Application Layer

DNS, TELNET

DNS: Domain Name System

"… The Domain Name System is a hierarchical distributed database. It stores information for mapping Internet host names to IP addresses and vice versa, mail routing information, and other data used by Internet applications.

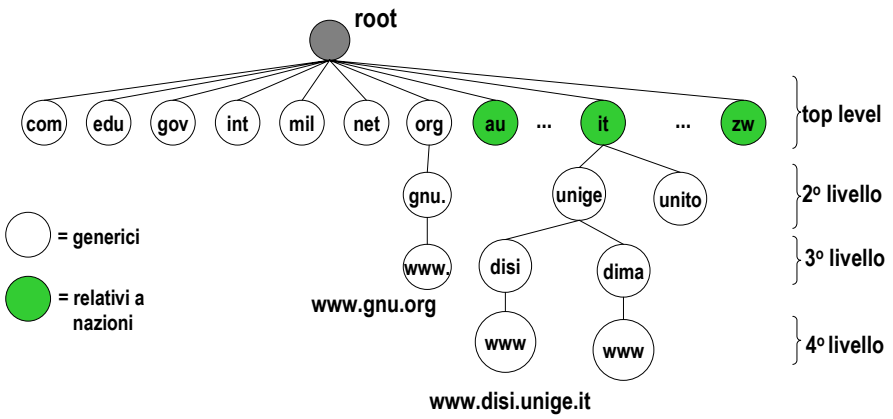
Clients look up information in the DNS by calling a resolver library, which sends queries to one or more name servers and intrerprets the responses …"

DNS: Domain Name System

- Gli host su Internet e i router possono essere identificati mediante
  - ✓ un indirizzo IP (32 bit) (130.251.152.126)
  - ✓ un nome logico (forum.educ.disi.unige.it)

Il DNS si occupa del mapping tra nomi logici e indirizzi IP

DNS: gerarchia dei domini



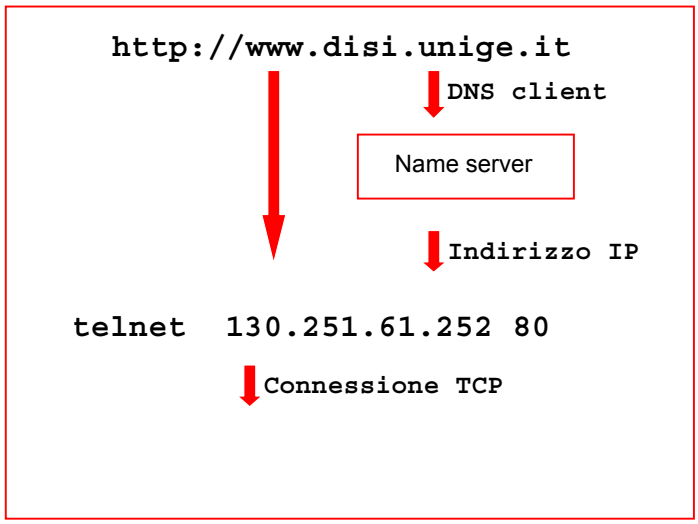
DNS: Domain Name System

- È un protocollo del livello Application usato dagli host per "risolvere" i nomi
- È caratterizzato da domande (query) e risposte (reply)
- Il DNS viene usato da altre applicazioni (basate su HTTP, SMTP, …) per tradurre i nomi logici forniti dall'utente in indirizzi IP

DNS: Domain Name System

- Sulla macchina client gira la parte client del DNS, cioè quella che inizia il meccanismo di interrogazione (query) per ottenere un indirizzo IP
- Il DNS usa i servizi che UDP fornisce a livello di trasporto
- Porta 53

DNS: Domain Name System



Perchè non centralizzare il DNS?

- Problema di tolleranza ai guasti
- Impensabile con l'attuale volume di traffico
- Non può essere "vicino" a tutti gli host che hanno bisogno di questo servizio
- Difficile da mantenere aggiornato

doesn't scale!

DNS name server

- Non esiste quindi un unico server che "conosce" il mapping di tutti i nomi logici negli indirizzi IP corrispondenti
- Si distingue tra

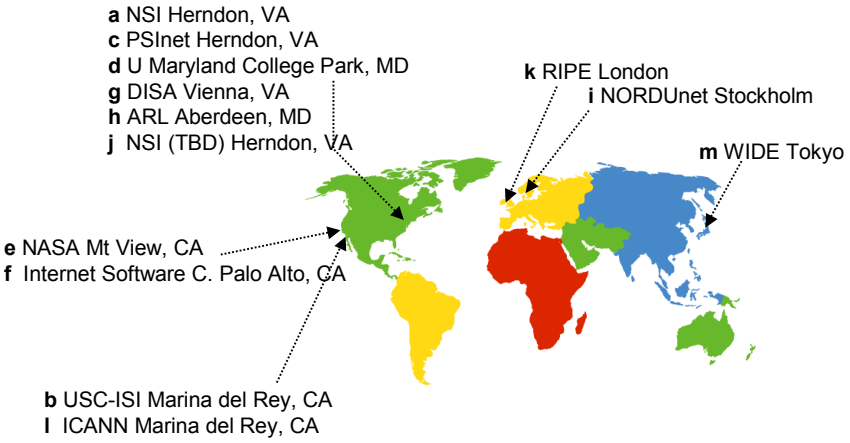
  1) local name server
  - ✓Ogni ISP, ogni azienda, ogni università ha il suo local name server
  - ✓Le richieste al DNS passano prima attraverso il name server locale

DNS name server

2) authoritative name server
- ✓Per un host memorizza la coppia <IP, nome logico>
- ✓Amministrano "gruppi" di host

3) root name server
- ✓Contengono gli IP degli authoritative name server (o sanno a chi chiedere) per ogni dominio registrato ufficialmente

DNS: root name server

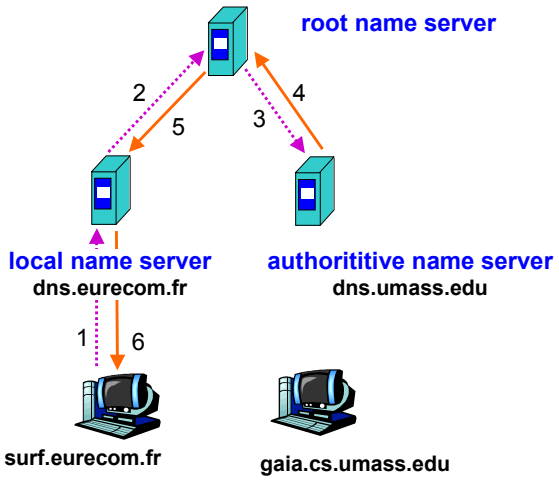


13 root name server

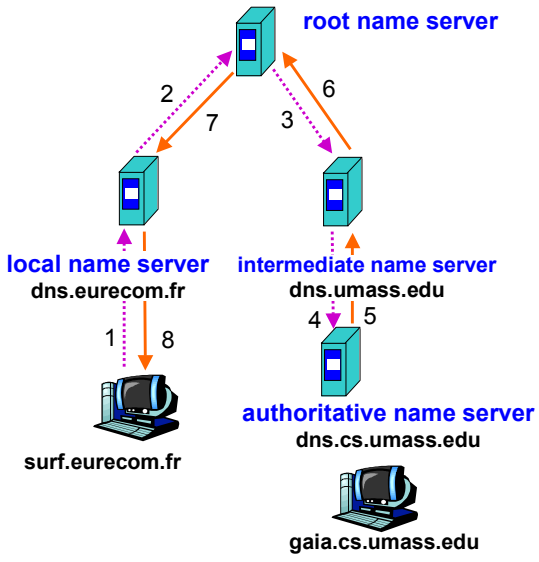
DNS: root name server

- Quando un root name server riceve una richiesta da un name server locale
  - ✓ Se conosce il mapping tra nome logico e indirizzo IP lo restituisce al name server locale
  - ✓ Altrimenti
    - ✓ interroga l'authoritative name server per quel name server locale
    - ✓ ottiene il mapping
    - ✓ lo restituisce al name server locale

Esempio (1)

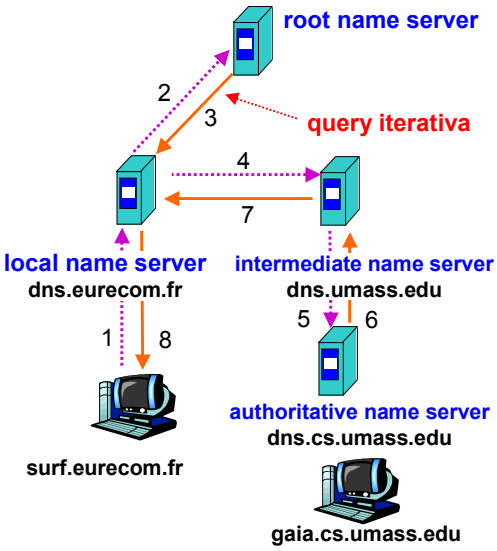


Esempio (2)



Esempio (3): query iterative

Il server contattato non restituisce un indirizzo IP ma il nome di un altro server: "non conosco questo nome, prova a chiedere a questo server"



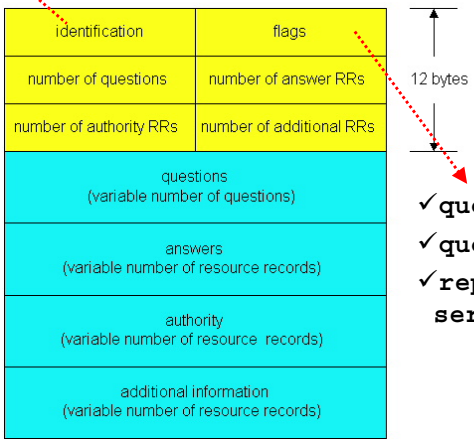
query reply

In realtà … DNS: caching

- I name server conservano nella memoria cache i mapping che ricevono (che valgono fino allo scadere di un timeout)
- Grazie al meccanismo del caching i name server locali possono evitare di interrogare i root name server ogni volta che necessitano di un indirizzo IP
- La richiesta viene invece passata al name server di livello superiore …

DNS: formato dei messaggi

numero a **16 bit** per **identificare la richiesta**
(la risposta usa lo stesso numero)



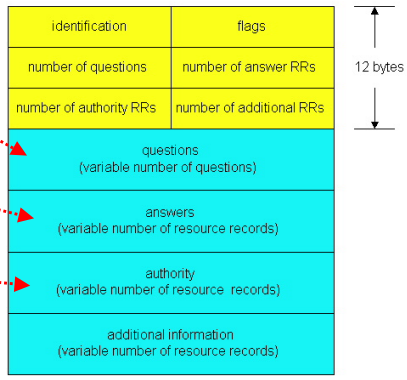
- ✓**query o reply**
- ✓**query ricorsiva**
- ✓**reply da un name server autoritativo**

DNS: formato dei messaggi

Name, Type per una query

record in risposta ad un query

record per i server autoritativi



DNS: resource record

RR: (Name, Value, Type, TTL)

TTL (Time to Live)

Determina la durata della validità di una risorsa in cache
(un mapping nome logico – IP)

DNS: resource record

RR: (Name, Value, Type, TTL)

Type: **A**
Name: **host**
Value: **indirizzo IP**

DNS: resource record

RR: (Name, Value, Type, TTL)

Type: NS

Name: dominio (es. unige.it)

Value: nome del name server autoritativo per il dominio

(serve per il routing delle query)

DNS: resource record

RR: (Name, Value, Type, TTL)

Type: CNAME

Name: è l'alias del nome canonico

es. webapp.educ.disi.unige.it è l'alias di puin.educ.disi.unige.it

Value: è il nome canonico

DNS: resource record

RR: (Name, Value, Type, TTL)

Type: MX

Name: alias del mail server

Value: nome canonico del mail server

Comando dig

```
$$ dig pianeta.di.unito.it

; <<>> DiG 8.3 <<>> pianeta.di.unito.it
;; res options: init recurs defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 2
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3,
   ADDITIONAL: 3

;; QUERY SECTION:
;; pianeta.di.unito.it, type = A, class = IN

;; ANSWER SECTION:
pianeta.di.unito.it.    8h33m39s IN A   130.192.239.1
```

Comando dig

```
;; AUTHORITY SECTION:
di.unito.it.            21m57s IN NS    amleto.di.unito.it.
di.unito.it.            21m57s IN NS    pianeta.di.unito.it.
di.unito.it.            21m57s IN NS    albert.unito.it.

;; ADDITIONAL SECTION:
amleto.di.unito.it.     21m57s IN A     130.192.239.30
pianeta.di.unito.it.    8h33m39s IN A   130.192.239.1
albert.unito.it.        23h49m31s IN A  130.192.119.1

;; Total query time: 10 msec
;; FROM: elios to SERVER: default -- 130.251.61.19
;; WHEN: Mon Oct  4 12:32:51 2004
;; MSG SIZE  sent: 37  rcvd: 157
```


TELNET [RFC 854]

- **Anche il TELNET è un protocollo del livello Application**
- **Permette la connessione remota (remote login)**

  " … The purpose of the TELNET protocol is to provide a fairly general, bi-directional, eight-bit byte oriented communications facility. Its primary goal is to allow a standard method of interfacing terminal devices and terminal-oriented processes to each other … "

TELNET [RFC 854]

telnet <macchina remota>

- **L'utente sulla macchina client digita i caratteri da tastiera (Network Virtual Terminal)**
- **I caratteri vengono inviati alla macchina server (<macchina remota>) che si occupa di rispedire indietro ogni carattere che viene visualizzato sullo schermo del client**
- **I comandi vengono eseguiti sulla macchina remota!**

Transport Layer

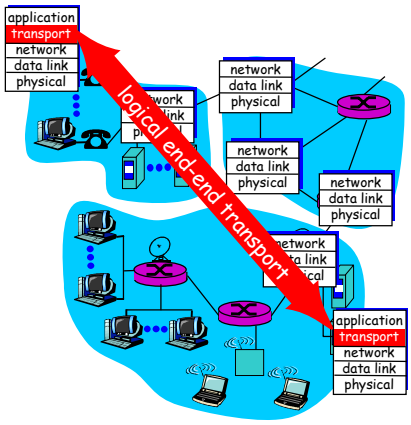
TCP

Transport layer

" … Residing between the application and the network layers, the transport layer is a central piece of the layered network architecture. It has the critical role of providing communication services directly to the application processes running on different hosts … "

Transport layer

- Fornisce una comunicazione logica tra applicazioni che girano su host diversi
- I protocolli del livello Transport girano sugli end system



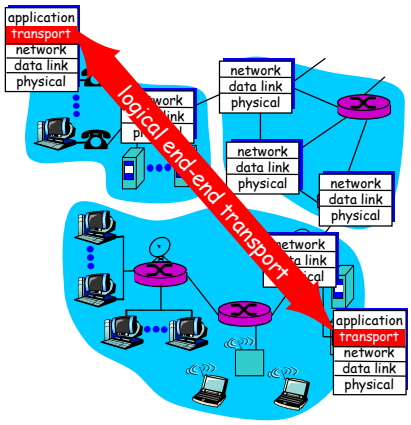
Transport layer

Lato sender

i messaggi prodotti al livello Application vengono suddivisi in segmenti

Lato receiver

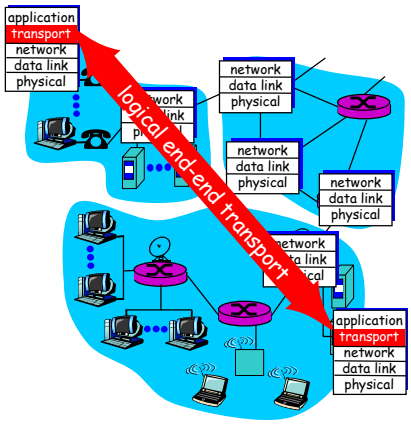
i segmenti vengono "riassemblati" per ricostruire il messaggio da passare al livello superiore



Internet: TCP e UDP

Transport layer: protocolli

- **TCP**
  - ✓ Orientato alla connessione
  - ✓ Affidabile
  - ✓ Controllo di flusso
  - ✓ Controllo della congestione
- UDP
  - ✓ Non affidabile
  - ✓ Non orientato alla connessione



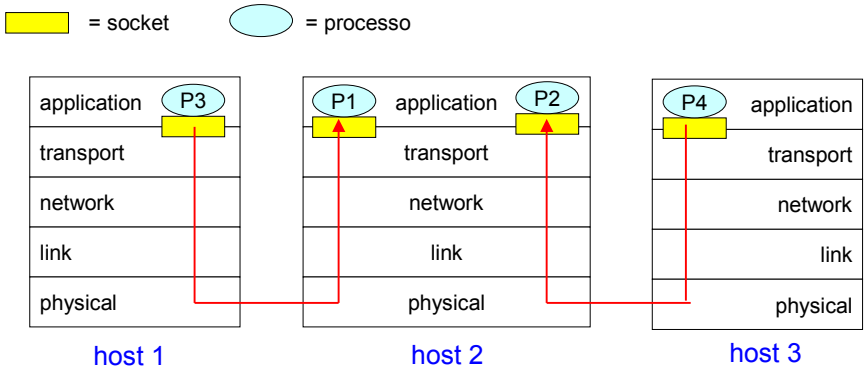
Multiplexing/demultiplexing

Multiplexing (sender)

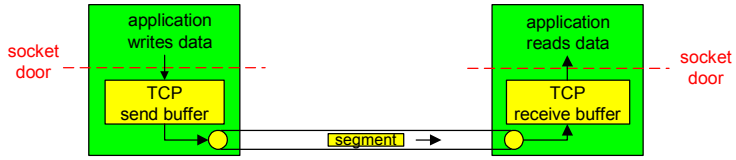
I dati in arrivo dai socket devono essere presi e trasformati in segmenti (con header opportuni)

Demultiplexing (receiver)

I segmenti devono essere inviati al socket corretto

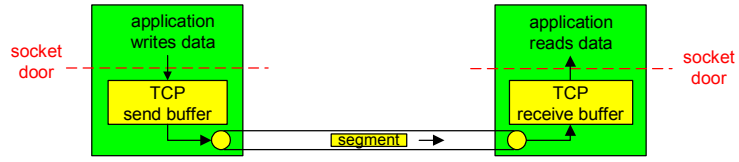


TCP: Transmission Control Protocol



- RFC: 793, 1122, 1323, 2018, 2581
- point-to-point
- reliable
- in-order byte stream

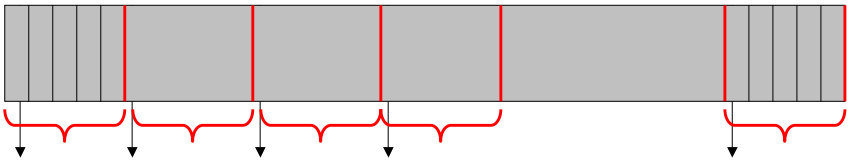
TCP: Transmission Control Protocol



- **full duplex**
  - ✓ **i dati fluiscono in entrambe le direzioni**
- **connection-oriented**
  - ✓ **meccanismo di handshaking**
- **flow control**
  - ✓ **il sender non sovraccarica di dati il receiver**

TCP: Struttura di un segmento

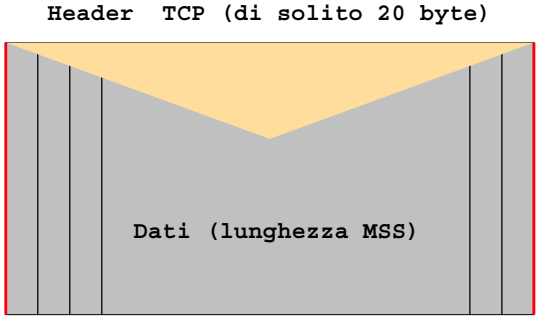
Dati in arrivo dal livello Application: stream di byte ordinati che vengono suddivisi in parti di dimensione MSS (Maximum Segment Size, 1500 byte, 512 byte, … **)**



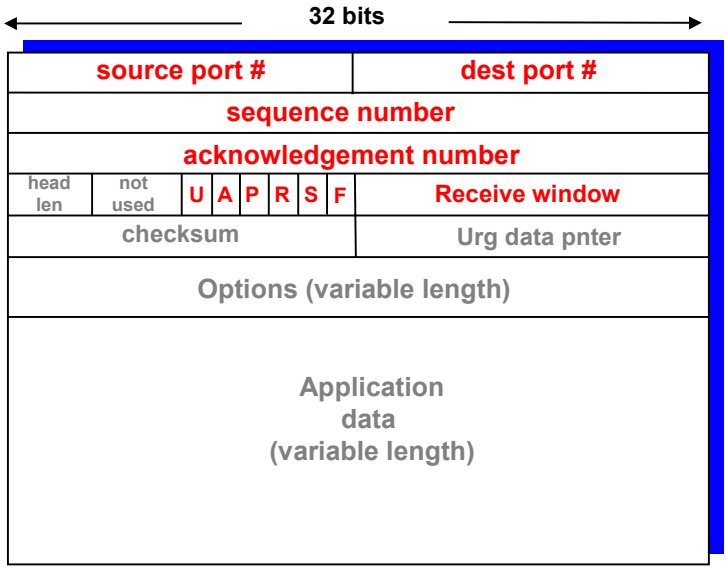
La posizione del primo byte in ogni parte viene usata come sequence number

In realtà viene generato un **numero casuale** per il **primo sequence number** che viene sommato alle varie posizioni iniziali di ogni parte. In questo modo **si evita di riconsiderare segmenti che sono scaduti** (ma che potrebbero avere lo stesso sequence number).

TCP: Struttura di un segmento



TCP: Struttura di un segmento



TCP: flag

- **U = URG** il livello superiore ha marcato dei dati come urgenti
- **A = ACK**
- **P = PSH** il receiver dovrebbe passare i dati immediatamente al livello superiore
- **R = RST**
- **S = SYN**
- **F = FIN**

TCP: Gestione della connessione

- Il sender e il receiver stabiliscono una connessione prima di iniziare lo scambio dei dati
- Vengono inoltre inizializzate numerose variabili e vengono creati i buffer per memorizzari i dati
- Il client inizia la connessione
- Il server accetta la connessione

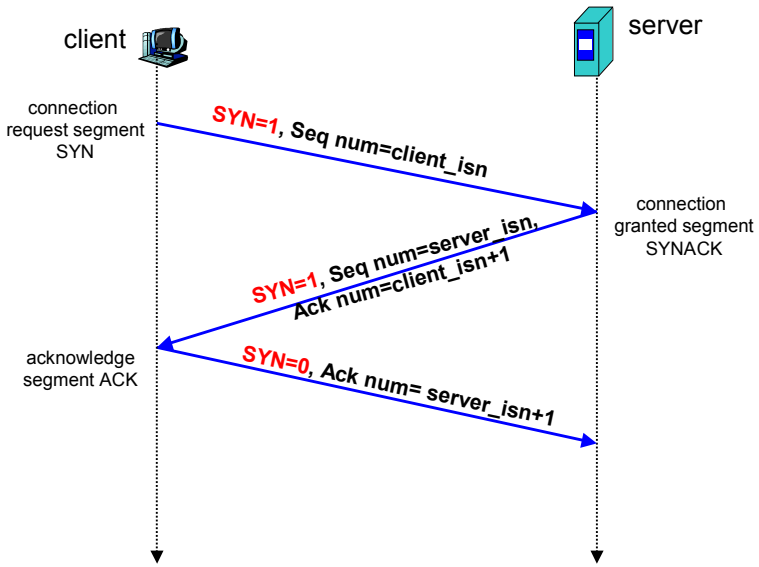
TCP: Three-way handshake

Passo 1: il client spedisce un segmento TCP di tipo **SYN** al server. Questo segmento non contiene dati ma un numero di inizio sequenza (**client_isn**) per il client

Passo 2: il server, ricevuto SYN, risponde con un segmento **SYNACK**. Inoltre, alloca i buffer e specifica un numero di inizio sequenza (**server_isn**) per il server

Passo 3: il client riceve SYNACK, risponde con un segmento **ACK**, che può contenere dei dati

TCP: Three-way handshake



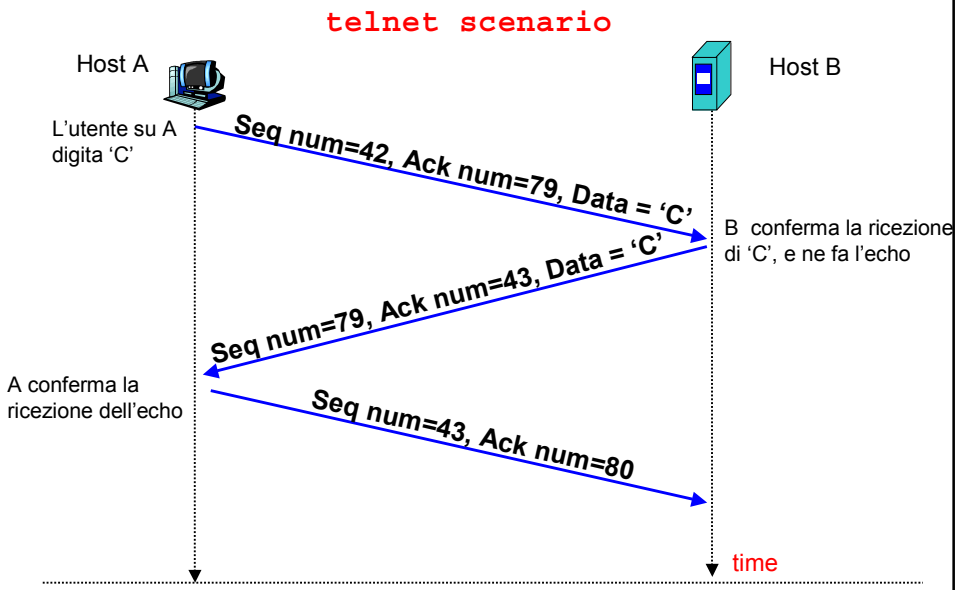
TCP: sequence e acknowledge numbers

- Ogni volta che un sender A invia un segmento ad un receiver B vi associa il suo sequence number
- Il receiver B, ricevuto il segmento, copia il sequence number incrementato di 1 nel campo acknowledge number
- Questo dirà al sender A che il receiver B è in attesa del "prossimo segmento"

TCP: sequence e acknowledge numbers

- Le cose sono un po' più complicate perchè TCP è full-duplex
- Inoltre, se arrivano segmenti fuori sequenza, ogni host continua a mandare l'acknowledge number del segmento che sta aspettando (cumulative acknowledge)
- Negli RFC di TCP non viene specificato cosa si deve fare dei segmenti fuori sequenza

TCP: sequence e acknowledge numbers



TCP: chiusura della connessione

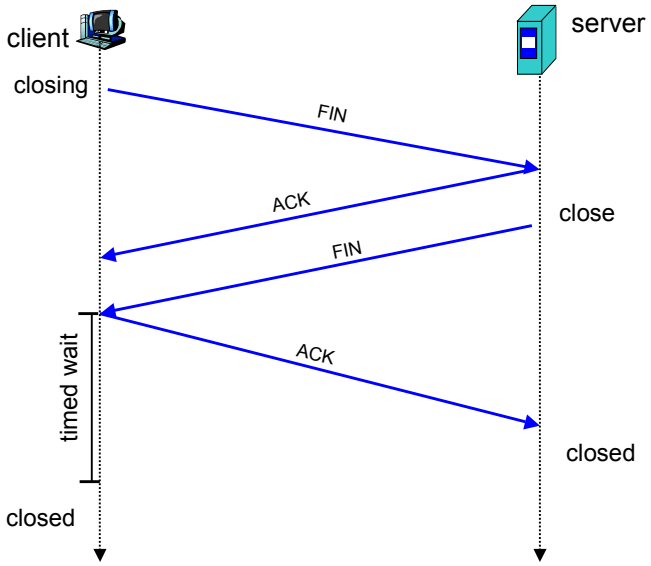
Passo 1: il client spedisce un segmento TCP di controllo detto **FIN**

Passo 2: il server, ricevuto FIN, risponde con **ACK**. Poi si prepara a chiudere la connessione ed invia a sua volta un segmento **FIN**

Passo 3: il client riceve FIN, e risponde con **ACK**. Entra quindi in uno stato di "timed wait": risponderà con ACK a tutti i FIN che riceve

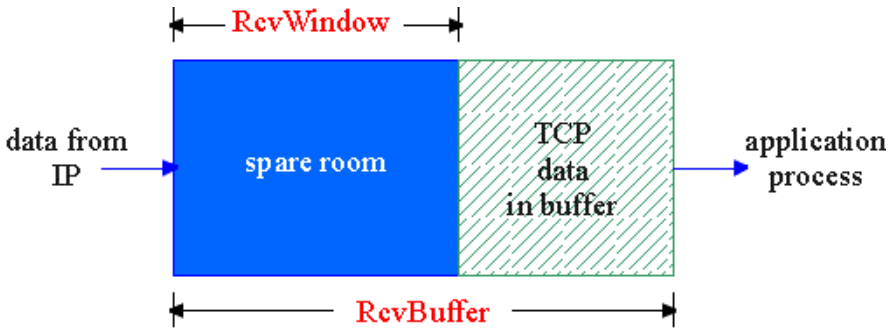
Passo 4: il server, ricevuto ACK, **chiude la connessione**

TCP: chiusura della connessione

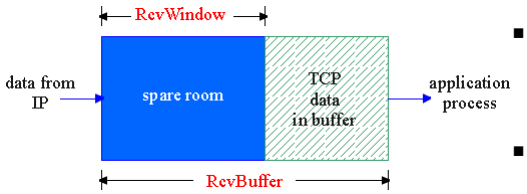


TCP: Flow Control

Il mittente non "inonda" il destinatario trasmettendo troppe informazioni
(overflow del buffer = perdita di dati)



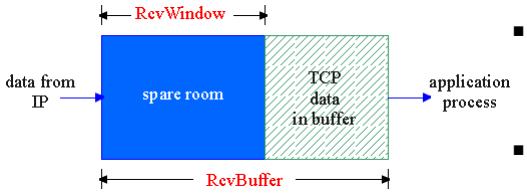
TCP: Flow Control



- Servizio di tipo "speed-matching"
- Il destinatario informa il mittente dello spazio libero nel buffer usando il campo Receive Window dell'header

RcvWindow = RcvBuffer
- [LastByteRcvd - LastByteRead]

TCP: Flow Control



- Servizio di tipo "speed-matching"
- Il mittente usa due variabili per sapere quanti sono i dati non confermati dal destinatario

LastByteSent - LastByteAcked
<= RcvWindow