HTML, quindi è il tipo di oggetto

Corpo ← (data data data data ...)

Codice di stato:

- 200 OK la richiesta ha avuto successo
- 301 MOVED PERMANENTLY l'oggetto è stato spostato presso l'URL specificato nel messaggio di risposta
- 400 BAD REQUEST la richiesta non è stata compresa dal server
- 404 NOT FOUND il documento richiesto non esiste sul server
- 505 HTTP Version not Supported il server non dispone della versione di protocollo HTTP richiesta
- 500 INTERNAL SERVER ERROR



consentono ai server di tener traccia degli utenti e permettono di gestire le sessioni. La tecnologia dei cookie presenta 4 componenti

- 1- Riga di intestazione nel messaggio di risposta HTTP
- 2- "richiesta"
- 3- Un file mantenuto sul sistema dell'utente e gestito dal browser
- 4- Database sul sito

I cookie permettono di registrare loggati tra le varie pagine e associano un identificativo ad un utente (Amazon, Ebay)

— Sono considerati una violazione della privacy.

## FTP (File Transfer Protocol)

trasferimento di file tra un host remoto. Il protocollo instaura 2 connessioni: da prima sulla porta 20 per scambiarsi dati e l'altra, adibita allo scambio di messaggi attraverso la porta 21 al fine di monitorare lo stato di connessione per decidere se mantenerla o no.

E' un protocollo che mantiene lo stato, ed è proprio questo la differenza da HTTP

## Web-Caching

una web-cache è un proxy-server ovvero un'entità che soddisfa le richieste HTTP al posto del web server effettivo

Il proxy ha una propria memoria in cui conserva copie di oggetti richiesti recentemente. Quando un client effettua una richiesta esse vengono dirette innanzitutto verso il proxy server, se quest'ultimo ha in cache l'oggetto richiesto allora egli lo invia al client, altrimenti effettua una richiesta al server e ritrasmette la risposta del server al client conservando in cache una copia dell'oggetto richiesto (agisce sia da client che da server). I proxy server hanno una grande importanza perché permettono di non sovraccaricare i server ed allo stesso tempo di fornire delle risposte più veloci.

**SMTP** (Simple mail transfer protocol) protocollo di push, in quanto invia messaggi SMTP rappresenta il principale protocollo per la posta elettronica a livello applicativo. Esso è basato su TCP al fine di garantire un trasferimento dati affidabile.

Il sistema postale è quindi costituito dallo user-agent (Outlook, gmail) e dal server di posta.

2 SIANO 2 HOST A E B E SUPponiamo che A voglia spedire una mail all'host B.

1. A invoca il proprio user-agent, fornisce l'indirizzo di posta B, scrive mess. e glielo invia
2. Lo user-agent di A invierà il messaggio al suo mail server, viene accodato ad una coda di message
3. Il lato client di SMTP eseguito sul mail server di A vede il messaggio e apre una connessione TCP verso il lato server SMTP in esecuzione su mailserver di B ed invia il messaggio.
4. Il lato server di B SMTP riceve il messaggio e lo ripone nella casella di B, il quale invoca il proprio user-agent per leggere il messaggio.

## Messaggi SMTP

Dopo una breve presentazione (handshake) in cui il client indica l'indirizzo di posta del destinatario, quest'ultimo invia i dati che costituiscono la mail più un messaggio terminale che consiste solo di un punto. Successivamente sempre il client invia un comando QUIT che chiude la conversazione. Ad ogni comando del client il server risponde con un codice ed una qualche spiegazione in inglese.

## POP3 e IMAP

Sono due protocolli usati dagli host per trasferire i messaggi dal web server al PC locale. SMTP infatti essendo di push trasferisce il messaggio dallo user agent del mittente al suo mail server e successivamente fra i mail server mentre non può essere usato per prelevare messaggi dal mail server.

POP3 è un protocollo piuttosto semplice che permette di prelevare dal server le mail e scaricarle sul proprio PC locale. Ciò risulta essere molto limitante soprattutto nel caso in cui noi volessimo nuovamente accedere alle mail però da un altro dispositivo.

A tal fine si è introdotto IMAP il quale ci permette di salvare la posta sul mail server e conservarne una copia in locale, offrendo dunque la possibilità di recuperarla successivamente anche tramite altri dispositivi.

## CONFRONTO TRA HTTPS E SMTP

HTTP è un protocollo pull: qualcuno cerca le informazioni su un web server e gli utenti usano HTTP per attirarle a sé dal server. Esso incapsula ogni oggetto nel proprio messaggio di risposta.

SMTP è un protocollo push: il mail server di invio spinge i file al main server in ricezione, la connessione TCP viene iniziata dall'host che vuole spedire il file.

Inoltre SMTP deve comporre l'intero messaggio in ASCII a 7 bit.

Essa racchiude tutti gli oggetti in un unico messaggio.

Intestazione: From, To, subject.

## DNS : il servizio di directory di Internet

I nomi degli host vengono identificati da indirizzi IP per essere identificati dai server, nonostante noi ci vediamo sotto forma di hostname.

consiste di quattro byte e presenta una gerarchica.

perché, leggendo da sinistra a destra ci darà informazioni.

più dettagliate sulla collocazione dell'host in Internet.

Noi identifichiamo gli host o del nome o IP.

IP DNS è un database distribuito implementato in una gerarchia di DNS server ed è anche un protocollo a livello di applicazione che consente agli host di interrogare i database, e permettono di ottenere le traduzioni di nome di host forniti dall'utente in indirizzi IP.

I DNS sono macchine UNIX che seguono un software chiamato BIND.

Il protocollo DNS utilizza UDP e la porta 53.

**ESEMPIO →** Consideriamo cosa succede quando un browser (ossia un client TCP) in esecuzione sull'host di un utente, chiede URL www.ciaq.edu / index.html. L'host dell'utente deve prima ottenere il suo indirizzo IP. Segue i seguenti passi:

- 1- da stessa macchina utente esegue il suo client dell'applicazione DNS.
- 2- Il browser estrae il nome dell'host, www.ciaq.edu, dall'URL e lo passa al suo client dell'applicazione DNS.
- 3- Il client DNS invia un'intervrogazione (query) contenente l'hostname a un DNS server.
- 4- Il client DNS prima o poi riceve una risposta che include l'IP corrispondente all'hostname.
- 5- Una volta ricevuto l'indirizzo IP dal DNS, il browser dà inizio una connessione TCP verso il processo server HTTP collegato alla porta 80 di quell'indirizzo IP.

Il DNS introduce un ritardo aggiuntivo alle applicazioni Internet. Possiamo ridurre il traffico DNS in rete e il ritardo medio utilizzando la cache del DNS.

## ALTRI SERVIZI CHE METTE A DISPOSIZIONE DNS

- **HOST ALIASING** → Un host dal nome complicato può avere uno o più sinonimi (alias) quindi il nome principale sarà chiamato un **host canone** che può essere richiesto invocando il DNS da un sinonimo.

- **MAIL SERVER ALIASING**

Nou l'ho compreso. Pagina 124 / 125

- **DISTRIBUZIONE DEL CARICO DI RETE**

Il DNS viene utilizzato per distribuire il carico tra server replicati. I siti con molto traffico vengono replicati su più server ognuno eseguito su un host diverso con un indirizzo IP differente, quindi si associa ad ogni hostname un insieme di indirizzi IP.

## PANORAMICA FUNZIONAMENTO DNS

Se il DNS fosse costituito da unico server si incorrerebbe in diversi problemi.

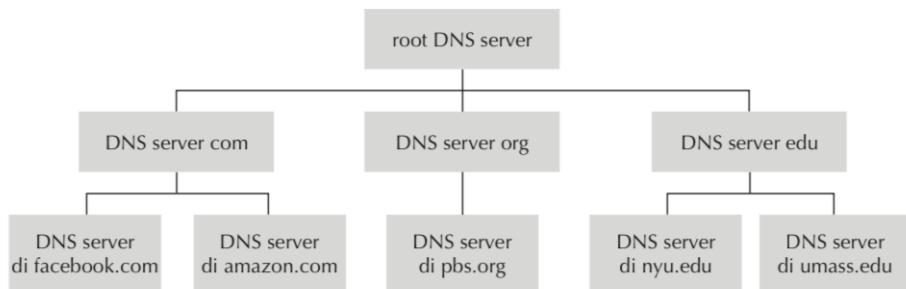
**VOLUME DI TRAFFICO** - 1 - un solo database dovrebbe far fronte a milioni di richieste al giorno.

**SOLI PUNTI DI FAILOVER** - 2 - Se il DNS server si guasta ne soffre tutta l'infrastruttura Internet.

**DATABASE CENTRALIZZATO** - 3 - Un unico database risulterebbe vicino a certe regioni ma lontano da altre distante il resto del mondo, questo causerebbe tempi di accesso lunghi e congesti causando ritardi significativi.

**MANTENIMENTO** → Il singolo DNS dovrebbe contenere record relativi a tutti gli host di Internet. Avremmo un enorme database che bisogna aggiornare frequentemente. Inoltre un database centralizzato non è scalabile.

Quindi, visto questi problemi abbiamo optato per un database distribuito.



Ie DNS utilizzano un grande numero di server, organizzati in maniera gerarchica e distribuiti nel mondo. Esistono 3 classi di DNS

### Root Server

Rappresentano il livello più alto nella gerarchia e si occupano di fornire gli indirizzi IP dei server Top-level domain. Sono 100.

Pochissimi (400 mondiali)

### Top-Level Domain (TLD)

Questi server si occupano dei domini di 1° livello quali .com, .org, .net, .edu e .gov e dei domini relativi ai vari paesi come .uk o .fr.

I servizi Top-level domain formiscono gli indirizzi IP dei server autorizzati.

### DNS server autorizzati

Ogni organizzazione dotata di host pubblicamente accessibili tramite Internet deve fornire record DNS pubblicamente accessibili che associno i nomi di tali host a indirizzi IP.

\* NON FA PARTE DELLA GERARCHIA DEI SERVER

### DNS Server locali

Sono associati a ciascun ISP, come università, dipartimenti, aziende. Questi server fanno da proxy fra host e gerarchia dei DNS server. Quando un host si connette a un ISP che fornisce un indirizzo IP, tratto da uno dei suoi DNS server locali, generalmente tramite DHCP.

Quando un host effettua una richiesta DNS, la query viene inviata al DNS locale, che opera da proxy e inoltre fa query alla gerarchia dei DNS server.

ESEMPIO pagina 29

TANTI A CUI CHIEDO E  
MI INDIRIZZANO AL MIO  
DNS Root

## DNS CASHING

In un DNS ha grande importanza il servizio di caching al fine di migliorare le prestazioni di ritardo e per ridurre il numero di messaggi DNS.

**Ma come funziona?** In una concatenazione di richieste, il DNS server che riceve una risposta DNS mette in cache le informazioni contenute quindi ogni server DNS mantiene in cache gli indirizzi IP delle ricerche recenti, così da comunicare subito all'host se chiede un IP chiesto poco tempo prima.

Grazie a questo servizio il DNS può essere visto come un grafo perché ogni server DNS su cui effettuiamo una richiesta di traduzione può avere in cache oppure a indirizzarà verso il DNS server che può farcela trovare.

I DNS server invalidano le informazioni in cache dopo un periodo di tempo fissato, in genere 2 giorni.

Un DNS server locale può memorizzare in cache gli indirizzi IP dei TLD server consentendogli di aggirare i root server nella catena di richieste.

Quando ricechiamo una traduzione i DNS server ci fornisce non uno ma una lista di indirizzi dalla quale scegliere. Tale lista, ad ogni richiesta, viene modificata al fine di non fornire ad ogni richiesta la stessa traduzione evitando così la congestione di un solo, ad esempio, web server ma si spartisca su tutti quelli disponibili come i server di Google.

\* Gli indirizzi IP variano con il tempo quindi il database ha bisogno di aggiornarsi.

I server che implementano IP database distribuito di DNS memorizzano i record di risorsa che forniscono le corrispondenze tra nomi e indirizzi.

Ogni messaggio di risposta DNS trasporta 1 o più record di risorse che contengono i seguenti campi:

(Name, Value, Type, TTL)

Time To Live ossia il tempo residuo di vita di un record e determina quando una risorsa va rimossa dalla cache.

- Se Type = A, allora Name è il nome dell'host e Value è il suo indirizzo IP. Pedendo un record di tipo A fornisce la corrispondenza tra hostname standard e indirizzo IP.
- Se Type = NS, Name è un dominio e Value è l'hostname del DNS server autoritativo. Viene usato per indirizzare le richieste DNS successive alla prima nella concatenazione delle query.
- Se Type = CNAME, Value è il nome canonico dell'host per il suo nome.

Se Type = MX, Value è il nome canonico di un mail server che ha il suo nome.

## livello di Trasporto

Un protocollo a livello di trasporto mette a disposizione una comunicazione logica tra processi applicativi di host differenti.

Per comunicazione logica si intende, dal punto di vista dell'applicazione, che tutto proceda come se gli host che eseguono i processi fossero direttamente connessi.

I processi applicativi usano la comunicazione logica fornita dal livello di trasporto per scambiare messaggi.

### dato Mittente

Il livello di Trasporto convierte i messaggi che riceve da un processo applicativo in pacchetti a livello di Trasporto noti come segmenti, questo avviene spezzando i messaggi e aggiungendo a ciascuna un'intestazione di Trasporto per creare un segmento.

Il livello di Trasporto passa il segmento al livello di rete dove viene incapsulato all'interno di un pacchetto a livello di rete (DATAGRAMMA) e viene inviato a destinazione.

Il compito di radunare i pacchetti provenienti dalle diverse socket, incapsularli per poi inviarli a livello di rete viene detto multiplexing.

**IMPORTANTE:** I ROUTER INTERMEDEI NON RICONOSCONO NÉ OPERANO SU ALCUNA INFORMAZIONE CHE IL LIVELLO DI TRASPORTO POSSA AVER AGGIUNTO AI MESSAGGI DI APPLICAZIONE.

INTERNET HA A DISPOSIZIONE 2 TIPI DI PROTOCOLLO: TCP E UDP, mentre basa il proprio livello di rete su protocollo IP.

→ protocollo inaffidabile che si basa sulla politica best-effort delivery service: IP fa del suo meglio per trasferire i file in maniera sicura ma potrebbero esserci eventuali ceppi.

## UDP

UDP (user datagram protocol), che fornisce alle applicazioni un servizio non affidabile e non orientato alla connessione.

Lato mittente, prendiamo i messaggi dal processo applicativo e li passiamo direttamente a livello di rete; lato ricevente, prendiamo i messaggi in arrivo dal livello di rete e li trasferiamo direttamente al processo applicativo.

- A parte la funzione di multiplexing/demultiplexing e una forma di controllo degli errori molto semplice, non aggiunge nulla a IP. Quando, infatti, lo sviluppatore sceglie UDP anziché TCP, l'applicazione dialoga quasi in modo diretto con IP.
- UDP prende i messaggi dal processo applicativo, aggiunge il numero di porta di origine e di destinazione per il multiplexing/demultiplexing, aggiunge due piccoli campi e passa il segmento risultante al livello di rete. Questi incapsula il segmento in un datagramma IP e quindi effettua un tentativo di consegnarlo all'host di destinazione in modalità best-effort.

Se il segmento arriva a destinazione, UDP utilizza il numero di porta di destinazione per consegnare i dati del segmento al processo applicativo corretto. Notiamo che in UDP non esiste handshaking tra le entità di invio e di ricezione a livello di trasporto. Per questo motivo, si dice che UDP è non orientato alla connessione.

### dato Ricevente

Il livello di rete estrae il segmento dal datagramma e lo passa al livello di Trasporto che elabora il segmento sudendo me disponibili i dati. Il livello di Trasporto esamina l'intestazione del segmento per capire a quale socket associarlo. Il compito di Trasportare i dati verso una destinazione prende il nome di demultiplexing.

## \*MULTIPLEXING richiede che

- Per socket abbiamo identificatori univoci
- ciascun segnale presente campi che indicano la socket a

Reg §82!

## ESEMPIO UDP

**DNS** → Quando si chiede una query attende una risposta se non ne riceve invia un query.

**SNMP** — Applicazioni.

**multimediali** — Streaming in tempo reale — Applicazioni. Possibilità in cui è sostenibile perdere dati durante la trasmissione.

## Reading

Applicazione	Protocollo a livello di applicazione	Protocollo di trasporto sottostante
Posta elettronica	SMTP	TCP
Accesso a terminali remoti	Telnet	TCP
Web	HTTP	TCP
Trasferimento file	FTP	TCP
Server di file remoti	NFS	generalmente UDP
Dati multimediali in streaming	generalmente proprietario	UDP o TCP
Telefonia su Internet	generalmente proprietario	UDP o TCP
Gestione di rete	SNMP	generalmente UDP
Traduzione di nomi	DNS	generalmente UDP

## Quando bisogna scegliere UDP?

Esso non introduce alcun stato per stabilire la connessione, infatti è più veloce a spedire i pacchetti, i quali avendo un'intestazione minima possono contenere <sup>a byte mentre TCP 20</sup> più dati. Non mantenendo alcun bit di stato una connessione UDP può generalmente supportare più client attivi.

UDP spara dati a raffica senza utilizzare handshake (TCP ne usa 1 a 3 vie).

Quindi UDP non è un protocollo end-to-end poiché non viene stabilita alcuna connessione.

## Struttura dei segmenti UDP

32 bit



Supponiamo 3 parole a 16bit  
 0110011001100000  
 0101010101010101  
 1000111100001100  
 Prima sommiamo le prime due e poi la terza.  
 Poi facciamo il complemento a 2 (diventa 0 e 0 diventa 1).

d'intestazione UDP presenta solo 4 campi di 2 byte ciascuno. Oltre la coppia che indica numero porta di origine e numero porta di destinazione abbiamo un campo lunghezza che specifica il numero di byte del segmento. Il campo checksum viene usato dall'host ricevente per verificare se sono avvenuti errori nel segmento. NEGLI UNICI I BIT  
 dato mittente UDP effettua il complemento a 2 della somma di tutte le parole a 16 bit del segmento e l'ha poi messo nel checksum. dato destinatario si calcola la somma delle parole a 16 bit e si sommano anche il checksum. Se il pacchetto è arrivato senza alterazioni avremo tutti i bit a 1.

Anche se UDP rileva un errore in rete non fa nulla per gestirlo, il pacchetto viene scartato o trasmesso con un avvertimento all'applicazione.

Il complemento a 2 solo in ricezione

HA UN PRINCIPIO END-TO-END (il controllo degli errori deve essere implementato nei nodi intermedi)

## PER VEDERE SE UN NUMERO DISPAR

- Il campo di lunghezza è fondamentale perché quando arriviamo a livello fisico non sapremo quando finisce un determinato datagramma.

### 3.4 Princìpi del trasferimento dati affidabili

Con un canale affidabile nessun bit dei dati trasferiti è corrotto o va perduto e tutti i bit sono conseguiti nell'ordine di invio.

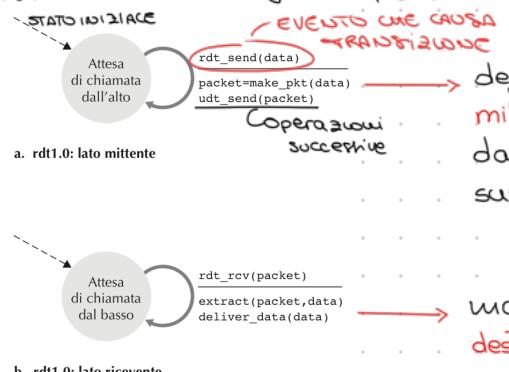
TCP (transmission control protocol) offre un servizio affidabile e orientato alla connessione.

Potiamo da una serie di protocolli che ci permettono di acciudere a TCP

- **Premessa:** Tratteremo mittente e destinazione come autonimi a stati finiti con unico stato e le transizioni hanno lo stesso stato e se stesso.

Perfettamente  
Affidabile  
Modello teorico

**1. RDT 1.0**  
Reliable  
Data  
Transfer



definisce le operazioni **de** mittente che trasferiscono i dati da consegnare al livello superiore sul lato ricevente.

a. rdt1.0: lato mittente

b. rdt1.0: lato ricevente

rdt1.0: protocollo per un canale completamente affidabile.



- RDT 1.0 riguarda il trasferimento dati su un canale perfettamente affidabile ed è uno dei protocolli banali.
- Il lato mittente di rdt accetta semplicemente dati dal livello superiore, crea un pacchetto e lo invia sul canale.
- Lato ricevente, raccoglie i pacchetti dal sottostante canale, rimuove i dati dai pacchetti e li passa al livello superiore.
- Con un canale perfetto non c'è bisogno che mittente e ricevente comunichino tra loro in quanto nulla può andare storto.

mostra come opera il **destinatario** dove è seduto. Giudica la chiamata a lato mittente di rdt. Per conseguire i dati a livello superiore fa chiamando `deliver_data()`.

Comunque supponiamo che i pacchetti arrivino in ordine giusto e non perda pacchetti.

**2. RDT 2.0** → trasferimento dati affidabile su un canale con errori sui bit

Un modello più realistico del canale è quello in cui i bit di un pacchetto possono essere corrotti. L'idea è quella di far sì che il mittente prima di spedire un ulteriore pacchetto, debba ricevere dal destinatario un messaggio di verifica: un ACK costituisce una verifica positiva (i dati sono arrivati integri), un NAK invece rappresenta una notifica negativa (i dati sono stati corrotti).

Il mittente risulta quindi composto da due stati. Uno che attende dati dal livello superiore per creare un pacchetto e spedirlo, ed un altro che (dopo aver spedito il pacchetto) attende dal destinatario un messaggio di notifica.

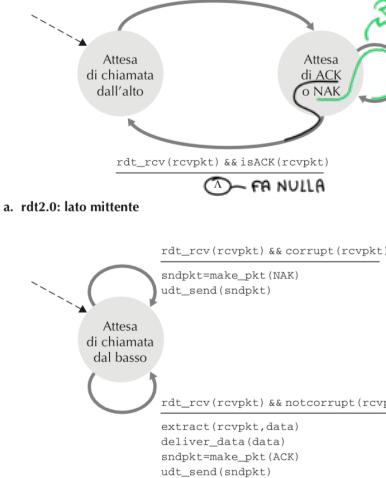
Durante l'attesa non può ricevere informazioni dal livello applicativo.

Si usa il principio dello stop and wait: **LIVELLO DI TRASPORTO** il mittente spedisce un pacchetto alla volta: se non riceve una notifica per quel pacchetto non può trasmetterne un altro.

I PROTOCOLLI DI TRASFERIMENTO DATI AFFIDABILI, BASESI SU RITRASMISSIONI SONO NOTI COME I PROTOCOLLI ARG che

- Rilevano gli errori sui bit
- Feedback del destinatario
- Ritrasmissioni.

- Rilevano gli errori sui bit  
- Feedback del destinatario  
- Ritrasmissioni.



In realtà bisogna tener conto anche della possibilità che anche i pacchetti ACK e NAK possano essere a loro volta alterati quando vengono ritrasmessi

La soluzione a questo problema consiste nel fatto che il mittente ritrasmetta il pacchetto di dati corrente a seguito della ricezione di un pacchetto NAK/ACK alterato aggiungendo un numero di sequenza di un bit.



## RDT 2.1

Vedi schema pagina 199/200

Una soluzione a questa problematica viene introdotta nel modello RDT 2.1:

l'idea consiste nell'obbligare il mittente a numerare i propri pacchetti dati con un numero di sequenza(0/1). Numero di sequenza 1 che è stato ritrasmesso mentre 0 che è un pacchetto nuovo.

Al destinatario sarà sufficiente controllare questo numero per sapere se il pacchetto rappresenti una ritrasmissione o meno. Il mittente esegue le stesse azioni di quello in RDT 2.0 ma ora sono presenti 4 stati al fine di gestire i pacchetti con numero di sequenza 0 e quelli con numero di sequenza 1.

Mi accorgo della ritrasmissione se due pacchetti consecutivi hanno lo stesso numero

Anche lato destinatario il numero di stati raddoppia.

Chiedere il numero di sequenze.



Il pacchetto ACK contiene il numero di sequenza.



## RDT 2.2

Un'altra modifica viene introdotta con RDT 2.2 : il destinatario, per segnalare al mittente di aver ricevuto un pacchetto corrotto, spedisce un ACK per l'ultimo pacchetto ricevuto correttamente, invece di inviare un segnale NAK. Così facendo il mittente che riceve due ACK per lo stesso pacchetto(ACK duplicati) sa che il destinatario non ha ricevuto quello successivo e quindi lo ritrasmette.

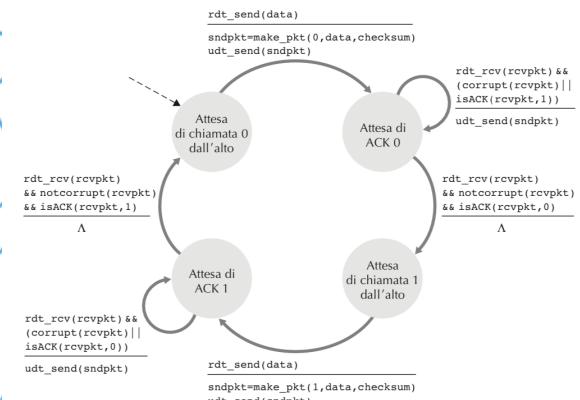
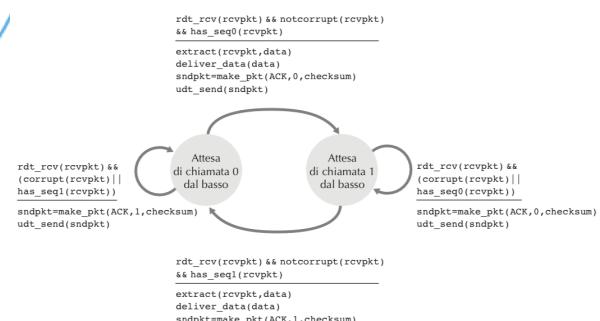


Figura 3.13 Mittente rdt2.2.



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

Supponiamo ora che il canale oltre a danneggiare i bit possa anche perdere pacchetti.

L'idea in questo caso è quella di utilizzare un timer: il mittente inizializza un contatore ogni volta che invia un pacchetto, se non si riceve un ACK per quel pacchetto prima che il timer scada allora lo ritrasmette.

Stimare la grandezza di un timer è molto difficile in quanto non si può mai avere la certezza che questo sia abbastanza grande da non introdurre troppi pacchetti duplicati ed abbastanza piccolo da non perdere troppo tempo prima di ritrasmetterlo.

RDT 3.0 introduce un timer e le primitive ad esso associate al fine di gestire la perdita di pacchetti.

~~PROTOCOLLO AD ADERNANZA DI BT = I NUMERI DI SEQUENZA DEI PACCHETTI SI ADERNANO TRA DI LORO~~

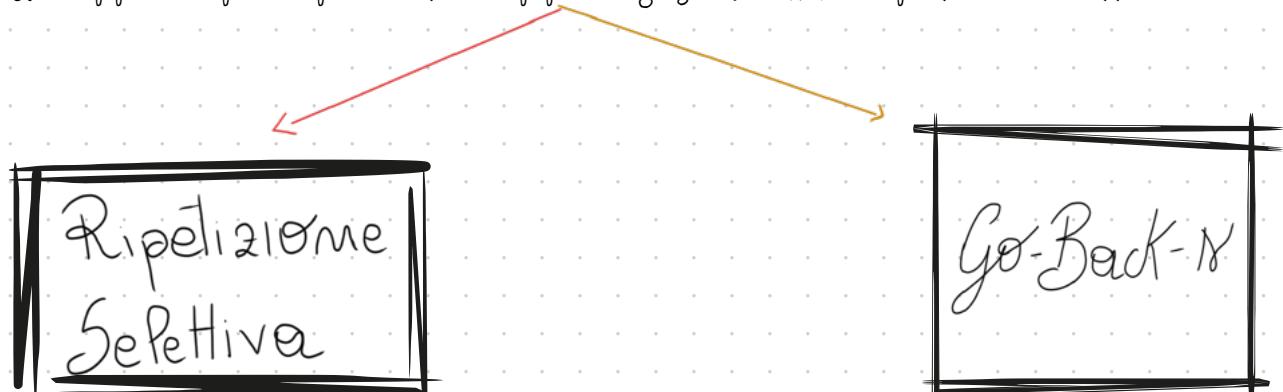
~~IL NUMERO D. SEQUENZA E UNIC E SI MUOVE~~

### Pipelining

Il protocollo RDT 3.0 risulta corretto dal punto di vista dell'affidabilità, ma risulta inefficiente in termini di prestazioni. Il problema sta nella strategia stop and wait: il mittente non potrà mai sfruttare tutta la banda disponibile se, prima di spedire un pacchetto, deve attendere un ACK per quello precedente.

Il pipelining è una metodologia che si basa sul continuo invio di pacchetti senza aspettare l'ACK di ricezione. Se ad esempio consentissimo al mittente di trasmettere tre pacchetti alla volta senza dover aspettare ACK di verifica, RDT 3.0 triplicherebbe il suo throughput.

Esistono due approcci per implementare il pipelining: Go-Back-N e ripetizione selettiva.



$$\begin{aligned} R &: \text{baud } 1 \text{ Gbps } (10^9 \text{ s}) \\ L &: \text{dimensione } 1000 \text{ byte} \\ \text{tempo richiesto per trasmettere 1 pacchetto} &= \frac{L}{R} \end{aligned}$$

### Ripetizione Selettiva

Il principale problema di GBN è dato dal fatto che quando l'ampiezza della finestra è grande, nella pipeline si possono trovare numerosi pacchetti. Se uno viene perso, questo porterà ad un grande numero di ritrasmissioni (spesso inutili in quanto non abbiamo la certezza che tutti gli altri siano stati persi).

I protocolli a ripetizione selettiva ritrasmettono solo quei pacchetti su cui esistono sospetti di errori. Si userà sempre una finestra di ampiezza  $N$ , al cui interno però (a differenza di GBN) si troveranno sia pacchetti per cui si è ricevuto un ack e sia quelli che non ne hanno ricevuto. Il destinatario infatti invia un riscontro anche per i pacchetti ricevuti fuori sequenza: questi vengono memorizzati in un buffer del ricevente finché non sono stati ricevuti quelli mancanti. Quando il ricevente riceve un pacchetto con numero di sequenza uguale alla 'base' della finestra di ricezione, allora lo spedisce al livello superiore insieme a tutti i pacchetti nel buffer con numeri di sequenza consecutivi. Lato mittente la finestra scorre solo nel caso in cui si è ricevuto un ack per il pacchetto più vecchio, rimane ferma nel caso in cui si riceve un ack per tutti gli altri.

## GO-Back-N(GBN)

Secondo questo modello il mittente può spedire più pacchetti senza dover attendere alcun ack, ma non può avere più di un numero massimo consentito di  $N$  pacchetti in attesa di un ack altrimenti potremmo causare congestione. Il protocollo GBN viene detto protocollo a finestra scorrevole. All'interno della pipeline avremo 4 intervalli:

- il primo che comprende i numeri di sequenza per quei pacchetti per cui abbiamo ricevuto un ack di ritorno
- il secondo comprende i numeri di sequenza dei pacchetti per cui non abbiamo ricevuto un ack di ritorno;
- il terzo comprende i numeri di sequenza dei pacchetti che possono essere inviati ma che non lo sono ancora stati;
- il quarto che comprende i numeri di sequenza dei pacchetti che non possono essere inviati.



Tale finestra è scorrevole: quando si riceve un ack per il pacchetto più vecchio all'interno della finestra, essa scorre verso destra, annettendo uno fra i pacchetti che possono essere spediti. Il mittente in GBN deve rispondere a tre diversi tipi di evento:

Invocazione dall'alto:

Se la finestra non è piena, crea e invia un pacchetto, altrimenti, mantiene il pacchetto nel buffer in attesa.

### Ricezione di un ack:

In questo modello la ricezione di un ack per il pacchetto con numero di sequenza  $n$  indica che tutti i pacchetti con un numero di sequenza minore o uguale a  $n$  sono stati correttamente ricevuti dal destinatario: in tal senso si parla di ack cumulativi.

In seguito alla ricezione di un ack la finestra di ricezione si sposta verso destra facendo così spazio a pacchetti che erano in attesa di essere spediti.

### Timeout:

Se il timer per un pacchetto (quello meno recente) scade allora il mittente ritrasmette tutti i pacchetti per cui non abbiamo ancora ricevuto un ack di verifica.

Lato destinatario se un pacchetto con numero di sequenza  $n$  viene ricevuto correttamente ed è in ordine, il destinatario invia un ack per quel pacchetto. In tutti gli altri casi il destinatario scarta il pacchetto e rimanda un ACK per il pacchetto più recente ricevuto correttamente. La scelta di non conservare pacchetti non in ordine deriva dal fatto che GBN, lato mittente, in caso di smarrimento del pacchetto  $n$  rispedirà tutti i pacchetti a partire da  $n$  in avanti (quindi anche  $n+1$ ). Ricordiamo che il destinatario non invierà ack in caso di ricezione del pacchetto  $n+1$  in mancanza del pacchetto  $n$ , in quanto essendo gli ack cumulativi, inviare un ack per il pacchetto  $n+1$  significherebbe dire al mittente di aver ricevuto anche il pacchetto  $n$  quando in realtà non è così. Se da un lato la ritrasmissione di tutti i pacchetti risulta dispendiosa, lato ricevente, egli deve memorizzare solo il numero di sequenza del pacchetto successivo.

**Trasporto Orientato alla Connessione TCP** Viene detto assemblato alla connessione. Prima di effettuare lo scambio dei dati, i processi devono effettuare l'handshake, ossia inviarsi reciprocamente alcuni segmenti preliminari per stabilire i parametri del successivo trasferimento dati.

Il protocollo TCP non include un circuito end-to-end, poiché lo stato dell'connessione si siede nei due sistemi periferici ed è anche per questo che non salva lo stato della connessione.

Una connessione TCP offre un servizio **FULL-DUPLEX**, cioè i dati possono viaggiare contemporaneamente in ambo le direzioni.

→ Una connessione TCP è anche punto a punto, cioè ha luogo da un singolo mittente a un singolo destinatario. Con TCP non possiamo effettuare il "multicast", cioè i trasferimenti dati da un mittente a più destinatari in un'unica soluzione.

Se 2 processi vogliono stabilire una connessione tra di loro per prima cosa

Il processo applicativo client innanzitutto informa il livello di trasporto client di voler stabilire una connessione verso un processo nel server. Il TCP in esecuzione sul client procede quindi a stabilire una connessione con il TCP del server. Verranno quindi trasportati 3 segmenti di cui i primi due servono per stabilire la connessione e il terzo potrebbe trasportare il payload (handshake a 3 vie).

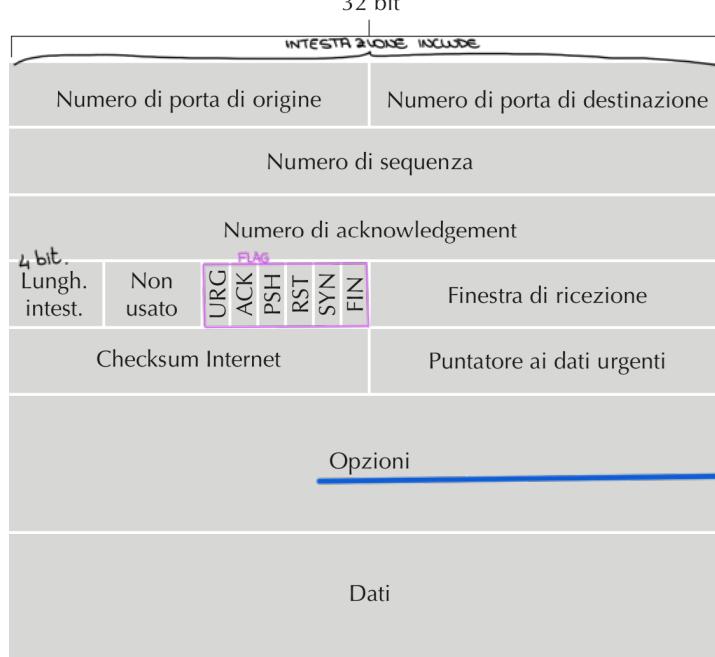
Una volta instaurata una connessione TCP, i due processi applicativi si possono scambiare dati. Il primo manda un flusso di dati attraverso la socket tramite TCP in esecuzione nel client. TCP dirige i dati al buffer di invio della connessione, uno dei buffer riservato durante l'handshake a tre vie, da cui, di tanto in tanto, preleverà blocchi di dati e li passerà al livello di rete.

d'intestazione TCP\IP  
normalmente è pari a 40 byte

La massima quantità di dati prelevabili e posizionabili in un segmento viene limitata dalla dimensione massima di segmento a livello di trasporto (MSS, maximum segment size). Questo valore viene impostato determinando prima la lunghezza del frame più grande che può essere inviato a livello di collegamento dall'host mittente locale, la cosiddetta unità trasmissiva massima compreso di intestazione di IP e datalink (MTU, maximum transmission unit) e poi scegliendo un MSS tale che il segmento TCP (una volta incapsulato in un datagramma IP) stia all'interno del singolo frame a livello di collegamento.

TCP accoppia ogni blocco di dati del client a una intestazione TCP, andando pertanto a formare segmenti TCP. Quando all'altro capo TCP riceve un segmento, i dati del segmento vengono memorizzati nel buffer di ricezione della connessione TCP. Ogni lato della connessione presenta un proprio buffer di invio e di ricezione.

# STRUTTURA SEGMENTI TCP



TCP frammenta i file in porzioni di dimensione MSS

3 dati affidabili

Utilizzato per il controllo di flusso

Facoltativo e di lunghezza variabile usato per la dimensione massima del segmento

- Il campo Flag è di 6 bit

1 bit ACK serve ad indicare che il campo acknowledgement è valido, il segmento è stato ricevuto.

RST }

SYN }

FIN }

CWR }

ECE }

PSH }

URG }

Impostati per chiudere e aprire la connessione.

usati per il controllo di congestione

Se = 1 invia i dati a livello superiore.

indica nel segmento la presenza di dati che l'entità mittente ha marcati come "urgenti".

In TCP il numero di sequenza indica il numero del primo byte del segmento. Se un host sta inviando segmenti di 1000 byte ciascuno affronta il primo segmento ha un numero di sequenza 0, il secondo 1000, il terzo 2000 e così via.

## TIMEOUT COME TCP

TCP usa i timer per fare fronte alla possibilità di perdere i pacchetti

**RTT** → Tempo che intercorre tra l'invio del segmento e la ricezione del suo ack di verifica. Il valore adatto viene calcolato attraverso una media ponderata tra il precedente valore del timer ed il nuovo valore di RTT. Tale media prende il nome di EWMA

$$\text{ESTIMATEDRTT}_n = (1 - \alpha) * \text{ESTIMATEDRTT}_{n-1} + \alpha * \text{SAMPLERTT}$$

EstimatedRTT è una combinazione ponderata del suo precedente valore e del nuovo valore di SampleRTT

Il valore raccomandato per  $\alpha$  è 0,125 e  $1 - \alpha = 1 - 0,125 = 0,875$ . Media esponenziale ponderata.

Telnet: un caso di studio per i numeri di sequenza e di acknowledgment

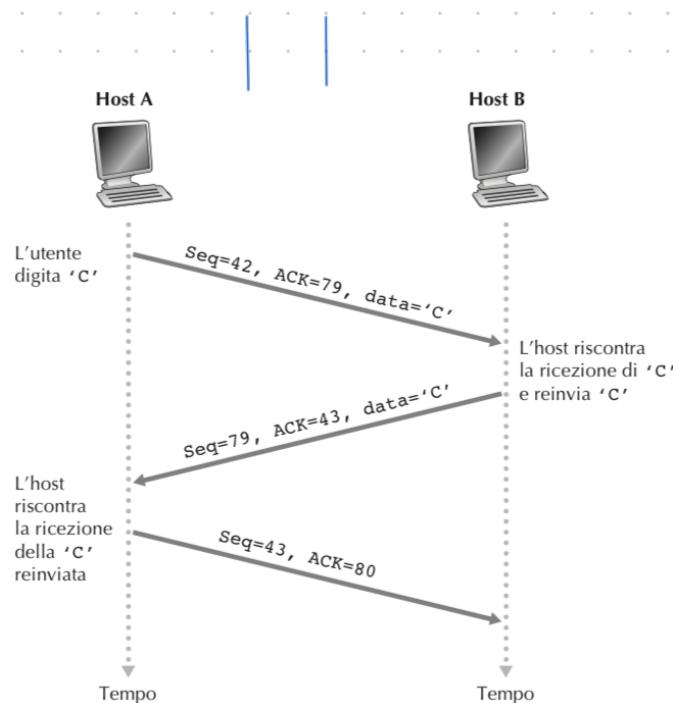
Telnet è impiegato per permettere la connessione in remoto che utilizza TCP. I dati, a differenza di SSH non vengono crittati e Telnet risulta vulnerabile agli attacchi di intercettazione.

E ogni carattere immesso dal Client verrà spedito all'host remoto che restituirà una copia di ciascun carattere che sarà mostrata nello schermo dell'utente. Quiudì ciascun carattere attraversa la rete per 2 volte.

RICORDA: NUMERO DI SEQUENZA È IL NUMERO DEL 1° BYTE NEL CAMPO DATI

I potremo dire che i numeri di sequenza sono 42 e 43.

RICORDA: IL NUMERO DI ACK È IL NUMERO DI SEQUENZA DEL SUCCESSIVO BYTE DI DATI CHE L'HOST STA ASPETTANDO.



## Esercizio

Connessione con verbale. 1 Megabit / secondo  $\rightarrow$  lunghezza di banda  
Host A deve mandare 5 frame da 2 Megabit  
Host B deve restituire un ack lungo 2 Kibit  
Calcola RTT di questo.

Messaggio lungo 20 Megabit + 2 Kibit

I Megabit sono  $10^3$  Kibit quindi  $10^4$  Kibit. Il messaggio che deve inviare  
 $\text{totale} = 10^4 + 2 = 10002 \text{ Kib} \Rightarrow 10,002 \text{ Megabit. } \Rightarrow \frac{10,002 \text{ Mb}}{\text{RTT}}$

$$\text{RTT} = 10,002 \text{ secondi.}$$

---

Potrebbe anche chiedere come calcolare la lunghezza di banda

---

Oltre la stima, viene stimata anche la distanza (DevRTT) tra la media pesata ed ogni campione misurato

$$\text{DevRTT} = (1-\beta) * \text{DevRTT} + \beta * |\text{Sample RTT} - \text{Estimated RTT}|$$

$\beta$  sarà pari a 0,25.

Per calcolare l'intervallo di timeout di trasmissione

$$\text{Timeout Interval} = \text{Estimated RTT} + 4 * \text{DevRTT}$$

Imizialmente il timer viene impostato ad 1 secondo, appena riceve il 1° ack viene ricalcolato

TCP usa anche il pipelining, consentendo al mittente di avere più segmenti trasmessi, ma non ancora riscontrati in ogni dato istante.

TCP crea un servizio di trasporto dati affidabile e best-effort di IP, assicurando che il flusso di dati in arrivo sia esattamente quello spedito (in ordine, non alterato)

Esistono tre eventi principali relativi alla trasmissione e ritrasmissione dei dati:

- dati provenienti dall'applicazione: incapsula i dati che gli giungono dall'applicazione in un segmento e lo passa a IP e se non fosse già avviato si attiva il timer.

- timeout

TCP risponde ritrasmettendo il segmento che lo ha causato e quindi riavviando il timer.

- ricezione di un ACK: è gestito dal mittente TCP, riguarda l'arrivo del segmento di acknowledgment con un valore valido nel campo ACK.

Se non ci sono segmenti che ancora necessitano di acknowledgment, riavvia il timer.

Non manda ACK

Ogni volta che TCP ritrasmette un segmento per cui è scaduto il timer e imposta quest'ultimo ad un valore doppio del precedente. Quindi gli intervalli crescono esponenzialmente ad ogni trasmissione.

da scadenza del Timer viene causata dalla congestione nella rete ossia troppi pacchetti arrivano presso una o più code dei router nel percorso tra l'origine e la destinazione

TCP usa ack cumulativi

TCP funziona come GBN ma senza ritrasmettere tutti i pacchetti in caso di scadenza del timer. Inoltre il destinatario TCP ha un buffer nel quale conserva i pacchetti che arrivano fuori sequenza: quando ne arriva uno, invece di spedire un ACK per quel pacchetto ne spedisce uno duplicato indicando il numero di sequenza del prossimo byte atteso

Se arriva un pacchetto con numero di sequenza pari a  $SendBase$ , ovvero il numero di sequenza del pacchetto più vecchio spedito, allora TCP può far avanzare la finestra di trasmissione perché è certo che i pacchetti precedenti siano stati ricevuti correttamente.

Potrebbe capitare che un ACK di conferma per il pacchetto con numero di sequenza  $SendBase$  venga perso. Se il mittente TCP riceve l'ACK per il pacchetto con numero di sequenza  $SendBase+1$ , capisce che l'ACK per il pacchetto precedente è stato perso e quindi aggiorna la sua variabile  $SendBase$  e fa scorrere la finestra: se il pacchetto con numero di sequenza  $SendBase$  non fosse stato ricevuto il destinatario TCP non avrebbe inviato l'ACK di conferma per  $SendBase+1$  ma avrebbe inviato un ACK duplicato indicando il numero di sequenza  $SendBase$ , ovvero il pacchetto atteso.

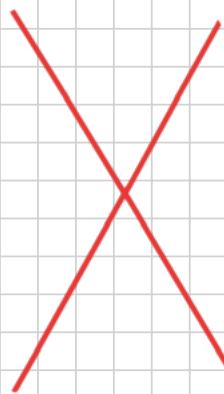
Se un ACK

## RITRASMISSIONE RAPIDA

Il periodo di timeout può riferirsi molto lungo e di conseguenza aumentano i tempi di ritrasmissione. Per questo, se il mittente riceve 3 ACK duplicati per un segmento egli considera il segmento che lo segue perduto e quindi lo ritrasmette ancor prima che il timer per il segmento smarrito scada.

Se il pacchetto viene perso ma vengono ricevuti i pacchetti  $n+1, n+2$ , questi ultimi vengono consegnati in un buffer e vengono inviati un ACK per il pacchetto atteso. Quando riceve i 3 ACK duplicati capisce che non si è perso e lo ritrasmette.

TCP offre un servizio di controllo di flusso (flow-control service) alle proprie applicazioni per evitare che il mittente saturi il buffer del ricevente che non riuscirebbe più a smaltire i pacchetti. Il controllo di flusso è pertanto un servizio di confronto sulla velocità, dato che paragona la frequenza di invio del mittente con quella di lettura dell'applicazione ricevente.



**NON Confondere con  
CONTROLLO DI Congestione**

\* Si riferisce alaffettamento che TCP applica sul mittente a causa dei ritardi dovuti al traffico in rete.

TCP offre controllo di flusso facendo mantenere al mittente una variabile chiamata finestra di ricezione (receive window) che, in sostanza, fornisce al mittente un'indicazione dello spazio libero disponibile nel buffer del destinatario. Dato che TCP è full-duplex, i due mittenti mantengono finestre di ricezione distinte.

La finestra di ricezione viene impostata uguale alla quantità di spazio disponibile nel buffer (scritto nel campo apposito del segmento TCP).

Per garantire che il buffer non venga saturato, è necessario mantenere la quantità di segmenti spediti per cui non si è ricevuto un ack di verifica, sotto al valore della finestra di ricezione.

Infatti, avremo un buffer che si occupa per la ricezione di dimensione **RcvBuffer**. Indichiamo con:

- **LastByteRead** numero dell'ultimo byte nel flusso dati che il processo applicativo ha letto dal buffer
- **LastByteRcvd** numero dell'ultimo byte che proviene dalla rete e che è stato copiato nel buffer di ricezione di 3.

Per evitare di mandare in overflow il buffer bisogna rispettare la seguente formula:

$$\text{LastByteRcvd} - \text{lastByteRead} \leq \text{RcvBuffer}.$$

La finestra di ricezione (**rwnd**) viene impostata alla quantità di spazio disponibile nel buffer.

$$(\text{rwnd}) = \text{RcvBuffer} - [\text{lastByteRcvd} - \text{lastByteRead}]$$

(dinamica)

## COME VIENE USATA LA VARIABILE RWND?

L'Host B comunica all'Host A quanto spazio disponibile sia presente nel buffer della connessione, scrivendo il valore corrente di rwnd nel campo apposito dei segmenti che manda ad A.

L'Host B inizializza rwnd con il valore di RcvBuffer e, ovviamente, deve tenere traccia di variabili specifiche per ogni connessione.

A sua volta, l'Host A tiene traccia di due variabili, LastByteSent e LastByteAcked il cui significato è rispettivamente "ultimo byte mandato" e "ultimo byte per cui si è ricevuto acknowledgment". Si noti che la differenza tra i valori di queste due variabili esprime la quantità di dati spediti da A per cui non si è ancora ricevuto un acknowledgment. Quindi per evitare di mandare in overflow il buffer l'host A deve rispettare la seguente diseguaglianza

$$\text{LastByte Sent} - \text{LastByte Acked} \leq \text{rwnd}$$

In alcuni casi rwnd potrebbe arrivare a 0, se succedesse TCP non deve permettere che l'host A blochi l'invio dei dati, ma inizia a inviare segmenti con un byte di dati a cui l'Host B risponderà con un ACK, prima o poi il buffer inizierà a svuotarsi e le varie di rwnd sarà diverso da 0.

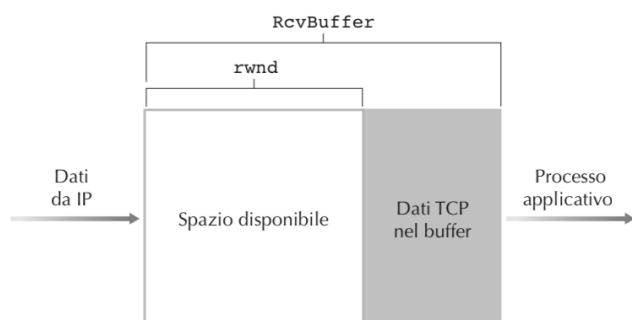


Figura 3.38 Finestra di ricezione (rwnd) e buffer di ricezione (RcvBuffer).

Sempre in questo caso, durante il riempimento del buffer ad un certo punto si libera dello spazio ed il destinatario lo comunica al mittente tramite un ACK: ma se questo messaggio viene perso? Tutti e due risultano bloccati.

Per evitare questo si usa un altro timer detto Persistent Timer che viene attivato quando viene ricevuto un messaggio di chiusura della finestra. Se il mittente non riceve più messaggi allora il timer manda un messaggio di sblocco/sveglia al destinatario "Sei ancora vivo?". Se non dovesse arrivare risposta allora la connessione viene chiusa.

### Sindrome della finestra stupida

Per evitare che ad ogni minimo spiraglio di buffer libero, il destinatario aggiorni il campo finestra, si cerca di mettere una pezza. La soluzione di Clark prevede che il ricevente non aggiorni subito la sua finestra di ricezione in caso di piccole variazioni.

**Algoritmo di Nagle:** al fine di evitare di spedire pacchetti i cui dati sono irrilevanti. L'idea di questo algoritmo è quella di non spedire un pacchetto per ogni tasto premuto ma finché non ricevo un ACK accumulo i byte e li spedisco appena possibile. L'algoritmo funziona bene quando il tempo di andata e ritorno (RTT) è elevato. Da un lato, migliora le prestazioni perché non spediamo pacchetti troppo frequentemente.

## Gestione della connessione TCP

Viene utilizzato Handshake a tre vie per dare inizio della connessione.

Il processo applicativo client informa il livello di trasporto client di voler stabilisce una connessione verso un processo nel server.

**Step 1:** TCP lato client invia uno speciale segmento al TCP lato server. Tale segmento non contiene dati ma ha il bit SYN posto a 1 (viene infatti chiamato segmento SYN). Inoltre il client sceglie a caso un numero di sequenza iniziale e lo pone nel campo numero di sequenza del segmento SYN che viene incapsulato in un datagramma IP e inviato al server.

**Step 2:** Il server TCP riceve il segmento, alloca variabili e buffer TCP e risponde con un altro segmento speciale, il quale non contiene dati, ma nella sua intestazione troviamo 3 dati fondamentali

- 1) il bit SYN è posto a 1
- 2) il campo ACK assume il valore del numero di sequenza successivo a quello scelto \*client-isn + 1\*
- 3) il server sceglie il proprio numero di sequenza iniziale \*server-isn).

Tale segmento, che indica che la connessione è stata approvata, è detto SYNACK.

**Step 3:** Ricevuto il segmento, anche il client TCP alloca variabili e buffer per la connessione TCP ed invia un terzo ed ultimo segmento speciale il quale ha il bit SYN posto a 0 ed il campo ACK con il numero di sequenza successivo a quello del server.

A differenza dei primi due segmenti, il terzo può contenere PAYLOAD ovvero dati a livello applicativo. (PIGGYBACKING) →

In realtà i numeri di sequenza di partenza non sono scelti a caso ma secondo alcuni criteri.

## Fine della connessione

Supponiamo che il client decida di chiudere la connessione.

Allora il processo applicativo client invia un comando di chiusura che forza il client TCP ad inviare un segmento TCP speciale al server il quale contiene il bit FIN con valore 1.

Il server, ricevuto il segmento, spedisce a sua volta un segmento speciale con il bit FIN a 1. Infine a sua volta il client manda un ack di verifica al server.

A questo punto tutte le risorse per la connessione risultano deallocate.

La chiusura è unidirezionale, rispetto all'apertura che invece è bidirezionale. Nel senso che, possiamo chiudere la connessione da una macchina verso l'altra, e tenere aperta la connessione dati nel senso opposto. Questo vuol dire che se il client chiude la connessione verso il server, il server può continuare a spedire dati al client, e ottenerne le relative risposte (gli ACK) senza però inviare dati.

In realtà non esiste un vero protocollo di chiusura e potrebbe accadere che da un lato la connessione rimanga aperta e dall'altro no. In realtà se il pacchetto FIN iniziale viene perso, l'host che vuole chiudere la connessione lo fa lo stesso, mentre l'altro continua ad inviare dati: quando riesce a capire che l'altro non vuole più comunicare allora la chiude.

Se il server riceve un pacchetto SYN Comporta di destinazione 80 ma al momento non sta accettando connessioni su quella porta invierà un segmento speciale di reset RST impostato a uno allo scopo di comunicare al cliente che non ha alcuna so che te per quel segmento e di non rimandare lo stesso messaggio.

*Chiedere a Riccardo cosa intendiamo con cancello di sfida nella Gestione delle connessioni TCP.*

## Controllo della Congestione

\* INDICO CON QUESTO COLORE I PROBLEMI CHE CAUSA LA CONGESTIONE

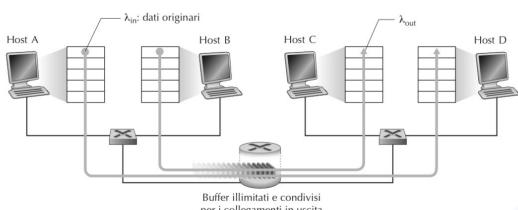


Figura 3.43 Scenario di congestione 1: due connessioni che condividono un hop con buffer illimitato.

Quando la congestione risulta molto alta è possibile che si vedano degli over flow router causando perdita di pacchetti.

Analizziamo il primo scenario due mittenti è un router con buffer illimitati

Entrambi gli host devono inviare dati sulla connessione ad una frequenza media di 17 byte. Per farlo passare attraverso il Router è un collegamento uscente condiviso di capacità R. Il Router contiene dei buffer che consentono di memorizzare i pacchetti entranti quando la loro velocità di arrivo supera la capacità del collegamento uscente. Ma se il throughput, cioè il numero di byte per secondo al ricevente supera una certa soglia e quindi il tasso di arrivo dei pacchetti si avvicina alla capacità del collegamento si rilevano lunghi ritardi di accodamento

246 Capitolo 3 – Livello di trasporto

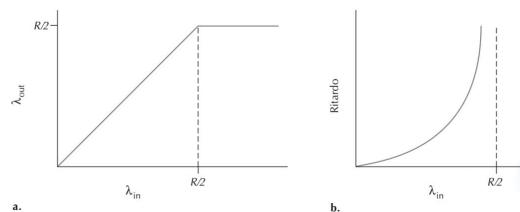


Figura 3.44 Scenario di congestione 1: throughput e ritardi in funzione della frequenza trasmittiva dell'host.

### Scenario 2: host Rutter con buffer limitato

Una conseguenza del buffer limitato è che pacchetti che giungono in un buffer già pieno saranno scartati. Di conseguenza il mittente utilizzando il Timeout li trasmetterà il pacchetto perché non ha ricevuto alcun ACK. E questo è un problema.

Un altro problema legato alla congestione è quello che potrebbero esserci delle ritrasmissioni non necessarie infatti a causa della congestione all'interno del router il pacchetto potrebbe arrivare in ritardo, è il mittente potrebbe avere il Timeout già scaduto e quindi potrebbe avere inviato un secondo pacchetto al destinatario.

Dal lato del destinatario non è un problema perché il secondo pacchetto copia vera scartato ma il router utilizza una larghezza di banda del collegamento per inviare un pacchetto già arrivato a destinazione mentre poteva utilizzarla per inviare un altro pacchetto.

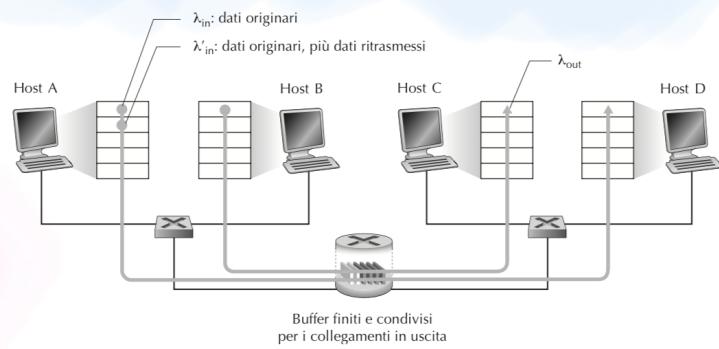


Figura 3.45 Scenario 2: due host (con ritrasmissioni) e un router con buffer di dimensione finita.

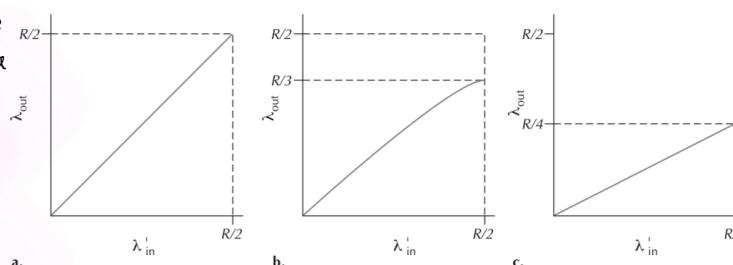
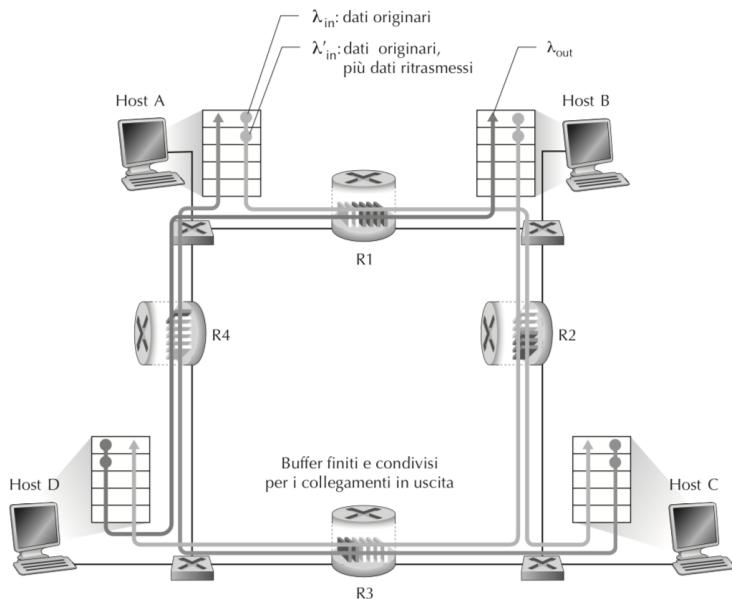
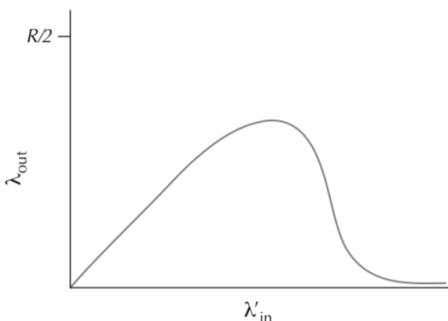


Figura 3.46 Prestazioni dello Scenario 2 con buffer di dimensione finita.



**Figura 3.47** Scenario 3: quattro mittenti, router con buffer di dimensione finita e percorsi multihop.



**Figura 3.48** Prestazioni dello Scenario 3 (buffer di dimensione finita e percorsi multihop).

da congestione viene controllato in 2 modi:

— **Controllo di congestione end-to-end**:

Ricordiamo che il mittente si rende conto della congestione a causa della perdita dei pacchetti tramite i timeout o i 3 ACK duplicati

Scenario 3.4 mittenti, Router con buffer finiti e percorsi composti da più collegamenti

Il pacchetto potrebbe perdere attraversando il Router successivi di conseguenza capiamo che è un'altra ulteriore problema è che quando un pacchetto viene scartato lungo il percorso la capacità trasmissiva utilizzata sui collegamenti velista dare il pacchetto fino a quel punto risulta essere sprecata

— **Last Byte Sent - last byte Acked < min (rwnd, cwnd)**

Modificando le varie di cwnd, il mittente può regolare la velocità di invio dei dati.

La dimensione della finestra di congestione dipende dal tempo in cui ci mettiamo ad arrivare gli ACK, quindi da RTT.

Se gli ACK arrivano con una frequenza alta  
la cwnd viene ampliata velocemente, per questo  
TCP si dice **AUTO-TEMPORIZZATO**

- Controllo di congestione assistito dalla rete tramite il controllo ABR  
i router possono comunicare quando si stanno riempendo criticamente ma ACP non vede il livello di rete e quindi non possiamo applicarci.

d'algoritmo di congestione di TCP si basa su 3 componenti principali:

### Slow Start

### Congestion Avoidance

### Fast Recovery

Quando si stabilisce una connessione TCP la finestra di congestione viene settata a 1 MSS il che comporta una velocità di invio pari a MSS/RTT.

Durante la fase iniziale detta di slow start, il valore della finestra si incrementa di 1 MSS ogni volta che un segmento trasmesso riceve un ack (prima che si verifichi un evento di perdita).

Se il valore della finestra è pari a 2 allora il mittente può inviare 2 segmenti. Se riceve gli ack per quei segmenti allora diventa 4, poi 8 e così via: la crescita è esponenziale.

Per il controllo della congestione si fa riferimento ad un'altra variabile aggiuntiva, una soglia. Quando si verifica un evento di perdita, la finestra di congestione viene settata a 1, mentre la soglia viene settata alla metà del valore che aveva la finestra prima dell'evento. Il modo in cui la slow start può terminare è quando vengono rilevati 3 ack duplicati, in quel caso utilizziamo la ritrasmissione rapida ed entra nello stato di fast recovery.

La crescita esponenziale della finestra potrebbe risultare molto pericolosa (da 16 arriviamo a 32, 64, 128...) e causare un forte aumento della congestione. Per prevenire ciò, quando il valore della finestra supera quello della soglia, la crescita passa da esponenziale a lineare: invece di raddoppiare il valore della finestra ad ogni RTT, esso viene incrementato di 1 ad ogni blocco (RTT) di trasmissione. Nel caso in cui si verifichi un evento di perdita, dopo aver aggiornato i valori della soglia e della finestra, quest'ultima riprende a crescere in maniera esponenziale.

Questo componente non è obbligatorio. In una prima versione TCP detta TCP Tahoe, qualunque fosse l'evento di perdita, il valore della finestra di ricezione veniva fatto ripartire da 1 (per poi crescere esponenzialmente).

In TCP Reno tramite Fast Recovery, quando l'evento di perdita è la ricezione di tre ack, invece di far ripartire la finestra di ricezione da 1 (e rientrare nella fase di slow start), essa riparte dal valore della soglia, seguendo un incremento lineare (congestion avoidance). Quindi dimezza la quantità di pacchetti inviati.

Il controllo di congestione di TCP è spesso indicato come incremento additivo, decremento moltiplicativo (AIMD) in quanto, in condizioni stabili la finestra viene incrementata linearmente, quando si verifica un evento di perdita, viene dimezzata. TCP Reno e Tahoe generalmente si comportano in maniera simile. Quando però essi si contendono una connessione, grazie al fast recovery, è TCP Reno ad avere la maggior porzione di banda, andando ad appropriarsi di quella che Tahoe rilascia quando avviene un evento di perdita (anche quando Reno parte dopo, vince sempre).

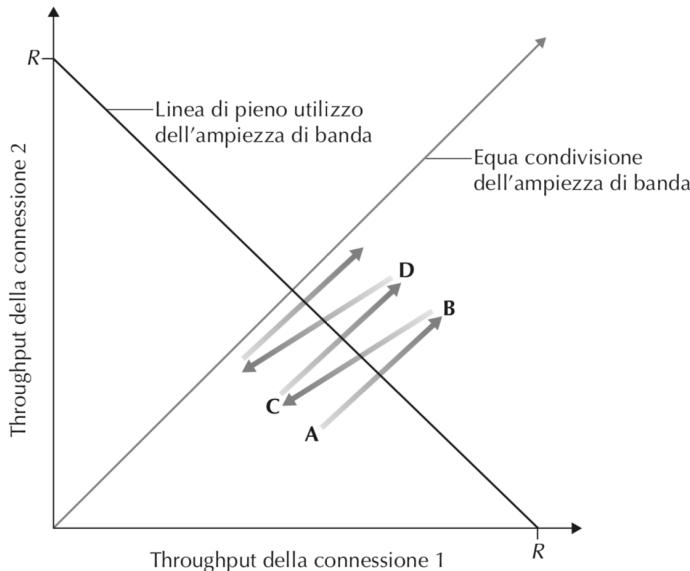
FairNess

DA TUTTO CON QUITO --> NON CI È CHIARA!

Consideriamo  $K$  connessioni TCP con un collegamento con capacità trasmissiva di  $R$  bps.

Sia dice che un meccanismo di controllo di congestione sia equo (fair) se la velocità trasmissiva media di ciascuna connessione è approssimativamente  $R/K$ ; in altre parole ciascuna connessione ottiene la stessa porzione di banda di collegamento.

Consideriamo il caso semplice di due connessioni TCP che condividono un collegamento con capacità trasmissiva  $R$ . Assumiamo che le connessioni abbiano gli stessi valori di MSS e RTT (e pertanto abbiano uguali finestre di congestione e lo stesso throughput), che debbano trasmettere una gran quantità di dati e che non vi siano altre connessioni TCP



Se TCP sta suddividendo la larghezza di banda del collegamento in modo uguale tra le due connessioni, allora il throughput dovrebbe cadere sulla bisettrice del primo quadrante. Idealmente, la somma dei due throughput dovrebbe esser uguale a  $R$ . l'obiettivo dovrebbe essere ottenere throughput situati vicino all'intersezione tra la bisettrice e la linea di massimo utilizzo della banda

I punti  $R_1$ (sotto)= $(R, 0)$  ed  $R_2$ (sopra)= $(0, R)$  indicano rispettivamente che la connessione 1 o la connessione 2 hanno preso tutta la banda disponibile. Infine il punto di coordinate  $(R/2, R/2)$  indica l'equa suddivisione della banda. Il segmento disegnato tra i punti  $R_1$  E  $R_2$  ha punti di coordinate  $(x,y)$  tale che  $x+y=R$  ovvero  $y=-x+R$ . Il coefficiente angolare di questa retta è quindi pari a -1 e risulta perpendicolare alla bisettrice. Se superiamo la bisettrice, tenderemo di nuovo verso il basso(zona a minimo potenziale). Tutti i punti che sono nel triangolo rettangolo di vertici  $OR_1R_2$  sono punti tale che la somma delle coordinate è minore o uguale a  $R$ . Quindi il segmento  $R_1R_2$  indica la linea di completa utilizzazione possibile della banda.

Anche se teoricamente non dovrebbe accadere, è possibile avere un throughput maggiore di  $R$  nel caso limite in cui ci sono dei buffer di ingresso in cui si sono accumulati dei pacchetti → può capitare di andare oltre la linea del massimo throughput.

Dimostriamo che c'è Fairness.

Supponiamo di trovarci in un punto sotto il segmento R<sub>1</sub>R<sub>2</sub>, ad esempio nel punto A oppure C. Dato che mi trovo al di sotto, le due connessioni aumentano entrambe il throughput.

Supponendo di essere nel punto P(x,y) allora il successivo punto sarà P<sub>1</sub>(x+1,y+1). Per trovare la retta passante per i due punti si fa il sistema e si trova che il coefficiente angolare di questa retta sarà 1. Quindi i punti delle connessioni si spostano su un segmento parallelo alla bisettrice (proprio per il valore del coefficiente angolare). La distanza dalla bisettrice rimane uguale quindi non miglioriamo ma nemmeno peggioriamo. Una volta che siamo nel punto B, supponiamo ci sia una perdita per entrambe le connessioni. Allora le due dimezzano i loro valori e si spostano al punto C. Questo punto C si trova nelle coordinate (x/2 e y/2). Anche qui facciamo un sistema e viene fuori che il termine noto dell'equazione della retta è zero. Quindi questa retta sarà del tipo  $y=mx$ . Dato che il termine noto è 0 allora la retta passa per l'origine. Quindi il punto C si trova esattamente a metà della retta che parte da B e passa per l'origine. Inoltre C è più vicino alla bisettrice rispetto ad A infatti: guardando il triangolo O<sub>B</sub>K ed il triangolo O<sub>C</sub>H ci accorgiamo che sono simili dato che hanno gli angoli uguali tra loro: l'angolo retto e l'angolo in H ed in K. Per questo motivo la distanza tra B e la bisettrice è esattamente il doppio della distanza tra C e la bisettrice. Quindi ci siamo avvicinati all'equità. Quando si verificano eventi di perdita allora la banda utilizzata si riduce (A), per poi ricominciare ad aumentare fino ad arrivare a B. Se a B si verificano degli eventi di perdita il tasso trasmittivo viene nuovamente ridotto fino ad arrivare a C, per poi aumentare di nuovo. Risulta importante sottolineare che le due connessioni TCP non comunicano in nessun modo. Esse sono infatti indipendenti, ma tutte le trattazioni ed accorgimenti precedentemente affrontati hanno portato alla realizzazione di un sistema che funziona da solo e che mantiene una sorta di rispetto reciproco fra le connessioni.

A differenza di TCP, le connessioni UDP non sono fair. Le applicazioni multimediali infatti, invece di dover ridurre il proprio traffico, preferiscono usare UDP e perdere pacchetti purché continuino a continuare a sostenere un alto tasso trasmittivo. Senza fairness non potrebbe esistere internet, infatti se non ci fosse questa proprietà, tutte quelle connessioni che arrivano dopo, non riuscirebbero a trovare spazio e non potrebbero comunicare.

# LIVELLO DI RETE

Il ruolo principale del livello di rete è quello di trasferire pacchetti da un host all'altro. Per fare questo bisogna identificare due concetti:

- **Inoltro (forwarding)** intendiamo l'inoltro da parte di un router quando riceve un pacchetto. Per farlo i router estraggono i campi dall'intestazione del pacchetto e li utilizzano come indice nella tabella di inoltro.

- **Instradamento (routing)** secondo cui il livello di rete deve determinare il percorso che i pacchetti devono seguire tramite algoritmi di instradamento. Si riferisce al processo globale di rete che determina i percorsi dei pacchetti nel viaggio che va dalla sorgente alla destinazione.

L'approccio tradizionale fa uso di algoritmi di routing che determinano i valori inseriti nelle tabelle di inoltro dei router.

Nella realtà oggi viene utilizzato l'approccio SDN (software-defined networking) dove un controller remoto, separata fisicamente dai router, calcola e distribuisce le tabelle di inoltro a tutti i router.

Ricordiamo che il livello di rete adotta un approccio best-effort.

**COMMUTATORE DI PACCHETTO** = indichiamo un generico dispositivo che si occupava del trasferimento dei pacchetti da un'interfaccia in ingresso a quella in uscita in base ai valori dell'intestazione del pacchetto.

All'interno del livello di rete abbiamo principalmente due tipi di servizio:



Il servizio datagram fa in modo che ogni pacchetto segua la strada indicata dal router. Questo significa che ogni pacchetto potrebbe cambiare percorso, di conseguenza la strada non è scelta a priori e per questo i pacchetti possono arrivare in ordine sparso a destinazione.

(Vantaggi) • Il guasto al sottoservizio è un enorme problema perché verrà inviato ed i pacchetti seguiranno un'altra strada

In questo metodo la strada che i pacchetti percorrono è scelta all'inizio, quindi si fa routing solo una volta e i pacchetti seguiranno esclusivamente quella strada.

- (Vantaggi)
- Non è necessario avere l'informazione completa (ogni sottoservizio).
  - Nessuna operazione in quanto tutto già stabilito.

- Gestione gestita in modo semplice. Il primo pacchetto trasporterà ad ogni sottoservizio l'informazione sulle quantità di dati che trasmetteremo così ogni router scommette alle istruzioni o no.

Possibile domanda di esame: differenza tra router, hub e switch

ROUTER: un router è un dispositivo di rete che si occupa di instradare i pacchetti di dati tra diverse reti. La sua funzione principale è quella di determinare il percorso più efficiente. Essi lavorano al livello di rete del modello OSI. I router sono in grado di controllare il traffico di rete, fornire funzionalità di sicurezza e gestire l'indirizzamento IP.

All'interno del router troviamo porte di ingresso, porte di uscita, struttura di commutazione che connette fisicamente le porte di ingresso a quelle d'uscita e un processo di instradamento.

Switch è un dispositivo di rete che viene utilizzato per creare una rete LAN connessa, consentendo la comunicazione di diversi dispositivi all'interno della stessa rete. Gli switch operano a livello di collegamento del modello OSI e utilizzano gli indirizzi MAC per instradare il traffico all'interno della rete. Gli switch sono in grado di gestire e suddividere il traffico in rete in base agli indirizzi MAC consentendo una comunicazione diretta tra i dispositivi collegati.

HUB è un dispositivo di rete passivo che funge da punto di connessione per i dispositivi in rete. Quando un dispositivo invia un pacchetto dati all'hub quest'ultimo lo trasmette a tutti gli altri dispositivi collegati ad esso. Gli hub operano a livello fisico. Tutti i dispositivi connessi all'hub condividono la lunghezza di banda e possono causare collisioni di pacchetti.

## IPv4

Il protocollo di rete viene chiamato protocollo IP e possiamo trovarci davanti a due versioni:

IPv4

IPv6

Il pacchetto a livello di rete è definito datagramma.

I datagrammi IP hanno 20 byte di intestazione (header), escludendo le opzioni. I datagrammi non frammentati che trasportano segmenti TCP ne hanno 40 byte complessivi di intestazione: 20 di intestazione IP più 20 di intestazione TCP, assieme al messaggio di livello applicativo.

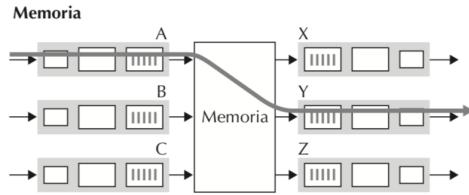
# Domanda

Perché TCP/IP effettua la verifica di errore sia a livello di trasporto che di rete?

Esistono vari motivi per questa ripetizione. Innanzitutto, notiamo che a livello IP il checksum riguarda soltanto l'intestazione, mentre il checksum TCP/UDP è calcolato sull'intero segmento. In secondo luogo, TCP/UDP e IP non appartengono necessariamente alla stessa pila di protocolli. TCP, in linea di principio, può essere usato su un protocollo diverso (per esempio, ATM) [Black 1995] e IP può trasportare dati che non verranno passati a TCP/UDP.

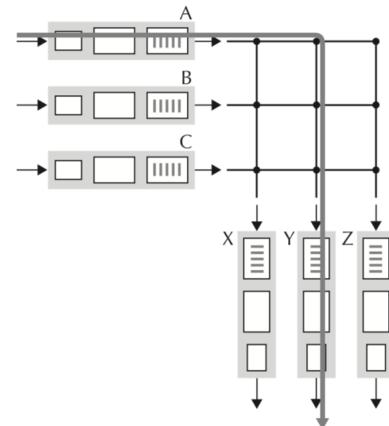
# COMMUTAZIONE

*In Memoria*

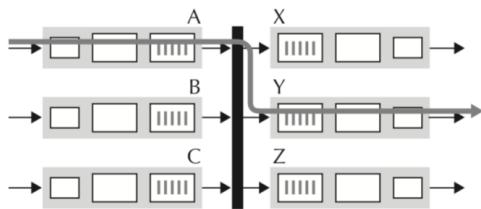


*Interconnessione*

Crossbar



Bus



Legenda:

Porta di ingresso      Porta di uscita

C viene posta un'etichetta e vengono spediti ad ogni porta di output e viene tenuto salvo quelli che corrispondono all'etichetta.

## ACCODAMENTO IN ENTRATA

→ FCFS (pag 266)  
B Poco in Testa alla code

## ACCODAMENTO IN USCITA

Quando vi sono più pacchetti accodati sulle porte di uscita, uno scheduler di pacchetti (packet scheduler) deve stabilire in quale ordine trasmetterli

Quindi, un collegamento a 10 Gbps con un RTT di 250 ms avrebbe bisogno di un buffer pari a  $B = RTT \times C = 2.5 \text{ Gbit}$ . Recenti studi teorici e sperimentali [Ap-penzeller 2004], tuttavia, suggeriscono che quando c'è un gran numero di flussi TCP ( $N$ ) che passano attraverso un collegamento, la quantità di buffer necessaria sia:

$$B = \frac{RTT \cdot C}{\sqrt{N}}$$

## SCHEDULAZIONE DEI PACCHETTI

TIFO

Coda con  
priorità

Round  
Robin

I pacchetti sono suddivisi in classi e le classi Robin prevede un'alternanza della trasmissione: prima viene inviato un pacchetto della classe 1 e poi uno della classe 2. Nella modalità conservativa i collegamenti non restano mai inattivi: quindi ci sono pacchetti.

Pag 311

# FORMATO DATAGRAMMA IPV4

Il numero di versione è formato da 4 bit e indica la versione del protocollo IP così che il router può interpretare correttamente il datagramma.

La lunghezza dell'intestazione contiene un numero variabile di opzioni (incluso nell'intestazione). Il datagramma IP ha un'intestazione di 20 byte.

Identificatore, flag, offset di frammentazione. Questi tre campi servono per la cosiddetta frammentazione.

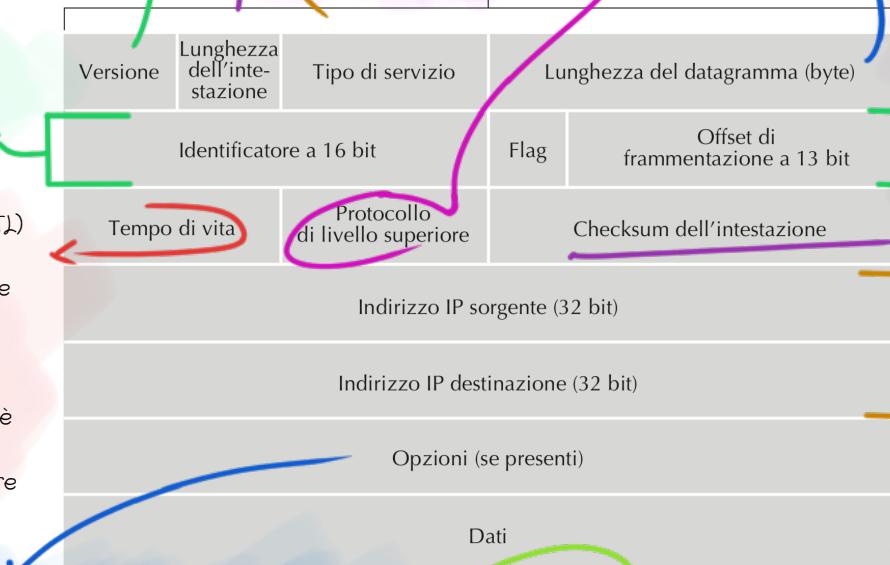
Tempo di vita. Il campo time-to-live (TTL) è stato incluso per assicurare che i datagrammi non restino in circolazione per sempre nella rete (per esempio, a causa di un instradamento ciclico). Questo campo viene decrementato di un'unità ogni volta che il datagramma è elaborato da un router; quando raggiunge 0, il datagramma deve essere scartato.

Opzioni. Questi campi consentono di estendere l'intestazione IP. Le opzioni dell'intestazione sono state concepite per un utilizzo sporadico le opzioni costituiscono un problema: dato che possono avere lunghezza variabile, non è possibile determinare a priori dove comincerà il campo dati. Inoltre, dato che i datagrammi possono richiedere o non richiedere l'elaborazione delle opzioni, il tempo necessario per questa operazione su un router può variare in modo significativo. E quindi sono state eliminate da IPv6.

Tipi di servizio distinguono i vari tipi di datagrammi (quelli che richiedono basso ritardo, alto throughput o affidabilità).

Protocollo. Questo campo è usato quando il datagramma raggiunge la destinazione finale. Il valore del campo indica lo specifico protocollo a livello di trasporto al quale vanno passati i dati del datagramma. È il collegamento tra livello di rete e livello di trasporto

32 bit



Dati

Dati (payload). Nella maggior parte dei casi, il campo dati contiene il segmento a livello di trasporto (TCP o UDP) da consegnare alla destinazione. Tuttavia, può trasportare anche altri tipi di dati, quali i messaggi ICMP.

Lunghezza del Datagramma è la somma tra intestazione e dati misurata in byte. La massima dimensione è di 65535 byte, anche se pur di non superare la lunghezza massima dei frame di Ethernet si limitano a 1500 byte.

Checksum dell'intestazione. Consente ai router di rilevare gli errori sui bit nei datagrammi ricevuti. È calcolato trattando ogni coppia di byte dell'intestazione come numeri che sono poi sommati in complemento a 1. Un router calcola tale valore per ciascun datagramma IP ricevuto e rileva una condizione di errore se il checksum trasportato nell'intestazione del datagramma non corrisponde a quello calcolato. I router normalmente scartano i datagrammi in cui si verifica un errore.

Indirizzi IP sorgente e destinazione. Quando un host crea un datagramma, inserisce il proprio indirizzo IP nel campo indirizzo IP dell'origine e quello della destinazione nel campo indirizzo IP di destinazione. Spesso l'host sorgente determina l'indirizzo di destinazione attraverso una ricerca DNS.

**MTU:** Maximum Transmission unit cioè la massima quantità di dati che un frame, a livello di collegamento, può trasportare da un router all'altro. Esso ha un limite ed è per questo che bisogna frammentare i dati del datagramma IP in 2 o più datagrammi IP, detti **frammenti**, e quindi trasferirli sul collegamento in uscita.

**Attenzione:** i DATAGRAMMI DOVRANNO ESSERE RIASSEMBLATI PRIMA DI RAGGIUNGERE IL LIVELLO DI TRASPORTO E DI QUESTO SE NE OCCUPANO I SISTEMI PERIFERICI ANZICHÉ I ROUTER INTERNI.

Affinché questo sia possibile nell'intestazione del datagramma IP sono stati messi i campi identificazione, flag e offset di frammentazione.



Quando crea un datagramma, l'host lo contrassegna con un numero identificativo e con gli indirizzi di sorgente e di destinazione. Generalmente, l'host di invio incrementa l'identificativo di ciascun datagramma che invia. Quando il router frammenta il datagramma, contrassegna i frammenti con gli indirizzi di sorgente e di destinazione e con l'identificatore numerico del datagramma originario.

alcuni frammenti potrebbero non giungere mai a destinazione. Per questo motivo, affinché l'host di destinazione sia assolutamente certo di aver ricevuto tutti i frammenti e sia in grado di riassestarli in modo corretto, l'ultimo ha il campo posto a 0, mentre in tutti gli altri è posto a 1. Si utilizza infine il campo di offset per specificare l'esatto ordine che i frammenti avevano originariamente all'interno del datagramma IP e per determinare se un frammento è stato perduto.



# INDIRIZZAMENTO IPV4

**INTERFACCIA:** IP confine tra host e collegamento fisico e ognuna ha un IP.

Un router deve essere connesso ad almeno 2 host. IP router presenta un'interfaccia per ogni suo collegamento.

Gli indirizzi IP sono di 32 bit (4 byte) e in totale ci sono  $2^{32}$  indirizzi IP, circa 4 miliardi. Tali indirizzi sono sovrapposti in notazione decimale puntata.

Tali indirizzi non possono essere scelti in modo arbitrario. Una parte dell'indirizzo di un'interfaccia è determinata dalla sottorete cui è collegata.

**SottoRete:** {rete che interconnette le interfacce di host con le interfacce di un router. IP assegna ad una sottorete l'indirizzo 223.1.2.0/24 dove la notazione /24 è detta anche maschera di sottorete dove i primi 24 bit, dopo essere stati convertiti in binario sono chiamati anche prefisso.

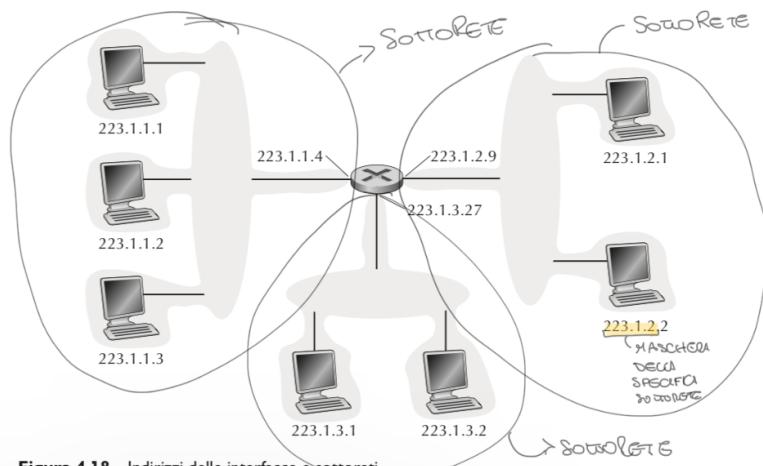


Figura 4.18 Indirizzi delle interfacce e sottoreti

**CLASSLESS INTERDOMAIN Routing:** IP CIDR (classless) generalizza la notazione di indirizzamento della sottorete.

d'indirizzo IP viene diviso in 2 parti e mantengono la forma decimale puntata a.b.c.d/24 dove 24 indica il numero di bit nella prima parte dell'indirizzo.

Gli 32-24 bit vengono utilizzati per identificare i dispositivi interconnessi all'organizzazione, che hanno tutti lo stesso prefisso.

**IP BROADCAST:** 255.255.255.255, il messaggio viene inviato a tutti gli host della stessa sottorete. I router si riservano la possibilità di inviare i messaggi alle sottoreti confinanti.

Blocco dell'ISP	200.23.16.0/20	<u>11001000 00010111 00010000 00000000</u>
Organizzazione 0	200.23.16.0/23	<u>11001000 00010111 00010000 00000000</u>
Organizzazione 1	200.23.18.0/23	<u>11001000 00010111 00010010 00000000</u>
Organizzazione 2	200.23.20.0/23	<u>11001000 00010111 00010100 00000000</u>
...	...	...
Organizzazione 7	200.23.30.0/23	<u>11001000 00010111 00011110 00000000</u>

# DHCP: ottenere l'indirizzo di un host

Gli indirizzi degli host possono essere configurati manualmente, ma più spesso questo compito è svolto utilizzando il dynamic host configuration protocol (DHCP) [1]. DHCP consente a un host di ottenere un indirizzo IP in modo automatico.

L'amministratore di rete può configurare DHCP di modo che un dato host riceva un indirizzo IP persistente o temporaneo (e quindi cambia ogni volta che l'host si connette alla rete).

DHCP viene spesso detto protocollo plug-and-play o zero-conf (zero-configuration) per la sua capacità di automatizzare la connessione degli host alla rete.

DHCP è un protocollo client-server. Per i nuovi host, il protocollo DHCP si articola in quattro punti:

Il primo compito di un host appena collegato è l'identificazione del server DHCP con il quale interagire. Questa operazione è svolta utilizzando un messaggio DHCP discover, che un client invia in un pacchetto UDP attraverso la porta 67. Il pacchetto UDP è encapsulato in un datagramma IP.

Il client DHCP crea un datagramma IP contenente il suo messaggio DHCP con l'indirizzo IP di destinazione broadcast di 255.255.255.255 e l'indirizzo IP sorgente di 0.0.0.0, cioè "questo host". Il client DHCP inoltra il datagramma IP al suo livello di collegamento, che invia il frame in broadcast a tutti i nodi collegati alla sottorete.

Un server DHCP, che riceve un messaggio di identificazione, risponde al client con un messaggio DHCP offer, che viene inviato in broadcast a tutti i nodi della sottorete, usando di nuovo l'indirizzo IP broadcast 255.255.255.255 così che il client possa ricevere tutte le proposte dei server e scegliere la migliore (in base per esempio al tempo in cui l'IP sarà valido).

## Richiesta DHCP.

Il client appena collegato sceglierà tra le offerte dei server e risponderà con un messaggio DHCP request

## Conferma DHCP

Il server risponde al messaggio di richiesta DHCP con un messaggio DHCP ACK, che conferma i parametri richiesti.

Quando il client riceve il DHCP ACK, l'interazione è completata e il client può utilizzare l'indirizzo IP fornito da DHCP per la durata della concessione.

# NAT (network address translation)

Lo spazio di indirizzamento 10.0.0/8 è una delle tre parti dello spazio di indirizzi IP riservato alle reti private [RFC 1918], o reame (realm) con indirizzi privati: ossia, una rete i cui indirizzi hanno significato solo per i dispositivi interni: quelli inviati sull'Internet globale non possono utilizzare questi indirizzi come sorgente o destinazione.

I router abilitati al NAT non appaiono come router al mondo esterno, ma si comportano come un unico dispositivo con un unico indirizzo IP.

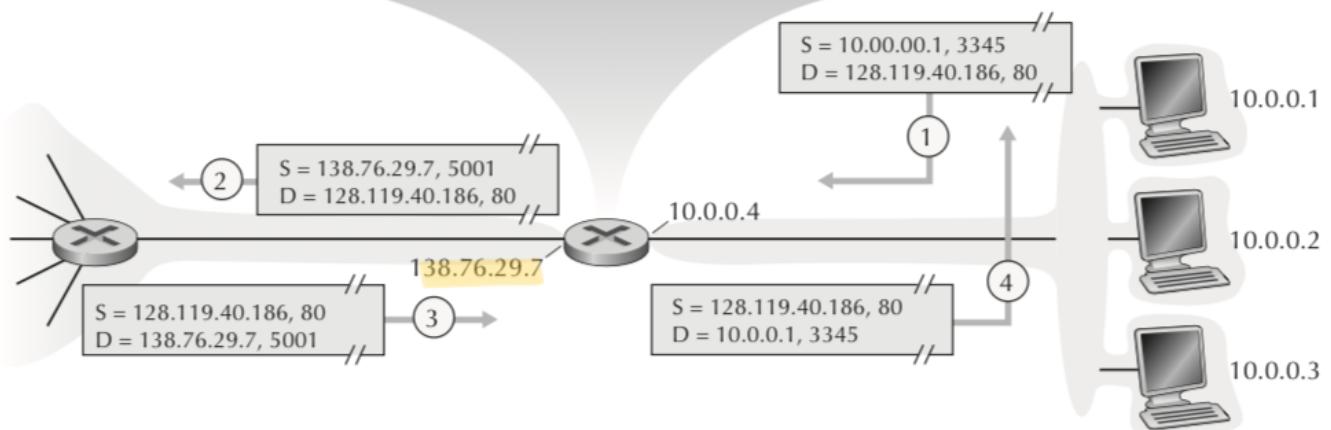
Il router abilitato al NAT nasconde i dettagli della rete domestica al mondo esterno.

Ma come fa il router a dirigere il traffico verso un determinato host se il suo indirizzo è sconosciuto dai dispositivi esterni? Viene effettuata una sorta di traduzione tramite la tabella di traduzione NAT in cui oltre agli indirizzi IP vengono usati anche i numeri di porta.

Supponiamo che l'host con IP 10.0.0.1 voglia effettuare una richiesta verso il server con IP 128.119.40.186. L'host assegna un numero di porta di origine arbitrario, ed invia il datagramma alla rete locale. Il router NAT riceve il datagramma, genera un nuovo numero di porta di origine e sostituisce l'indirizzo IP di origine dell'host con il proprio IP sul lato WAN (ossia quello riconosciuto dal mondo esterno).

Il NAT del router aggiunge un nuovo record alla propria tabella di traduzione NAT che associa all'IP del router con quel numero di porta, l'IP dell'host client con il suo numero di porta. Il router infine spedisce il datagramma al server, il quale ignaro della manipolazione avvenuta, risponde con l'oggetto richiesto. Quando il datagramma originato dal server arriva al router, questo consulta la tabella di traduzione NAT per ottenere l'indirizzo IP corrispondente al numero di porta del pacchetto, ed invia all'host client il datagramma ricevuto dal server esterno.

Tabella di traduzione NAT	
Lato WAN	Lato LAN
138.76.29.7, 5001	10.0.0.1, 3345
...	...



## Problemi di NAT

1. In pratica per quanto concerne il mondo esterno, è come se tutte le operazioni venissero eseguite dalla stessa macchina.
2. Utilizzo della porta: essendo il numero di porta un numero a 16bit il router può scegliere tra circa 65000 porte, ma in caso di LAN molto grandi, egli potrebbe saturarsi per via delle troppe connessioni.
3. Una volta usato un numero di porta per una connessione, esso non potrà essere utilizzato per comunicare fintanto che non sarà cancellato dalla tabella NAT.
4. Se un pacchetto arriva dal mondo esterno il NAT, non essendoci un record in tabella, non sa che traduzione fare e quindi lo scarta. Con NAT la comunicazione deve sempre partire dalla LAN al mondo esterno.

Soluzioni:

- Mettere una riga statiche nella tabella, indicando un aparto di entrata che gira in automatico il pacchetto ad una data macchina nel lato LAN: UpnP.

### UpnP

Protocollo che permette ad un host di mandare un messaggio al router NAT in modo settare di default che il traffico proveniente dall'esterno verso la LAN venga indirizzato all'host stesso.

Tali record nella tabella NAT hanno una scadenza.

## IPv6 M1 Im due sema IPv6 ha un'etichettatura di flussi.

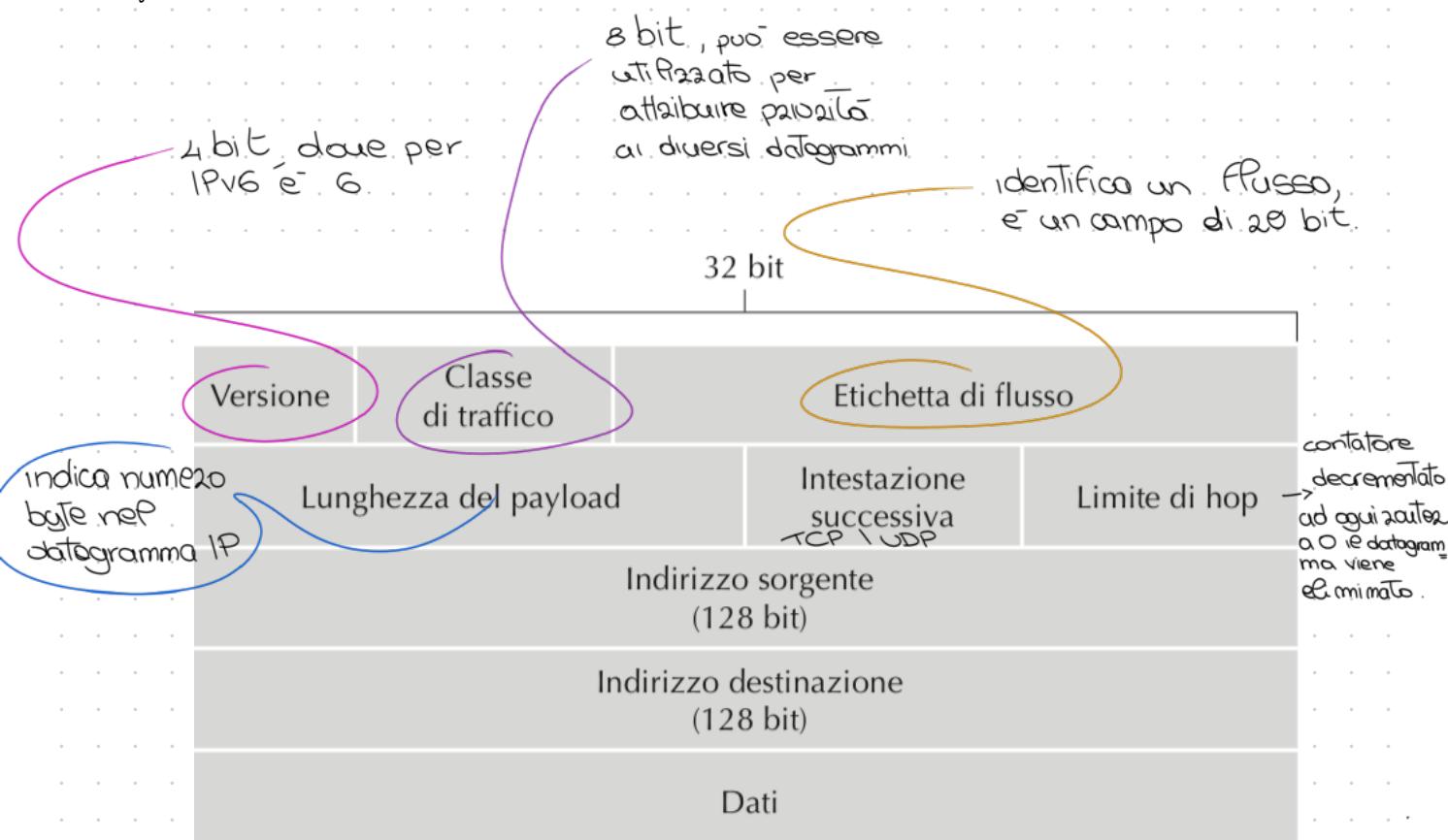
Venne creato perché lo spazio IP di 32 bit stava iniziando a sostituirsi.

### FORMATO DATAGRAMMI IPv6.

IPv6 aumenta la dimensione dell'indirizzo IP da 32 a 128 bit. IPv6 supporta anche indirizzi anycast, che consentono di consegnare un datagramma a un qualsiasi host all'interno di un gruppo. Questa caratteristica potrebbe essere usata, per esempio, per inviare una GET HTTP al più vicino di una serie di siti contenenti un dato documento.

La lunghezza dell'intestazione è di 40 byte

IPv6 presenta una definizione elusiva di flusso (flow). L'RFC 2460 afferma che ciò consente "l'etichettatura di pacchetti che appartengono a flussi particolari per i quali il mittente richiede una gestione speciale, come una qualità di servizio diversa da quella di default o un servizio in tempo reale". La trasmissione audio e video potrebbe essere trattata per esempio come un flusso



IPv6 non consente frammentazione né riassemblaggio sui router intermedi; queste operazioni possono essere effettuate soltanto da sorgente o destinazione. Se un router riceve un datagramma IPv6 che risulta troppo grande per essere inoltrato sul collegamento di uscita, non fa altro che eliminarlo e inviare al mittente un messaggio d'errore ICMP "Pacchetto troppo grande". La frammentazione e il riassemblaggio sono operazioni che consumano tempo; trasferire l'onere di questa funzionalità dai router ai sistemi periferici rende assai più rapido l'instradamento IP all'interno della rete.

A differenza di IPv4 non troviamo in IPv6 la checksum dell'intestazione poiché considerata una funzione ridondante e troppo onerosa poiché a controllo fare ci pensava già a livello di collegamento e a livello di trasporto.

È stato eliminato il campo **Ozioni**.

Passaggio da IPv4 a IPv6



Il problema è che, mentre i nuovi sistemi IPv6 sono retrocompatibili, ossia sono in grado d'inviare, instradare e ricevere datagrammi IPv4, i sistemi IPv4 esistenti non sono in grado di gestire datagrammi IPv6. Per ovviare a questo handicap esistono diverse opzioni



Una via sarebbe quella di dichiarare una "giornata campale" in cui tutte le macchine Internet vengano spente e aggiornate da IPv4 a IPv6. È una via impercorribile



Usiamo **tunnelling**.

Supponiamo che due nodi IPv6 vogliano utilizzare datagrammi IPv6, ma siano connessi da un insieme di router intermedi IPv4, che chiameremo tunnel.

Il Nodo B, al lato di invio del tunnel, prende l'intero datagramma IPv6 pervenutogli da A e lo pone nel campo dati di un datagramma IPv4. Quest'ultimo viene quindi indirizzato al Nodo E, al lato di ricezione del tunnel e inviato al primo nodo nel tunnel (C). I router IPv4 intermedi instradano il datagramma IPv4, come farebbero per qualsiasi altro datagramma, ignari che questo contenga un datagramma IPv6 completo. Il nodo IPv6, sul lato di ricezione del tunnel, riceverà quindi il datagramma IPv4, determinerà che questo ne contiene uno IPv6 osservando che il valore del campo numero di protocollo nel pacchetto IPv4 è 41 [RFC 4213] corrispondente a payload IPv6, lo estrarrà e lo instradera esattamente come se l'avesse ricevuto da un nodo IPv6 adiacente.



# Inoltro generalizzato e SDN

l'inoltro basato sulla destinazione è caratterizzato da due passi: ricerca dell'indirizzo IP di destinazione ("match"), quindi invio del pacchetto attraverso la struttura di commutazione a una specifica porta di uscita ("action").



Nell'inoltro generalizzato, una tabella match-action generalizza il concetto di tabella di inoltro basata sulla destinazione

Ogni occorrenza (riga) in una tabella di inoltro match-action, nota come tabella dei flussi (flow table) in OpenFlow, contiene quanto segue.

→ Un insieme di valori dei campi dell'intestazione con i quali il pacchetto entrante verrà confrontato. Il confronto implementato a livello hardware è effettuato molto più rapidamente nelle memorie TCAM, che permettono più di un milione di occorrenze di indirizzi di destinazione. Un pacchetto che non abbia riscontro in alcuna occorrenza della tabella dei flussi può essere scartato o inviato a un controller remoto per ulteriori elaborazioni. Nella pratica, una tabella dei flussi può essere implementata attraverso molteplici tabelle per ragioni di prestazioni o di costo

→ Un insieme di contatori che vengono aggiornati quando i pacchetti vengono associati a un'occorrenza nella tabella dei flussi. Tali contatori possono contenere il numero di pacchetti associati a tale occorrenza e l'informazione temporale sull'ultimo aggiornamento dell'occorrenza

→ Un insieme di azioni che devono essere intraprese quando un pacchetto è associato a un'occorrenza della tabella dei flussi. Tali azioni potrebbero essere

- l'inoltro di un pacchetto a una data porta di uscita, o inviato in broadcast a tutte le porte eccetto quella a cui è entrato o inviato in multicast ad un insieme selezionato di porte
- lo scarto del pacchetto,
- modifica dei campi nell'intestazione

Vedi Esempi del paradigma match-action in OpenFlow pag 340

SDN opera una chiara separazione tra il piano dei dati e il piano di controllo, implementando le funzioni di quest'ultimo in un controller separato e remoto rispetto alle componenti di inoltro dei router che controlla.

# Livello di rete: piano di controllo

OSPF è il protocollo di instradamento che opera all'interno della rete di un singolo ISP.

BGP è il protocollo di instradamento che serve a interconnettere tutte le reti in Internet.

SDN opera una chiara separazione tra il piano dei dati e il piano di controllo, implementando le funzioni di quest'ultimo in un controller separato e remoto rispetto alle componenti di inoltro dei router che controlla.

Esistono 2 tipi di controllo:

- Controllo locale illustra il caso in cui l'algoritmo di instradamento viene eseguito su ogni singolo router, all'interno del quale vengono effettuate sia le funzioni di inoltro (piano dei dati) che quelle di instradamento (piano di controllo). Ogni router ha una componente di instradamento che comunica con le componenti di instradamento degli altri router per calcolare la propria tabella di inoltro.

i protocolli OSPF e BGP, sono basati su tale approccio.

- Controllo logicamente centralizzato, dove il controller calcola e distribuisce le tabelle di inoltro che devono essere utilizzate da ogni router. (SDN)

ALGORITMI DI INSTRADAMENTO: Il scopo è determinare i percorsi attraverso la rete dei sottoservizi.

Per formulare i problemi di instradamento si utilizza un grafo. Ricordiamo che un grafo  $G = (N, E)$  è un insieme  $N$  di nodi e un insieme  $E$  di archi (edge), ove ciascun arco collega una coppia di nodi di  $N$ .

Nel contesto dell'instradamento a livello di rete, i nodi del grafo rappresentano i router e gli archi i loro collegamenti fisici.

PROBLEMA determinare un percorso tra l'origine e la destinazione che abbia il costo minimo.

gli algoritmi di instradamento sono classificabili come

Un algoritmo di instradamento centralizzato calcola il percorso a costo minimo tra una sorgente e una destinazione avendo una conoscenza globale e completa della rete, ha infatti informazioni complete su connettività e costi. Gli algoritmi con informazioni di Stato globali sono detti algoritmi link State

algoritmo di instradamento decentralizzato, il percorso a costo minimo viene calcolato in modo distribuito e iterativo. Nessun nodo possiede informazioni complete sul costo di tutti i collegamenti di rete. Inizialmente i nodi conoscono soltanto i costi dei collegamenti a loro incidenti. Poi, attraverso un processo iterativo e lo scambio di informazioni con i nodi adiacenti, un nodo gradualmente calcola il percorso a costo minimo verso una destinazione o un insieme di destinazioni. Un esempio è distance-vector

Bisogna anche classificare gli algoritmi in altri 2 modi:

### STATICI

i percorsi cambiano raramente, per esempio dopo il risultato di un intervento umano (modifica manuale di una Tabella di indirizzi di un router).

### DINAMICI

determinano gli stradamenti a variazioni delle condizioni del traffico. Rispondono meglio ai cambiamenti della rete ma sono anche soggetti a problemi quali l'instradamento in loop.

Un terzo modo per classificare:

### ALGORITMO SENSIBILE AL CARICO

DELLA RETE cioè ai costi dei collegamenti che variano dinamicamente a causa della congestione.

### ALGORITMO NON SENSIBILE

Instradamento "link-state" (LS) *Maggiori informazioni pag. 354*

In un instradamento link-state la topologia di rete e tutti i costi dei collegamenti sono noti, ossia disponibili in input all'algoritmo. Ciò si ottiene facendo inviare a ciascun nodo pacchetti sullo stato dei suoi collegamenti a tutti gli altri nodi della rete.

Viene utilizzato Dijkstra  $\rightarrow O(n^2)$ , con tempo logaritmico

Quando l'algoritmo Dijkstra termina, abbiamo per ciascun nodo il suo predecessore lungo il percorso a costo minimo dal nodo origine. Per ciascun predecessore abbiamo il rispettivo predecessore, e in questo modo riusciamo a costruire l'intero percorso dall'origine a tutte le destinazioni.

Im questo modo formiamo la tabella di imoēto.

Quale sono le problematiche?

link-state tiene conto delle oscillazioni causate dalla congestione in rete. Per risolvere abbiamo 2 soluzioni:

I COSTI DEL COLLEGAMENTO  
NON DIPENDONO DALLA  
QUANTITÀ DI TRAFFICO  
TRASPORTATO

I ROUTER NON DEVONO  
LANCIARE L'ESECUZIONE  
CONTTEMPORANEA NEI  
NEANCHE A DISTANZA DI  
POCHI SECONDI (AUTO-  
SINCRONIZZAZIONE) MA IN  
MODO RANDOMICO.

## Instradamento "distance-vector" (DV)

Ogni router ha un vettore di costi che viene aggiornato dinamicamente comunicando tra di loro.

Distance vector utilizza l'algoritmo Bellman-Ford ed è un algoritmo decentralizzato.

Distance - vector ha 3 caratteristiche:

DISTRIBUITO

Ciascun nodo riceve posta dell'informazione da 1 o più vicini direttamente connessi a cui restituisce i risultati.

ITERATIVO

Si ripete finché non si aggiornano le tabelle di costo.

AUTO-TERMINANTE

IP capisce se blocca spontaneamente.

Ricordiamo che l'instradamento LS è centralizzato nel senso che richiede a ciascun nodo di ottenere innanzitutto una mappa completa della rete prima di mandare in esecuzione l'algoritmo di Dijkstra.

L'algoritmo Bellman Ford è invece decentralizzato e non usa tale informazione globale. Infatti le sole informazioni detenute dal nodo sono il costo dei collegamenti verso i vicini direttamente connessi e quelle ricevute da questi vicini. Ogni nodo attende aggiornamenti dai suoi vicini quando riceve un aggiornamento calcola il proprio nuovo vettore delle distanze (riga 14) e lo distribuisce ai suoi vicini (righe 16 e 17). L'instradamento DV viene utilizzato in molti protocolli reali tra cui RIP e BGP per Internet, ISO IDRP, IPX di Novell e ARPAnet originale.

Dopo la ricezione degli aggiornamenti, i nodi ricalcolano i propri vettori delle distanze e aggiornano le tabelle di instradamento.

Dato che non vengono più inviati messaggi di aggiornamento, non si verificano ulteriori computazioni nella tabella di instradamento e l'algoritmo entra in uno stato quiescente.

Distribuzione costo Area vengono aggiornati i vari vettori.

Aumento Costo di un Area sussiste il problema di conteggio all'infinito che può essere evitato con una tecnica nota come inversione avvenente (poisoned reverse).

Se 2 strade tramite y per giungere a z, dirà una bugia a y dicendo che  $D(z) = +\infty$  anche se in realtà è una buona.

ATTENZIONE: I cicli che non riguardano 2 nodi adiacenti non verranno rilevati nella tecnica dell'inversione avvenente e quindi il conteggio all'infinito non si risolverà completamente.

## CONFRONTO TRA LS E DV

### • COMPLESSITÀ DEI MESSAGGI

LS

Ciascun nodo conosce il costo di ogni collegamento della rete e quanto implica è invio di  $O(N \times |E|)$  messaggi.

DV

C dipende dai costi dei collegamenti.

### • VELOCITÀ DI CONVERGENZA

(LS) Algoritmo  $O(N^2)$  che richiede  $O(|N||E|)$  messaggi.

### • ROBUSTEZZA

Che succede se un router si guasta, funziona male o viene sabotato?

LS → I calcoli di instradamento sono in qualche modo isolati, perché i nodi si occupano di calcolare le proprie tabelle di routing.

DV → Un errore può diffondersi all'intero rete

## OSPF

è un protocollo link-state che utilizza il flooding per inviare in broadcast le informazioni riguardo lo stato dei collegamenti e Dijkstra per trovare i percorsi minimi.

In OSPF, un router costruisce una mappa topologica, cioè un grafo, dell'intero sistema autonomo e manda in esecuzione (locale) l'algoritmo di Dijkstra per determinare un albero dei percorsi minimi verso tutte le sottoreti

In OSPF, ogni qualvolta che si verifica un cambiamento nello stato di un collegamento (per esempio, una variazione di costo o un cambiamento di disponibilità), il router manda informazioni di instradamento via broadcast a tutti gli altri router nel sistema autonomo. Inoltre, invia periodicamente lo stato dei collegamenti (almeno ogni 30 minuti), anche se questo non è cambiato.

## VANTAGGI

Sicurezza

Percorsi con lo stesso costo:

Supporto integrato per l'instradamento unicast e multicast

Supporto alle gerarchie in un dominio di instradamento

## Instradamento tra ISP: BGP

**BGP** (Border Gateway Protocol) è un protocollo di routing utilizzato per lo scambio di informazioni di routing tra le reti all'interno di Internet. Il suo ruolo è quello di consentire i e routing tra reti autonome (cioè insieme di router che appartengono ad una unica organizzazione).

È fondamentale in Internet perché è un protocollo che consente la connettività tra le reti autonome e il routing efficiente su scala globale.

**ICMP** (Internet Control Message Protocol) viene usato da host e router per scambiarsi informazioni a livello di rete: il suo uso più tipico è la notifica degli errori.

Durante una sessione di HTTP se incontrassimo il messaggio "Rete di destinazione irraggiungibile" ha origine di ICMP cioè da qualche parte un router IP non è stato in grado di trovare un percorso verso l'host specificato.

Dal punto di vista dell'architettura si trova esattamente sopra IP, dato che i suoi messaggi vengono trasportati nei datagrammi IP: ossia, i messaggi ICMP vengono trasportati come payload di IP, esattamente come i segmenti TCP o UDP.

I 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, in modo che il mittente possa determinare il datagramma che ha causato l'errore.

Il noto programma "ping" invia un messaggio ICMP di tipo 8 e codice 0 verso l'host specificato: l'host destinazione risponde con un messaggio ICMP di tipo 0 e codice 0 (echo).