

Rete: Una rete è un'interconnessione di dispositivi in grado di scambiarsi informazioni, quali sistemi terminali, router, switch e modem.

Sistemi terminali (host): Macchine di utenti finali o server

Router: Dispositivi che interconnettano reti

Switch: Dispositivi che collegano tra loro più host a livello locale

Collegamenti (link): Mezzi trasmissivi

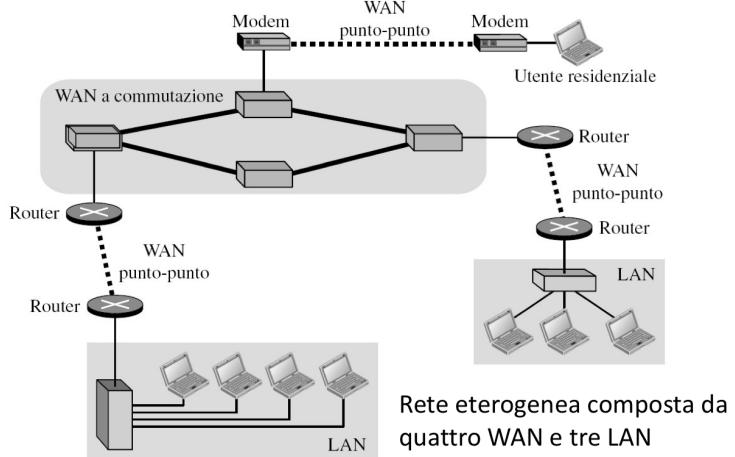
Tipi di Reti:

- 1) LAN (Local Area Network): Reti locali di computer circoscritte ad un'area limitata
- 2) WAN (Wide Area Network): Reti geografiche il quale compito è di interconnettere LAN o singoli host separati da distanze geografiche.

WAN punto-a-punto: Collega 2 dispositivi tramite mezzo trasmissivo

WAN a commutazione: Collega più di 2 punti di terminazione

INTERCONNESSIONE DI RETI:



In particolare, i dispositivi si distinguono in:

- Sistemi terminali (host)
- Dispositivi di interconnessione che si trovano nel percorso (o rotta) tra i sistemi sorgente e destinazione nella comunicazione tra host.

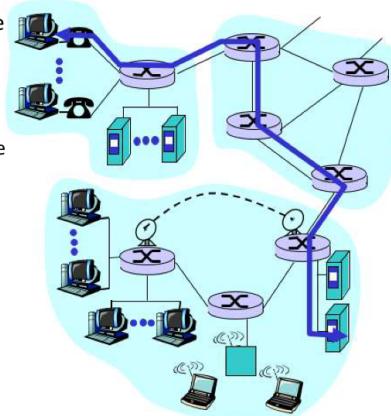
TECNICHE DI COMMUTAZIONE:

Modalità con cui viene determinato il percorso sorgente - destinazione e verso dedicato sono le risorse delle reti

Commutazione di circuito: Istantia un cammino dedicato tra i due dispositivi che vogliono comunicare. Le risorse allocate sono garantite per tutta la durata della comunicazione.

Svantaggi

- Necessaria una fase di instaurazione (setup) della comunicazione
- le risorse rimangono inattive se non utilizzate (non c'è condivisione)
 - Ad es. silenzi durante conversazione telefonica



Vantaggi

- Performance (garantita)
- Esempio: rete Telefonica fissa tradizionale (Public Switched Telephone Network)

Gli operatori di telefonia stanno però migrando le reti PSTN verso sistemi All-IP (commutazione di pacchetto).

16

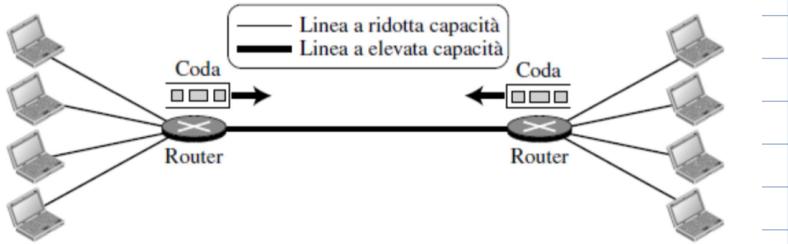
Commutazione di pacchetto: Il flusso di dati punto-punto viene suddiviso in pacchetti.

- I pacchetti degli utenti A e B condividono le risorse di rete
- Ogni pacchetto è indirizzato singolarmente e indipendentemente dagli altri pacchetti della stessa comunicazione
- Le risorse vengono usate a seconda delle necessità però:
 - 1) la richiesta delle risorse può eccedere il quantitativo disponibile
 - 2) posso avere una congestione dei pacchetti

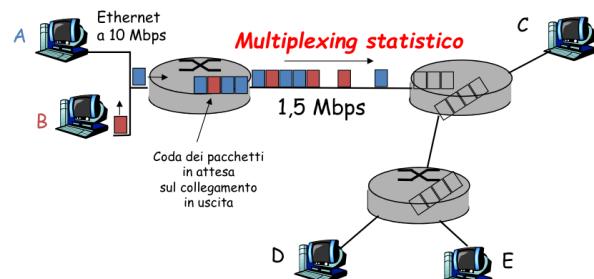
Ritardi e perdite:

1) La comunicazione deve ricevere l'intero pacchetto prima di cominciare a trasmettere sul canale di uscita → **Ritardo di Store e Forward**.

- 2) Attendo dei pacchetti in code di output (Buffer) → **Ritardo di Coda**
- 3) I buffer hanno dim. finita → **Perdita di Pacchetti**



- Non c'è un canale dedicato, gli host comunicano scambiandosi pacchetti
- I router possono memorizzare i pacchetti nelle code (buffer)
- Se il collegamento tra i due router è usato alla massima capacità, gli ulteriori pacchetti che arrivano vengono messi in coda
- Utilizzo efficiente delle risorse ma non c'è garanzia nelle prestazioni (es. ritardi)

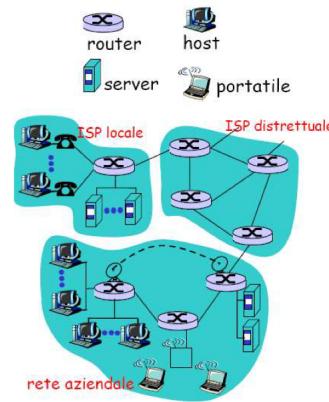


- Mentre la commutazione di circuito prealloca l'utilizzo del collegamento trasmissivo con collegamenti garantiti, nella commutazione di pacchetto, pacchetto dopo pacchetto la capacità trasmissiva dei collegamenti sarà condivisa solo tra gli utenti che devono trasmettere sul collegamento
- La sequenza dei pacchetti A e B sul collegamento a 1,5 Mbps non segue uno schema prefissato → condivisione di risorse su richiesta (detta anche multiplexing statistico delle risorse)

DIVERSE VISTE INTER NET

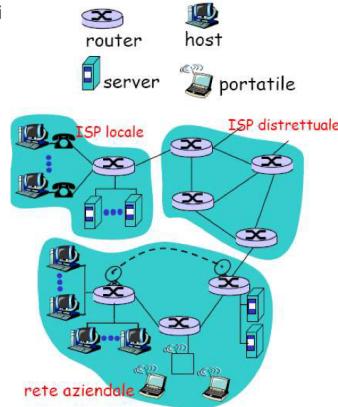
1) Vista dei Componenti :

- Milioni di dispositivi interconnessi : hosts, end-systems
 - pc, workstations, servers
 - PDA, telefoni, etc.
 che supportano le applicazioni (utenti, provider di contenuti, ecc.)
- links di comunicazione
 - fibre ottiche, doppini telefonici, cavi coassiali, onde radio...
- routers: che instradano pacchetti (sequenze) di dati attraverso la rete



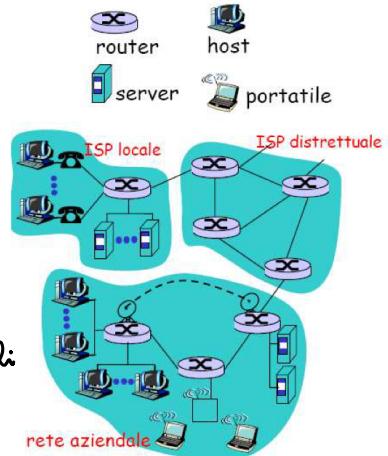
2) Vista dei Servizi :

- Infrastruttura che fornisce servizi di comunicazione alle applicazioni:
 - WWW, email, giochi, e-commerce, database, controllo remoto, Voice over IP, ecc.
 - Applicazioni distribuite che coinvolgono più host
- Fornisce servizi di comunicazione per le applicazioni:
 - Senza connessione (Connectionless): senza garanzia di consegna
 - Orientati alla connessione (connection-oriented): garantiscono integrità, completezza ed ordine

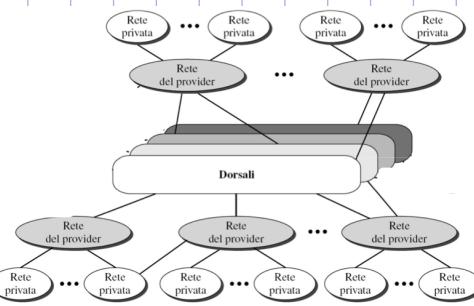


3) Vista delle Entità SW:

- Applicazioni: elaborano e si scambiano le informazioni
- Protocolli: regolamentano la trasmissione e la ricezione di messaggi
 - es., TCP, IP, HTTP, FTP, PPP
- Interfacce: definite in seguito, sono le "membrane" che separano gli strati della pila protocollare
- Gli standard Internet e del Web:
 - IETF: Internet Engineering Task Force : **standard dei protocolli**
 - W3C : **standard del web**
 - ICANN: Gestione dominio**



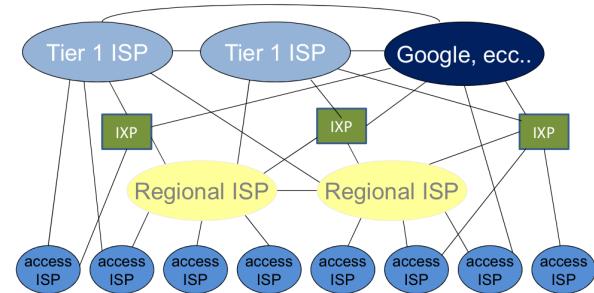
INTERNET :



- Le reti degli host sono connessi a Internet attraverso una gerarchia di fornitori di servizi Internet (Internet Service Provider)
- Le dorsali sono ISP di livello 1
 - Poche dorsali (circa 11), interconnesse tra di loro
 - ISP di secondo e terzo livello (ISP Regionali e ISP di accesso)

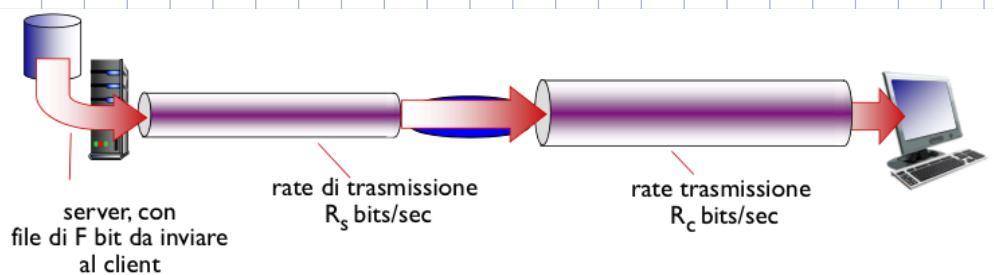
28

- Peering point: accordo tra due ISP di accettare e inoltrare il traffico che ricevono dall'altro
 - IXP: internet eXchange Point: punto d'incontro (può essere gestito da un'azienda terza) per il peering tra due o più ISP



METRICHE:

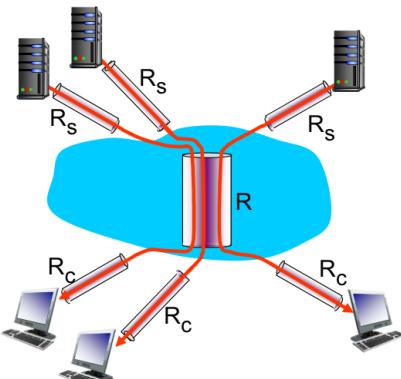
- 1) larghezza di banda: larghezza dell'intervallo di frequenze utilizzato dal sistema trasmittente. Si misura in Hz
- 2) Velocità di trasmissione: Quantità di bits che sono state trasmesse nell'unità di tempo su un certo collegamento.
- 3) throughput: Quantità di dati che possono essere trasmesse da un nodo sorgente a un nodo destinazione in un certo intervallo di tempo.



Dati i 2 canali di trasmissione

$$\text{throughput} = \min(R_s, R_c)$$

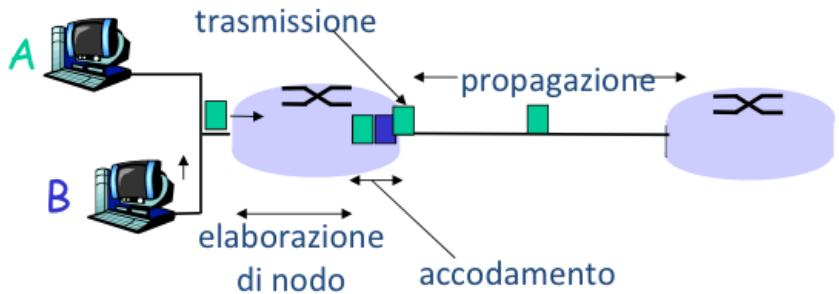
Nel caso in cui lo communi spettro (equamente) il collegamento di backbone con rate R bit/sec



$$\text{throughput} = \min(R_c, R_s, \frac{R}{10})$$

4) Latenza: tempo richiesto affinché un messaggio arrivi a destinazione dal momento in cui il primo bit parte dalla sorgente

LATENZA = ritardo di propagazione + ritardo di trasmissione +
(ritardo
nodo) ritardo di accodamento + ritardo di elaborazione



1) Ritardo di elaborazione del nodo:

Dovuto al controllo erroni sul bit, determinazione canale di uscita

2) Ritardo di accodamento:

Dovuto all'attesa di trasmissione, numero di pacchetti in entrata e uscita.

3) Ritardo di trasmissione:

Si tratta del tempo impiegato per trasmettere un pacchetto sul link

Ritardo di trasmissione = L / R

$$\downarrow \quad \downarrow \\ \text{lunghezza} \quad \text{bit rate} \\ \text{pacchetto} \quad (\text{bps}) \\ (\text{bits})$$

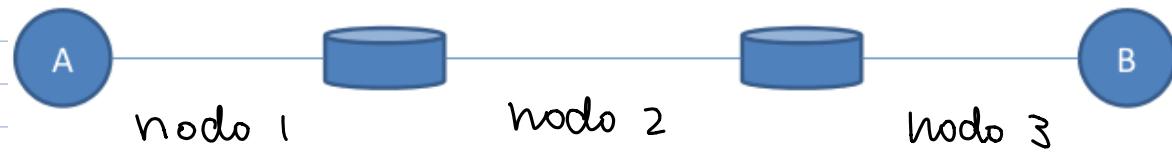
4) Ritardo di propagazione:

Tempo impiegato da 1 bit per essere propagato da un nodo all'altro

Ritardo di propagazione: d / s

$$\downarrow \quad \downarrow \\ \text{lunghezza} \quad \text{Velocità di propag.} \\ \text{Collegamento fisico} \quad \text{del mezzo} (\approx 3-2 \times 10^8 \text{ m/s}) \\ (\text{m})$$

OSS: Un un collegamento end-to-end con più nodi, si somma ciascun ritardo



$$\text{Ritardo}_{A-B} = \underset{\text{nodo 1}}{\text{Ritardo}} + \underset{\text{nodo 2}}{\text{Ritardo}} + \underset{\text{nodo 3}}{\text{Ritardo}}$$

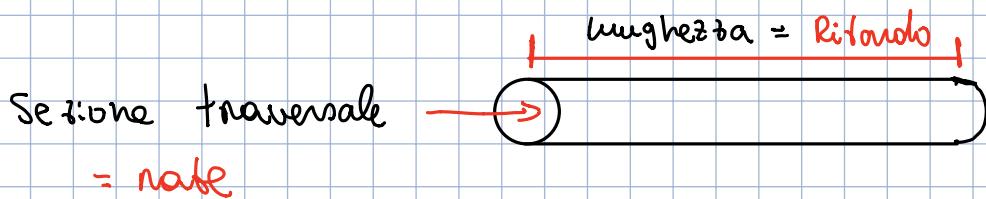


Ritardo di propagazione + ritardo di trasmissione +
ritardo di accodamento + ritardo di elaborazione
Per ciascun nodo

5) Prodotto rate - ritardo : Numero massimo di bit che un link può contenere.

esso corrisponde al volume del tubo

$$\text{Volume} = \text{rate} \times \text{ritardo}$$



Modelli Stratificati: Modelli divisi su più livelli i quali comunicano solo con livelli sopra-sotto o equivalenti

Perche stratificare: 1) la struttura esplicita permette l'identificazione delle relazioni tra gli elementi di un sistema complesso

2) la modularizzazione facilita la manutenzione e aggiornamento del sistema

Modulo: Svolge un insieme delimitato di compiti

Ciascun livello offre servizi allo strato superiore
implementandone azioni all'interno del livello interno,
utilizzando servizi del livello inferiore

3) Separazione tra servizi offerti (interfaccia) e implementazione
perché il cambiamento dell'implementazione di un servizio
in un livello è trasparente per il resto del sistema.

Come si realizza:

- **Separation of Concern**

Separazione degli interessi e delle responsabilità, fare ciò che compete, delegando ad altri tutto ciò che è delegabile

- **Information Hiding**

Nascondere tutte le informazioni che non sono indispensabili a che il committente possa comodamente definire l'operazione

Vantaggi:

- scomponere il problema in sottoproblemi più semplici da trattare -> il singolo strato è più semplice del sistema nel suo complesso
 - Semplificazione della progettazione, implementazione e manutenzione del software
- rende i vari livelli indipendenti
 - Posso modificare l'implementazione di uno strato senza dover cambiare gli altri strati (adiacenti e non), a patto che l'interfaccia non cambi
 - I servizi forniti dagli strati inferiori possono essere usati da più entità negli strati adiacenti superiori
- definendo solamente servizi e interfacce, i livelli diversi possono essere sviluppati da soggetti diversi

Criteri di Strettificazione:

- Ogni livello logico di astrazione è realizzato in un apposito strato
 - Un livello viene creato quando si rende necessario un diverso grado di astrazione
- Ogni strato svolge una sola e ben definita funzione
- Il flusso dati attraverso le interfacce di ogni strato deve essere minimizzato
- Il numero degli strati deve essere minimizzato, compatibilmente con la loro complessità
 - Numero sufficientemente alto per garantire che nessun livello sia troppo complesso e contenga troppe funzioni, ma anche sufficientemente basso per non rendere troppo onerosa l'integrazione e l'architettura poco flessibile

Protocollo: Un insieme di regole che permettono a due entità di comunicare

Tipi di sistemi

1) **Chiusi**: Tutti i componenti della rete devono essere dello stesso costruttore.

Problema:

- gli apparati non riescono ad interpretare i segnali degli altri in altre reti (parlano linguaggi diversi)

Interoperabilità

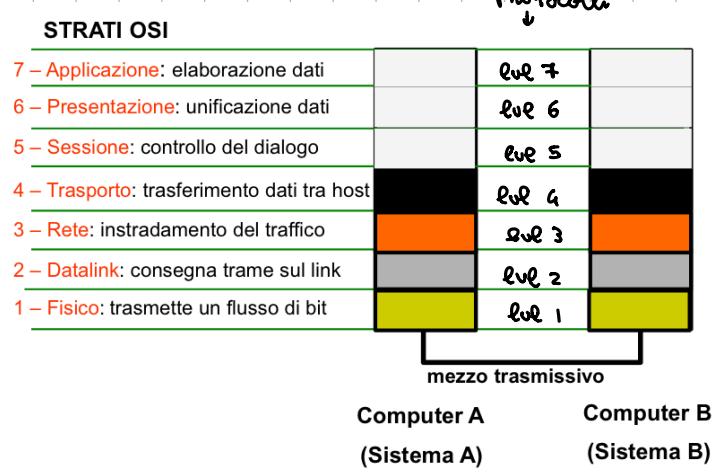
- i programmi applicativi non riescono ad operare in ambiente distribuito.

2) **Aperti**: Sistemi che implementano protocolli aperti



- Dettagli protocollo pubblici
- Cambiamenti gestiti da un'organizzazione pubblica

Modello ISO-OSI



- Ogni strato fornisce servizi allo strato superiore e riceve servizi dallo strato inferiore.
- La comunicazione avviene tramite **interfacce**

- L'n-esimo strato di una entità comunica con lo strato n-esimo di un'altra entità secondo un **protocollo** ammesso

Strato: Modulo definito attraverso i servizi, protocolli e interfacce (livello)

Servizio: Un insieme di operazioni che uno strato fornisce ad uno strato sovrastante

Interfaccia: Un insieme di regole che governano il formato e il significato delle unità dati che vengono scambiati tra due strati adiacenti della stessa entità.

Protocollo: Permette a due entità omologhe uno scambio efficace ed efficiente delle info.

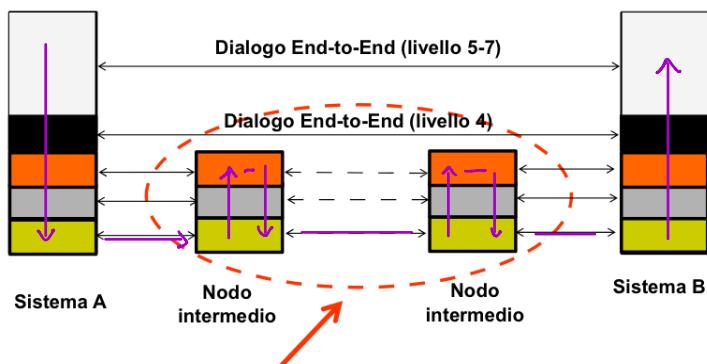
Definiscono il formato, l'ordine dei dati e le azioni.

- efficace: raggiungere scopo prefissato con maggior frequenza possibile
- efficienti: raggiungere scopo prefissato con minor sforzo possibile

Nel protocollo bisogna specificare:

- La sintassi di un messaggio
 - che campi contiene?
 - in quale formato?
- La semantica del messaggio
 - cosa significa il messaggio?
 - Es: not-OK significa che il ricevitore ha ricevuto il msg corrotto
- Le azioni da intraprendere dopo la ricezione di un msg
 - Es: dopo avere ricevuto not-OK, ritrasmettere il msg

Collegamento tra end-systems



4 nodi intermedi non hanno tutti i livelli
ma solo i primi 3/4 a seconda del dispositivo

Tipi di connessioni

1) Connection-oriented:

- Instaurazione Connessione
- Trasferimento dati
- Chiusura Connessione

2) Connection-less:

Dati trasferiti senza connessione

Flusso dell'informazione

Le informazioni hanno origine a livello applicativo, le quali poi discendono per i vari livelli fino alla trasmissione sul canale fisico.

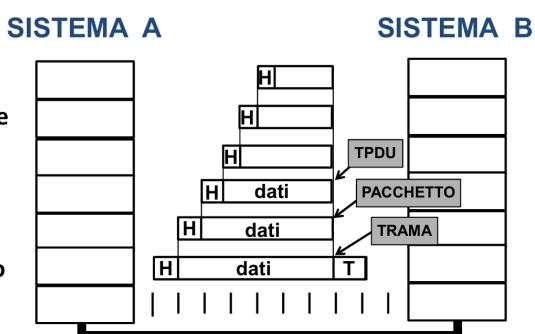
Ogni livello effettua l'incapsulamento (e viceversa) sui dati ricevuti dal livello superiore, ai quale aggiunge un header (pacchetto dati di quel livello).

I dati saranno quindi forniti così:



↓

Per le trattamenti degli errori



H = Header
T = Trailer

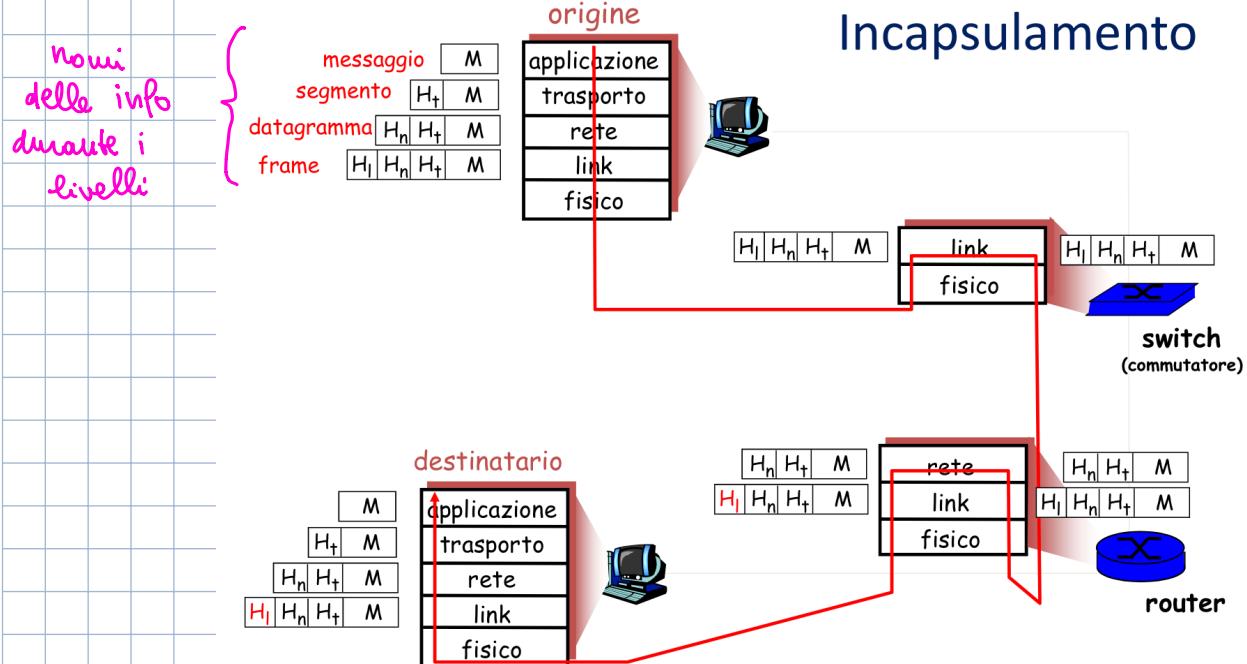
PDU = Protocol Data Unit
TPDU = Transport PDU

STACK PROTOCOLARE TCP/IP



- **applicazione:** supporta le applicazioni di rete, collegamento logico end-to-end: scambio di messaggi tra due processi
 - ftp, smtp, http
- **trasporto:** trasferimento dati end-to-end (da un host sorgente all'host destinatario)
 - tcp, udp
- **rete:** instradamento dei datagrammi dalla sorgente alla destinazione
 - Ip, ICMP
- **link:** trasferimento dati in frame attraverso il collegamento tra elementi di rete vicini
 - ppp, ethernet, ... qualunque cosa
- **Fisico:** trasferimenti dei bit di un frame sul mezzo trasmittivo

Flusso dell' informazione:



ISO/OSI vs TCP/IP

- | | |
|---|---|
| • ISO/OSI | • TCP/IP |
| + generale | + standard de facto |
| + definizione di servizio, interfaccia e protocollo | + / - implementation driven |
| - Poco efficiente, alcuni livelli poco utili | - Specifiche non astratte e rigorose |
| - Standard difficili | - Modello non generale |
| - Telco-oriented | - Oltre a TCP/IP, presenza di protocolli minori (per problemi ad-hoc), difficili da rimpiazzare |
| - Poca tempestività | |

Strato Applicativo

Applicazioni di rete: Applicazioni formate da processi distribuiti comunicanti

↓
Programmi eseguiti dai dispositivi terminali di una rete

DSS: i livelli applicazione dei due host agiscono come se esistesse un collegamento diretto attraverso il quale poter inviare e ricevere messaggi

Protocolli strato di applicazione:

- 1) Definiscono i tipi di messaggi scambiati a livello applicativo
- 2) Definiscono la sintassi dei vari tipi di messaggio
- 3) Definiscono la semantica dei campi
- 4) Definiscono le regole per determinare quando e come un processo invia messaggi o risponde ai messaggi

Paradigma:

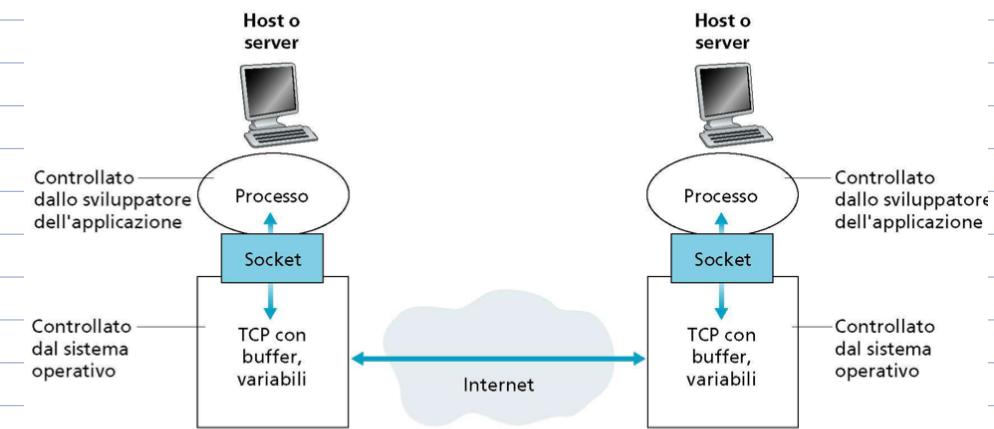
- 1) Client-Server: Client richiede un servizio, server lo offre
- 2) Peer-to-peer: Offrire servizi e inviare richieste
- 3) Mixto: Sia Client-Server che peer-to-peer

API (Application programming interface): Gruppo di regole che un programmatore deve rispettare per utilizzare delle risorse

Esempio di API:

```
connection TCPopen (IPaddress, int) //per aprire una connessione  
void TCPsend (connection, data) //per spedire dati su una connessione  
data TCPreceive (connection) //per ricevere dati su una connessione  
void TCPclose (connection) //per chiudere una connessione  
  
int TCPbind (int) //per rich. assegna. porta su cui attendere rich. di conn.  
void TCPunbind (int) //per liberare una porta  
connection TCPaccept (int) //per attendere richieste di connessione
```

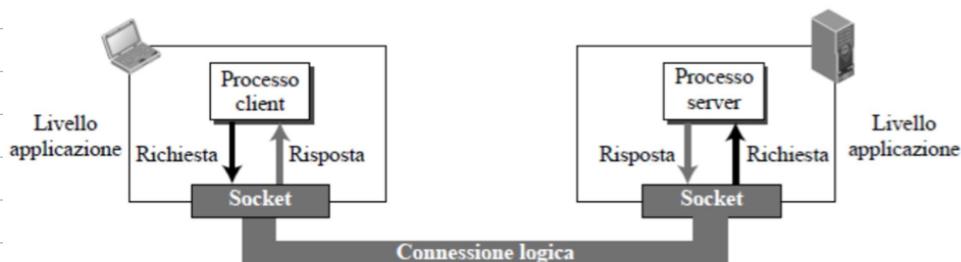
Interfaccia socket: API che funge da interfaccia tra gli strati di applicazione e di trasporto.



Due processi comunicano mandando dati alla socket, e leggendoli da questa

- Connessione logica tra i due processi, l'invio/ricezione dei dati è responsabilità dei restanti quattro livelli dello stack TCP/IP nel sistema operativo
- Socket è un'astrazione. Si tratta di una struttura dati creata e utilizzata dal programma applicativo.
- Comunicare tra un processo client e server significa comunicare tra le due socket (**endpoint della comunicazione, socket = connettore, presa**)
- Il client considera la socket come l'entità che riceve le richieste e fornisce le risposte, per il server la socket è l'entità da cui ~~riceve~~ **riceve** le richieste e a cui ~~invia~~ **invia** le risposte

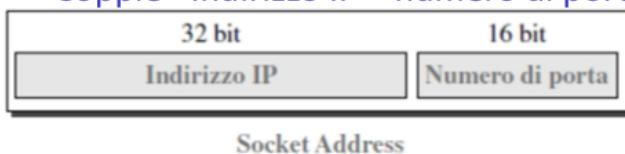
11



12

Per identificare sia l'host che il processo serve un identificativo:

Copie <Indirizzo IP + numero di porta>



Utilizzo servizi di trasporto per la comunicazione (Protocolli TCP/UDP)

Protocollo TCP:

- Connection-oriented: setup richiesto tra client e server
- Trasporto affidabile tra processo mittente e destinatario
- Controllo del flusso: il mittente non "inonderà" di dati il destinatario
- Controllo di congestione: "strangola" il mittente quando la rete è sovraccarica
- Non offre garanzie di timing né di ampiezza minima di banda

Protocollo UDP:

- Non orientato alle connessioni
- Trasporto NON affidabile
- NO controllo di flusso
- NO controllo congestione
- No garanzie timing né ampiezza minima di banda

Applicazioni Internet

applicazione	protocollo di livello applicazione	Protocollo di trasporto
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	HTTP (e.g., YouTube), RTP [RFC 1889]	TCP or UDP
Internet telephony	SIP, RTP, proprietary (e.g., Skype)	TCP or UDP

WEB

Enorme collezione di informazioni nella quale le risorse web sono distribuite e collegate l'una all'altra.

Pagine web: oggetti formati da:

- 1) File HTML di base
- 2) Diversi oggetti riferiti
- 3) Ciascun oggetto è individuabile tramite una URL (Uniform Resource Locator)

Uniform resource identifier (URI): Forma generale per identificare una risorsa presente sulla rete

- Uniform: uniformità della sintassi dell'identificatore anche se i meccanismi per accedere alle risorse possono variare
- Resource: qualsiasi cosa abbia un'identità (!)
 - Documento, servizio, immagine, collezione di altre risorse
- Identifier: informazioni che permettono di distinguere un oggetto da altri oggetti (entro un ambito definito di identificazione)

Ci sono due tipi di URI:

1) Uniform Resource Locator (URL):

Sottoinsieme di URI che identifica le risorse attraverso il loro meccanismo di accesso

- Esempi:
- <https://doi.org/10.1109/LCN.1988.10239>
 - <ftp://ftp.is.co.za/rfc/rfc1808.txt>
 - <http://www.apple.com/index.html>

Sintesi URL: <schema>://<host>:<port>/<path>

- scheme: protocollo di accesso alla risorsa
- host = nome di dominio di un host o indirizzo IP (in notazione decimale puntata),
- port = numero di porta del server. Molti protocolli hanno porte di default (es. 80 per http)
- Path: contiene dati specifici per l'host (o schema) e identifica la risorsa nel contesto di quello schema e host. Può consistere in una sequenza di segmenti separati da "/". **Ad esempio path nel file system del server ma non solo!** Il path specifica la Request-URI*

HTTP URL: <http://<host>:<port>/<path>?<query>>

- <query> stringa di informazioni che deve essere interpretata dal server

2) Uniform Resource Name:

Sottoinsieme di URI che devono rimanere globalmente unici e persistenti

anche quando la risorsa cessa di esistere e diventa non disponibile

- <urn:oasis:names:specification:docbook:dtd:xml:4.1.2>
- <urn:doi:10.1109/LCN.1988.10239>

URI Relativa e Assoluta:

1) URI assoluta: Identifica una risorsa indipendentemente dal contesto in cui è usata.

2) URI Relativa: Informazioni per identificare una risorsa in relazione ad un'altra URL

Sia <http://a/b/c/d;p?q> il documento di partenza, allora:

g = <http://a/b/c/g>
/g = <http://a/g>
//g = <http://g>
?y = <http://a/b/c/?y>
#s = (**current document**)#s
g;x?y#s = <http://a/b/c/g;x?y#s>
.. = <http://a/b/>
.../g = <http://a/g>

<https://www.ietf.org/rfc/rfc3986.txt>

Le URI Relative...

- NON viaggiano sulla rete
- Sono interpretate dal browser in relazione al documento di partenza

HTTP (HyperText transfer Protocol)

Protocollo di livello applicazione per sistemi di informazione distribuiti collaborativi ed intermediali

1) E' un protocollo client/server:

- Il client initia la connessione inviando un messaggio di richiesta (**request**)
- Il server risponde con un messaggio di risposta (**response**)

Client HTTP

- programma che stabilisce una connessione con un server HTTP e invia una richiesta per effettuare operazioni su risorse web

Server HTTP

- programma che accetta connessioni per servire richieste inviando messaggi di risposta

2) E' un protocollo stateless (Senza memoria di stato):

- Coppie richiesta/risposta sono indipendenti
- Ogni richiesta viene eseguita indipendentemente da quelle che l'hanno preceduta

3) Usa TCP (connection-oriented):

- Il client richiede l'istituzione di una connessione TCP col server
- I processi si scambiano messaggi attraverso le rispettive socket

Tipica interazione HTTP client-server:

```
//esempio client  
c = TCPopen("131.115.7.24", 80);  
TCPsend(c,"GET /index.html");  
d = TCPreceive(c);
```

```
//esempio server  
p = TCPbind(80);  
d = TCPaccept(p);  
r = TCPreceive(d)  
...  
TCPsend(d,pag)  
TCPclose(d)
```

Tipi di connessione

Una connessione è un circuito logico di livello trasporto stabilito tra due programmi applicativi per comunicare tra loro.

Connessione non persistente: Viene stabilita una connessione TCP separata per recuperare ciascuna URL.

Connessione persistente: Il client e server continuano a comunicare fin quando non avviene la chiusura della connessione.

- Vantaggi:
- 1) Vengono aperte e chiuse meno connessioni TCP
 - 2) I ritardi delle richieste successive sono minimi perché non si richiede nuovamente l'istituzione della connessione
 - 3) Gestione più efficace del controllo della congestione.

Esempio di Connessione non persistente:

Supponiamo che l'utente digiti la URL
www.someSchool.edu/someDepartment/home.index

(file html contenente testo e riferimenti a 10 immagini jpg)

1a. Il client http inizia la connessione TCP verso il server http al www.someSchool.edu. Socket Porta 80 è default per il server http.

2. Il client http invia un **messaggio di richiesta http** (che contiene la URL) alla socket di connessione con lo strato di trasporto (TCP)

1b. Il server http dell'host www.someSchool.edu aspetta le richieste di connessione TCP connection alla porta 80. "Accetta" la connessione, e lo notifica al client

3. Il server http riceve il msg di richiesta, compila un **messaggio di risposta** che contiene l'oggetto richiesto (someDepartment/home.index), manda il messaggio alla socket

4. Il server http chiude la connessione TCP

5. Il client http riceve il messaggio di risposta che contiene il file html e lo visualizza. Percorrendo il file trova il riferimento a 10 oggetti jpg

6. Ripete i passaggi da 1-5 per ognuno dei 10 oggetti jpg

Ipotesi: connessione non persistente

Il protocollo http è "stateless"

ogni risposta è collegata solo alla richiesta che l'ha generata, indipendentemente dalle richieste precedenti

tempo

- La stessa connessione HTTP può essere utilizzata per una serie di richieste e una serie corrispondente di risposte
- Il server lascia aperta la connessione TCP dopo aver spedito la risposta e può quindi ricevere le richieste successive sulla stessa connessione
- Il server HTTP chiude la connessione quando viene specificato nell'header del messaggio (desiderata da parte del cliente) oppure quando non riceve richieste per un certo intervallo di tempo (time out)

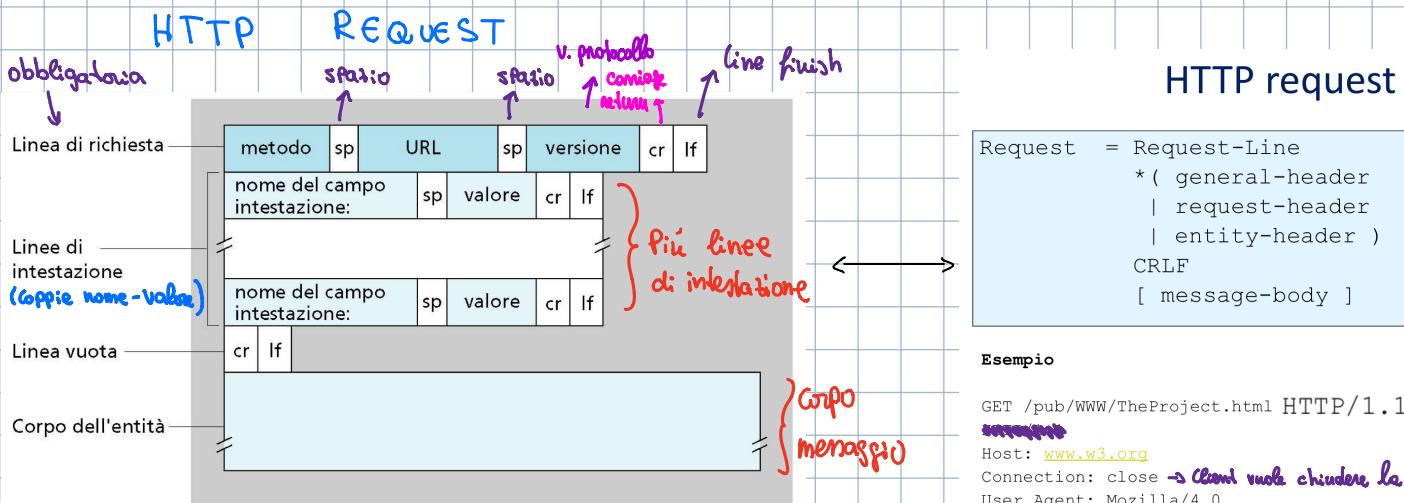
Pipelining: Consiste nell'invio da parte del client di molteplici richieste senza aspettare la ricezione di ciascuna risposta. Il server deve inviare le risposte nello stesso ordine in cui sono state ricevute le richieste.

Le richieste devono usare metodi HTTP idem-potenti.

Menaggi HTTP

I messaggi HTTP possono essere di due tipi: **Request** e **Response**

↳ Metodi che usati più volte ottengono la stessa modifica



1) Request-line

```
Request-Line = Method SP
              Request-URI SP
              HTTP-Version CRLF
```

- Method: operazione che il client richiede venga effettuata sulla risorsa identificata dalla Request-URI. Definizione della semantica dei metodi nella RFC 2616
- HTTP-Version – il mittente indica il formato del messaggio e la sua capacità di comprendere ulteriori comunicazioni HTTP

```
Method = "OPTIONS" | "GET"
       | "HEAD"    | "POST"
       | "PUT"     | "DELETE"
       | "TRACE"   | extension-method
```

- Gli header sono insiemi di coppie (nome: valore) che specificano alcuni parametri del messaggio trasmesso o ricevuto:
- **General Header** – relativi alla trasmissione
 - Es. Data, codifica, connessione,
- **Entity Header** – relativi all'entità trasmessa
 - Content-type, Content-Length, data di scadenza, ecc.
- **Request Header** – relativi alla richiesta
 - Chi fa la richiesta, a chi viene fatta la richiesta, che tipo di caratteristiche il client è in grado di accettare, autorizzazione, ecc.
- **Response Header** – nel messaggio di risposta
 - Server, autorizzazione richiesta, ecc.

↓ Esempio

General Headers	
DATA	Date: Tue, 15 Nov 1994 08:12:31 GMT Connection: close Transfer-Encoding: chunked → Max mandato su + max HTTP
Request Headers	
	Preferenza sul tipo di formato
Accept	text/plain; q=0.5, text/html, image/png, application/json
Accept-Charset	iso-8859-5, unicode-1-1;q=0.8
Accept-Encoding	compress, gzip

- Accept: specifica il formato del corpo del messaggio accettabile per una risposta. "q" indica una sorta di preferenza. Default =1
- Accept-Charset: set di caratteri accettabile per la risposta
- Accept-Encoding: tipo di codifica del contenuto accettabile per la risposta



Request methods:

1) OPTIONS: Richiediamo quali sono le operazioni offerte dal server su quelle determinata URL

```
OPTIONS http://192.168.11.66/manual/index.html  
HTTP/1.1  
host: 192.168.11.66  
Connection: close  
  
HTTP/1.1 200 OK  
Date: Sun, 14 May 2000 19:52:12 GMT  
Server: Apache/1.3.9 (Unix) (Red Hat/Linux)  
Content-Length: 0  
Allow: GET, HEAD, OPTIONS, TRACE  
Connection: close
```

→ Op. disponibili sul server

2) GET: Metodo che richiede il trasferimento delle rappresentazioni di una risorsa identificata da una URL o operazioni associate all'URL stessa

Richiesta

```
GET http://192.168.11.66 HTTP/1.1  
host: 192.168.11.66  
Connection: close
```

Solitamente la req non ha il body.

Headers

Ha solo una req. linea e degli header

body

```
HTTP/1.1 200 OK  
Date: Sun, 14 May 2000 19:57:13 GMT  
Server: Apache/1.3.9 (Unix) (Red Hat/Linux)  
Last-Modified: Tue, 21 Sep 1999 14:46:36 GMT  
ETag: "f2fc-799-37e79a4c"  
Accept-Ranges: bytes  
Content-Length: 1945 → Contenuto in byte  
Connection: close  
Content-Type: text/html  
  
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2  
Final//EN">  
<HTML> ...
```

Sono possibili:

1) Conditional get: Header contiene "if-..." → If-Modified-Since, If-Unmodified-Since (date); If-Match, If-None-Match (ETag); or If-Range

Request

```
GET http://192.168.11.66 HTTP/1.1  
Host: 192.168.11.66  
If-Modified-Since: Tue, 21 Sep 1999 14:46:36 GMT
```

Response

```
HTTP/1.1 304 Not Modified  
Date: Wed, 22 Sep 1999 15:06:36 GMT  
Server: Apache/1.3.9 (Unix) (Red Hat/Linux)
```

RISORSA NON MODIFICATA

2) Partial get: Header contiene range

3) HEAD: È simile al GET, ma il server non trasferisce il message body nella risposta.

Utile per controllare lo stato dei documenti (Validità, modifiche cache refresh)

REQUEST

```
HEAD http://192.168.11.66 HTTP/1.1  
host: 192.168.11.66  
Connection: close
```

RESPONSE

```
HTTP/1.1 200 OK  
Date: Sun, 14 May 2000 20:02:41 GMT  
Server: Apache/1.3.9 (Unix) (Red Hat/Linux)  
Last-Modified: Tue, 21 Sep 1999 14:46:36 GMT  
ETag: "f2fc-799-37e79a4c"  
Accept-Ranges: bytes  
Content-Length: 1945  
Connection: close  
Content-Type: text/html
```

4) POST: Il metodo POST serve per inviare dal client al server informazioni inserite nel body del messaggio.

Aggiungi una
struttura con le info
parametri

La funzione eseguita dal POST è determinata dal server e dipende dalle tipicamente dalla REQUEST-URI.

In teoria lo standard dice che...

Il metodo POST è usato per chiedere che il server accetti l'entità (risorsa) nel corpo della richiesta come una nuova subordinata della risorsa identificata dalla Request-URI nella Request-Line.

5) PUT: Il client richiede al server di creare/modificare una risorsa specificata con la request URI. Il server può recuperarla con la GET.

Modifica una URI
esistente

6) DELETE: Il client chiede di cancellare una risorsa identificata dalla URI.

Metodi sicuri: Metodi che non hanno effetti collaterali:

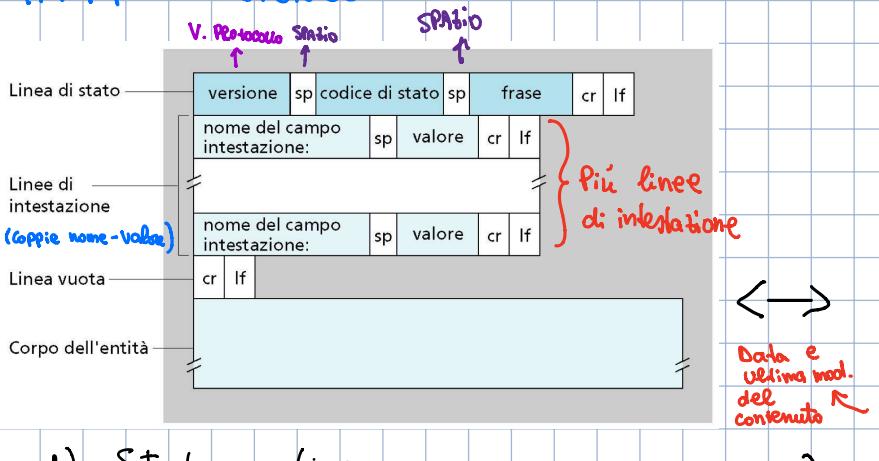
(GET, HEAD, OPTION, TRACE)

Metodi idempotenti: Metodi idempotenti negli effetti collaterali se NON richieste identiche hanno lo stesso effetto di una richiesta singola.

(GET, HEAD, PUT, DELETE, OPTIONS, TRACE)

Esempio: Cancellare più volte una risorsa → si cancella solo una volta!

HTTP RESPONSE



1) Status - line

Status-Line
HTTP-Version SP
Status-Code SP
Reason-Phrase CRLF

Esempio: HTTP/1.1 200 OK

- Status-Line: prima riga di un messaggio di risposta
 - Status-Code: intero a tre cifre che rappresenta un codice di risultato dell'operazione richiesta
 - Reason-Phrase: descrizione testuale dello Status code (pensata per l'utente umano)

HTTP response

```
Response = Status-Line
*( general-header
| response-header
| entity-header )
CRLF
[ message-body ]
```



Dopo è
ultima mod.
del
contenuto

HTTP/1.1 200 OK
Date: Sun, 14 May 2000 23:49:39 GMT
Server: Apache/1.3.9 (Unix) (Red Hat/Linux)
Last-Modified: Tue, 21 Sep 1999 14:46:36 GMT

2) Response - header

- Informazioni relative alla risposta che non possono essere inserite nella Status Line
- Esempi:

Location: http://www.w3.org/pub/
WWW/People.html

Server: CERN/3.0 libwww/2.17

Location: usato per ridirezionare il ricevente a una URI differente dalla Request-URI per completare la richiesta.

Server: informazioni sul software usato dal server per gestire la richiesta

status - code

3) Entity - Header

1xx: Informational - Request received, continuing process
2xx: Success - The action was successfully received, understood, and accepted
3xx: Redirection - Further action must be taken in order to complete the request
4xx: Client Error - The request contains bad syntax or cannot be fulfilled
5xx: Server Error - The server failed to fulfill an apparently valid request

"100" - Continue	"101" - Switching Protocols
"200" - OK	"201" - Created
"202" - Accepted	"203" - Non-Authoritative Information
"204" - No Content	"205" - Reset Content
"206" - Partial Content	
"300" - Multiple Choices	"301" - Moved Permanently
"302" - Moved Temporarily	"303" - See Other
"304" - Not Modified	"305" - Use Proxy
"400" - Bad Request	"401" - Unauthorized
"402" - Payment Required	"403" - Forbidden
"404" - Not Found	"405" - Method Not Allowed
"406" - Not Acceptable	"407" - Proxy Authentication Required
"408" - Request Time-out	"409" - Conflict
"410" - Gone	"411" - Length Required
"412" - Precondition Failed	"413" - Request Entity Too Large
"414" - Request-URI Too Large	"415" - Unsupported Media Type
"500" - Internal Server Error	"501" - Not Implemented
"502" - Bad Gateway	"503" - Service Unavailable
"504" - Gateway Time-out	"505" - HTTP Version not supported

Content-Base

URI assoluta da usare per risolvere le URL relative contenute nell'entity body

Content-Encoding

codifica dell'entity body (es: gzip)

Content-Language

lingua dell'entity body (es: en, it)

Content-Type

tipo dell'entity body (es: text/html)

Expires

(utile per caching) validità temporale del contenuto

Last-Modified

(utile per caching) data dell'ultima modifica sul server

Esempio richiesta / risposta

```

GET http://192.168.11.66/ HTTP/1.1
host: 192.168.11.66
Connection: close

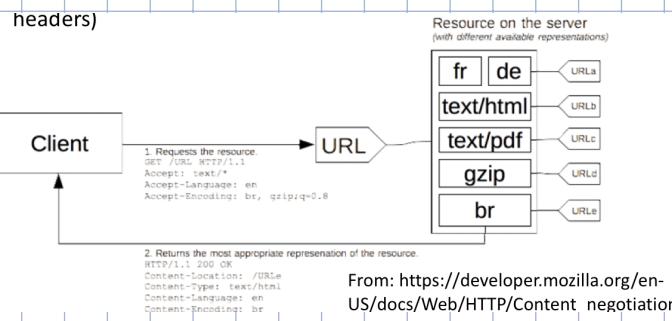
HTTP/1.1 200 OK
Date: Sun, 14 May 2000 23:49:39 GMT
Server: Apache/1.3.9 (Unix) (Red Hat/Linux)
Last-Modified: Tue, 21 Sep 1999 14:46:36 GMT
ETag: "f2fc-799-37e79a4c"
Accept-Ranges: bytes
Content-Length: 1945
Connection: close
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 3.2 Final//EN">
<HTML>
<HEAD> <TITLE>Test Page for Red Hat Linux's Apache Installation</TITLE> </HEAD>
<H1 ALIGN="CENTER">It Worked!</H1>
<P>
If you can see this, it means that the installation of the <A
HREF="http://www.apache.org/">Apache</A> software on this <a
href="http://www.redhat.com/">Red Hat Linux</a> system was successful. You
may now add content to this directory and replace this page.
</P>
</BODY>
</HTML>

```

Content Negotiation

Le risorse possono essere disponibili in più rappresentazioni quindi si usa il meccanismo di content negotiation per selezionare la rappresentazione più appropriata quando viene rivolta una richiesta.



WEB CACHING

Serve per soddisfare la richiesta di un client senza contattare il server.

Memorizzare copie temporanee di risorse WEB e servirle al client per ridurre l'uso di risorse e il tempo di risposta.

USER AGENT CACHE: Il browser mantiene una copia delle risorse visitate dall'utente.

Proxy Cache: Il proxy intercetta il traffico e mette in cache le risposte per poi fornire servizi da essa senza inviare le richieste al server.



Bisogna usare gli headers per controllare che la copia in cache non sia troppo vecchia.

- 1) 1° cerca fatto, va su internet e il proxy salva la risposta.
- 2) 2° cerca dopo non troppo tempo, la risposta viene servita dal proxy.

Cookie: Servizio per "numerare" i clienti e obbligarli a "farsi riconoscere" ogni volta presentando un "cookie".

Funzionamento:

- cliente C invia a server S normale richiesta HTTP
- server invia a cliente normale risposta HTTP + linea **Set-cookie: 1678453**
- cliente memorizza cookie in un file (associandolo a S) e lo aggiunge con una linea **cookie: 1678453** a tutte le sue successive richieste a quel sito
- server confronta cookie presentato con l'informazione che ha associato a quel cookie

Otobizzo:

- autenticazione
- ricordare profilo utente, scelte precedenti (cfr. "carte-soci")
- in generale creare sessioni sopra un protocollo stateless (es. shopping carts, Webmail)
- ... "non accettare dolci dagli sconosciuti" ... visitate cookiecentral.com

TELNET

Telnet è un protocollo di terminale remoto il cui scopo è quello di permettere l'uso interattivo di macchine remote.

Funzionamento

- TELNET permette ad un utente su una macchina di stabilire una connessione con un login server remoto
- In seguito, passa le battute dei tasti della macchina locale alla macchina remota (i comandi vengono eseguiti come se fossero stati battuti al terminale della macchina remota)
- L'output della macchina remota viene trasportato al terminale dell'utente

N.B. Nello sviluppo il suo funzionamento

erro:

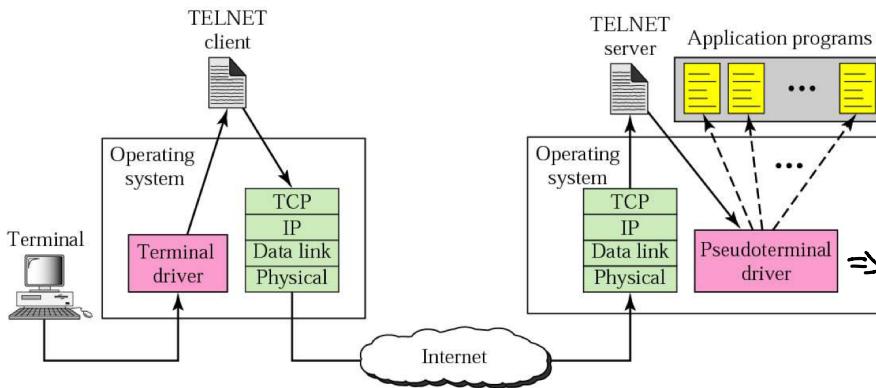
- Maschera non la rete che i S.O
- Utilizza interfaccia minima e veloce

È un protocollo client-server:

- client: 1) Stabilisce la connessione TCP
2) Prende i tasti in input, li manda al server, riceve l'output e lo visualizza su terminale client

- Server: 1) Accetta connessione TCP
2) Tramette i dati al S.O. locale del server.

la porta utilizzata è la 23.



Permette di trasferire caratteri a cui processo come se provengono dal terminale.

Per risolvere la differenza (sette codifiche di caratteri, lunghezza linea & lung. pagina ...) che ci può essere tra i 2 terminali: client e server, viene considerato un unico terminale virtuale con il quale conversano gli host (sia client che server).

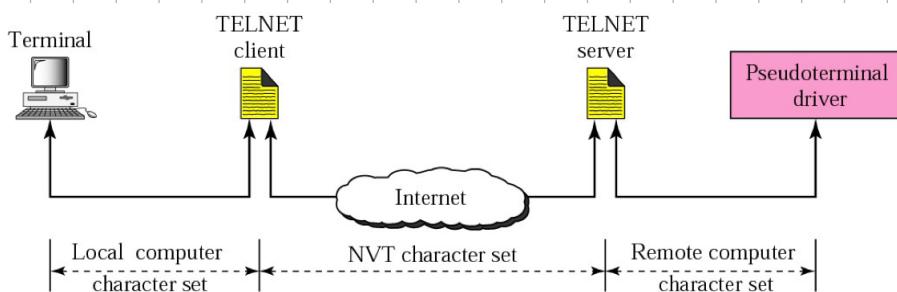
E' uno viene chiamato NVT (Network virtual terminal).

↳ RAPPRESENTAZIONE ASINTOTICA DI UN TERMINALE CANONICO

Il Network Virtual Terminal (NVT) definisce un set di caratteri e di comandi universali che permette di trasformare il set di caratteri in uso localmente in un set di caratteri universale

→ Client e server trasformano il set locale nel set globale

Telnet avviene su sugli host sia in esecuzione una NVT, quindi la connessione TCP è instaurata tra questi 2 NVT.



Funzionamento:

I terminali NVT si scambiano dati in formato 7-bit US-ASCII. Ogni carattere è inviato come un ottetto con il primo bit settato a zero.

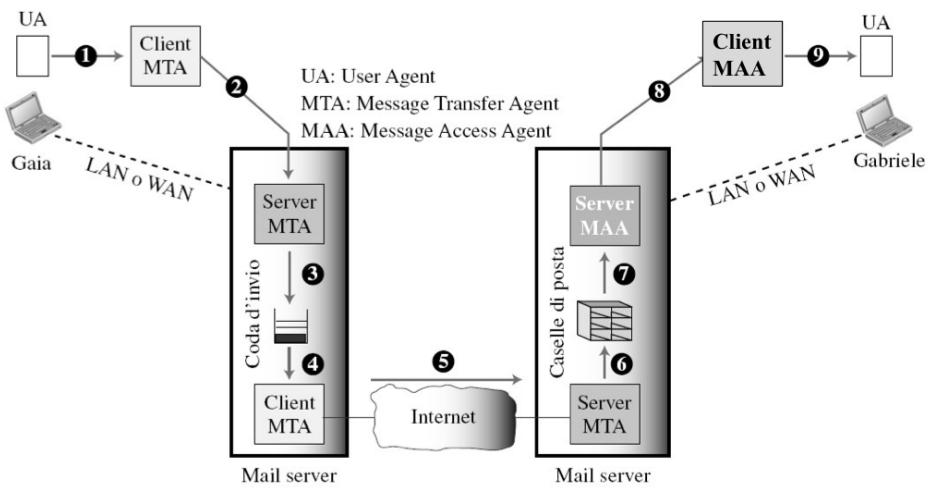
I byte con il bit più significativo a 1 usati per le sequenze di comandi

I comandi (es. end-of-line trasmesso come la sequenza CR-LF linefeed) iniziano con un ottetto speciale (Interpret as Command - IAC) di 1

-> Inband signalling (comandi e dati sulla stessa connessione)

- I messaggi di controllo iniziali sono usati per scambiare informazioni sulle caratteristiche degli host (Telnet option negotiation).
- Il client invia un login identifier e una password.
- Il client interagisce con la macchina remota
- Telnet non è sicuro -> SSH Secure Shell (SSH) Connection Protocol ¹¹

Posta Elettronica



Mail Server

Utilizzano una tecnica chiamata spooling:

- L'utente invia un messaggio, il sistema ne pone una copia in memoria (*spool* – o area di accodamento della posta), insieme con id mittente, id destinatario, id macchina di destinazione e tempo di deposito.
- Il sistema avvia il trasferimento alla macchina remota. Il sistema (client) stabilisce una connessione TCP con la macchina destinazione
- Se la connessione viene aperta, inizia il trasferimento del messaggio.
- Se il trasferimento va a buon fine il client cancella la copia locale della mail
- Se il trasferimento non va a buon fine, il processo di trasferimento scandisce periodicamente l'area di spooling, e tenta il trasferimento dei messaggi non consegnati. Oltre un certo intervallo di tempo (definito dall'amministratore del server) se il messaggio non è stato consegnato, viene inviata una notifica all'utente mittente.

Indirizzi email:

local-part: Specifica la cassetta di posta nel mail server.

local-part @ domain-name

Domain - Part : Specifica un mail server

Componenti Principali:

1) User - Agent : Composizione, editing, lettura messaggi
(Outlook, Gmail...)

2) Mail - Server:

- i messaggi in entrata ed uscita vengono archiviati sul server
- le mailbox (CASSETTE di POSTA) contengono messaggi in ingresso (che devono ancora essere letti) diretti all'utente
- una coda di messaggi in uscita (che devono essere inviati)

3) Protocollo SMTP: Dialogo tra mail - server

"client": mail server che invia

"server": mail server che riceve

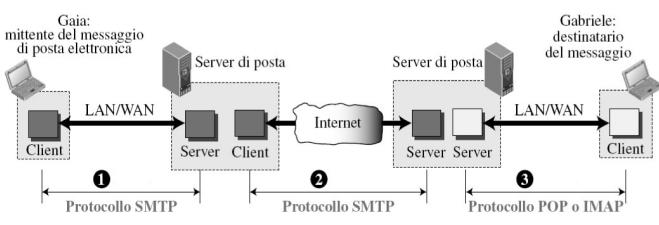
SMTP

È un protocollo il cui obiettivo è il trasferimento efficiente ed efficiente di mail.

Era e' indipendente dal riserbo di trasmissione usato e richiede solo il trasferimento di stream di byte ordinato e affidabile

Quando un client SMTP vuole trasferire un messaggio, stabilisce un canale di trasmissione bidirezionale con un server SMTP. La responsabilità di un client è di trasferire la mail a un server SMTP, o comunicare un eventuale insuccesso (scambio formale di responsabilità).

Un client SMTP determina l'indirizzo di un host appropriato che ospita un server SMTP risolvendo il nome della destinazione in un un mail server destinazione (risoluzione di nome in indirizzo IP attraverso il DNS).



19

- Potibili
- ⇒
- connessione con mailserver del mittente (server inesistente o irraggiungibile)
 - connessione con mailserver destinatario (server inesistente o irraggiungibile)
 - inserimento in mailbox destinatario (user unknown, mailbox full)
... ma in tutti questi i casi mittente riceve una notifica!

Destinatario può non ricevere (senza che mittente sia avvisato) solo se qualcuno (intruso, filtro antispam) rimuove messaggio

Era utilizza il protocollo TCP con porta 25.

Si suddivide in tre fasi nelle quali il client manda comandi in testo ASCII e la risposta è un codice di stato + una descrizione (Facoltativa).

1) HANDSHAKING

- Il client stabilisce la connessione e attende che il server invii: 220 READY FOR MAIL
- Il client risponde con il comando HELO
- Il server risponde identificandosi
- A questo punto il client può trasmettere messaggi

```
S: 220 Beta.GOV Simple Mail Transfer Service ready  
C: HELO Alfa.EDU  
S: 250 Beta.GOV
```

Comandi possibili (Alcuni):

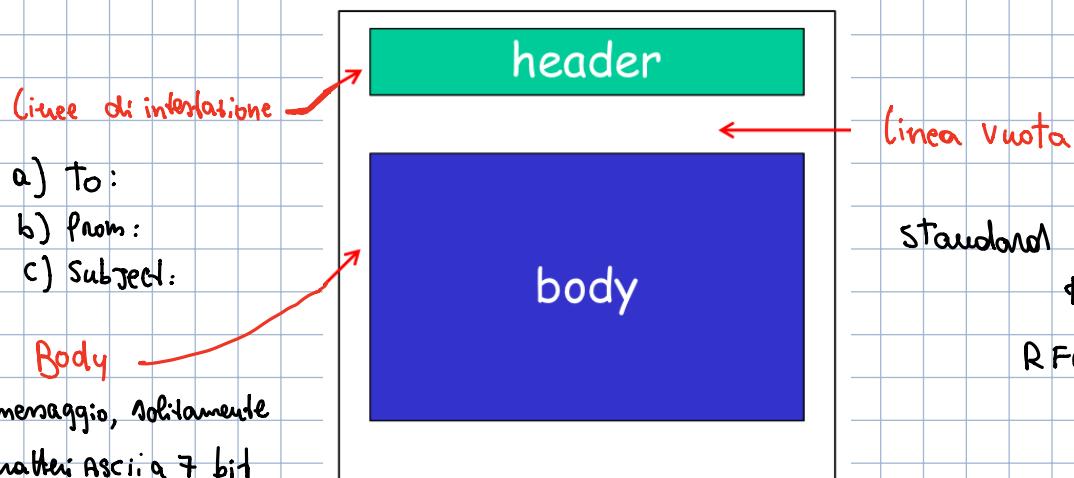
Alcuni comandi:

- HELO <client identifier>
- MAIL FROM:<reverse-path> [SP <mail-parameters>]
<CRLF>
- RCPT TO:<forward-path> [SP <rcpt-parameters>] <CRLF>
- DATA
- QUIT

N.B. <CRLF>.<CRLF> per determinare la fine di un messaggio

2) Trasferimento messaggio

Formato:

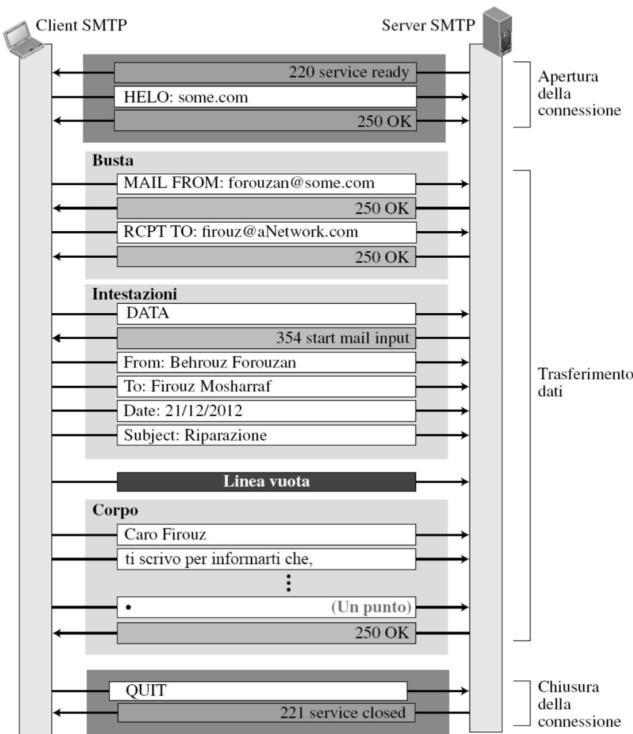


Il messaggio, solitamente
in caratteri ASCII a 7 bit

standard per il formato di
testo:
RFC 2822

3) Termination

Esempio di interazione:



Il formato descritto in precedenza (STD. RFC 2822) non permette di inviare file multimediali.

Nasce il formato MIME (RFC 2045), cioè uno standard che estende il formato per le email.

- per supportare:
- * testo in set di caratteri diversi da US-ASCII;
 - * allegati in format non testuale;
 - * corpo del messaggio con più parti (multi-part)
 - * header in set di caratteri non-ASCII

Quanto è possibile perché viene aggiunta una struttura al message body e vengono definite le regole di codifica per il trasferimento di testo non-ASCII.

Per dichiarare il tipo di contenuti MIME vengono aggiunte linee di intestazione



Formato:

Versione MIME
metodo usato per codifica dati
Dati multimediali tipo, sottotipo, dichiarazione parametri
Dati codificati

```

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

```

- Client e server SMTP si aspettano messaggi ASCII (caratteri ASCII che usano 7 degli 8 bit di un byte). I dati binari usano invece tutti gli 8 bit di un byte (es. immagini, eseguibili, set estesi di caratteri).
- MIME fornisce cinque schemi di transfer encoding, tra cui:
 - ASCII encoding of binary data: base 64 encoding
 - Gruppi di 24 bit sono divisi in 4 unità da 6 bit e ciascuna unità viene inviata come un carattere ASCII
 - Quoted-printable encoding: per messaggi con pochi caratteri non-ASCII, più efficiente

Tipi di MIME:

Content-Type: type/subtype; parameters

Specificano la natura dei dati nel corpo di un'entità MIME

Text

esempi di subtypes: plain, html

Image

esempi di subtypes: jpeg, gif

Audio

esempi di subtypes: basic (8-bit mu-law encoded), 32kadpcm (32 kbps coding)

Video

esempi di subtypes: mpeg,

Application

altri dati che devono essere processati da un'applicazione prima di essere visualizzabili

esempi di subtypes: msword, octet-stream (dati arbitrari binari)

Tipo "Multipart"

```

From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=98766789

```

--98766789

Content-Transfer-Encoding: quoted-printable
Content-Type: text/plain

Dear Bob,

Please find a picture of a crepe.

--98766789

Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data

.....base64 encoded data

--98766789--

Protocolli di accesso alla posta:

1) POP3: lo user agent apre una connessione TCP (Porta 110) verso il server di posta e poi si negoziano 3 fasi:

1) Autorizzazione:

comandi del client:

- user: specifica la username
- pass: specifica la password

il server risponde

- OK
- ERR

comandi del client:

- list: visualizza la lista dei messaggi
- retr: preleva il messaggio per numero
- dele: elimina il messaggio dal server
- quit: chiude la sessione

2) Transazione:

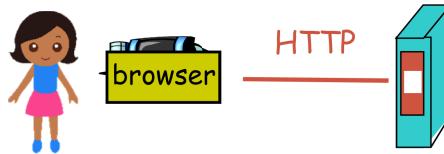
Dopo quit il server cancella i messaggi marcati per la rimozione

3) Aggiornamento:

2) IMAP :

- più feature (e più complesso) di POP3 (che non mantiene informazioni di stato tra sessioni – solo cancellazioni permesse)
- manipolazione di messaggi memorizzati sul server (p.e. folder)
- comandi per estrarre solo alcuni componenti dei messaggi (p.e. solo intestazioni se si sta usando una connessione lenta)

3) HTTP : Hotmail , Yahoo , Gmail ecc...



DNS (Domain Name System)

L'obiettivo del DNS è di associare ad ogni nome di dominio un indirizzo IP

Il DNS è un protocollo dello strato applicativo il quale non comunica con gli utenti

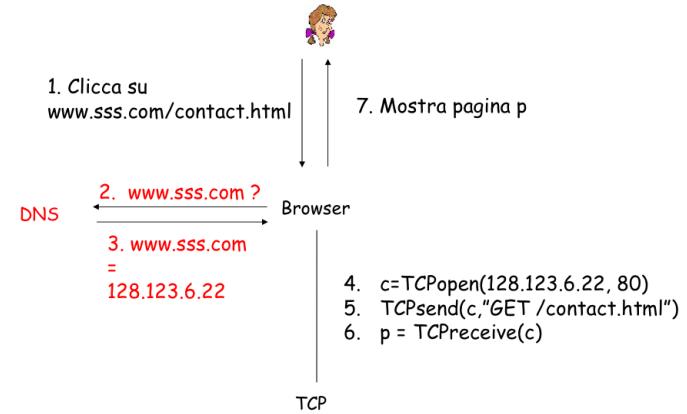
- e :
 - 1) Gira sui sistemi terminali, adotta il paradigma client-server
 - 2) Si affida al notevolmente protocollo di trasporto punto-punto per trasferire i messaggi tra host.

Il DNS è un meccanismo che deve:

- specificare la sintassi dei nomi e le regole per gestirli
- consentire la conversione da nomi a indirizzi e viceversa

Esso è costituito essenzialmente da:

1. uno **schema di assegnazione dei nomi** gerarchico e basato su domini
2. un **database distribuito** contenente i nomi e le corrispondenze con gli indirizzi IP implementato con una gerarchia di name server
3. un **protocollo** per la distribuzione delle informazioni sui nomi tra name server
 - host, router, name server comunicano per **risolvere** nomi (traduzione nome/indirizzo)
 - utilizzando **UDP** (porta 53) [oppure **TCP**]



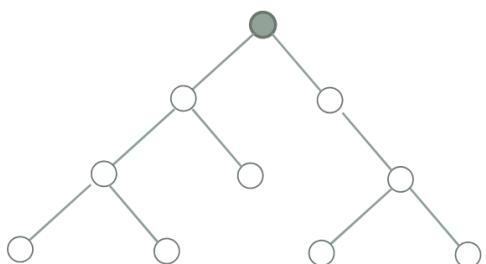
I servizi DNS devono offrire :

- 1) Risoluzione di hostnames in indirizzi IP
- 2) Host Aliasing : Un host può avere più nomi (nome canonico + Sintomi o alias)
- 3) Mail Server Aliasing : Sintomi per mail server
- 4) Gestire la distribuzione di carico :
 - Distribuzione carico tra server replicati
 - Ad un hostname canonico corrispondono più indirizzi IP
 - Il DNS restituisce la lista di IP variandone l'ordinamento a ogni risposta

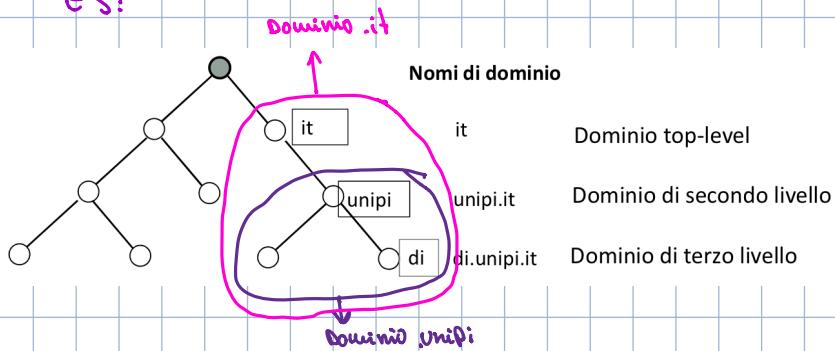
Struttura dei Nomi

La struttura è gerarchica:

- Struttura ad albero
- Oggi nodo è individuato da una etichetta di max 63 caratteri
- Alla radice l'etichetta è vuota



E.S.:



DOMINIO: sottoalbero nello spazio dei nomi di dominio che viene identificato dal nome di dominio del nodo in cima al sottoalbero.
Un dominio può essere suddiviso in ulteriori domini, detti sottodomini

Domini

Internet è divisa in diverse centinaia di domini il quale è partitionato in sottodomini ecc...

Nomi di dominio. Un internet: nomi gerarchici delle macchine sono assegnati in base alle

strutture delle organizzazioni che offengono l'autorità per posizioni dello spazio dei

nomi. La struttura gerarchica permette autonomia nella scelta dei nomi all'interno

di un dominio. (Posso scegliere il sottodominio che voglio, nel mio dominio)

Esempi di top-level domain

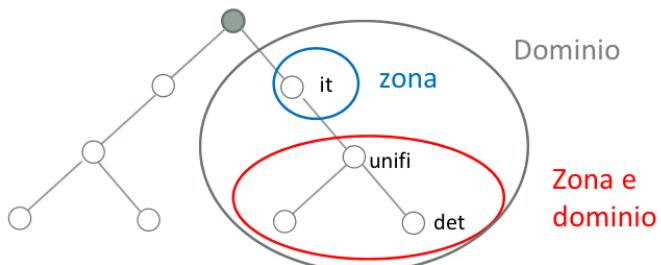
com	Organizzazioni commerciali
edu	Istituti di istruzione (università, scuole)
mil	Gruppi militari
gov	Istituzioni governative (USA)
net	Principali centri di supporto alla rete
org	Organizzazioni diverse dalle precedenti
Codice geogr. (ir, uk, us, fr, etc.)	Schema geografico per nazioni

Name Server: Programma che gestisce la conversione da nome di dominio a indirizzo IP

Le info sui domini sono ripartite su più name server.

Zona: Regione di cui è responsabile un name server.

Un name server immagazina le info relative alla propria zona inclusi i riferimenti ai server dei domini di livello inferiore.



Gerarchia dei server:

- 1) **Server Radice:** Responsabile dei record della zona radice. Restituisce info su nome (primo nodo gerarchia) server di TLD
- 2) **Server top-level-Domain:** Mantiene le info dei nomi di domio che appartengono a un certo TLD.

Restituisce le info sui name server di competenza dei sottodomini

3) **Server di competenza:**

- Autorità per una certa zona
- memorizza nome e indirizzo IP di un insieme di host
- può effettuare traduzioni nome/indirizzo per quegli host
- Per una certa zona ci possono essere server di competenza primari e secondari
 - Server primari mantengono il file di zona
 - Server secondari ricevono il file di zona e offrono il servizio di risoluzione

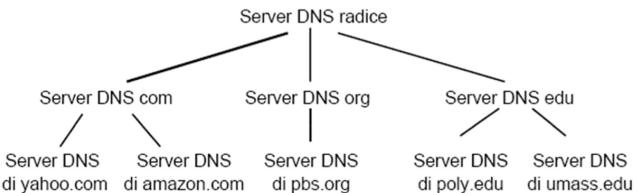
4) **Local name servers:**

- Non appartiene strettamente alla gerarchia dei server
- ogni ISP (università, società, ISP) ha il suo (*default*) name server locale
- Le query DNS vengono prima rivolte al name server locale
 - Il server DNS locale opera da proxy e inoltra la query in una gerarchia di server DNS

Quando un prog. deve trasformare un nome in indirizzo IP chiama un prog. detto RESOLVER, passando il nome come parametro di ingresso.

Il RESOLVER interroga il local name server, il quale cerca il nome nelle sue tabelle e restituisce l'indirizzo al resolver oppure invia la query alla gerarchia DNS.

Risoluzione dei nomi



Il client vuole l'IP di www.amazon.com; 1^a approssimazione:

- Il client interroga il server radice per trovare il server DNS com
- Il client interroga il server DNS com per ottenere il server DNS amazon.com
- Il client interroga il server DNS amazon.com per ottenere l'indirizzo IP di www.amazon.com

Esempi

Esempio

- host www.tintin.fr cerca l'indirizzo IP di www.topolino.it
1. Contatta il suo DNS locale dns.tintin.fr
 2. dns.tintin.fr contatta il root name server, se è necessario
 3. root name server contatta l'autoritative name server, dns.topolino.it, se è necessario

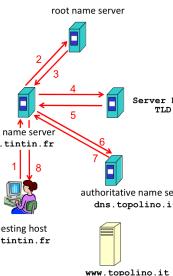
Query RICORSIVA: Il local name server ha richiesto una conversione completa

Venne aggiunto perciò perché
dai suoi name server

Esempio

- Query ITERATIVA:
 - Le risposte sono restituite direttamente al client
 - Viene restituito il riferimento al server da contattare successivamente

local name
server
gestisce
la query



CACHING ed AGGIORNAMENTO RECORDS

Una volta che un Name Server ha acquisito una associazione, la mette in cache i quali vengono cancellati dopo un certo tempo.

RECORD DNS

4 record nel database sono chiamati Resource Records ed hanno il seguente formato:

RR format: (name, value, type, ttl)

4 significati di name e value dipendono da type:

Type=A

- Name: hostname
- Value: indirizzo IP

Type=CNAME

- Name: hostname (sinonimo)
 - Value: nome canonico dell'host
- ↑ nome
che funziona
de
sinonimo
- ↑ nome
principale
dell'host

Type=NS

- Name: nome di dominio (e.g. unipi.it)
- Value: hostname dell'autoritative name server per quel dominio ↗: dns.unipi.it

Type=MX

- Name: nome di dominio
- Value: nome canonico del server di posta associato a name

ci sono altri type...

TTL: Quando la risorsa dovrà essere rimossa dalla cache

Esempio:

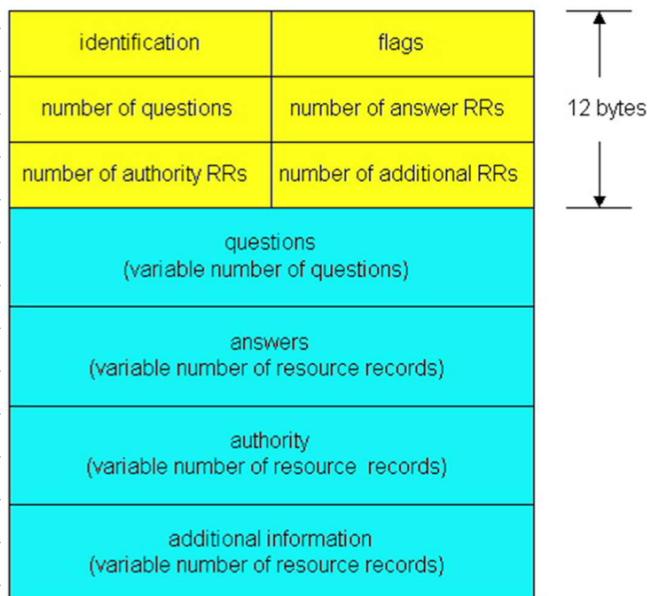
- (HostName, indirizzoIPdiHostName, A, ...)
(www.cnn.com, 157.166.224.25, A, ...)
 - (HostName, indirizzoIPdiHostName, A, ...)
(www.cnn.com, 157.166.224.26, A, ...)
 - (Dominio, NomeDiAuthoritativeServerPerDominio, NS, ...)
(cli.di.unipi.it, nameserver.cli.di.unipi.it*, NS, ...)
- [* il messaggio di risposta potrà contenere anche (nameserver.cli.di.unipi.it, 131.114.120.2, A, ...)]
- (Alias, HostNameMailServerConTaleAlias, MX, ...)
(cli.di.unipi.it, mailserver.cli.di.unipi.it*, MX, ...)
- [* il messaggio di risposta potrà contenere anche (mailserver.cli.di.unipi.it, 131.114.11.39, A, ...)]

Protocollo DNS

Il protocollo DNS utilizza UDP sulla porta 53. Permette anche l'uso TCP.

I messaggi di query e reply hanno entrambi lo stesso formato di messaggio.

Formato:



Identification: 16 bit di identificazione per la query, la risposta alla query usa lo stesso ID

FLAGS:
Query = 0
Reply = 1

- Domande: campi per il nome richiesto e il tipo di query
- Risposte: RR nella risposta alla domanda
- Competenza: record relative ai server di competenza
- Informazioni aggiuntive

} Possono essere 0 o più

Esempio:

dns.poly.edu -----> TLD server
Question: QName=gaia.cs.umass.edu QType=NS

dns.poly.edu <---- TLD server
Answer: (umass.edu. dns.umass.edu, NS) -----
Additional: (dns.umass.edu, 128.115.40.41, A)

OSS:

Ogni organizzazione dotata di host Internet pubblicamente accessibili (es server web e server di posta) deve fornire i record DNS di pubblico dominio che mappano i nomi di tali host in indirizzi IP (server mantenuti dall'organizzazione o ISP).

Tali record possono essere mantenuti in un proprio DNS o tramite un gestore di servizi

livello di TRASPORTO

Il livello di trasporto è il 4° del modello TCP/IP

I suoi obiettivi sono:

- 1) Realizzare una comunicazione logica fra processi residenti in host system diversi.
- 2) Offrire servizi al livel applicazione → I processi a livel applicativo si comportano come se gli host fossero direttamente collegati.
- 3) Utilizzazione i servizi del livel di rete

Oltre il seguente servizio:

- 1) Fornire la comunicazione logica fra processi applicativi di host differenti
- 2) Multiplexing/demultiplexing
- 3) Controllo degli errori

Tipi di servizi offerti:

(processo di applicativo)

- 1) Servizio privo di connessione: Il mittente consegna i messaggi al livel di trasporto uno per uno
I segmenti possono non essere consegnati o non arrivare in ordine
Il livel di trasporto è considerato come entità singola senza relazioni fra essi!
- 2) Servizio orientato alla connessione: Client e server stabiliscono una connessione logica
Connessione fra processo e livel di trasporto!

Protocolli:

- 1) UDP:
 - Senza connessione
 - Non affidabile, consegne senza ordine
 - Estensione "senza fronzoli" del servizio di consegna "host to host" di IP

- 2) TCP:
 - Gestione della connessione
 - Consegna affidabile (priva di errori, completezza e ordine)
 - Flow Control
 - Controllo di congestione

SERVIZI MULTIPLEXING e DEMULTIPLEXING (Si basano sui socket address dei processi)

- 1) MULTIPLEXING: Accorpamento dei flussi dati dai processi verso la rete con un prefisso (liv app.) (liv rete) (Header)
- 2) DEMULTIPLEXING: Smista i pacchetti che arrivano dal livel di rete ai vari processi.

Demultiplexing

nell'host ricevente:

consegnare i segmenti ricevuti alla socket appropriata

Multiplexing

nell'host mittente:

raccogliere i dati da varie socket, incapsularli con l'intestazione (utilizzati poi per il demultiplexing)

Ogni comunicazione di trasporto (TCP o UDP) è identificata in maniera

univoca grazie alle coppie numero IP / porta degli host. (Indirizzo socket)

- La "porta" è un numero (unsigned int – 16 bit [0-65535]) che viene assegnato a un processo, o più precisamente a un punto di demultiplexing dei protocolli TCP o UDP.
- L'indirizzo IP è un indirizzo di 32 bit presente nello stack TCP/IP

le porte si suddividono in:

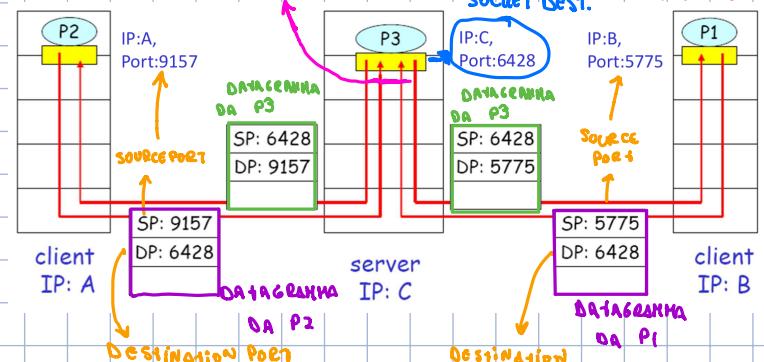
- 1) System ports (well known ports): range 0-1023, assegnate da IANA, identificate ai router servizi
- 2) User ports (registered ports): range 1024-49151, assegnate da IANA, richieste da aziende che producono applicazioni
- 3) Dynamic ports (private ports): range 49152-65535, non assegnate da IANA

Le porte vengono assegnate dinamicamente ai processi dal S.O

DEMULPLEXING SENZA CONNESSIONE - UDP

- es. creazione automatica di socket DatagramSocket
`mySocket = new DatagramSocket()`
- Lo strato di trasporto dell'host ricevente consegna il segmento UDP alla socket identificata da IP e porta destinazione
- I datagrammi con IP e/o porta mittente differenti ma stessi IP e porta destinatari vengono consegnati alla stessa socket
- Socket UDP identificata dalla coppia (IP, porta)

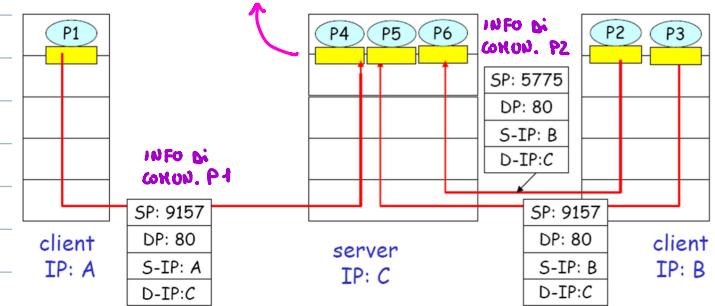
TUTTI i hex che arrivano vengono inviati alla stessa socket e uguali vengono inviati dalla stessa socket



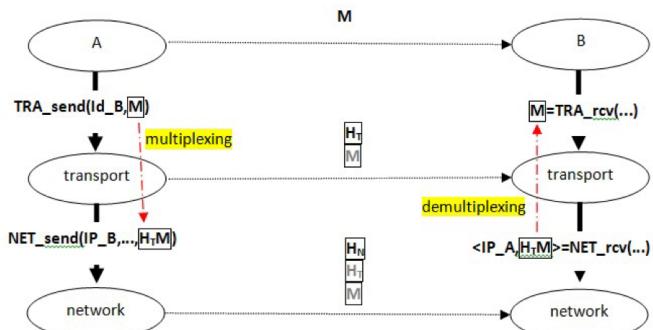
DEMULPLEXING ORIENTATO ALLA CONNESSIONE - TCP

- La socket TCP è identificata da 4 parametri:
 - Indirizzo IP di origine
 - Numero di porta di origine
 - Indirizzo IP di destinazione
 - Numero di porta di destinazione
- L'host ricevente usa i 4 parametri per inviare il segmento alla socket appropriata
- Un host server può supportare più socket contemporanee
- Es. server Web: socket differenti per ogni client

1 socket x ogni comunicazione instaurata



Interfaccia tra i livelli dello stack



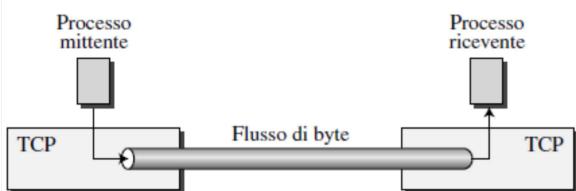
INFO DI COLLEG. P3
 SP = source port
 DP = destination port
 S-IP = Source IP
 D-IP = destination IP

TCP

PROPRIETÀ DEL SERVIZIO TCP:

1) ORIENTAMENTO ALLO STREAM:

- Lunghezza di byte indefinita a priori. Il servizio di consegna delle stream nella macchina di destinazione passa esattamente la medesima sequenza di ottetti che il trasmettitore ha passato al servizio di consegna nella macchina origine **O RIGINA I SEGNALI**
- TCP vede i dati come un flusso di byte ordinati, ma non strutturati **NON MANIACHE NÉ SISTEMATI**



2) ORIENTATO ALLA CONNESSIONE

- I processi effettuano un handshake prima dello scambio dei dati. Invio di informazioni preliminari per preparare lo scambio dei dati
- **Orientato** perché lo stato della connessione risiede sui punti terminali, non sugli elementi intermedi della rete (ad es. router).
- La connessione è vista dagli applicativi (USERS) come un circuito fisico dedicato, quindi il TCP è capace di fornire servizi del tipo CONNECTION ORIENTED mentre il protocollo IP su cui appoggia, è in grado di fornire servizi CONNECTION LESS.

3) COMUNICAZIONE FULL-DUPLEX

- il flusso dati tra due host può avvenire contemporaneamente nelle due direzioni. Le due direzioni sono slegate
- connessione punto-punto

5) MULTIPLEXING

- consente di assegnare una data **connessione** ad un particolare processo (permette una comunicazione da processo a processo)

4) FUNZIONI BASE PER IL TRASM. DI DATI

- capacità di trasferire un flusso continuo di byte
- trasferimento bidirezionale (full duplex)

6) CONTROLLO DELLA CONNESSIONE

- meccanismi di inizio e fine trasmissione (controllo di sessione)

7) TRASFERIMENTO DATI ORDINATO E AFFIDABILE

- si intende la capacità di correggere tutti i tipi di errore, quali:
 - dati corrotti
 - segmenti persi
 - segmenti duplicati
 - segmenti fuori sequenza

8) CONTROLLO DI FWSNO

- evitare di spedire più dati di quanti il ricevitore sia in grado di trattare

9) CONTROLLO DI CONGESTIONE

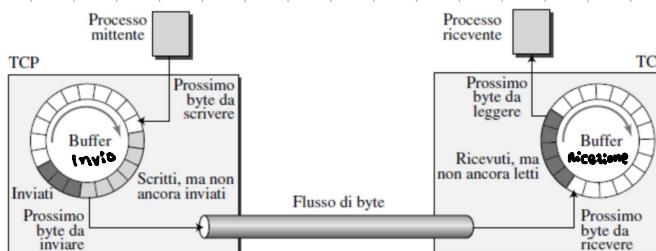
- ha lo scopo di recuperare situazioni di sovraccarico nella rete

10) TRASFERIMENTO BUFFERIZZATO

- Il software del protocollo TCP è libero di suddividere il flusso di byte in **segmenti in modo indipendente dal programma applicativo** che li ha generati. Per fare questo è necessario **disporre di un BUFFER** dove immagazzinare la sequenza di byte. Appena i dati sono sufficienti per riempire un segmento ragionevolmente grande, questo viene trasmesso attraverso la rete.
- La bufferizzazione consente una riduzione del traffico sulla rete "ottimizzando" in qualche modo il numero di segmenti da trasmettere

9 dati sono inseriti dal processo applicativo nel buffer.
 Poi prelevati dal buffer dal livello di trasporto.
 Vengono inviati quando sono stati impackettati alla rete.

TRASF. BUFFERIZZATO



I processi a livello applicativo scrivono e leggono byte nel/dal buffer. Questo può avvenire a velocità diverse
oss: Si noti che dest. hanno buffer invio e ricezione!

Numeri di sequenza e di riscontro

TCP numero i byte anziché i segmenti

Numeri di sequenza associati a un segmento:

Numeri del primo byte del segmento calcolato sul numero di byte totale da inviare su quella connessione!

Numeri di riscontro:

- 1 + numero ultimo byte correttamente ricevuto $\Rightarrow N.S$ ultimo byte segmento + 1
- Riscontri interpretati come "cumulativi"
- "ACK=y" significa "aspetto il byte y (ho ricevuto tutti i byte fino a y-1)"

24

numero (nel flusso) del primo byte (di dati) del segmento



- In genere si parte da un initial sequence number generato in modo casuale (e quindi $\neq 0$)

SEGMENTI TCP



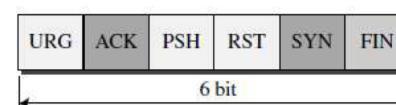
Numeri di seq., Numero di riscontro, Finestra

permettono il flow control, meccanismo di ritrasmissione ed il riordino dei pacchetti

Ricevuti.

→ **Bit codice:** sono 6 flag e servono per (da sinistra a destra):

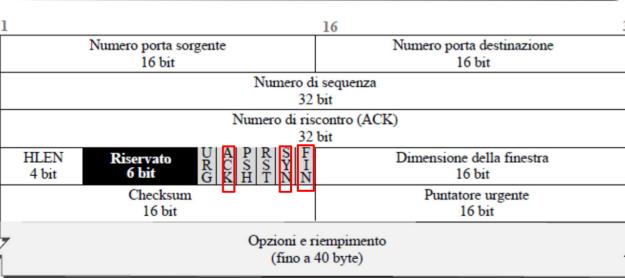
- URG: Il campo Puntatore Urgente contiene dati significativi da trasferire in via prioritaria
- ACK: Il campo Numero di Riscontro contiene dati significativi
- PSH: Funzione Push (trasferimento immediato dei dati in un segmento dal trasporto al livello applicativo)
- RST: Reset della connessione = Chiusura connessione brusca RST=1
- SYN: Sincronizza il Numero di Sequenza = Usato per l'HANDSHAKE
- FIN: Non ci sono altri dati dal mittente – chiusura della connessione



E' inoltre presente un campo URGENT che permette la trasmissione di dati "fuori banda", ovvero a priorità maggiore degli altri (la loro gestione però è affidata all'applicazione)

→ **Puntatore Urgente (16 bits):** questo campo è un offset positivo a partire dal Numero di Sequenza del segmento corrente. E' interpretato solo se il bit URG è uguale ad 1. Punta al primo byte di dati non urgenti a partire dal Numero di Sequenza, e consente di far passare i dati urgenti in testa alla coda di ricezione. Nel segmento contenente dati urgenti deve essere presente almeno un byte di dati.

- Ad esempio se un segmento contiene 400 byte di dati urgenti e 200 byte di dati non urgenti, URG vale 400

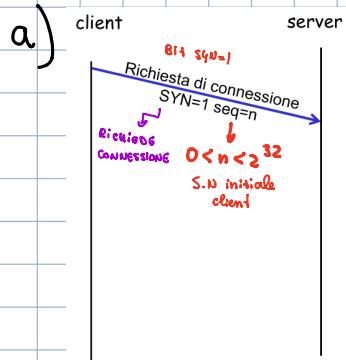


- Porta (16bit): numeri di porta della comunicazione.
- Numero di sequenza (32bit): è il numero di sequenza nello stream del primo byte di dati di questo segmento. Se il flag SYN è settato il numero di sequenza è ISN (initial sequence number) e il primo byte di dati è ISN+1.
- Numero di riscontro (32bit): se il bit ACK è settato, questo campo contiene il valore del prossimo numero di sequenza che il mittente del segmento si aspetta di ricevere dall'altro host. Una volta che la connessione è stabilita è sempre inviato.
- Hlen (4bit): lunghezza dell'header TCP espresso in parole da 4 byte (la lunghezza dell'header può variare tra 20 e 60 byte)
- Finestra di ricezione (16bit): indica il numero di byte di dati a partire da quello indicato nel campo Numero di Riscontro che il mittente di questo segmento è in grado di accettare. Serve per il controllo di flusso *yndica spazio che ha dopo nel buffer*
- Checksum (16bit): checksum dell'intero pacchetto (dati, header TCP + parte dell'header IP) per rilevare errori (se i bit del segmento sono stati alterati). Si calcola come per UDP (per TCP è obbligatorio). *Se vengono ricevuti errori, scatta pacchetto*
- Opzioni (facoltativo, lunghezza variabile, max 40 byte): negoziazione di vari parametri: ad es, dimensione massima segmento (MSS), selective acknowledgement supportato e blocchi di dati riscontrati selettivamente. Le opzioni sono sempre multipli di 8 bit e il loro valore è incluso nel checksum.

26

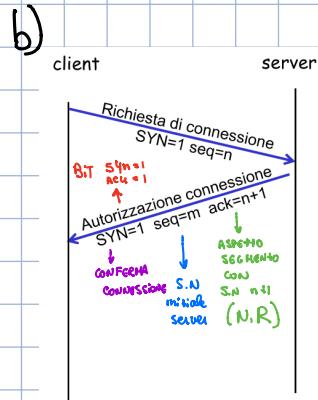
Gestione della connessione:

1) HANDSHAKE : IMPORTANTE: DURANTE HANDSHAKE NON POSSONO ESSERE TRANSF. DATI



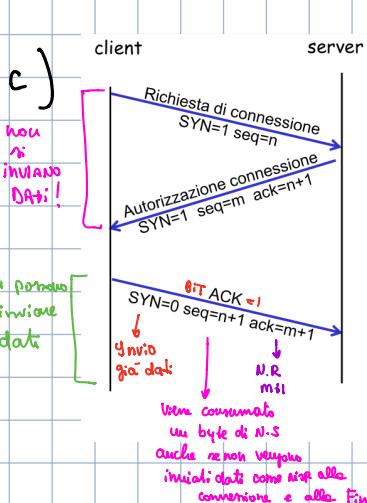
Il client invia una richiesta di connessione a un server TCP
è attivo il bit SYN, il segmento non contiene dati
si trasmette anche un numero di sequenza iniziale casuale (ISN)

Ad esempio:
Flag SYN=1 client_isn = 41



Il server estrae il segmento, alloca buffer e le variabili TCP per la connessione
invia in risposta un segmento di connessione garantita al client (chiamato SYNACK)

- è attivo SYN, il numero di sequenza è il valore iniziale (es. server_isn = 78)
- è attivo ACK, il server aspetta client_isn+1 (es. 42)
- Esempio SYN=1, ACK = client_isn +1, proprio numero di sequenza iniziale server_isn. Segmento SYNACK



- il client alloca buffer e variabili di connessione
manda un riscontro positivo del messaggio del server.
- SYN è inattivo. Questo segmento può già trasportare dati
 - il prossimo dato sarà client_isn+1 (42) ed il client attende server_isn+1 (79)
 - SYN=0, riscontro server_isn+1.
 - Inizia lo scambio dati, SYN=0

OSS:

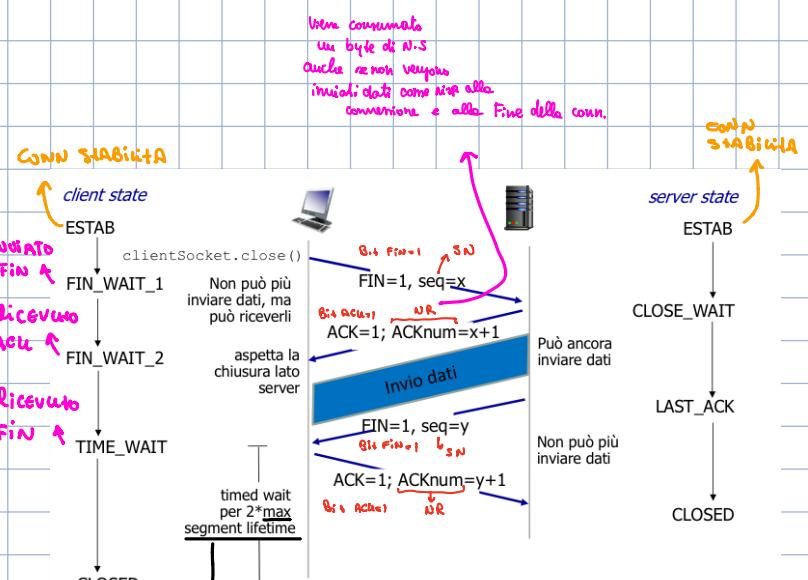
- I primi segmenti non hanno carico utile.
- All'arrivo del primo segmento il server inizializza due buffer (memorie di scambio) e le variabili, necessari per il controllo del flusso e della congestione.
- All'arrivo del riscontro del primo segmento il client alloca due buffer e le variabili, necessari per il controllo del flusso e della congestione.
- Alla ricezione del terzo segmento la connessione è instaurata.
- Non c'è più diff. dopo l'HANDSHAKING tra client e server!

E' importante la terza via per essere sicuri che sia il client che server siano pronti per lo scambio!

2) TRASFERIMENTO DATI:

3) CHIUSURA CONNESSIONE

- Client e server chiudono ciascuno il loro lato della connessione
 - Invio di segmento TCP con bit FIN = 1
- Ciascuno risponde all'ACK ricevuto con un ACK
 - Quando viene ricevuto un FIN, l'ACK può essere combinato con il proprio FIN
- È possibile anche lo scambio simultaneo di FIN



HSL è 2 min. secondo RFC, ma può variare.

STATO TIME_WAIT

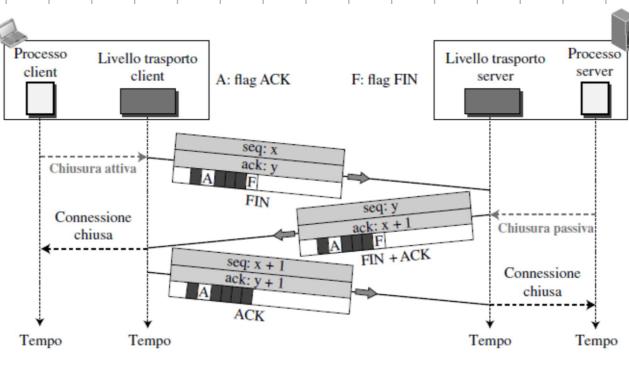
TIME_WAIT è lo stato finale in cui il capo di una connessione che esegue la chiusura attiva resta prima di passare alla chiusura definitiva della connessione

- due volte la MSL (Maximum Segment Lifetime). **2 * MSL**
- La MSL è la stima del massimo periodo di tempo che un pacchetto IP può vivere sulla rete; questo tempo è limitato perché ogni pacchetto IP può essere ritrasmesso dai router un numero massimo di volte (detto hop limit).
- Ogni implementazione del TCP sceglie un valore per la MSL (RFC 793 2 minuti, Linux 30 o 60 secondi).

Lo stato TIME_WAIT viene utilizzato dal protocollo per due motivi principali:

- implementare in maniera affidabile la terminazione della connessione in entrambe le direzioni.
 - Se l'ultimo ACK della sequenza viene perso, chi esegue la chiusura passiva manderà un ulteriore FIN, chi esegue la chiusura attiva deve mantenere lo stato della connessione per poter reinviare l'ACK
- consentire l'eliminazione dei segmenti duplicati in rete.

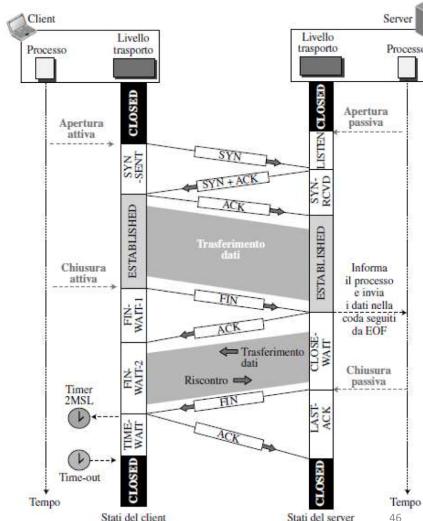
2° tipo di chiusura (three-way Handshake)



- Un FIN che non trasporta dati consuma comunque un numero di sequenza
- Idem per FIN+ACK

APERTURA e CHIUSURA CONNESSIONE (riassunto)

Apertura e chiusura connessione



Stati TCP

- LISTEN - represents waiting for a connection request from any remote TCP and port.
- SYN-SENT - represents waiting for a matching connection request after having sent a connection request.
- SYN-RECEIVED - represents waiting for a confirming connection request acknowledgment after having both received and sent a connection request.
- ESTABLISHED - represents an open connection, data received can be delivered to the user. The normal state for the data transfer phase of the connection.
- FIN-WAIT-1 - represents waiting for a connection termination request from the remote TCP, or an acknowledgment of the connection termination request previously sent.
- FIN-WAIT-2 - represents waiting for a connection termination request from the remote TCP.
- CLOSE-WAIT - represents waiting for a connection termination request from the local user.
- CLOSING - represents waiting for a connection termination request acknowledgment from the remote TCP.
- LAST-ACK - represents waiting for an acknowledgment of the connection termination request previously sent to the remote TCP (which includes an acknowledgment of its connection termination request).
- TIME-WAIT - represents waiting for enough time to pass to be sure the remote TCP received the acknowledgment of its connection termination request.
- CLOSED - represents no connection state at all.

TRASFERIMENTO DATI AFFIDABILE

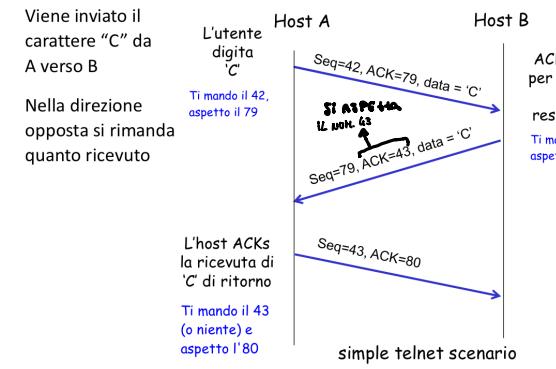
TCP crea un servizio di trasferimento dati affidabile sul servizio infidabile di IP,

checksum: Controllo sugli errori, i segmenti corrotti vengono scartati

Riscontri:

- Numero di sequenza di un segmento è il numero del primo byte del segmento nel flusso di byte. N.B. i numeri di sequenza si applicano ai byte, non ai segmenti trasmessi
- Numero di riscontro: il numero di sequenza del byte successivo che l'host attende dall'altro.
- Riscontro cumulativo: si effettua il riscontro dei byte fino al primo byte mancante nel flusso
- Timer: Avviato alla trasmissione di un segmento

TELNET su TCP

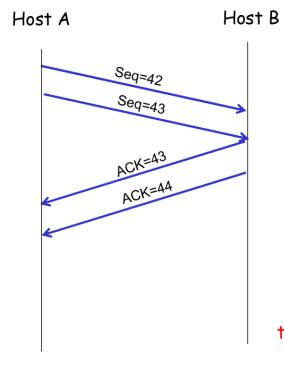


Kurose, Fig. 3.31 (3.a ed.)

Pipeline

Pipeline:

- il mittente può inviare più segmenti senza attendere il riscontro
- Permette di aumentare la produttività



52
time
53

Azioni LATO MITTENTE

1) TCP riceve dati dall'applicazione, gli incapsula in uno o più segmenti e assegna un numero di sequenza.
Avvia il timer di trasmissione (RTO)

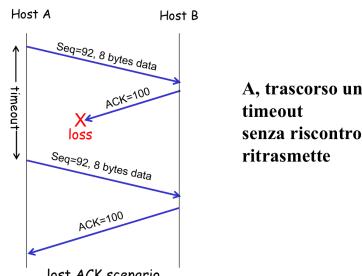
Se non in funzione per un altro segmento

2) Ritrasmissione dei segmenti in caso di:

a) Timeout: Ritrasmette il segmento che ha causato il timeout e riavvia il timer

b) ACK duplicato:

Se il mittente riceve ACK duplicati (3 o più) il segmento successivo a quello riscontrato è escluso. E' quindi ritrasmesso velocemente, prima della scadenza del timer.



A, trascorso un timeout senza riscontro, ritrasmette

3)

Segmenti fuori sequenza

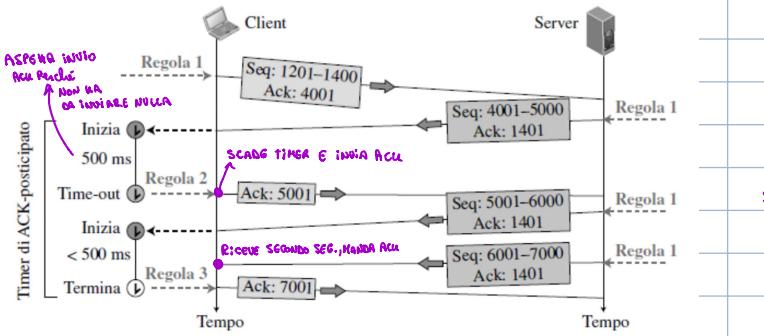
- I dati possono arrivare fuori sequenza ed essere temporaneamente memorizzati dall'entità TCP destinataria
- Il TCP non dice come il destinatario deve gestire i pacchetti fuori sequenza, dipende dall'implementazione

Nelle versioni più recenti si implementa SACK

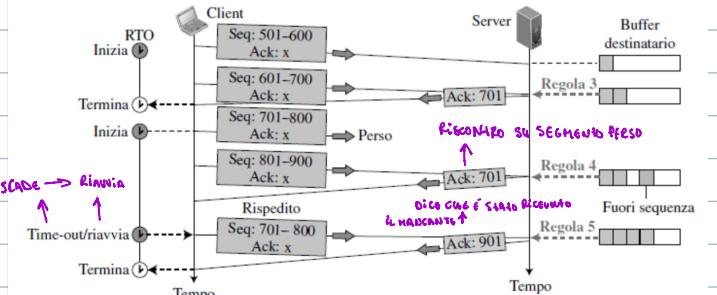
- i pacchetti ricevuti fuori sequenza vengono memorizzati
- riscontro di pacchetti fuori sequenza e duplicati inviato in OPTIONS

CASI

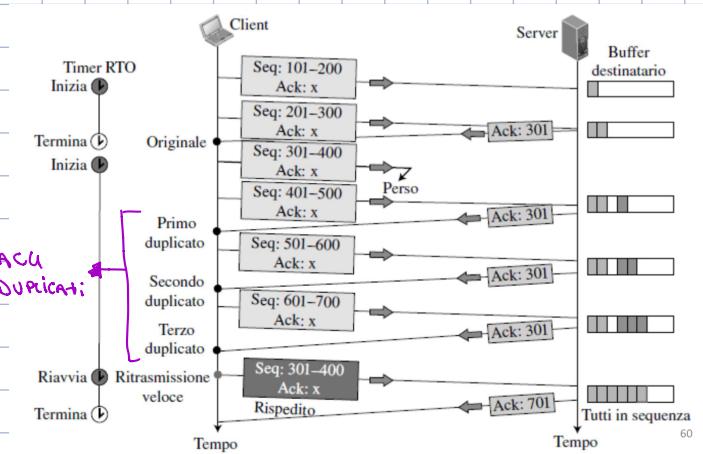
1) OPERATIVITÀ NORMALE



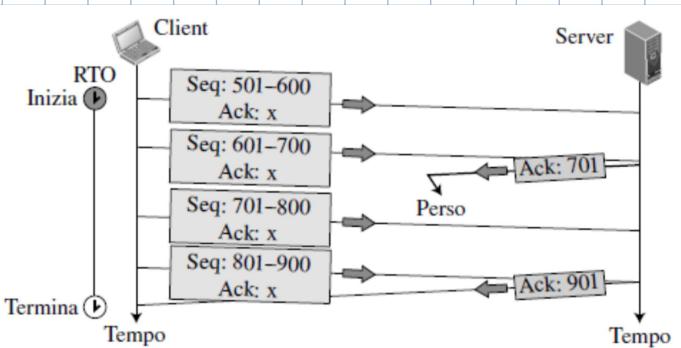
2) SEGUIMENTO SMARRITO



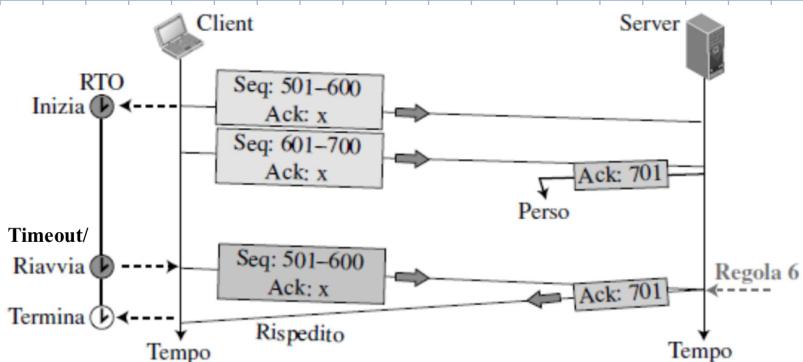
3) RITRASMISSIONE VELOCE



4) RISCONTRO SMARRITO (NO-TIMEOUT)



5) RISCONTRO PERSO CORRETTO DA RITRASMISSIONE (CON TIMEOUT)



CALCOLO DEL TIMEOUT

Il tempo del Timeout (RTO) deve essere maggiore di RTT (Round trip time)

↓

Viene calcolato analizzando gli

RTT dei segmenti non riconosciuti

Tempo trascorso da quando si invia un segmento a quando ne ne riceve il riscontro.

Sì considera ESTIMATED RTT cioè la combinazione tra i precedenti valori EstimatedRTT con il nuovo valore Sempl RTT

$$\text{Estimated RTT} = (1-\alpha) * \text{Estimated RTT} + \alpha * \text{Sample RTT}$$

↓
Stimale per un segmento trasmesso,
non per ogni invio.
E' molto più fluctuante.

Il valore di α viene posto a $1/8$ in modo da rendere via via meno importanti gli RTT dei pacchetti più vecchi

$$\text{Estimated RTT} = 0,875 * \text{Estimated RTT} + 0,125 * \text{Sample RTT}$$

Oltre all' EstimatedRTT è necessario anche una stima della variabilità di RTT

$$\text{RTT}_{\text{dev}} = (1-\beta) \text{RTT}_{\text{dev}} + \beta | \text{RTT}_{\text{sample}} - \text{RTT}_{\text{estimated}} |$$

↓
Stima di quanto si discosta da Estimated RTT

$$\beta \text{ viene posto} = 1/4$$

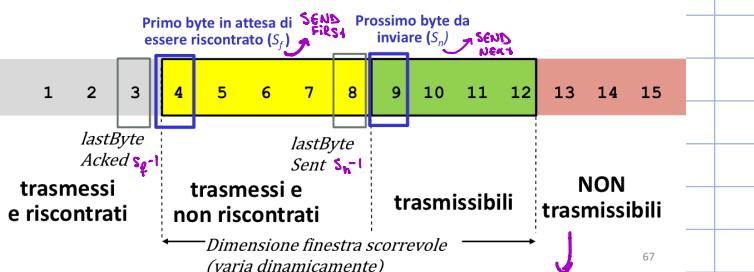
Infine il timeout viene calcolato come:

$$RTO = \text{RTT}_{\text{estimated}} + 4 \text{RTT}_{\text{dev}}$$

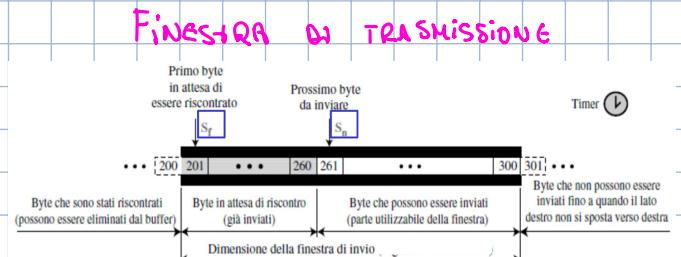
In alcune implementazioni si raddoppia l'RTO dopo un errore (es: ACK non ricevuto)

FINESTRA DI TRASMISSIONE

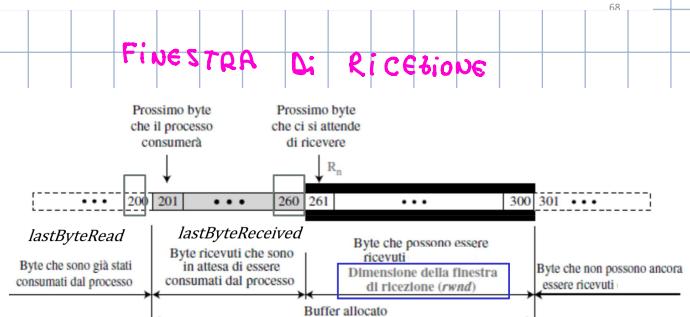
- I dati inviati dal processo a livello applicativo sono mantenuti nel buffer di invio
- La trasmissione dei dati si basa sulla finestra di trasmissione (*sliding window*).
 - finestra sovrapposta sulla sequenza da trasmettere
 - negoziata dinamicamente
 - viene fatta avanzare alla ricezione di un ACK



Non mentiamo
nella finestra
di trasmissione,
bisogna aspettare
che un byte
venga riscontrato



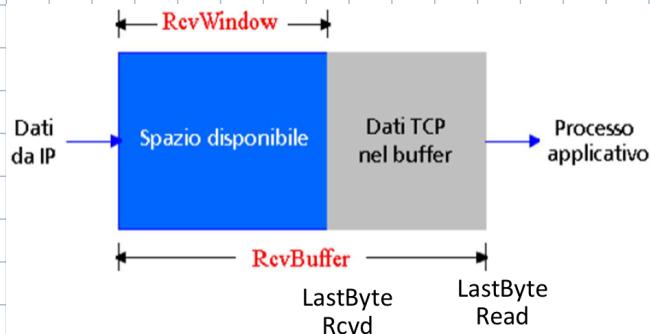
S_f (SendFirst): numero di sequenza del primo byte in attesa di essere riscontrato (chiamato SendBase sul Kurose, SND.UNA nelle RFC)
 S_n (Send Next): prossimo byte da inviare (prossimo numero di sequenza da inviare, chiamato NextSeqNum sul Kurose, SND.NXT nelle RFC)



CONTROCCO DI FLUSSO = Capacità del mittente di evitare possibilità di saturare il buffer del ricevitore

- 1) Oggi host impone buffer di invio e ricezione
- 2) Il processo applicativo destinatario legge i dati dal buffer di ricezione.

Viene implementato tramite la finestra di ricezione mantenuta nel mittente, la quale fornisce un'idea di quanto spazio c'è ancora a disposizione nel buffer del ricevitore. Tale valore è comunicato nel campo window dell'header TCP.



Spazio disponibile nel buffer del destinatario: $Rwnd = RcvBuffer - (LastByteReceived - LastByteRead)$

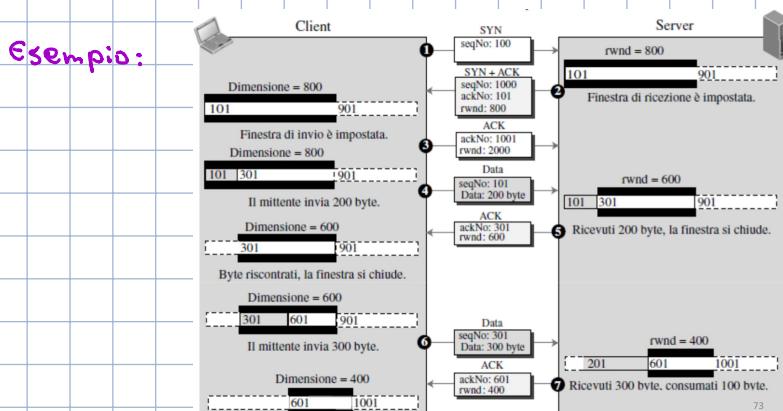
Rwnd:

- 1) È dinamico
- 2) L'Host destinatario comunica la dim. di Rwnd al mittente
- 3) Il mittente si accorghe che $LastByteSent - LastByteAcked < Rwnd$

Quantità di dati trasmessi e ancora non riscontrati

OSS: Se RWN=0, il mittente manda segmenti <>sonda>> di 4 byte per ricevere

l'aggiornamento sulla dimensione di RWN



CONTROLLO DI CONGESTIONE

Il protocollo TCP prevede il controllo della congestione, imponendo a ciascun mittente di limitare la freq. di invio di pacchetti sulla connessione, in funzione della congestione percepita. Questo permette alla capacità di TCP di adattarsi alla velocità della rete.

Controllo di congestione punto-punto:

- 1) Nessun supporto esplicito delle reti
- 2) La congestione è debolita dai sistemi terminali

Finestra di congestione: Basata su una stima della capacità della rete

la dimensione della finestra di invio sarà quindi: $\min(RWND, CWND)$

Finestra di congestione

Si misura in MSS (Maximum Segment Size)



1 MSS = quantità massima di dati trasportabili da un segmento

- Determinato in base alla MTU (unità trasmissiva massima) – lunghezza massima del payload del frame di collegamento inviabile dall'host mittente
- MSS scelto in modo tale che il segmento TCP, incapsulato in pacchetto IP, stia dentro un singolo frame di collegamento
- Esempi tipici di MSS 1460, 536,512 byte (tolgo dall'MTU 20 byte header TCP + 20 byte header IP)

Algoritmo per il controllo della congestione (calcolo dim. Finestra di congestione)

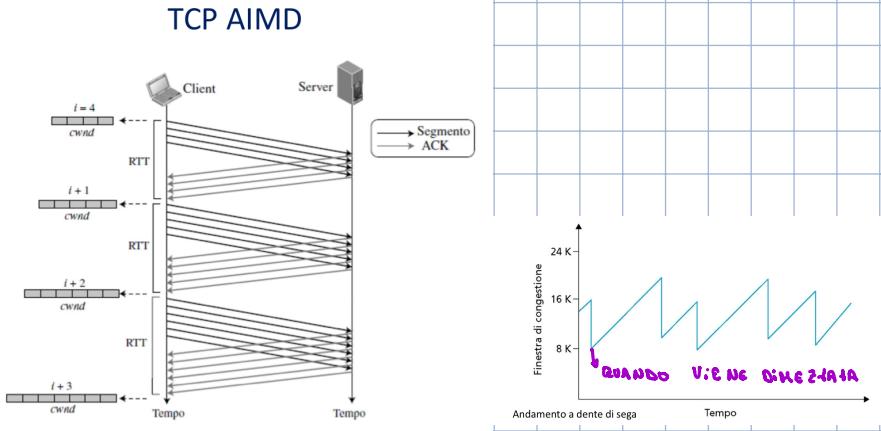
- 1) Partenza lenta (Slow Start)
- 2) Incremento additivo e decremento moltiplicativo (AIMD)
- 3) Ripresa veloce (Fast Recovery)
- 4) Reazione al time-out

AIMD

TCP del mittente aumenta proporzionalmente la propria finestra di congestione ad ogni ACK ricevuto.

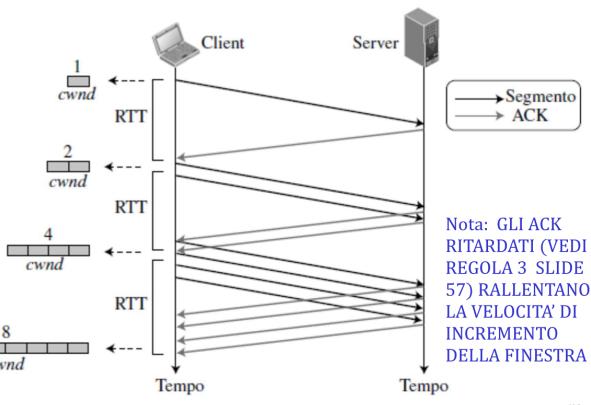
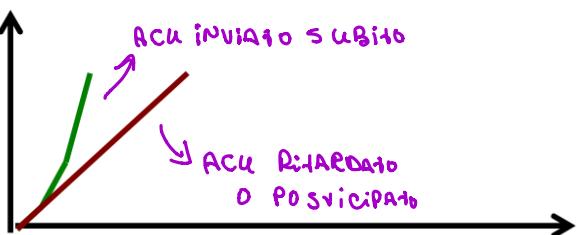
- Di quanto viene incrementata la finestra?
- Ad ogni ACK la finestra di congestione viene incrementata in modo che si abbia un crescita pari ad 1 MSS per ogni RTT (Congestion avoidance).
 - Ad ogni ACK $cwnd = cwnd + 1/cwnd$
 - Es $cwnd = 4 \text{ MSS} \rightarrow \text{incremento di } 1 \text{ MSS}$

TCP del mittente dimezza la propria CongWin ad ogni evento di perdita



Slow Start

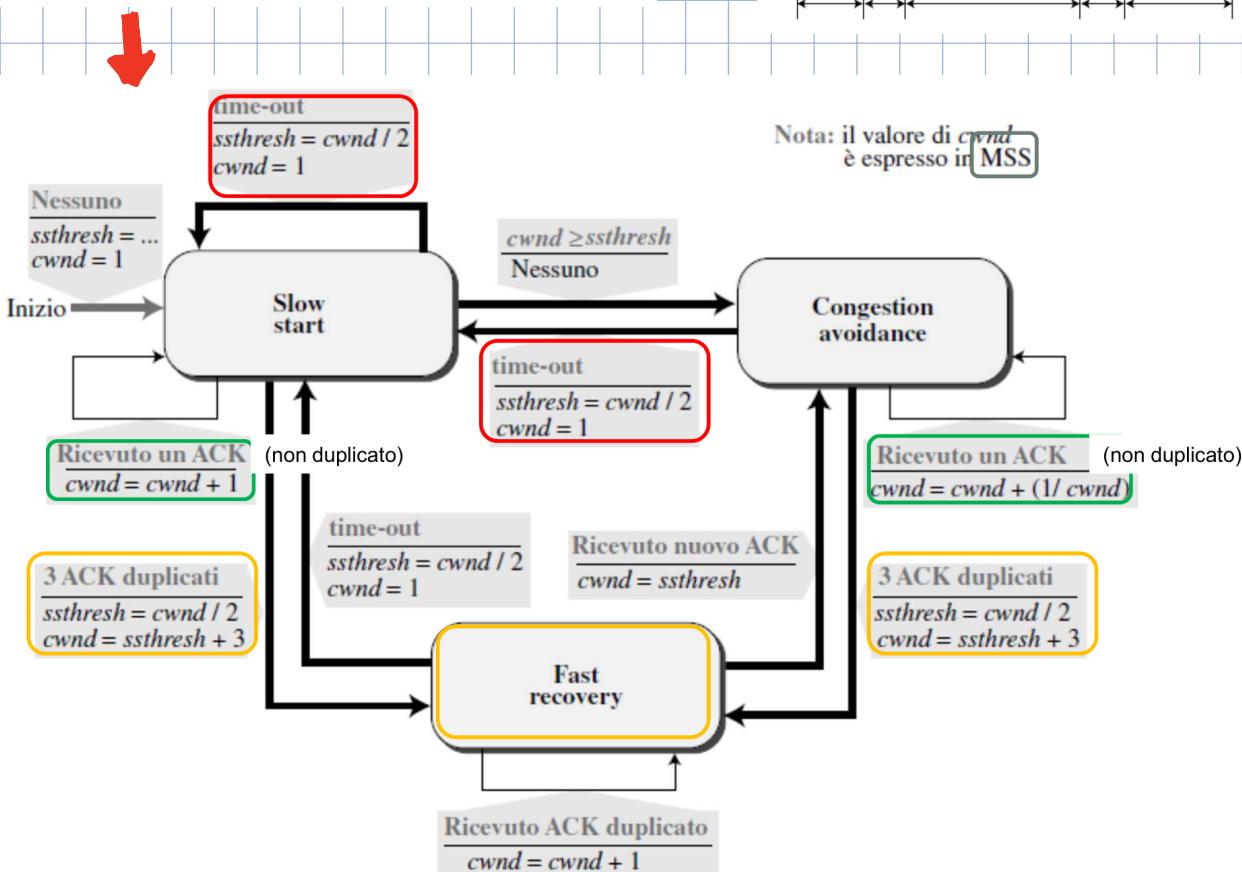
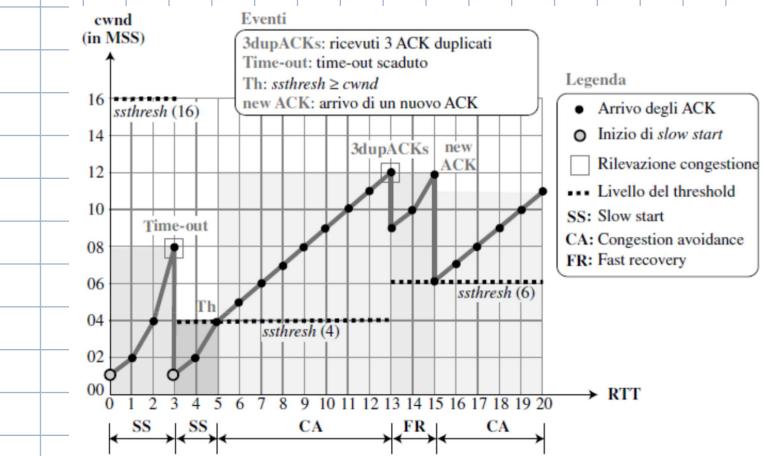
- All'inizio la finestra di congestione, CongWin, è posta pari a 1 MSS (frequenza di invio 1 MSS/RTT)
 - Se MSS=500 byte e RTT=200ms si ha una frequenza di invio di circa 20kb/s: se ho 1Mb/s di banda, impiego molto tempo ad arrivarci con incremento lineare (vedi AIMD).
- Incremento Congwin di 1MSS ad ogni ACK (**non duplicato**)
- L'effetto è che CongWin raddoppia ad ogni RTT → crescita esponenziale



TCP RENO (Algoritmo)

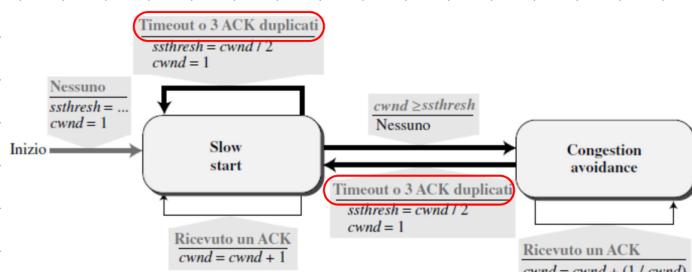
- Inizialmente viene definita una variabile "soglia", alla quale è assegnato un valore alto (es. 64 KB).
 - La soglia determina quando termina la partenza lenta (slow start) ed inizia la fase di congestion avoidance (CA)
- Se $cwnd < \text{soglia}$, cwnd aumenta esponenzialmente (**slow start**)
- Se $cwnd > \text{soglia}$, cwnd aumenta linearmente (**Additive Increase**)
- Evento di perdita
 - Se ho **3 ACK duplicati** pongo prima la soglia a metà di cwnd e poi $cwnd = \text{soglia} + 3 \text{ MSS}$ (**fast recovery**)
 - Se ho un ACK perso per **timeout** pongo la soglia a metà di cwnd e pongo $cwnd = 1 \text{ MSS}$ (**slow start**)
- Nella fase fast recovery:
 - se avviene un timeout si va in slow start
 - finché continuano ad arrivare ACK duplicati $cwnd=cwnd+1$
 - se arriva un nuovo ack (non duplicato) si va in congestion avoidance e $cwnd=\text{soglia}$

Esempio:



Seconda parte del programma

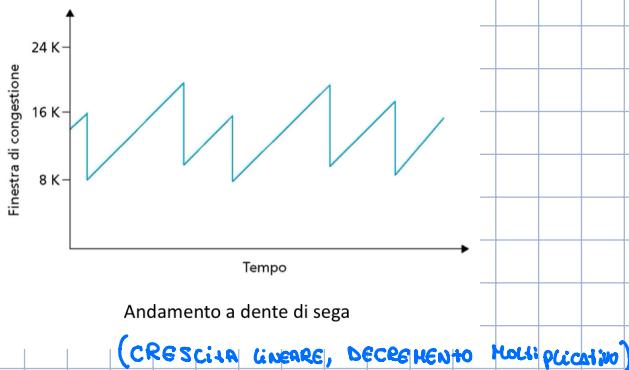
TCP Tahoe (Algoritmo)



IN GENERALE:

Andamento macroscopico della finestra di congestione

- Nella maggior parte dei casi la congestione viene rilevata tramite 3 ack duplicati
- Trascurando slow start e Fast recovery
 - > **incremento additivo** (finestra aumenta di $1/cwnd$ ad ogni riscontro non duplicato)
 - e **decremento moltiplicativo** ($cwnd$ dimezza)



TCP Throughput

w = Valore massimo in byte della finestra di congestione (ovvero quando si verifica l'errore)

$$\text{Throughput} = \frac{0,75 \cdot W}{RTT}$$

Medio

Lo 0,75 si ottiene perché:

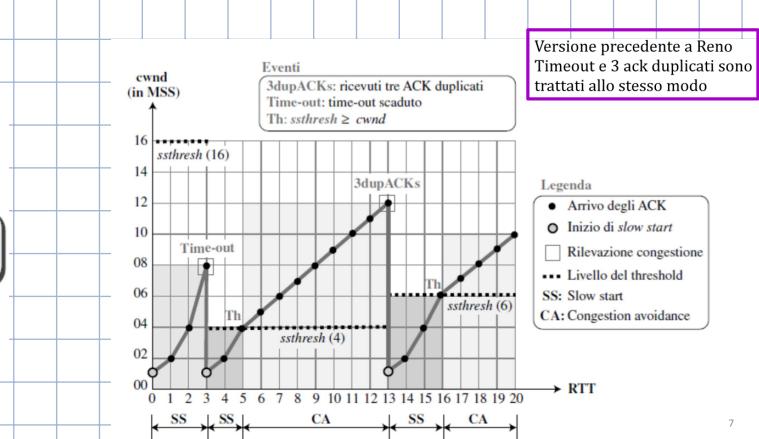
- Quando la finestra è w il throughput è W/RTT

- Dopo un evento di perdita, la finestra va a $w/2$ \Rightarrow throughput = $W/2RTT$

Quindi il throughput medio vale: $0,75 W/RTT$



È interessante perché la quantità di dati che posso essere inviati dipende dalla finestra, e il RTT indipendentemente dalla rate di trasmissione sul collegamento

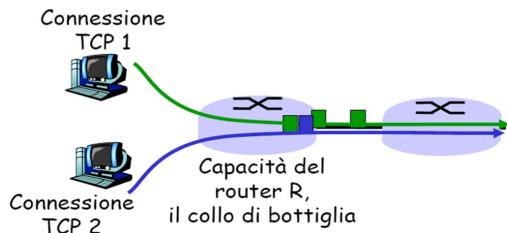


Versione precedente a Reno
Timeout e 3 ack duplicati sono trattati allo stesso modo

7

EQUILIBRIO DEL TCP

- Ipotesi:
 - K connessioni TCP insistono su un unico link di capacità R bit/s
 - Le connessioni hanno gli stessi valori di MSS e RTT
 - Non ci sono altri protocolli che insistono sullo stesso link
- Risultato:
 - Ognuna delle connessioni TCP tende a trasmettere R/K bit/s



REALITÀ
⇒

NOTA: nella pratica connessioni con RTT più piccolo variano più velocemente congwin e raggiungono throughput superiori a connessioni con RTT maggiore

UDP

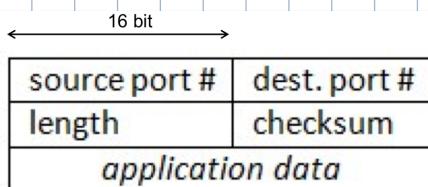
- Servizio di consegna a massimo sforzo
 - I datagrammi UDP possono essere perduti o consegnati fuori sequenza
 - Nota: L'affidabilità può essere aggiunta al livello applicazione
- Orientamento al messaggio (*4e TCP è orientato allo stream di byte*)
 - Ogni datagramma UDP indipendente dall'altro
 - I processi devono inviare messaggi di dimensioni limitate, che possono essere incapsulati in un datagramma UDP

Rispetto al TCP l'UDP è meno complesso, offre meno servizi, ma è più indicato in contesti dove occorre un completo controllo della temporizzazione (applicazioni time-sensitive come la trasmissione di dati multimediali)

Proprietà:

- 1) Nessuna connessione (non introduce ritardo)
- 2) Checksum facoltativo
- 3) Facile e leggero da gestire
- 4) Utilizzato in applicazioni multimediali, DNS, SNMP, telefonia ecc...
 ↗ No controllo di flusso
 ↗ No controllo di congestione

Datagramma UDP



- 8 byte di intestazione
- **Porta:** numeri di porta della comunicazione (per il demultiplexing è usato solo quello di destinazione).
- **Lunghezza del messaggio:** lunghezza totale (header + dati) del datagramma UDP (16 bit -> max 65535 byte)

- **Checksum:** checksum dell'intero datagramma
- E' opzionale in UDP.
 - Controllo errore end-to-end
 - Il pacchetto corrotto viene scartato, ma il mittente non ne riceve notifica → **Nel caso di UDP, nel caso di TCP viene inviato nuovamente.**
- Calcolata sull'intero datagramma UDP più lo pseudo-header (ovvero parte dell'header IP)

Pseudoheader – usato per calcolo checksum

0	8	16	31
Indirizzo IP di Provenienza			
Indirizzo IP di Destinazione			
Zero	Proto (17)	Lunghezza UDP	

↓
Se viene usato
TCP o UDP

Mittente

- Campo checksum a 0
- Tratta il contenuto del datagramma UDP come una sequenza di parole da 16 bit
- checksum: somma le parole di 16 bit in complemento a uno
- Complemento a uno del risultato della somma
- Il mittente pone il valore della checksum nel campo checksum del datagramma UDP

a capo	1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
	1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
somma	1 0 1 1 1 0 1 1 1 0 1 1 1 0 0 1
	0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 1

- Analogico per TCP
- In TCP la checksum è obbligatoria

19

Riepilogo :

Servizio UDP:

- trasferimento dati non affidabile tra mittente e destinatario
- non fornisce: setup della connessione, affidabilità, controllo di flusso, controllo della congestione, timing, o garanzia di banda
- Richiede minor overhead

Servizio TCP:

- **connection-oriented:** richiesto handshake tra client e server
- **trasporto affidabile** tra i processi mittente e destinatario
- **controllo di flusso:** il mittente non saturerà il destinatario
- **controllo della congestione:** limita il mittente quando la rete è sovraccarica
- **non fornisce:** timing, banda minima garantita

Nella programmazione di rete si deve ricordare che:

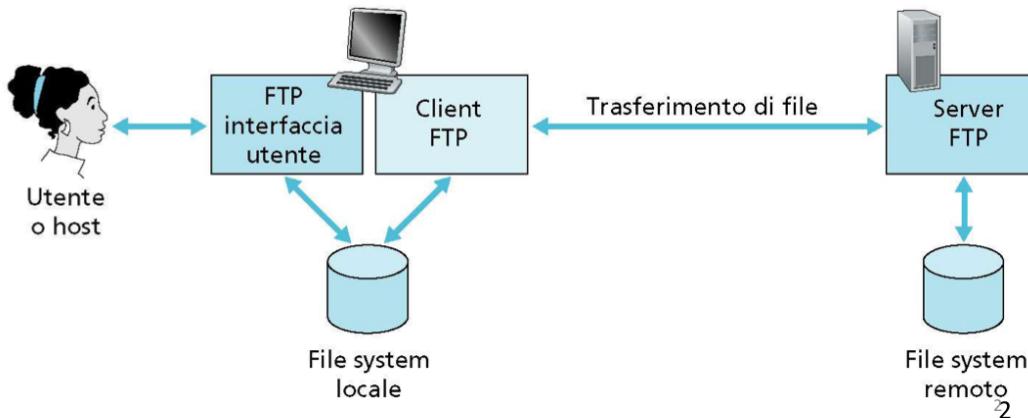
- il TCP offre un servizio di trasporto a stream, quindi si può leggere da un input di rete quanti byte si desiderano.
- l'UDP offre un servizio a messaggi, quindi occorre leggere tutto il messaggio in arrivo.
- **UDP adeguato per**
 - processi che richiedono uno scambio di dati con volume limitato con scarso interesse al controllo di flusso e degli errori
 - processi che hanno meccanismi interni di controllo di flusso e degli errori
 - trasmissioni multicast (destinatari multipli)
 - applicazioni interattive in tempo reale che non tollerano ritardi variabili

FTP (File transfer protocol)

Protocollo utilizzato per il trasferimento di file a/di un host remoto.

Modello client/server: a) client: richiede il trasferimento

b) server: host remoto



FTP è lo standard per il trasferimento di file in reti TCP/IP.

I file vengono ottenuti come copia locale, quindi per modificare il file su host remoto bisogna effettuare il trasferimento del file modificato.

FTP fornisce funzionalità aggiuntive rispetto al semplice trasferimento file

- accesso interattivo: l'utente può navigare e cambiare/modificare l'albero di directory nel file system remoto
- Specifica del formato dei dati da trasferire (es. File di testo o file binari)
- Autenticazione: il client può specificare login e password

Con il protocollo FTP possono essere eseguiti 2 tipi di connessione (**Utilizza TCP su porta 21**)

control connection: scambio di comandi e risposte tra client e server.

Segue il protocollo Telnet.

data connection: Connessione su cui i dati sono trasferiti con modi e tipi specificati. I dati trasferiti possono essere parte di un file, un file o un set di file

FTP è un protocollo **STATEFUL** →

Il server deve tener traccia dello stato dell'utente (connessione di controllo associata ad un account, directory attuale, ecc..)

Control connection:

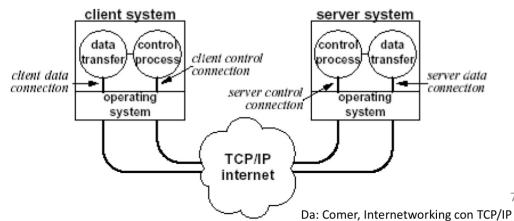
- Il client FTP contatta il server FTP alla porta 21
- Il client ottiene l'autorizzazione sulla connessione di controllo
- Il client invia i comandi sulla connessione di controllo (es. cambio directory, invio file, ecc.)
- La connessione di controllo è **persistente**



Quando il client attiva la connessione di controllo con il server, usa un numero di porta assegnato localmente in modo casuale e contatta il server ad una porta nota (21)

FTP usa la connessione di controllo per permettere a client e server di coordinare l'uso delle porte assegnate dinamicamente **per il trasferimento dati**

La comunicazione sulla connessione di controllo avviene per mezzo di caratteri con una codifica standard NVT ASCII, sia per i comandi che per le risposte (Telnet)



Comandi di controllo

Comandi

- USER username
- PASS password
- LIST elenca i file della directory corrente
- NLST richiede elenco file e directory (ls)
- RETR filename: recupera (get) un file dalla directory corrente
- STOR filename: memorizza (put) un file nell'host remoto

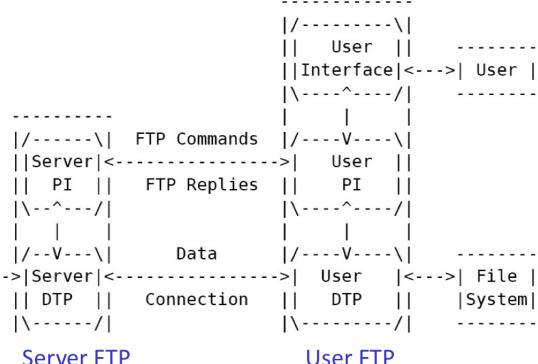
- ABOR interrompe l'ultimo comando ed i trasferimenti in corso
- PORT indirizzo e numero di porta del client
- SYST il server restituisce il tipo di sistema
- QUIT (quit): chiude la connessione

Codici di ritorno

- 125 data connection already open; transfer starting
- 225 Data connection open
- 226 Closing data connection. Requested file action successful (for example, file transfer or file abort).
- 425 Can't open data connection
- 452 Error writing file
- 200 Comando OK

Codici di ritorno

Modello FTP



- DTP: Data Transfer Process
- PI: Protocol Interpreter

Ancora su Modello FTP

- I dispositivi dove risiedono client e server FTP sono diversi:
 - Sistema operativo, strutture per gestire i file, diversi formati dei file
 - Per effettuare il trasferimento file, il client deve definire il tipo di file, la struttura dati e la modalità di trasmissione al fine di risolvere i problemi di eterogeneità tra client e server.
 - Il trasferimento file viene preparato attraverso uno scambio di informazioni lungo la connessione di controllo
- Modalità di trasmissione**
 - Stream mode: FTP invia i dati a TCP con unflusso continuo di bit
 - Block mode: FTP invia i dati a TCP suddivisi in blocchi. Ogni blocco è preceduto da un header
 - Compressed mode: si trasmette il file compresso

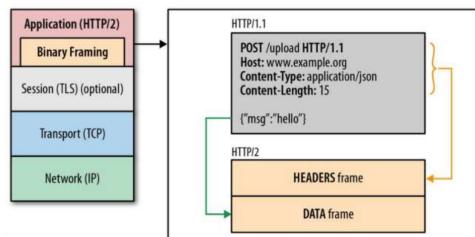
Apprendimento HTTP

HTTP/2

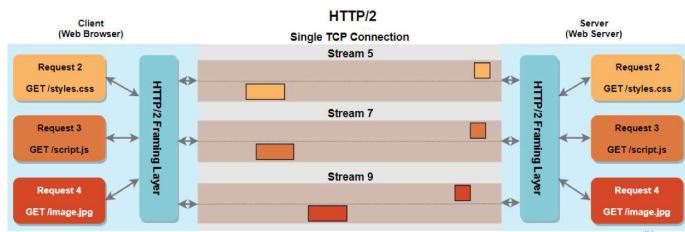
Caratteristiche HTTP-2

1) Multiplexing delle richieste su una unica connessione TCP

- Frame:** l'unità di comunicazione in HTTP/2. Una sequenza di frame mappa un messaggio HTTP (es. un frame può essere un HEADER frame, DATA frame..)



- Stream:** un flusso bidirezionale di frame all'interno di un'unica connessione TCP, rappresenta una comunicazione richiesta risposta
- Mediante l'astrazione degli stream è possibile effettuare il **multiplexing delle richieste** (più stream sulla stessa connessione TCP)



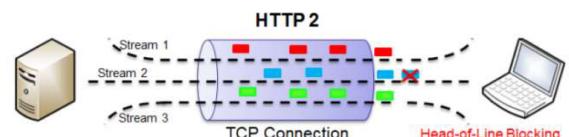
2) Definizione delle priorità

E' possibile associare ad ogni stream:

- Un **peso** che ne indica la priorità
- Una **dipendenza** verso altri stream

5)

- HTTP/2 risolve il problema di uso inefficiente di una connessione TCP
 - In particolare per pagina con tante risorse
 - Vedi demo <https://http2.akamai.com/demo>
 - https://http1.akamai.com/demo/h1_demo_frame.html
 - https://http2.akamai.com/demo/h2_demo_frame.html
 - ma...
- LIMITE: **Head Of Line Blocking:** Una perdita (3 ack/timeout) provoca lo stall della connessione (e quindi tutti gli stream)



3) Comprensione delle interazioni

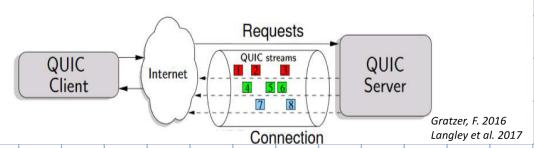
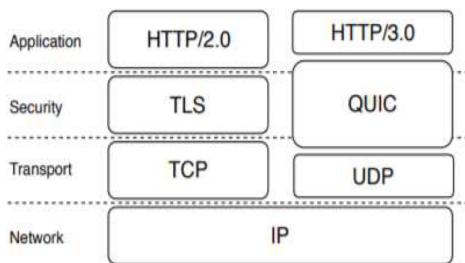
4) Server Push: Il meccanismo di **Server Push**, permette al server di inviare risorse aggiuntive per una singola richiesta da parte del client

HTTP/3

- Evoluzione di HTTP/2 che usa i servizi di **QUIC**
- Mantiene la semantica di HTTP/2
- Gli stream non sono trasparenti al livello di trasporto
- <https://datatracker.ietf.org/doc/draft-ietf-quic-http/>

Quic

- Protocollo di trasporto che sfrutta UDP
- Aggiunge:
 - Controllo del flusso e della congestione
 - Rilevazione delle perdite e ritrasmissione
- IMPORTANTE:**
 - Astrazione dello stream gestito a livello di trasporto
 - Gli stream sono indipendenti tra loro, la perdita o rallentamento di uno stream non influisce sul progredire degli altri streams



Livello di RETE

- Responsabile della consegna dei datagrammi tra gli host
- Offre servizi allo strato di trasporto.
- Utilizza i servizi dello strato collegamento

Comunicazione a livello di rete

Servizi offerti

Lista di Servizi che possono essere offerti da un livello di rete generico -> prossima slide IP

1. L'entità a livello di rete riceve i segmenti dal livello di trasporto nell'host mittente, e incapsula i segmenti in datagrammi

2. I datagrammi sono inoltrati al prossimo nodo (host o router)

3. Il router esamina i campi intestazione in tutti i datagrammi IP che lo attraversano e li inoltra da un collegamento in ingresso ad un collegamento in uscita

4. Sul lato destinatario, consegna i segmenti al livello di trasporto (demux TCP o UDP)

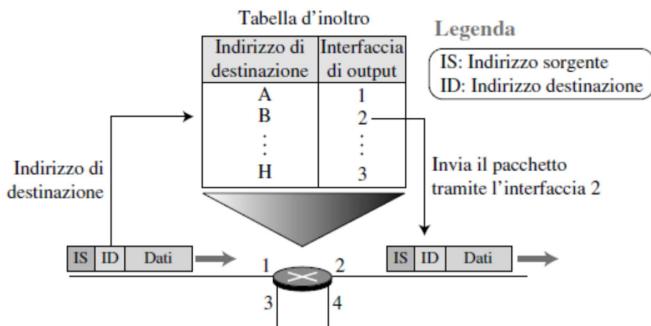
- Suddivisione in pacchetti
- Instradamento (routing)
- Inoltro (forwarding)
- Controllo errori
- Controllo flusso
- Controllo congestione
- Qualità del servizio
- Sicurezza

Protocollo IP (INTERNET PROTOCOL)

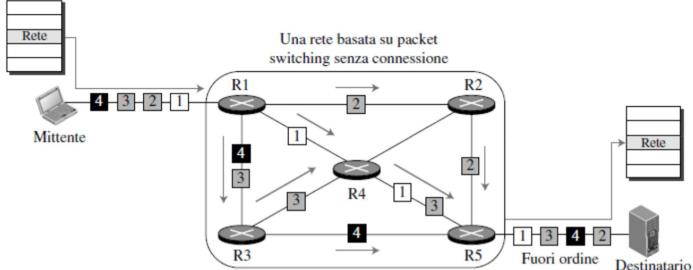
- 1) Protocollo connection-less (Nessun circuito virtuale e fisico a livello IP)
- 2) Servizio BEST-EFFORT (Nessimo impegno fornibile).
 - a) Non c'è garanzia sull'ordine di arrivo dei pacchetti
 - b) Non c'è garanzia la consegna dei pacchetti
- 3) Non è affidabile, nessun meccanismo di recupero di errore
- 4) Non prevede garanzie sulle qualità del servizio (QoS), tempo di consegna e controllo di flusso (Possono essere aggiunti mediante estensioni)

Funzionalità IP :

- Inoltro (Forwarding): trasferimento del pacchetto sull'appropriato collegamento di uscita



- Instradamento: processo decisionale di scelta del percorso verso una destinazione (algoritmi di instradamento o routing)



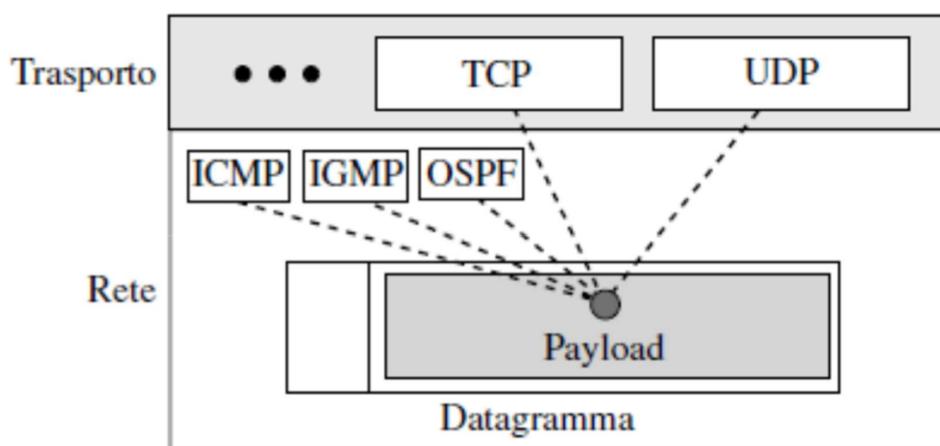
NB: la tabella di inoltro viene costituita con la funzione di instradamento.

MECCANISMI IP

- Indirizzamento
 - Strumento per identificare gli host nell'internet
- Modello datagram (senza connessione) per la consegna dei dati
 - Segmentazione e riunificazione dei pacchetti (adattamento al data-link)
 - Controllo degli errori (solo header)
 - Verifica TTL

MULTIPLEXING E DEMULTIPLEXING

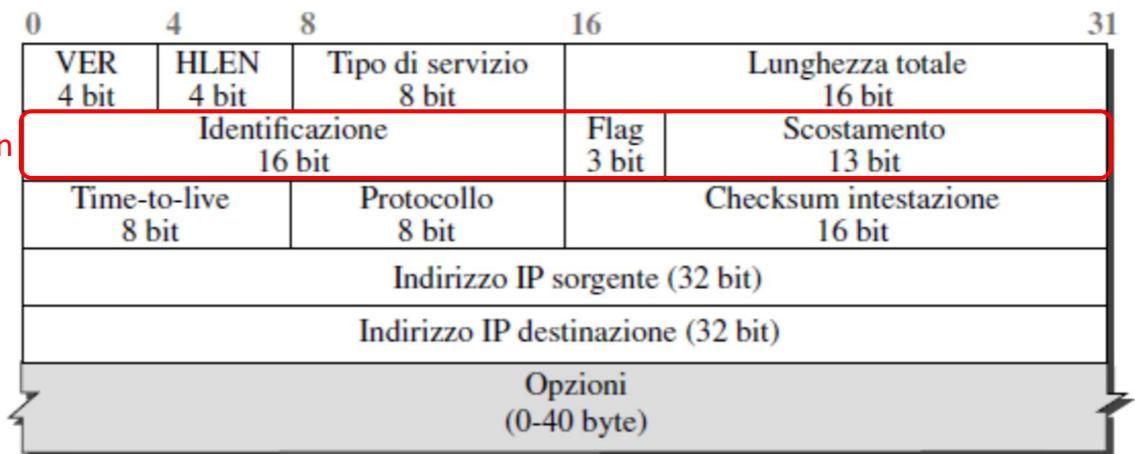
Anche a livello di rete si effettua multiplexing e demultiplexing.



Alcuni valori dei protocolli

ICMP: 01
IGMP: 02
TCP: 06
UDP: 17
OSPF: 89

DATAGRAMMA IP



- **Versione (4 bit)**: specifica la versione di IP usata (attualmente IPv4 o IPv6)
- **Hlen (lunghezza dell'header, 4 bit)**: lunghezza dell'intestazione espressa in parole da 32 bit. Tipicamente vale 0101 ovvero 20 byte.
- **Tipo di servizio (8 bit)**: serve per "colorare" il datagramma IP (basso ritardo, affidabilità, ecc...). Inizialmente Type of Service, poi :
 - 4 bit per Differentiated Services
 - I pacchetti vengono marcati in base alla classe di servizio (telefonia, controllo, multimedia streaming, dati a bassa priorità)
 - Differenti politiche di accodamento ai router
 - 2 bit Explicit Congestion Notification (ECN) – supporto a livello rete e trasporto per la notifica di eventi di congestione
- **Protocollo (8 bit)**: in ricezione all'host destinatario indica quale protocollo dello strato superiore deve ricevere i dati. Inizialmente la tabella era nella RFC1700 (STD2), attualmente è un database online che si trova su iana.org
 - Demux, es. TCP 6, UDP 17
- **Checksum dell'intestazione (16 bit)**: viene calcolato il checksum della sola intestazione (ponendo questo campo pari a zero) ad ogni router (il TTL cambia ad ogni hop!). Se si ottiene un errore si scarta il datagramma.
- **Lunghezza del datagramma (16 bit)**: è la lunghezza di tutto il datagramma in byte, header incluso.
 - Dim. Max 65535 byte (nella pratica 1500 byte)
- **Identificatore, flag, offset**: sono campi per la frammentazione (vedi più avanti).
- **Tempo di vita (8 bit)**: ad ogni passaggio da un router viene decrementato, quando raggiunge lo zero viene scartato. Assicura che eventuali percorsi ad anello non provochino traffico perpetuo nella rete
 - Valore iniziale dipende dal S.O.: esempi 30, 64, 128, 255
- **Indirizzi sorgente e destinazione (32 bit)**: indirizzi IP del mittente e del destinatario.
- **Opzioni (lunghezza variabile, multiplo di 32 bit)**: uso sporadico, da 0 a 40 byte
- **Dati**: dati trasportati dal datagramma IP.

Frammentazione

MTU (Maximum Transfer Unit) : Quantità massima di dati trasportata dal protocollo di collegamento in un frame.

Il

Dimensione massima del payload del frame.

Pone un limite alla lunghezza dei datagrammi IP e traesse diverse potranno porre limiti differenti.

Mecanismo di frammentazione:

- Se il router riceve un datagramma la cui dimensione supera l'MTU della rete verso cui deve inoltrare quel datagramma, il router frammenta il datagramma IP in due o più datagrammi più piccoli, detti frammenti.
- Il riassemblaggio viene effettuato dall'entità rete nel sistema terminale (destinazione).
- Se qualche frammento non arriva a destinazione?
 - si butta via tutto il datagramma

OSSERVAZIONI:

1) Un pacchetto diviso in n frammenti. Come sono inoltrati questi frammenti?

- Ciascun frammento è a sua volta un datagramma IP completo che viene trasmesso attraverso una serie di reti fisiche indipendentemente dagli altri

2) I frammenti possono arrivare fuori ordine, come fa il livello di rete a ricomporre i frammenti nel pacchetto originale?

Usa identificazione, flag, e offset

- Identificatore (16 bit)** è associato al datagramma dall'host sorgente. Associato a IP sorgente e IP destinazione identifica quel datagramma in un intervallo di tempo ragionevolmente lungo.
 - i frammenti di quel datagramma mantengono il valore di questo campo. Il destinatario riconosce i frammenti che vanno assemblati insieme
- Offset (13 bit)** indica la posizione relativa come multiplo di 8 byte. Serve ad ordinare i frammenti nell'assemblaggio.
 - I frammenti devono essere multipli di 8 byte
 - Primo frammento Offset =0
- Flag (3 bit)** serve ad identificare l'ultimo frammento – vedi slide seguente

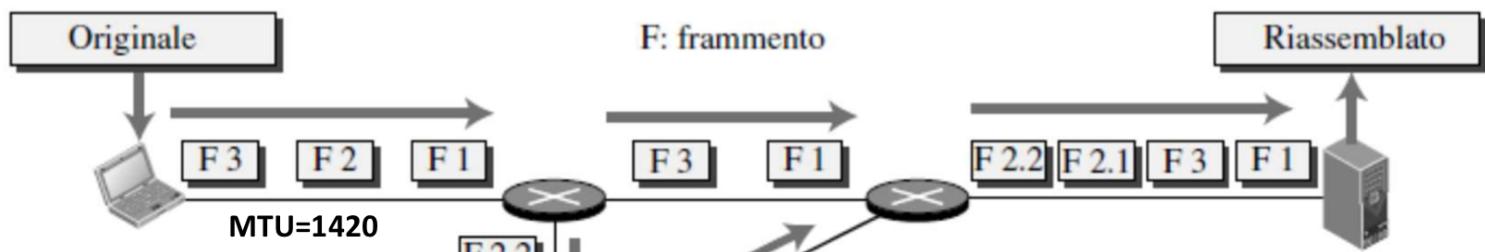
Flag

- Il bit 0 è riservato (per ora vale sempre 0)
- Il bit 1 *do not fragment* vale:
 - 0 se il pacchetto può essere frammentato
 - 1 se il pacchetto non deve essere frammentato
- Il bit 2 *more fragments* vale:
 - 0 se il pacchetto è l'ultimo del frammento
 - 1 se il pacchetto non è l'ultimo del frammento

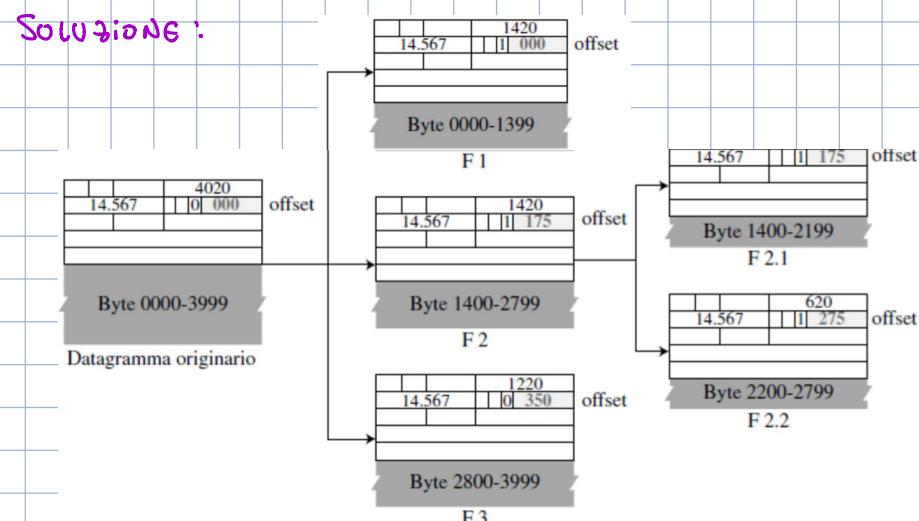
3) Il processo di frammentazione può essere ripetuto

es: Se un frammento deve essere inviato su un collegamento MTU ancora più piccolo

Esempio:



SOLUZIONE:



INDIRIZZAMENTO IP

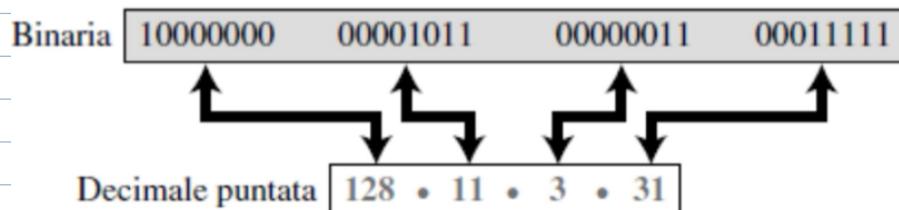
INDIRIZZI IPv4

Ogni host è connetto ad internet attraverso un'interfaccia di rete la quale è il confine fra l'host ed il collegamento su cui vengono inviati i datagrammi.

Ancor ogni interfaccia di rete è assegnato un indirizzo IP.

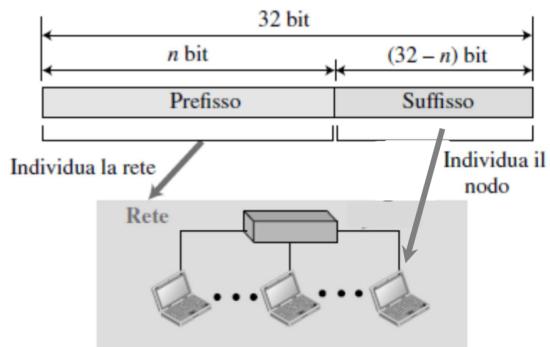
I router devono essere necessariamente connetti ad almeno 2 collegamenti.

Gli indirizzi IP sono costituiti da 32 bit (4 byte), rappresentati in notazione decimale puntata.



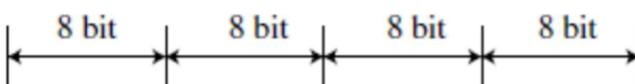
Ogni host ha un indirizzo IP univoco diviso in 2 parti:

Network ID + Host ID, che identificano una rete IP su Internet e l'host su quella rete IP.



CLASS FULL ADDRESSING (INDIRIZZAMENTO CON LE CLASSI)

Le reti sono divise in classi, ogni classe ha i prefissi di lunghezza fissa



Classe A	0 Prefisso	Suffisso
Classe B	10 Prefisso	Suffisso
Classe C	110 Prefisso	Suffisso
Classe D	1110 Indirizzi multicast	
Classe E	1111 Riservati per uso futuro	

Classe	Prefissi	Primo byte
A	$n = 8$ bit	Da 0 a 127
B	$n = 16$ bit	Da 128 a 191
C	$n = 24$ bit	Da 192 a 223
D	Non applicabile	Da 224 a 239
E	Non applicabile	Da 240 a 255

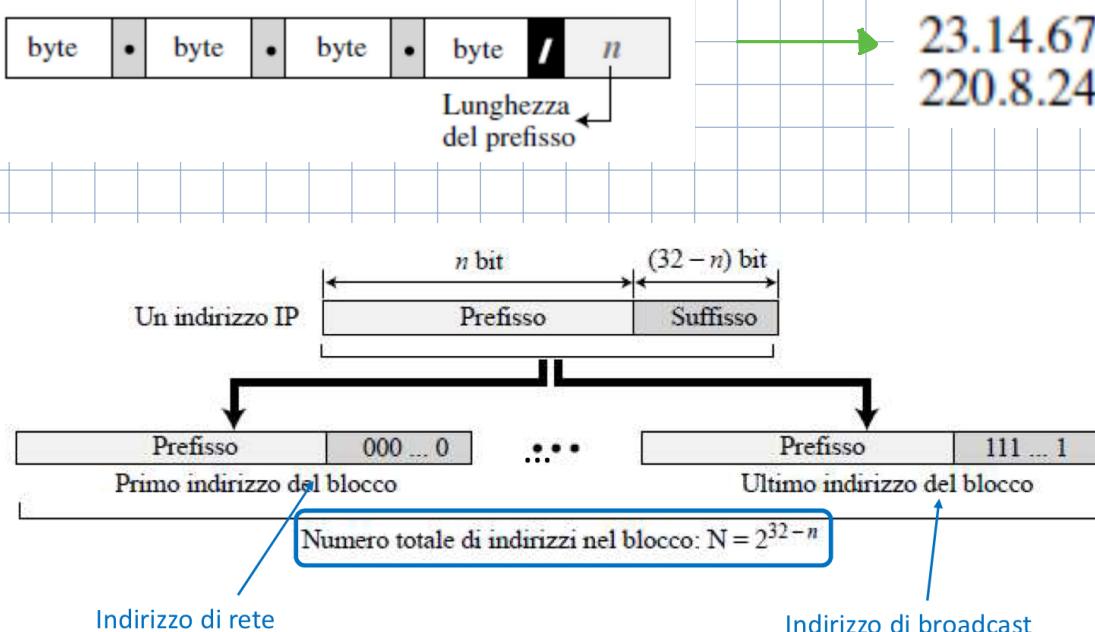
5 classi di indirizzi IP

- **Classe A:** 7 bit per 128 reti IP, 24 bit
 - 0.0.0.0 - 127.255.255.255
 - Reti di 16 milioni di host
- **Classe B:** 14 bit per ca. 16000 reti IP, 16 bit host id
 - 128.0.0.0 - 191.255.255.255
 - Reti di circa 64000 host
- **Classe C:** 21 bit per reti IP e 8 bit per host id
 - 192.0.0.0 - 223.255.255.255
 - Reti di circa 256 host
- **Classe D,** riservata a multicasting,
 - 224.0.0.0 - 239.255.255.255
- **Classe E,** riservata per usi futuri
 - 240.0.0.0 – 255.255.255

Esempi:

- 15.10.10.90: E' un indirizzo di classe A, poiché il primo numero è compreso fra 0 e 127, dunque i seguenti campi indicano l'host 10.10.90 nella rete IP 15.
- 130.250.42.53: E' un indirizzo di classe B, poiché il primo numero è compreso fra 128 e 191; dunque i primi due campi indicano la rete IP 130.250, gli altri indicano l'host 42.53.
- 196.234.12.14: E' un indirizzo di classe C, poiché il primo numero è compreso fra 192 e 223; dunque i primi tre campi indicano la rete IP 196.234.12, e l'ultimo indica l'host 14.

INDIRIZZAMENTO SENZA CLASSI:



Esempi:
12.24.76.8/8
23.14.67.92/12
220.8.24.255/25

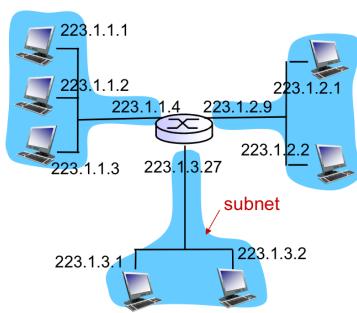
- Numero massimo di host?
 - Una rete con un prefisso di rete di n bit può avere al massimo **$2^{32-n}-2$** host (devo sottrarre indirizzo di rete e indirizzo di broadcast)

Esempio

Rete data dall'interconnessione di tre sottoreti

24 bit per l'indirizzo di rete:
/24

Indicare l'indirizzo di ciascuna sottorete



Subnetting

Indirizzo

a.b.c.d/n

n bit più a sinistra costituiscono la parte di indirizzo di rete

Subnet mask (maschera di sottorete)

Numero composto da 32 bit in cui i primi n bit a sinistra sono impostati a 1 e i rimanenti a 0

Serve a distinguere quale parte di un indirizzo ip identifica la rete e quale l'host

Subnet mask e indirizzo di rete: esempio

- Ipotesi: l'indirizzo IP [150.217.8.42](#) ha netmask [255.255.255.0](#)
- Qual è l'indirizzo di rete?

Si effettua un'operazione di AND

Indirizzo IP	150.217.8.42	10010110 11011001 00001000 00101010
Subnet Mask	255.255.255.0	11111111 11111111 11111111 00000000
AND	150.217.8.0	10010110 11011001 00001000 00000000

La rete ha indirizzo
[150.217.8.0/24](#)