



Reti

(già “Reti di Calcolatori”)

Protocolli di Livello Applicativo

Renato Lo Cigno

<http://disi.unitn.it/locigno/teaching-duties/reti>

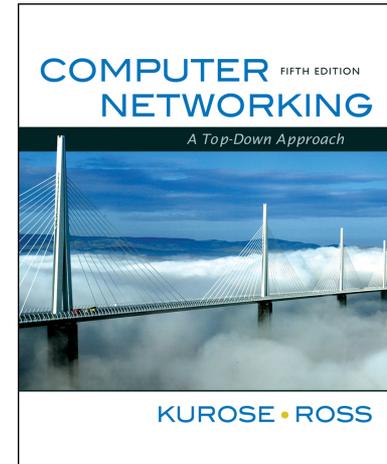
A note on the use of these slides:

These slides are an adaptation from the freely available version provided by the book authors to all (faculty, students, readers). The originals are in PowerPoint and English.

The Italian translation is originally from
Gianluca Torta, Stefano Leonardi, Francesco Di Tria

Adaptation is by Csaba Kiraly, Renato Lo Cigno, and Fabrizio Granelli

All material copyright 1996-2016
J.F Kurose and K.W. Ross, All Rights Reserved



*Computer Networking:
A Top Down Approach,
6-7th edition.
Jim Kurose, Keith Ross
Addison-Wesley*



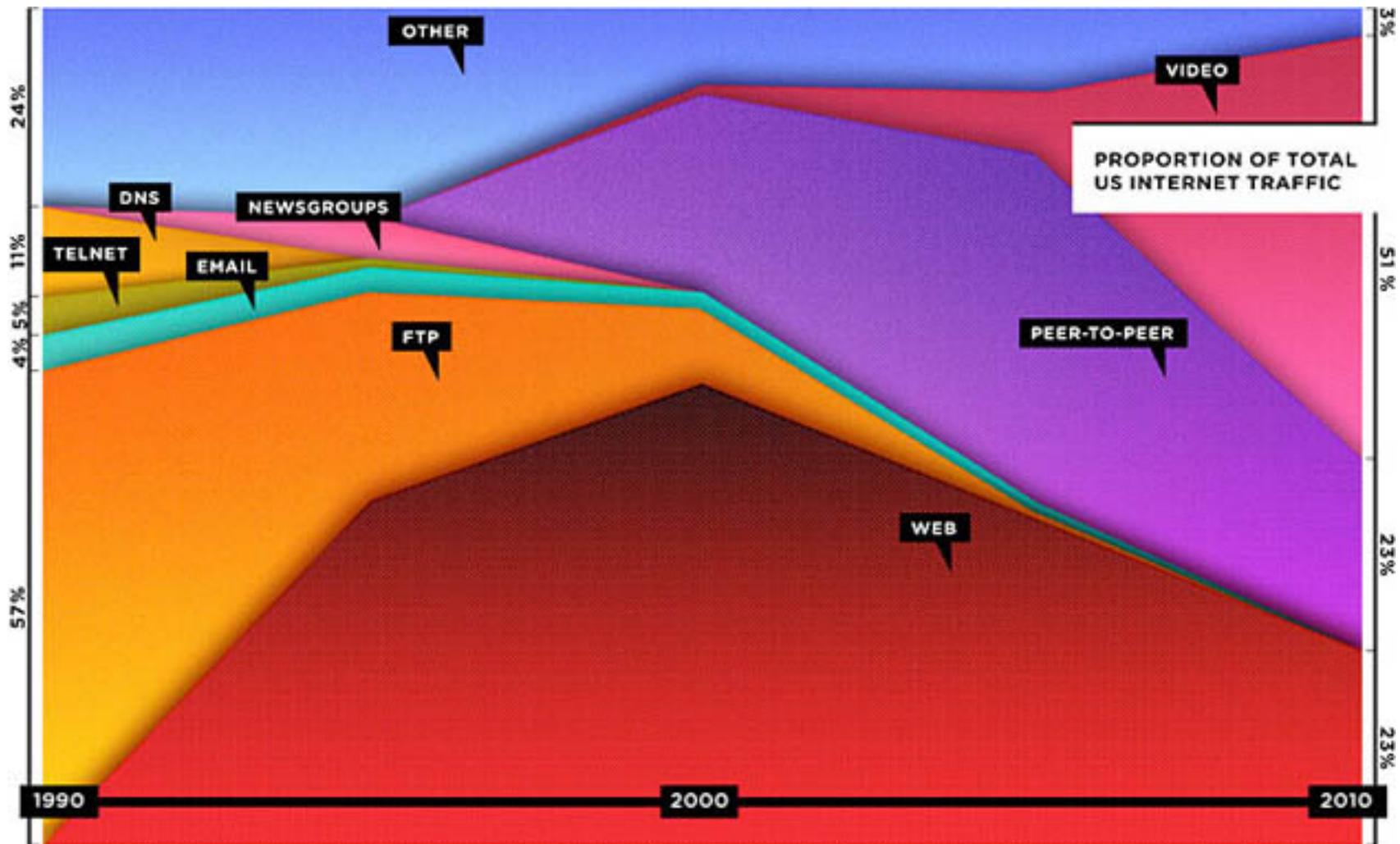
- ❑ 2.1 Principi delle applicazioni di rete
- ❑ 2.2 Web e HTTP
- ❑ 2.3 FTP
- ❑ 2.4 Posta Elettronica
 - ❑ SMTP, POP3, IMAP
- ❑ 2.5 DNS
- ❑ Protocolli per applicazioni multimediali



- ❑ Posta elettronica
- ❑ Web
- ❑ Messaggistica istantanea
- ❑ Autenticazione in un calcolatore remoto
- ❑ Condivisione di file P2P
- ❑ Giochi multiutente via rete
- ❑ Telefonia via Internet
- ❑ Videoconferenza in tempo reale
- ❑ Grid computing
- ❑ Streaming di video-clip memorizzati
- ❑ Social Networks



Chi usa Internet?



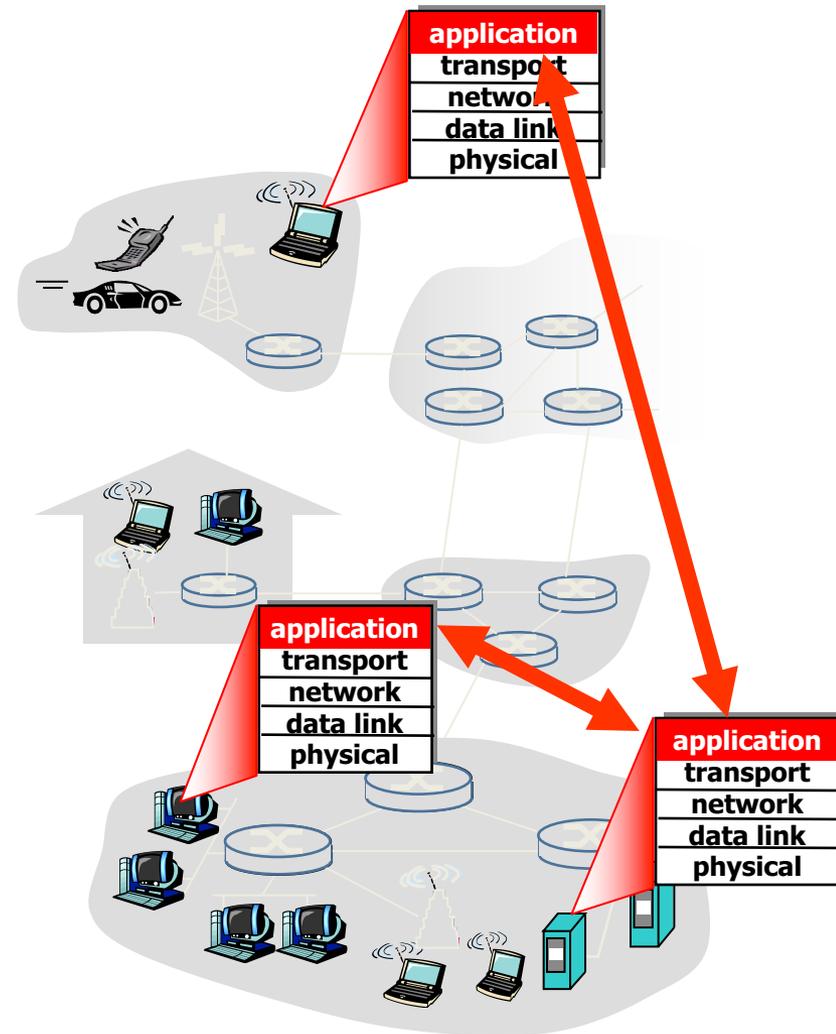
Fonte: Wired US, 2011

Scrivere programmi che

- girano su *end systems*
- comunicano sulla rete
- sfruttano un **protocollo a livello applicazione** (non l'applicazione stessa)

Non è necessario scrivere software per dispositivi interni alla rete

- I dispositivi di rete non eseguono applicazioni utente
- Rapido sviluppo di applicazioni





- ❑ **2.1 Principi delle applicazioni di rete**
- ❑ 2.2 Web e HTTP
- ❑ 2.3 FTP
- ❑ 2.4 Posta Elettronica
 - ❑ SMTP, POP3, IMAP
- ❑ 2.5 DNS
- ❑ Protocolli per applicazioni multimediali



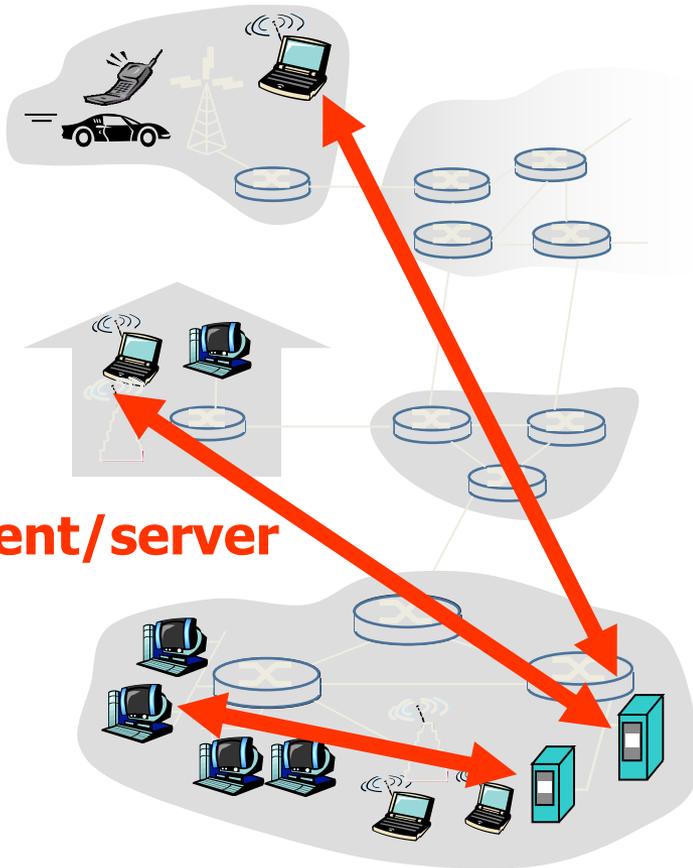
- ❑ Client-server
- ❑ Peer-to-peer (P2P)
- ❑ Architetture ibride (client-server e P2P)
- ❑ “Cloud”

server:

- host sempre attivo
- indirizzo IP fisso e noto al client
- server farm (=un hostname con più indirizzi IP) per creare un potente server virtuale

client:

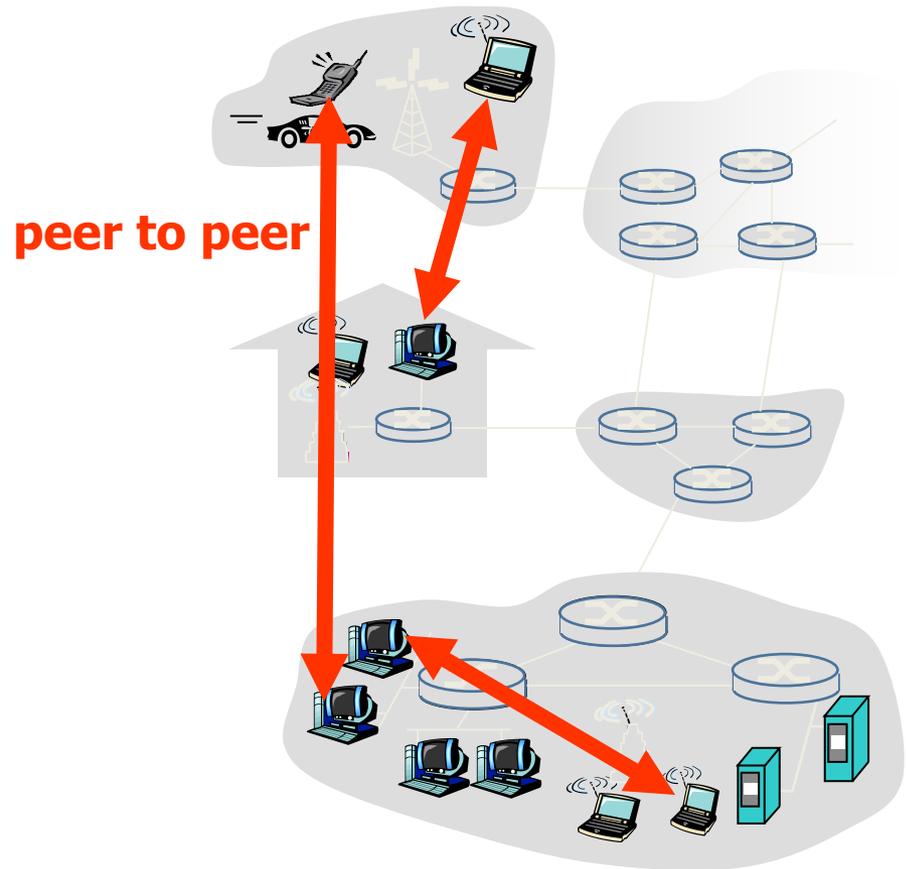
- comunica con il server
- può contattare il server in qualunque momento
- può avere indirizzi IP dinamici
- non comunica direttamente con gli altri client



- ❑ non c'è un server sempre attivo
- ❑ coppie arbitrarie di host (peer) comunicano direttamente tra loro
- ❑ i peer non devono necessariamente essere sempre attivi, e cambiano indirizzo IP

Facilmente scalabile

Difficile da gestire





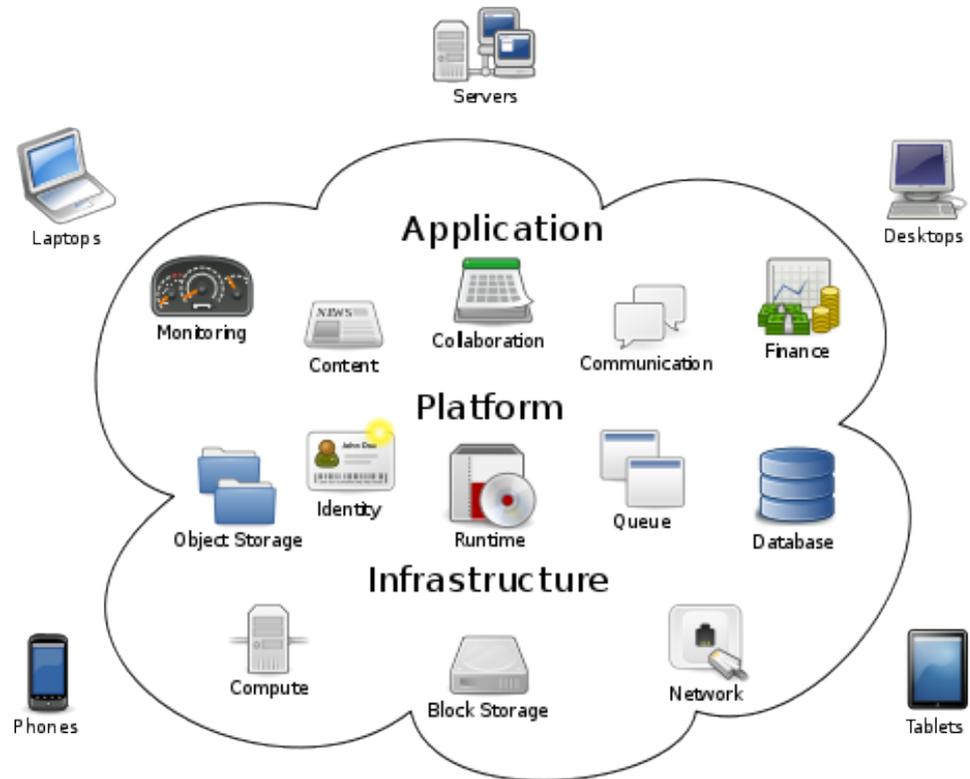
Skype

- Applicazione P2P di Voice over IP
- Server centralizzato: Autenticazione
- Ricerca utenti e indirizzi (Rubrica telefonica): P2P, con l'aiuto di SuperPeer che normalmente hanno indirizzi pubblici
- Connessione client-client: diretta o attraverso SuperPeer (non attraverso il server)

Messaggistica istantanea

- La chat tra due utenti è del tipo P2P
- Individuazione della presenza/location centralizzata:
 - l'utente registra il suo indirizzo IP sul server centrale quando è disponibile online
 - l'utente contatta il server centrale per conoscere gli indirizzi IP dei suoi amici

- ❑ Un insieme di tecnologie che permettono **sia di memorizzare/archiviare dati che di elaborarli** (con CPU o software) tramite l'utilizzo di risorse distribuite e virtualizzate in rete
- ❑ La creazione di una copia di sicurezza (backup) è automatica e **l'operatività si trasferisce tutta online**
- ❑ I dati sono memorizzati in server farm generalmente localizzate nei Paesi di origine del service provider



Processo: programma in esecuzione su di un host.

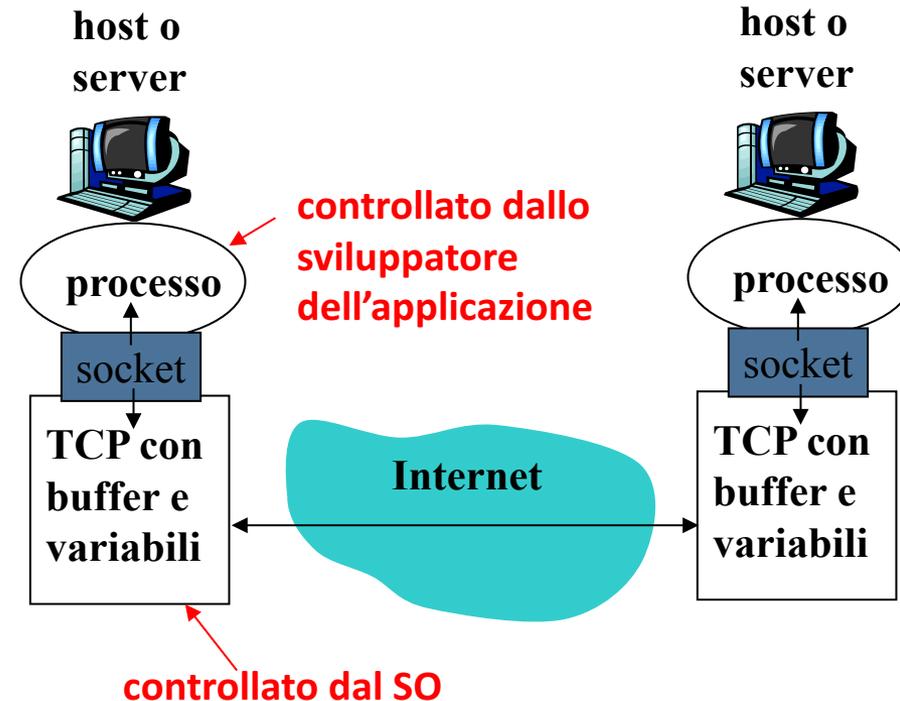
- ❑ All'interno dello stesso host, due processi comunicano utilizzando **schemi interprocesso** (definiti dal SO)
- ❑ processi su host differenti comunicano attraverso lo scambio di **messaggi**

Processo client: processo che dà inizio alla comunicazione

Processo server : processo che attende di essere contattato

le applicazioni con architetture P2P hanno processi client e processi server, ma i dettagli del server possono non essere noti a priori

- ❑ un processo invia/riceve messaggi a/da la sua **socket**
- ❑ un socket è analogo a un punto di accesso/uscita (SAP)
 - un processo che vuole inviare un messaggio, lo fa uscire dalla propria “interfaccia” (socket)
 - il processo presuppone l’esistenza di un’infrastruttura esterna che trasporterà il messaggio attraverso la rete fino alla “interfaccia” del processo di destinazione
- Si usano API che consentono:
 - scelta del protocollo di trasporto
 - capacità di determinare alcuni parametri



Le chiamate ai socket sono le “primitive” del protocollo



- ❑ Affinché un processo su un host invii un messaggio a un processo su un altro host, il mittente deve identificare il processo destinatario
- ❑ Un host ha un indirizzo IP univoco a 32 bit
- ❑ **Domanda:** È sufficiente conoscere l'indirizzo IP dell'host su cui è in esecuzione il processo per identificare il processo stesso?
- ❑ **Risposta:** No, sullo stesso host possono essere in esecuzione molti processi
- ❑ L'identificatore comprende sia l'indirizzo IP che i **numeri di porta** associati al processo in esecuzione su un host
- ❑ Esempi di numeri di porta:
 - HTTP server: 80
 - Mail server: 25
- ❑ Per inviare un messaggio HTTP al server `gaia.cs.umass.edu`:
 - **Indirizzo IP:** 128.119.245.12
 - **Numero di porta:** 80



- ❑ Tipi di messaggi scambiati, ad esempio messaggi di richiesta e di risposta
- ❑ Sintassi dei tipi di messaggio: quali sono i campi nel messaggio e come sono descritti
- ❑ Semantica dei campi, ovvero significato delle informazioni nei campi
- ❑ Regole per determinare quando e come un processo invia e risponde ai messaggi

Protocolli di pubblico dominio:

- ❑ Definiti nelle RFC
- ❑ Consentono l'interoperabilità
- ❑ Ad esempio, HTTP, SMTP

Protocolli proprietari:

- ❑ Ad esempio, Skype



Che servizio di trasporto richiede un'applicazione?

Perdita di dati

- alcune applicazioni (ad esempio, audio) possono tollerare qualche perdita
- altre applicazioni (ad esempio, trasferimento di file, telnet) richiedono un trasferimento dati affidabile al 100%

Temporizzazione

- alcune applicazioni (ad esempio, telefonia Internet, giochi interattivi) per essere "realistiche" richiedono piccoli ritardi e sincronia

Throughput

- alcune applicazioni (ad esempio, quelle multimediali) per essere "efficaci" richiedono un'ampiezza di banda minima
- altre applicazioni ("le applicazioni elastiche") utilizzano l'ampiezza di banda che si rende disponibile

Sicurezza

- Cifratura, integrità dei dati, ...



Requisiti del servizio di trasporto di alcune applicazioni comuni

| Applicazione | Tolleranza alla perdita di dati | Throughput | Sensibilità al tempo e tolleranza ai ritardi |
|-----------------------------------|--|---|---|
| Trasferimento file | No | Variabile | No |
| Posta elettronica | No | Variabile | No |
| Documenti Web | No | Variabile | No |
| Audio/video in tempo reale | Sì | Audio: da 5 kbps a 1 Mbps Video: da 10 kbps a 5 Mbps | Sì, centinaia di ms |
| Audio/video memorizzati | Sì | Come sopra | Sì, pochi secondi |
| Giochi interattivi | No | Fino a pochi kbps | Sì, centinaia di ms |
| Messaggistica istantanea | No | Variabile | Sì e no |

Servizio di TCP:

- ❑ *orientato alla connessione:* è richiesto un setup fra i processi client e server (handshaking)
- ❑ *trasporto affidabile* fra i processi d'invio e di ricezione
- ❑ *controllo di flusso:* il mittente non vuole sovraccaricare il destinatario
- ❑ *controllo della congestione:* "strozza" il processo d'invio quando la rete è sovraccaricata
- ❑ *non offre:* temporizzazione, garanzie su un'ampiezza di banda minima, sicurezza

Servizio di UDP:

- ❑ trasferimento dati inaffidabile fra i processi d'invio e di ricezione
- ❑ *non offre:* setup della connessione, affidabilità, controllo di flusso, controllo della congestione, temporizzazione né ampiezza di banda minima e sicurezza
- ❑ applicazioni in tempo reale (tollerano perdita di dati ma non ritardo o variazioni di throughput); applicazioni di transazione semplici



| Applicazione | Protocollo di applicazione | Protocollo di trasporto |
|-----------------------------------|---|------------------------------------|
| Posta elettronica | SMTP [RFC 2821] | TCP |
| Accesso a terminali remoti | Telnet [RFC 854] | TCP |
| Web | HTTP [RFC 2616] | TCP |
| Trasferimento file | FTP [RFC 959] | TCP |
| Multimedia in streaming | HTTP (es. YouTube) RTP [RFC 1889] | TCP o UDP |
| Telefonia Internet | SIP, RTP, proprietario (es. Skype) | Tipicamente UDP |



- ❑ 2.1 Principi delle applicazioni di rete
- ❑ **2.2 Web e HTTP**
- ❑ 2.3 FTP
- ❑ 2.4 Posta Elettronica
 - ❑ SMTP, POP3, IMAP
- ❑ 2.5 DNS
- ❑ Protocolli per applicazioni multimediali

Terminologia HTML (da non confondere con HTTP!)

- ❑ Una **pagina web** è costituita da **oggetti**
- ❑ Un oggetto può essere un file HTML, un'immagine JPEG, un'applet Java, un file audio, ...
- ❑ Una pagina web è formata da un **file base HTML** che include diversi oggetti referenziati
- ❑ Ogni oggetto è referenziato da un **URL (Universal Resource Locator)**
- ❑ Esempio di URL:

`http://www.someschool.edu/someDept/pic.gif`

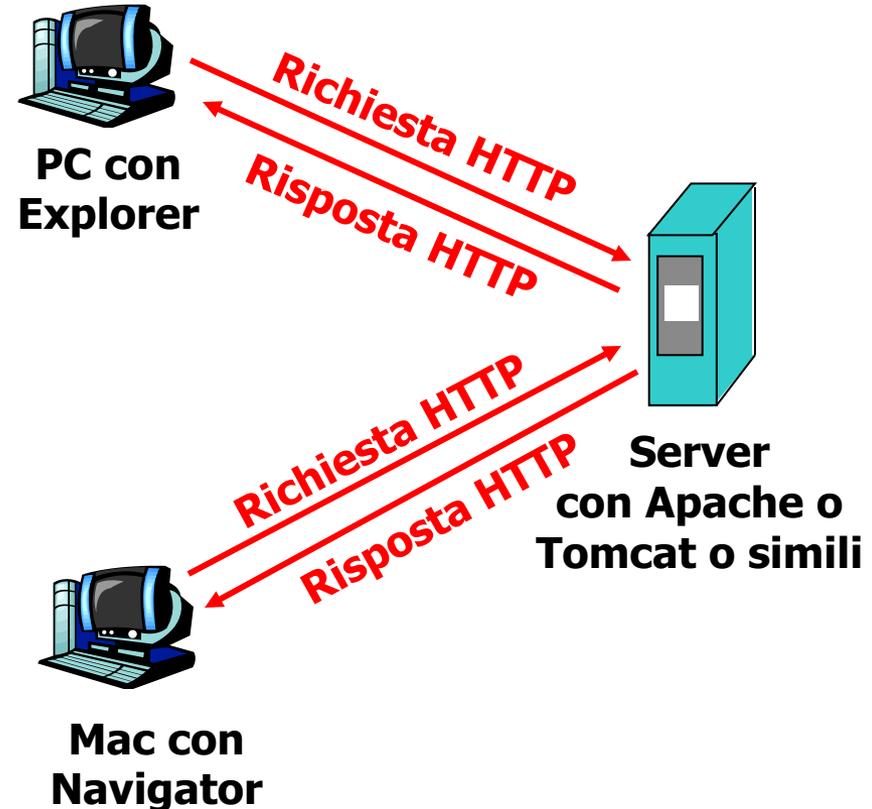
protocol

nome dell' host

nome del percorso

HTTP: hypertext transfer protocol

- Protocollo a livello di applicazione del Web
- Modello client/server
 - *client*: il browser che richiede, riceve, “visualizza” gli oggetti del Web
 - *server*: il server web invia oggetti in risposta a una richiesta





Usa TCP:

- ❑ Il client inizializza la connessione TCP (crea una socket) con il server, la porta 80
- ❑ Il server accetta la connessione TCP dal client
- ❑ Messaggi HTTP scambiati fra browser (client HTTP) e server web (server HTTP)
- ❑ Connessione TCP chiusa

HTTP è un protocollo “senza stato” (stateless)

- ❑ Il server non mantiene informazioni sulle richieste fatte dal client

nota

I protocolli che mantengono lo “stato” sono complessi!

- ❑ La storia passata (stato) deve essere memorizzata
- ❑ Se il server e/o il client si bloccano, le loro viste dello “stato” potrebbero essere contrastanti e dovrebbero essere riconciliate



Connessioni non persistenti

- ❑ Un singolo oggetto per volta viene trasmesso su una connessione TCP

Connessioni persistenti

- ❑ Più oggetti possono essere trasmessi su una singola connessione TCP tra client e server

**(contiene testo,
riferimenti a 10
immagini jpeg)**

Supponiamo che l'utente immetta l' URL

`www.someSchool.edu/someDepartment/home.index`

1a. Il client HTTP inizializza una connessione TCP con il server HTTP (processo) a `www.someSchool.edu` sulla porta 80

1b. Il server HTTP sull'host `www.someSchool.edu` in attesa di una connessione TCP alla porta 80 "accetta" la connessione e avvisa il client

2. Il client HTTP trasmette un *messaggio di richiesta* (con l'URL) nella socket della connessione TCP. Il messaggio indica che il client vuole l'oggetto `someDepartment/home.index`

3. Il server HTTP riceve il messaggio di richiesta, forma il *messaggio di risposta* che contiene l'oggetto richiesto e invia il messaggio nella sua socket

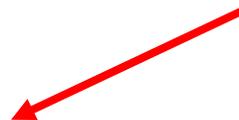
tempo

4. Il server HTTP chiude la
connessione TCP

5. Il client HTTP riceve il messaggio
di risposta che contiene il file
html e visualizza il documento
html. Esamina il file html, trova i
riferimenti a 10 oggetti jpeg

6. I passi 1-5 sono ripetuti per
ciascuno dei 10 oggetti jpeg

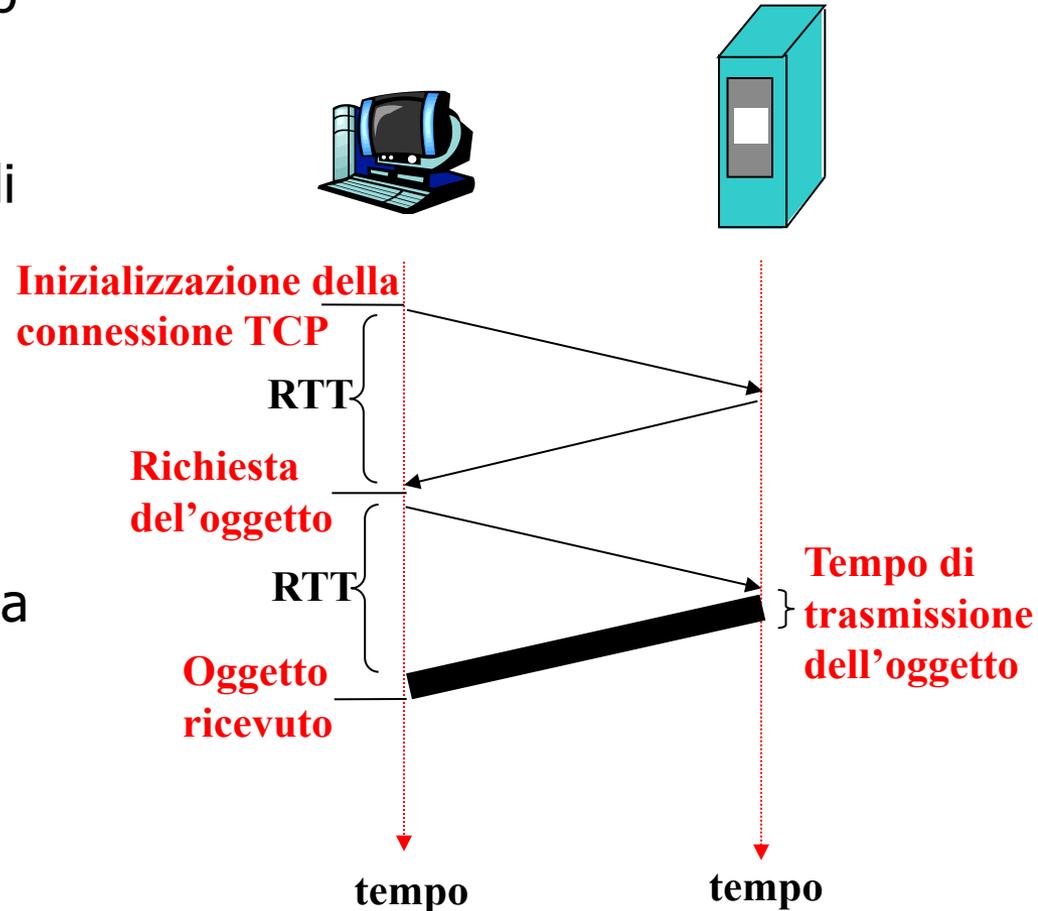
tempo



Definizione di RTT: tempo impiegato da un piccolo pacchetto per andare dal client al server e per una eventuale risposta (breve) di ritornare al client

Tempo di risposta:

- un RTT per inizializzare la connessione TCP
- un RTT perché ritornino la richiesta HTTP e i primi byte della risposta HTTP
- tempo di trasmissione del file



totale = 2RTT + tempo di trasmissione



Connessioni non persistenti:

- ❑ richiedono 2 RTT + tempo di trasmissione per oggetto
- ❑ overhead del sistema operativo per *ogni* connessione TCP
- ❑ i browser spesso aprono connessioni TCP parallele per caricare gli oggetti referenziati
- ❑ Si crea competizione tra le connessioni dello stesso host in caso di congestione

Connessioni persistenti

- ❑ il server lascia la connessione TCP aperta dopo l'invio di una risposta
- ❑ i successivi messaggi tra gli stessi client/server vengono trasmessi sulla connessione aperta
- ❑ il client invia le richieste non appena incontra un oggetto referenziato
- ❑ un solo RTT per ogni oggetto richiesto
- ❑ Con pipelining:
 - Il client invia le richieste in sequenza senza aspettare i precedenti oggetti
 - Un solo RTT di attesa per tutti gli oggetti, gli oggetti sono trasferiti in sequenza

- ❑ due tipi di messaggi HTTP: *richiesta*, *risposta*
- ❑ **Messaggio di richiesta HTTP:**
 - ASCII (formato leggibile dall'utente)

Riga di richiesta
(comandi GET,
POST, HEAD)

Righe di
intestazione

Un carriage return
e un line feed
indicano la fine
dell'intestazione
del messaggio

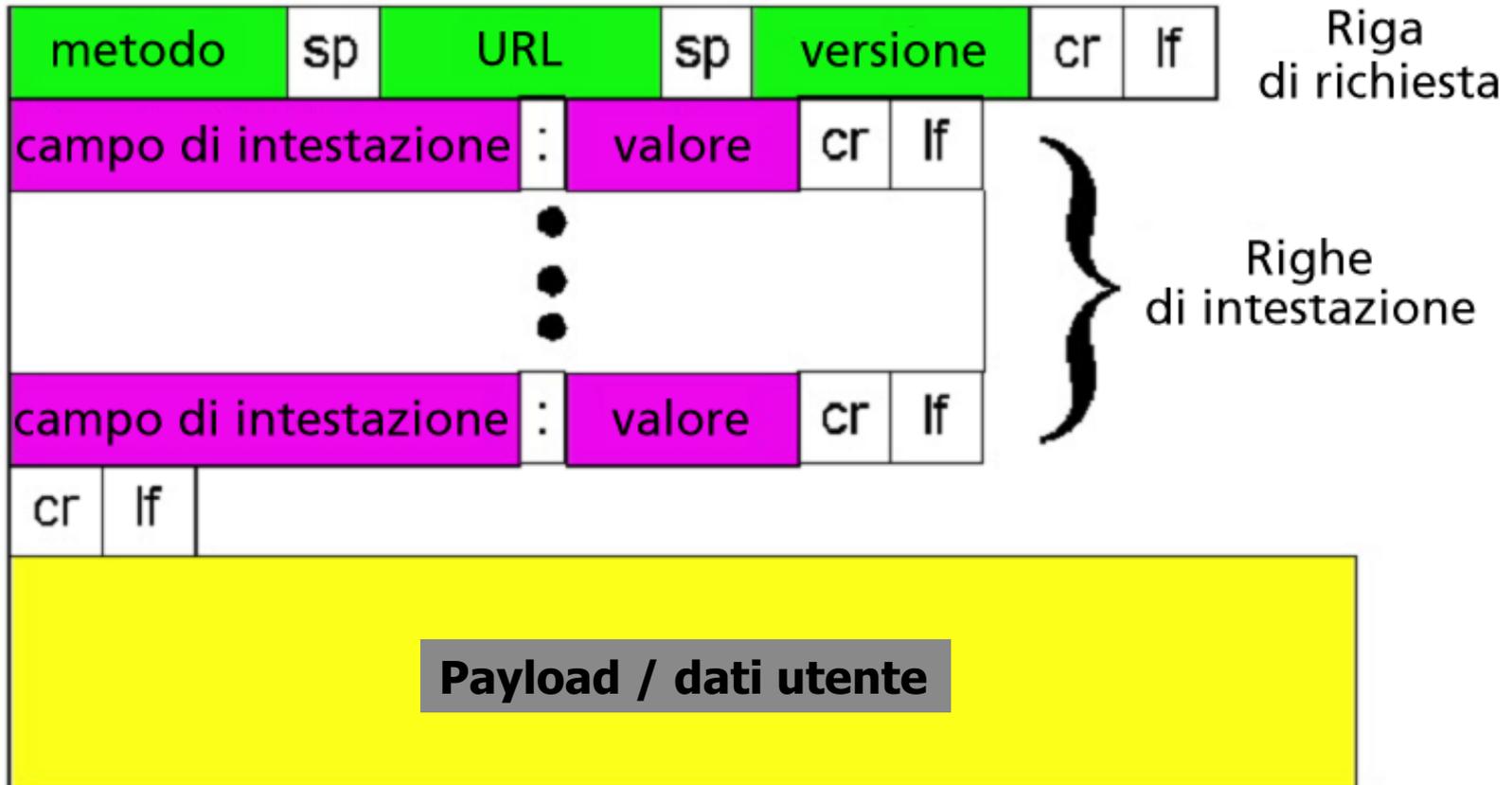
GET /somedir/page.html HTTP/1.1

```
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language: fr
```

(carriage return e line feed extra)



Messaggio di richiesta HTTP: formato generale





Metodo Post:

- ❑ Una pagina web a volte può includere spazi e campi per consentire "input" di dati da parte dell'utente
- ❑ I dati di input arrivano al server nel payload

Metodo GET:

- ❑ Non richiede in genere dati utente e arriva al server nel campo URL della riga di richiesta:

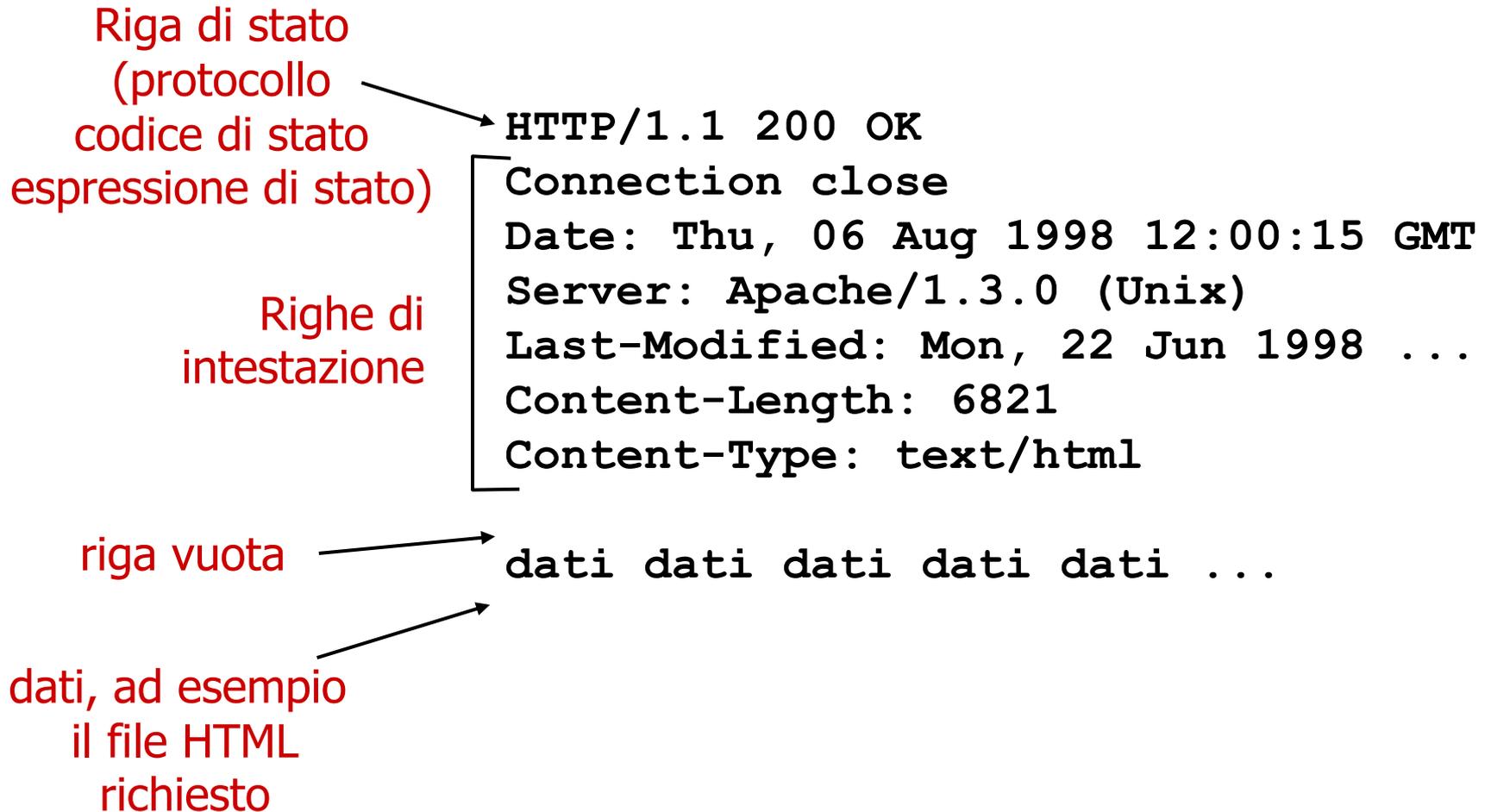
`www.somesite.com/search?a=2&b=5`

HTTP/1.0

- GET
- POST
- HEAD
 - chiede al server di escludere l'oggetto richiesto dalla risposta

HTTP/1.1

- GET, POST, HEAD
- PUT
 - include il file (o oggetto) specificato nel payload e lo invia al percorso specificato nel campo URL del messaggio
- DELETE
 - cancella il file specificato nel campo URL





Alcuni codici di stato e relative
espressioni:

Sono sempre il contenuto della prima
riga nel messaggio di risposta
server->client.

200 OK

- La richiesta ha avuto successo; l'oggetto richiesto viene inviato nella risposta

301 Moved Permanently

- L'oggetto richiesto è stato trasferito; la nuova posizione è specificata nell'intestazione `Location`: della risposta

400 Bad Request

- Il messaggio di richiesta non è stato compreso dal server

404 Not Found

- Il documento richiesto non si trova su questo server

505 HTTP Version Not Supported

- Il server non ha la versione di protocollo HTTP



1. Collegatevi via Telnet al vostro server web preferito:

```
telnet cis.poly.edu 80
```

Apri una connessione TCP alla porta 80 (porta di default per un server HTTP) dell'host cis.poly.edu. Tutto ciò che digitate viene trasmesso alla porta 80 di cis.poly.edu

2. Digitate una richiesta GET:

```
GET /~ross/ HTTP/1.1  
Host: cis.poly.edu
```

Digitando questo (premete due volte il tasto Invio), trasmettete una richiesta GET minima (ma completa) al server HTTP

3. Guardate il messaggio di risposta trasmesso dal server HTTP!

Bell' esempio ... ma non funziona perchè gli amministratori di rete non consentono queste operazioni per questioni di sicurezza (giustamente!!)

- Downloads Ctrl+Shift+Y
- Add-ons Ctrl+Shift+A
- Set Up Sync...
- Web Developer**
- Page Info Ctrl+I
- Start Private Browsing Ctrl+Shift+P
- Clear Recent History... Ctrl+Shift+Del
- Manage Content Plug-ins

- Web Console Ctrl+Shift+K
- Inspect Ctrl+Shift+I
- Scratchpad Shift+F4
- Page Source Ctrl+U
- Error Console Ctrl+Shift+J
- Get More Tools



Add-ons for Firefox > Collections > Mozilla > Web Developer's Toolbox

Web Developer's Toolbox

by Mozilla

About this Collection

Speed up the development process by using add-ons to troubleshoot, edit, and debug web projects without ever clicking away from Firefox.

Share this Collection

711 67
 12,243 followers
 Updated March 4, 2012

What are Collections?

Collections are groups of related add-ons that anyone can create and share.

[Explore Collections](#)

Common Tags

- css
- javascript
- privacy
- recommended
- web

13 Add-ons in this Collection

Sort by: Popularity

Firebug
 by Joe Hewitt, Jan Odvarko, others

+ Add to Firefox

Firebug integrates with Firefox to put a wealth of development tools at your fingertips while you browse. You can edit, debug, and monitor

★★★★★ 1,180 reviews
 3,149,949 users



The screenshot shows the Mozilla Firefox browser interface. The address bar contains the URL `http://gеп.disi.unitn.it/`. The **Tools** menu is open, displaying the following options:

- Downloads (Ctrl+Shift+Y)
- Add-ons (Ctrl+Shift+A)
- Set Up Sync...
- Web Developer >
- Page Info (Ctrl+I)
- Start Private Browsing (Ctrl+Shift+P)
- Clear Recent History... (Ctrl+Shift+Del)
- Manage Content Plug-ins

The **Web Developer** sub-menu is also open, showing:

- Firebug >
- Web Console (Ctrl+Shift+K)
- Inspect (Ctrl+Shift+I)
- Scratchpad (Shift+F4)
- Page Source (Ctrl+U)
- Error Console (Ctrl+Shift+J)
- Get More Tools

The **Firebug** sub-menu is open, showing:

- Open Firebug (F12)
- Firebug UI Location >
- Deactivate Firebug for This Site (Shift+F12)
- Open With Editor >
- Text Size >
- Options >
- Firebug Online >
- Inspect Element (Ctrl+Shift+C)
- Profile JavaScript
- Command Line (Ctrl+Shift+L)
- Search
- Customize Shortcuts
- About... 1.9.1

It works!

It works!

Console HTML CSS Script DOM Net

Clear Persist All HTML CSS JS XHR Images Flash Media

| URL | Status | Domain | Size | Remote IP | Timeline |
|-----------------------|--------|-------------------|------|-------------------|----------|
| GET gep.disi.unitn.it | 200 OK | gep.disi.unitn.it | 56 B | 193.205.213.86:80 | 33ms |

Headers Response Cache HTML

Response Headers [view source](#)

Accept-Ranges bytes
Connection Keep-Alive
Content-Encoding gzip
Content-Length 56
Content-Type text/html
Date Mon, 05 Mar 2012 11:39:03 GMT
Etag "1a92fe-2d-46f604f7d5500"
Keep-Alive timeout=15, max=100
Last-Modified Thu, 23 Jul 2009 14:29:08 GMT
Server Apache/2.2.16 (Ubuntu)
Vary Accept-Encoding

Request Headers [view source](#)

Accept text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Charset ISO-8859-1,utf-8;q=0.7,*;q=0.7
Accept-Encoding gzip, deflate
Accept-Language en-us,en;q=0.5
Cache-Control no-cache
Connection keep-alive
Host gep.disi.unitn.it
Pragma no-cache
User-Agent Mozilla/5.0 (Ubuntu; X11; Linux x86_64; rv:9.0.1) Gecko/20100101 Firefox/9.0.1

0 Request start time since the beginning

Request phases start and elapsed time relative to the request start:

| | | |
|-------|------|------------|
| 0 | 5ms | DNS Lookup |
| +5ms | 11ms | Connecting |
| +16ms | 0 | Sending |
| +16ms | 17ms | Waiting |
| +33ms | 0 | Receiving |

Event timing relative to the request start:

| | |
|-------|------------------|
| +60ms | DOMContentLoaded |
| +72ms | load |



Molti dei più importanti siti web usano i cookie

Quattro componenti:

- 1) Una riga di intestazione nel messaggio di *risposta* HTTP
- 2) Una riga di intestazione nel messaggio di *richiesta* HTTP
- 3) Un file cookie mantenuto sul sistema terminale dell'utente e gestito dal browser dell'utente
- 4) Un database sul sito

Esempio:

- Susan accede sempre a Internet dallo stesso PC
- Visita per la prima volta un particolare sito di commercio elettronico
- Quando la richiesta HTTP iniziale giunge al sito, il sito crea un identificativo unico (ID) e una *entry* nel database per ID

File cookie sul client

Server Amazon



A cosa possono servire i cookie:

- autorizzazione
- carrello elettronico
- suggerimenti
- stato della sessione dell'utente

Lo "stato"

- Mantengono lo stato del mittente e del ricevente per più transazioni
- Livello di sessione utente al di sopra di HTTP privo di stato

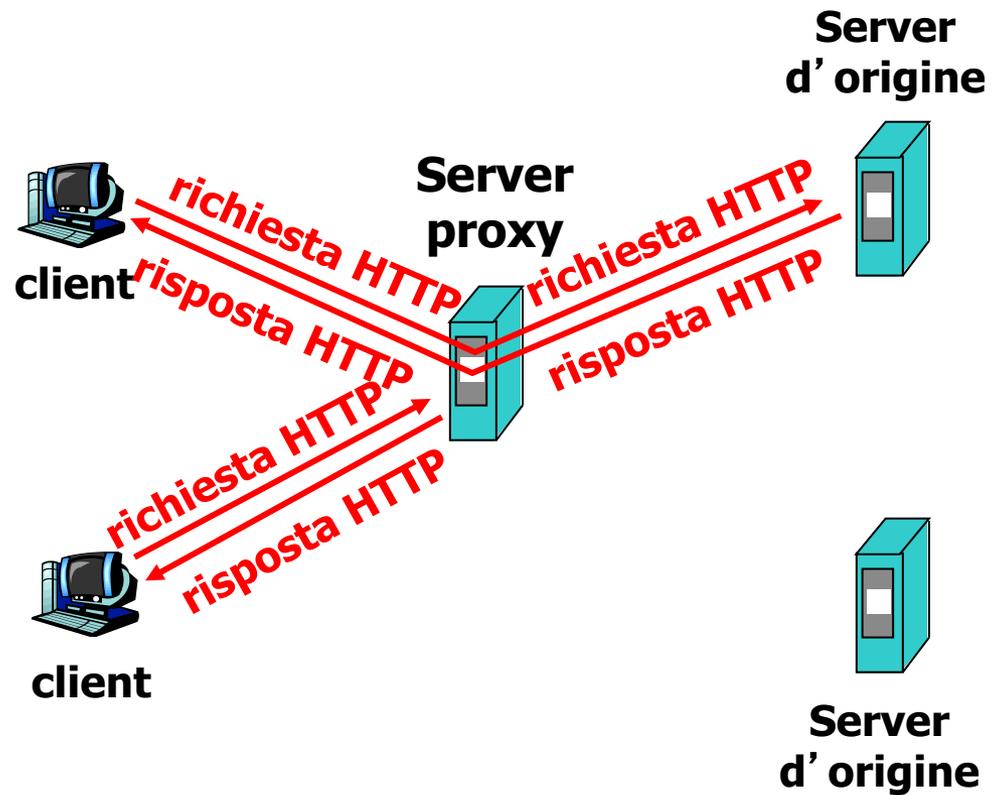
nota

Cookie e privacy:

- i cookie permettono ai siti di imparare molte cose sugli utenti
- l'utente può fornire al sito il nome e l'indirizzo e-mail
- Il comportamento del browser è influenzato dal sito in modo "personalizzato", quindi esiste un serio rischio di manipolazione

Obiettivo: soddisfare la richiesta del client senza coinvolgere il server d'origine

- L'utente configura il browser: accesso al Web tramite la cache
- Il browser trasmette tutte le richieste HTTP alla cache
 - oggetto nella cache: la cache fornisce l'oggetto
 - altrimenti la cache richiede l'oggetto al server d'origine e poi lo inoltra al client





- ❑ La cache opera come client e come server
- ❑ Tipicamente la cache è installata da un ISP (università, aziende o ISP residenziali)
- ❑ Limita la libertà dell'utente
- ❑ Può essere un punto di controllo forte (livello applicativo) degli utenti

Perché il caching web?

- ❑ Riduce i tempi di risposta alle richieste dei client
- ❑ Riduce il traffico sul collegamento di accesso a Internet
- ❑ Internet arricchita di cache consente ai provider con bassa ampiezza di banda di fornire dati con efficacia e velocità
- ❑ L'accesso alla rete è fortemente controllato e si riducono problemi di sicurezza

- ❑ **Obiettivo:** non inviare un oggetto se il client ha una copia aggiornata dell'oggetto
- ❑ **Cache** del browser: tiene una copia dell'oggetto già scaricato
- ❑ client: specifica la data della copia dell'oggetto nella richiesta HTTP
 - If-modified-since: <data>
- ❑ server: la risposta non contiene l'oggetto se la copia nella cache è aggiornata:
 - HTTP/1.0 304 Not Modified

oggetto non modificato

client

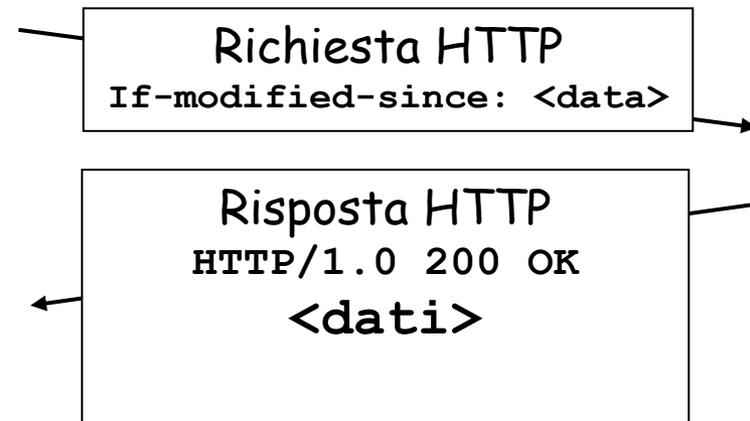
server



oggetto modificato

client

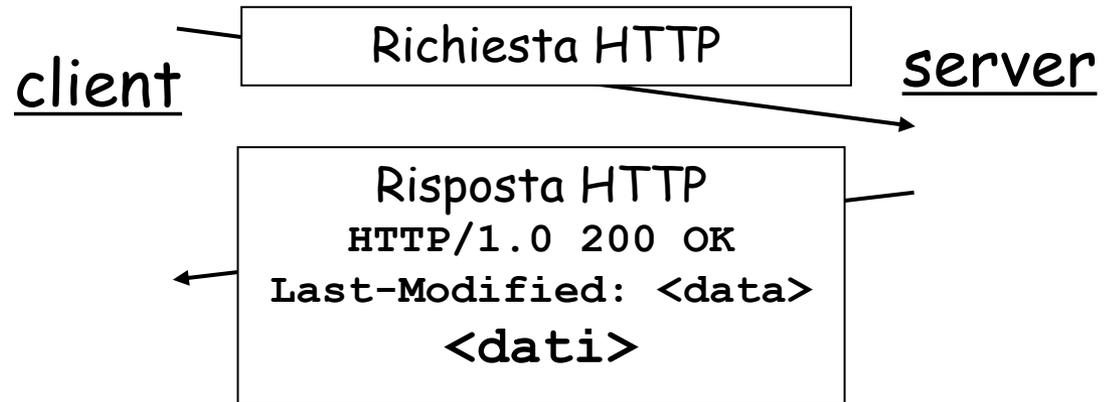
server



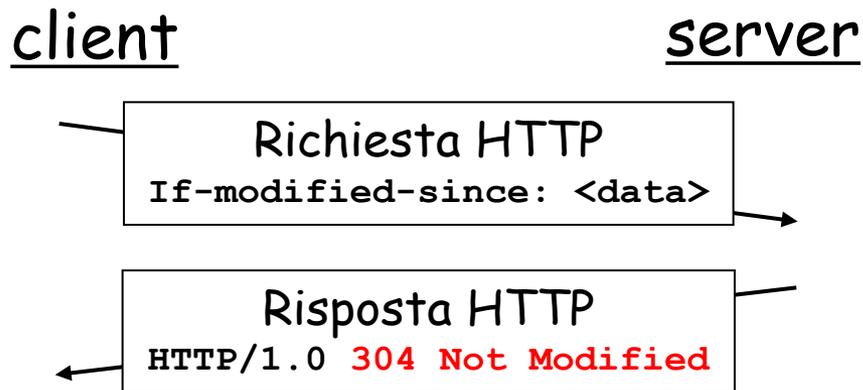


GET condizionale

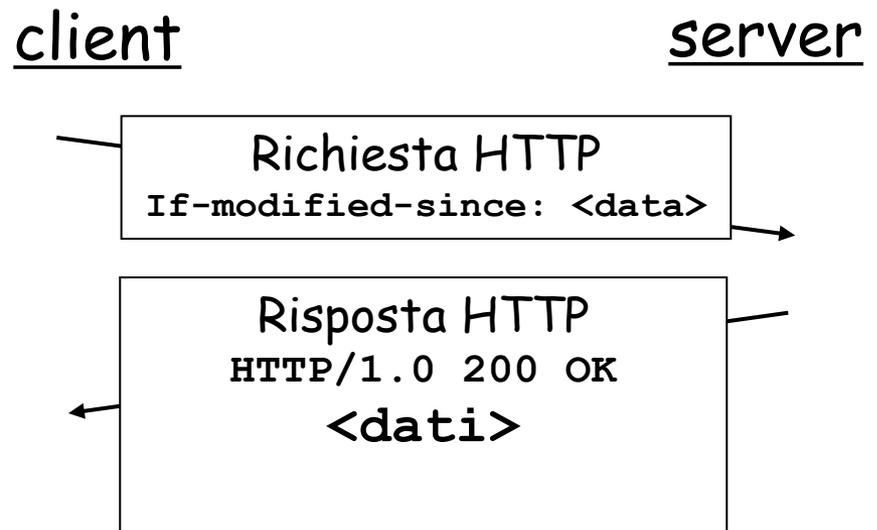
- D: Qual' è la data utilizzata?
- R: la data nella risposta originale



oggetto non modificato



oggetto modificato



EDITION: INTERNATIONAL U.S. MÉXICO ARABIC

TV: CNNi CNN en Español

Set edition preference



Home Video World U.S. Africa Asia Europe Latin America Middle East Business World Sport Entertainment

Console HTML CSS Script DOM Net

Clear Persist All HTML CSS JS XHR Images Flash Media

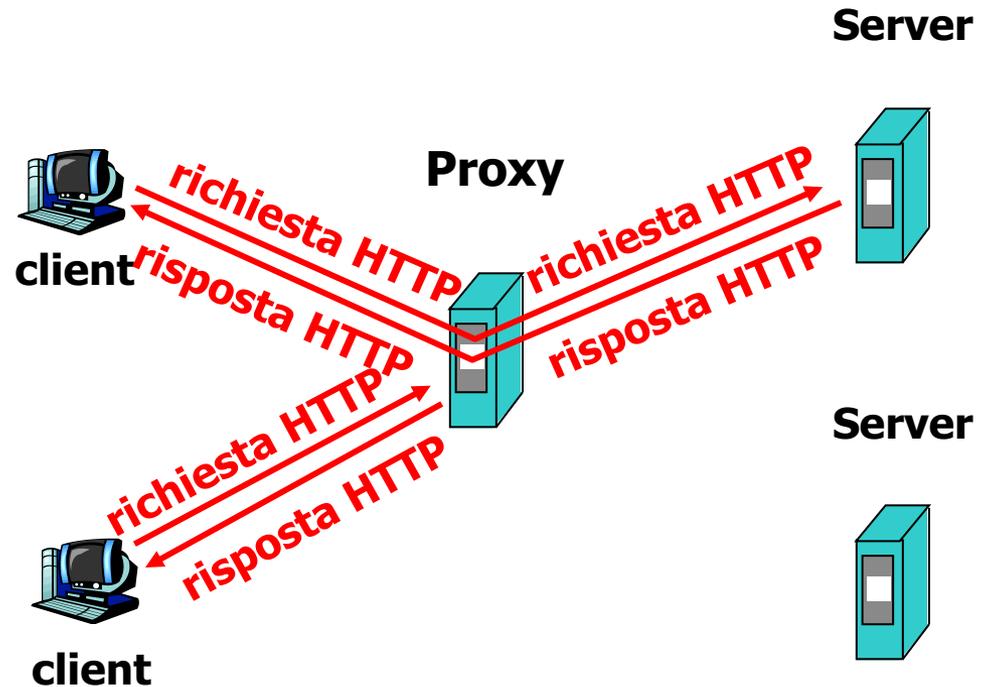
| URL | Status | Domain | Size | Remote IP | Timeline |
|-------------------|------------------|-------------------|----------|--------------------|----------|
| GET edition.cnn. | 200 OK | edition.cnn.com | 22 KB | 157.166.255.32:80 | 480ms |
| GET intlhplib-mi | 304 Not Modified | z.cdn.turner.com | 31.1 KB | 95.101.34.9:80 | 18ms |
| GET intlhplib-mi | 304 Not Modified | z.cdn.turner.com | 111.2 KB | 95.101.34.9:80 | 18ms |
| GET hdr-globe-c | 304 Not Modified | i.cdn.turner.com | 4.4 KB | 192.221.106.126:80 | 18ms |
| GET btn_search_ | 304 Not Modified | i.cdn.turner.com | 858 B | 198.78.208.254:80 | 36ms |
| GET site=cnn_in | 200 OK | ads.cnn.com | 2.3 KB | 157.166.226.207:80 | 245ms |
| GET banner.html | 200 OK | edition.cnn.com | 255 B | 157.166.255.32:80 | 120ms |
| GET 1203050810 | 200 OK | i2.cdn.turner.com | 17.6 KB | 198.78.208.254:80 | 77ms |
| GET video_icon. | 304 Not Modified | i.cdn.turner.com | 138 B | 192.221.106.126:80 | 27ms |
| GET 1px.gif | 304 Not Modified | i.cdn.turner.com | 43 B | 192.221.106.126:80 | 39ms |
| GET 120305104: | 200 OK | i2.cdn.turner.com | 41.5 KB | 198.78.208.254:80 | 92ms |
| GET 1201250240 | 200 OK | i2.cdn.turner.com | 3.5 KB | 198.78.208.254:80 | 39ms |
| GET 1203050210 | 200 OK | i2.cdn.turner.com | 5.9 KB | 198.78.208.254:80 | 56ms |
| GET 120222055: | 200 OK | i2.cdn.turner.com | 5.2 KB | 198.78.208.254:80 | 57ms |
| GET 120304112: | 200 OK | i2.cdn.turner.com | 5.8 KB | 198.78.208.254:80 | 57ms |
| GET 120305022! | 200 OK | i2.cdn.turner.com | 3.9 KB | 198.78.208.254:80 | 57ms |
| GET 120228023! | 200 OK | i2.cdn.turner.com | 4.9 KB | 198.78.208.254:80 | 75ms |
| GET 120305093! | 200 OK | i2.cdn.turner.com | 3.6 KB | 198.78.208.254:80 | 75ms |
| GET 120224122! | 200 OK | i2.cdn.turner.com | 6.3 KB | 198.78.208.254:80 | 76ms |
| GET 120305103: | 200 OK | i2.cdn.turner.com | 7.6 KB | 198.78.208.254:80 | 75ms |
| GET site=cnn_in | 200 OK | ads.cnn.com | 6.4 KB | 157.166.226.207:80 | 363ms |
| GET advertisement | 304 Not Modified | i.cdn.turner.com | 94 B | 192.221.106.126:80 | 38ms |
| GET 35x35_gene | 304 Not Modified | i.cdn.turner.com | 229 B | 192.221.106.126:80 | 37ms |
| GET close_bt.gif | 304 Not Modified | i.cdn.turner.com | 293 B | 192.221.106.126:80 | 38ms |

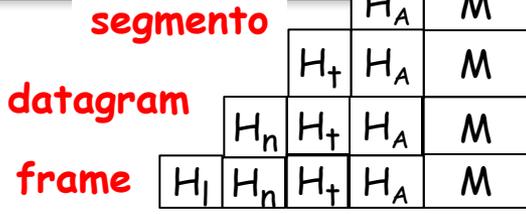
Proxy:

- interpone tra un client ed un server facendo da tramite tra i due
- inoltra le richieste e le risposte dall'uno all'altro

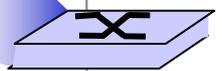
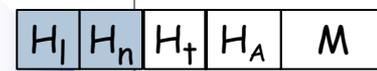
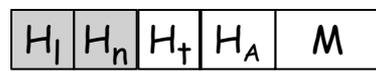
Obiettivo:

- Caching proxy
- Connettività
- Controllo/filtraggio/modifiche
- Privacy

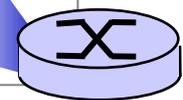
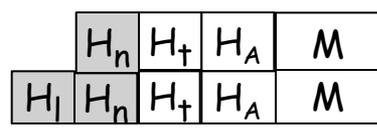




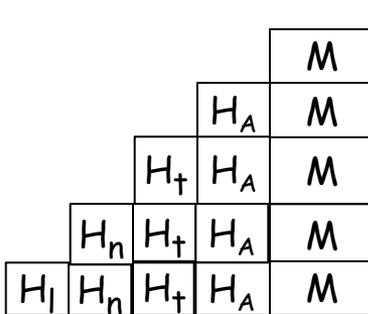
Incapsulamento (richiamo)



switch
(commutatore)



router





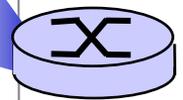
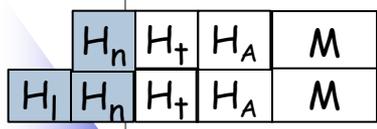
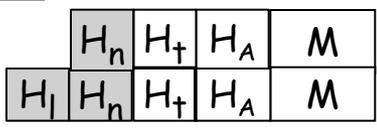
messaggio
segmento
datagram
frame



Client

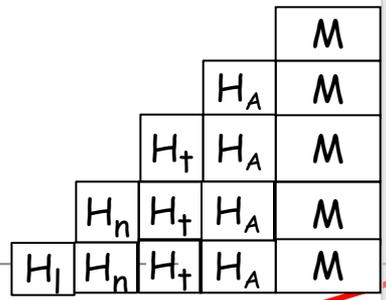
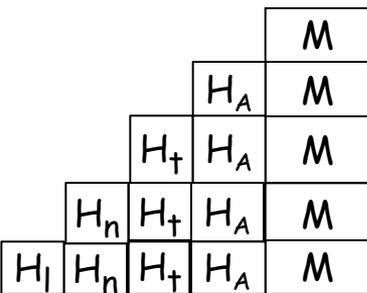


Server Proxy

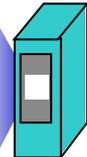


router

Server



Proxy





- Richiesta senza Proxy

GET /pub/WWW/TheProject.html HTTP/1.0

- Richiesta con Proxy

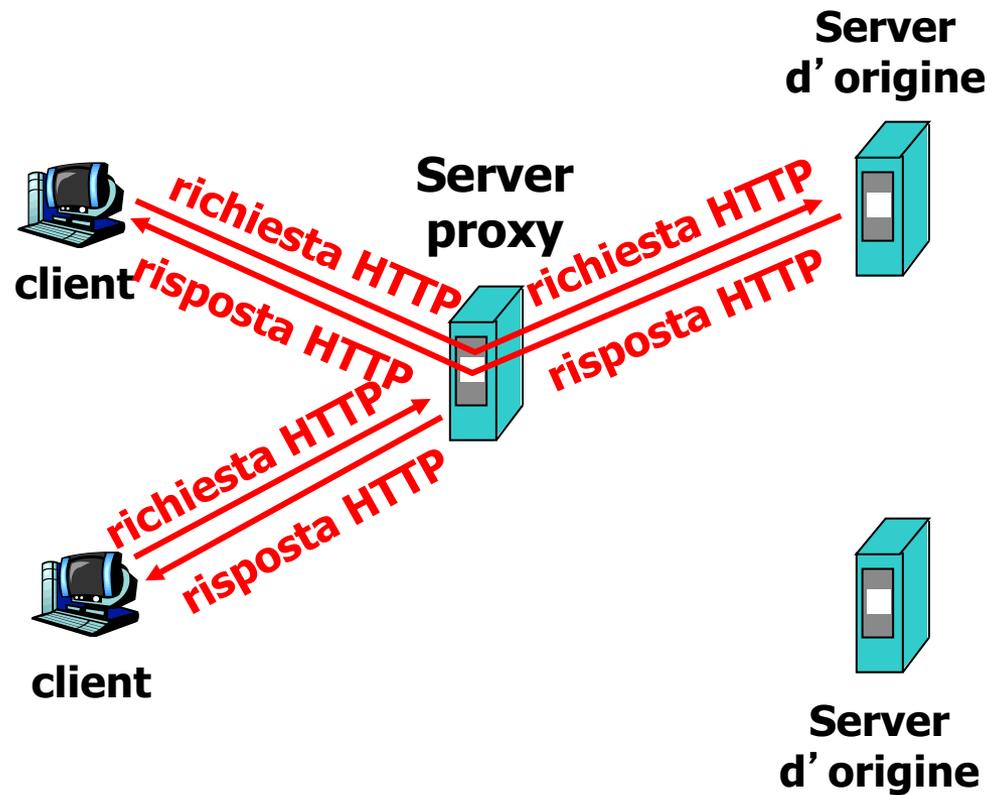
GET <http://www.w3.org/pub/WWW/TheProject.html> HTTP/1.0

absolute URL

- Absolute URL: necessario per aprire una connessione TCP verso il Proxy server

Obiettivo: soddisfare la richiesta del client senza coinvolgere il server d'origine

- ❑ L'utente configura il browser: accesso al Web tramite la cache
- ❑ Il browser trasmette tutte le richieste HTTP alla cache
 - oggetto nella cache: la cache fornisce l'oggetto
 - altrimenti la cache richiede l'oggetto al server d'origine e poi lo inoltra al client



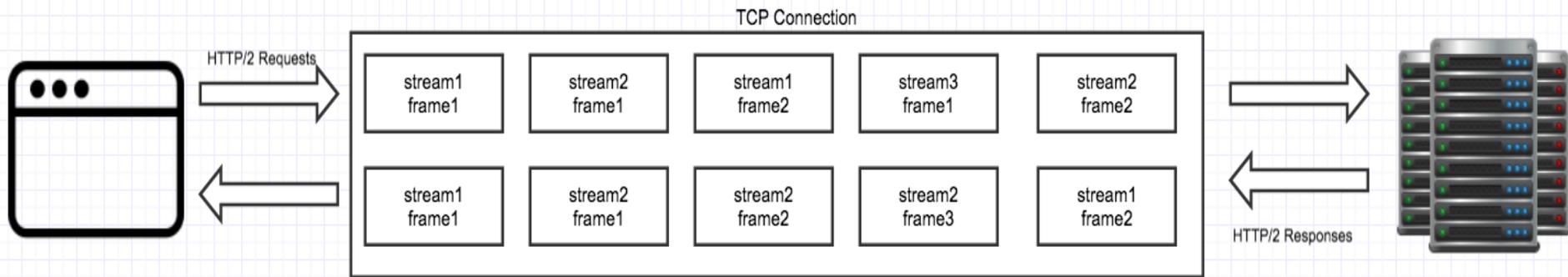


- RFC 7540 (May 2015)
- Cambio di “filosofia” per il supporto del web
- Protocollo codificato binario (header compression)
- Scopo dichiarato: Migliorare la latenza e la manipolazione dei contenuti
- Cerca di limitare a una sola connessione TCP persistente la comunicazione tra un client e un server
- Evoluzione del protocollo proprietario SPDY di Google



- Cerca di costruire dei tunnel TLS “on the fly” tra client e server
 - L’operatore di rete non vede più le diverse connessioni
- Multiplazione delle diverse request http sulla stessa connessione TCP
 - Riduzione del fenomeno di Head of the Line blocking se il servizio responsabile di una request non risponde
- Server Push: Il server può manipolare a suo piacimento l’ordine delle richieste
 - Priorità alle informazioni più importanti ... o alla pubblicità
 - Minore controllo del servizio da parte del client (browser)

- La maggior parte dei browser lo supporta
- I server non ancora in modo diffuso
- Introduce la nozione di “stream” e “frame” per fare il multiplexing delle richieste sulla stessa connessione TCP



- ❑ 2.1 Principi delle applicazioni di rete
- ❑ 2.2 Web e HTTP
- ❑ 2.3 FTP
- ❑ **2.4 Posta Elettronica**
 - ❑ SMTP, POP3, IMAP
- ❑ 2.5 DNS
- ❑ Protocolli per applicazioni multimediali

Componenti principali:

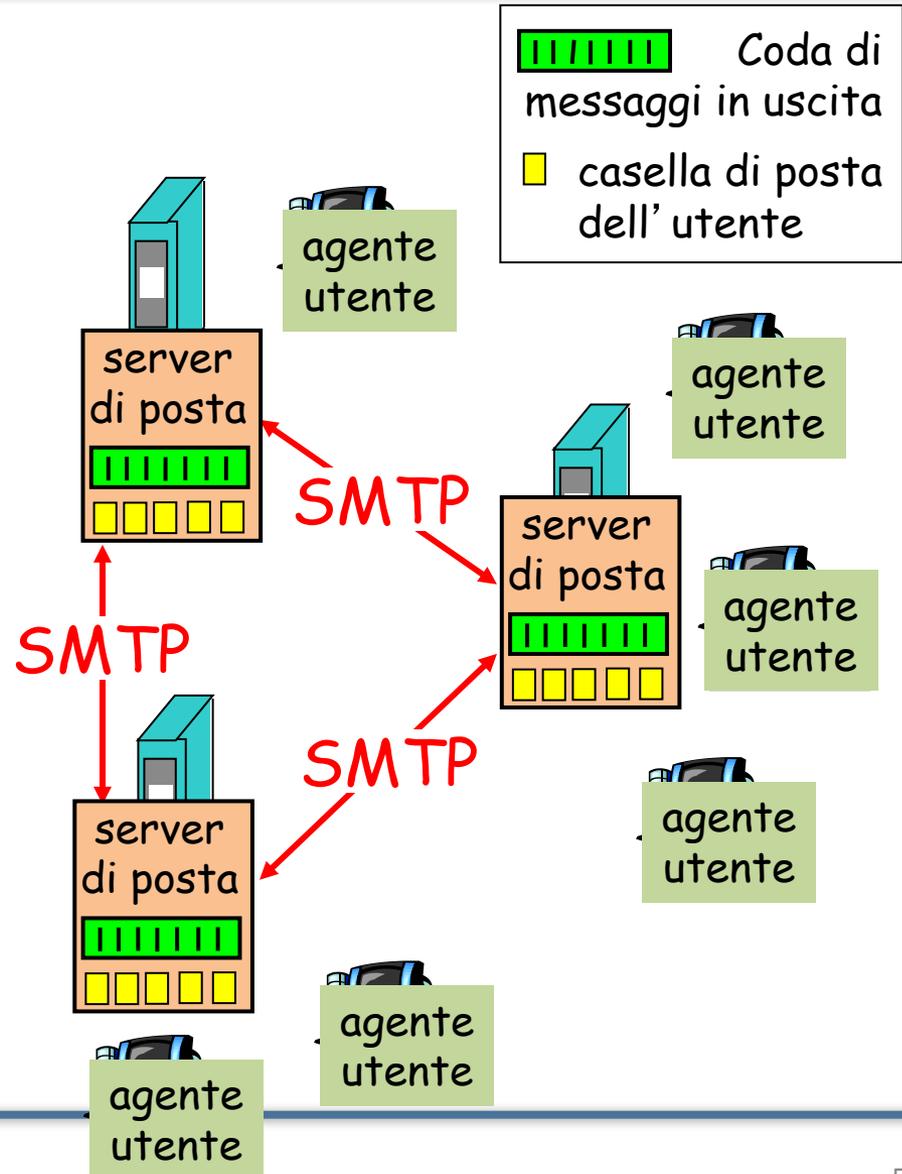
- agente utente
- server di posta

Protocolli principali:

- SMTP: Simple Mail Transfer Protocol
- POP3: Post Office Protocol
- IMAP: Internet Mail Access Protocol

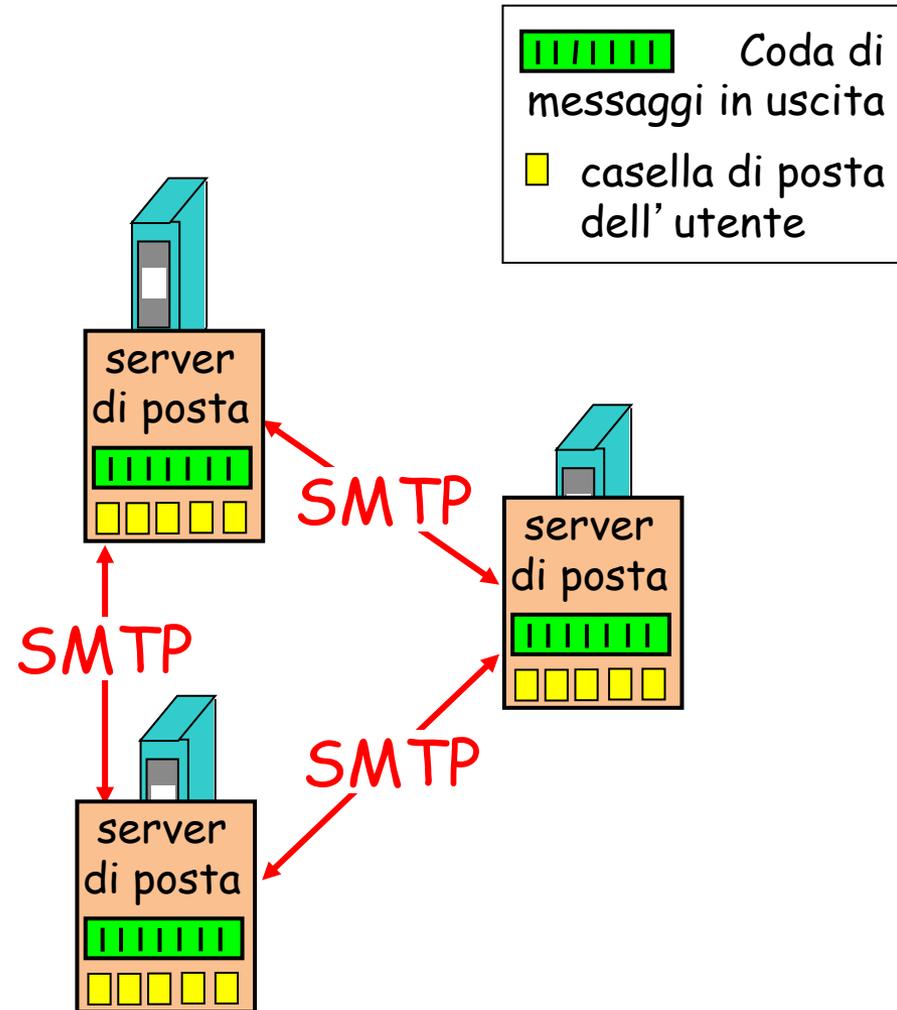
Agente utente

- detto anche "mail reader"
- composizione, editing, lettura dei messaggi di posta elettronica
- esempi:
 - Eudora, Outlook, Mozilla Thunderbird
 - pine, elm
 - Web browser!
- i messaggi in uscita o in arrivo sono memorizzati sul server



Server di posta

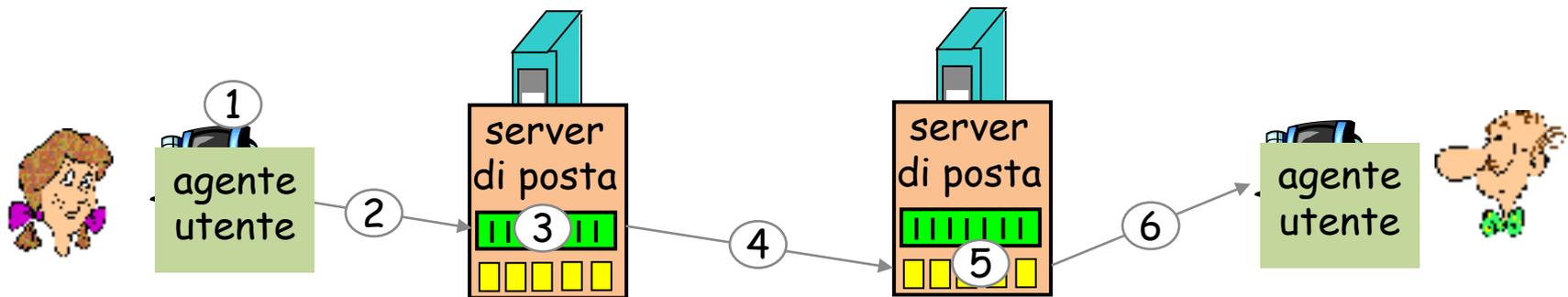
- ❑ **Casella di posta** (*mailbox*)
contiene i messaggi in arrivo per l'utente
- ❑ **Coda di messaggi** da trasmettere
- ❑ **Protocollo SMTP** tra server di posta per inviare messaggi di posta elettronica
 - client: server di posta trasmittente
 - "server": server di posta ricevente





- ❑ usa TCP per trasferire in modo affidabile i messaggi di posta elettronica dal client al server, porta 25
- ❑ trasferimento diretto: il server trasmittente al server ricevente (di solito)
- ❑ tre fasi per il trasferimento
 - handshaking (saluto)
 - trasferimento di messaggi
 - chiusura
- ❑ interazione comando/risposta
 - **comandi**: testo ASCII
 - **risposta**: codice di stato ed espressione
- ❑ i messaggi devono essere nel formato ASCII a 7 bit

- 1) Alice usa il suo agente utente per comporre il messaggio da inviare "a" `rob@someschool.edu`
- 2) L'agente utente di Alice invia un messaggio al server di posta di Alice; il messaggio è posto nella coda di messaggi
- 3) Il lato client di SMTP apre una connessione TCP con il server di posta di Roberto
- 4) Il client SMTP invia il messaggio di Alice sulla connessione TCP
- 5) Il server di posta di Roberto pone il messaggio nella casella di posta di Roberto
- 6) Roberto invoca il suo agente utente per leggere il messaggio





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



- SMTP usa connessioni persistenti
- SMTP richiede che il messaggio (intestazione e corpo) sia nel formato ASCII a 7 bit
- Il server SMTP usa CRLF.CRLF per determinare la fine del messaggio

Confronto con HTTP:

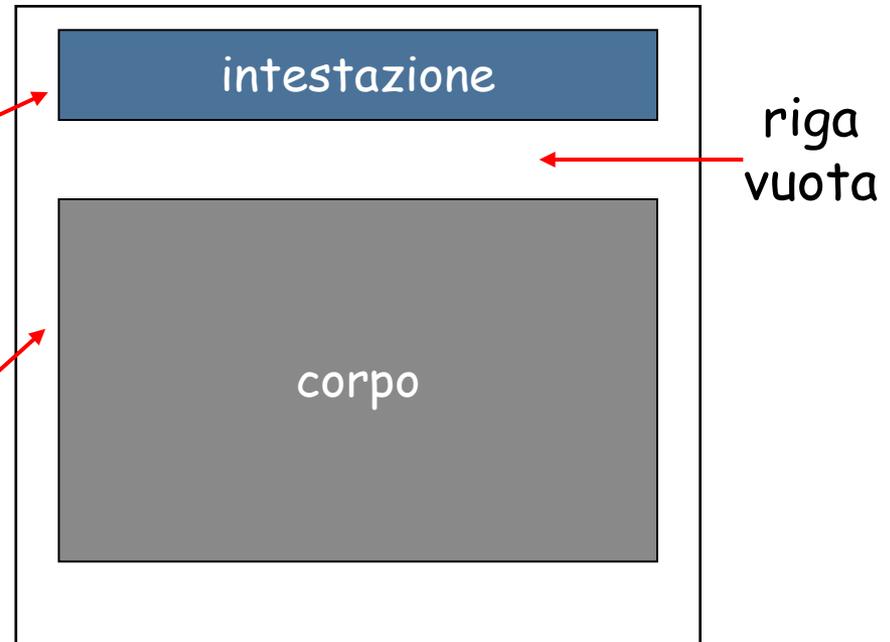
- HTTP: pull
- SMTP: push
- Entrambi hanno un'interazione comando/risposta in ASCII, codici di stato
- HTTP: ciascun oggetto è incapsulato nel suo messaggio di risposta
- SMTP: più oggetti vengono trasmessi in un unico messaggio

SMTP: protocollo per scambiare
messaggi di posta elettronica

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

- Righe di intestazione, per
esempio
 - To/A:
 - From/Da:
 - Subject/Oggetto:

*differenti dai
comandi SMTP !*
- corpo
 - il “messaggio”, soltanto
caratteri ASCII



- ❑ MIME: estensioni di messaggi di posta multimediali, RFC 2045, 2056
- ❑ Alcune righe aggiuntive nell'intestazione dei messaggi dichiarano il tipo di contenuto MIME

Versione MIME

metodo usato
per codificare i dati

Tipo di dati
multimediali, sottotipo,
dichiarazione
dei 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
    
```



Chi ha
inviato

Chi ha
ricevuto

Quando

Received: from crepese.fr by hamburger.edu;
12 Oct 98 15:27:39 GMT

From: alice@crepes.fr

To: bob@hamburger.edu

Subject: Picture of yummy crepe.

MIME-Version: 1.0

Content-Transfer-Encoding: base64

Content-Type: image/jpeg

base64 encoded data

.....

.....base64 encoded data



Return-Path: <alberto.fondriest@gmail.com>

Received: by disi.unitn.it with ESMTP id r2C8VAML002967; Tue, 12 Mar 2013 09:31:10 +0100

Received: from mail2.unitn.it (mail2.unitn.it [193.205.206.22])

by mailhub1.unitn.it (Postfix) with ESMTP id 21253AE5466

for <locigno@disi.unitn.it>; Tue, 12 Mar 2013 09:31:10 +0100 (CET)

Received: from mail2.unitn.it (localhost.localdomain [127.0.0.1])

by localhost (Email Security Appliance) with SMTP id 05DFDBD8DA_13EE7CEB

for <locigno@disi.unitn.it>; Tue, 12 Mar 2013 08:31:10 +0000 (GMT)

Received: from mail-vc0-f171.google.com (mail-vc0-f171.google.com [209.85.220.171])

by mail2.unitn.it (Sophos Email Appliance) with ESMTP id 8B476BBAD7_13EE7CDF

for <locigno@disi.unitn.it>; Tue, 12 Mar 2013 08:31:09 +0000 (GMT)

Received: by mail-vc0-f171.google.com with SMTP id fk10so268115vcb.16

for <locigno@disi.unitn.it>; Tue, 12 Mar 2013 01:31:09 -0700 (PDT)



DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed;
d=gmail.com; s=20120113;
h=mime-version:x-received:in-reply-to:references:date:message-id
:subject:from:to:cc:content-type;
bh=4jgE9Hzcnpa4A7/B83etPn58dldy88QmnY7qogg8t4Q=;
b=zMQLFc71F+Ub4nqxZbIU2mtOrJCirxZ3zK/eyih2G3MJT/lmgvPgi9daMKF9QZ6Qi
Svrnq+VUjentQHfWmQrMrG3zxiWAJFQ2ys642+yDn9m8ru9xMJDxkNx8a7duufMpgsnN
C54/UPUnMdlyjwN0m7EMBnAlthUk4Q8yPaTM2MQ/OglYoLoXaNqUnUtRmlN/+WuD4XP3
wN3vmEK6c2P6sTNyUEZ+n0G7HaPBs7fM3swSDRPSzM7oeC8Qdolb9hF3VUUeP5oA2k6d
xPpc+jwgUw33Q4ezdN1YY78nCWbVTvD7XN/wk3shNn5GxY9vQ8VeSfcx7AvqdepUBbo0
rOnQ==



MIME-Version: 1.0

X-Received: by 10.58.137.34 with SMTP id qf2mr6338871veb.25.1363077068936;
Tue, 12 Mar 2013 01:31:08 -0700 (PDT)

Received: by 10.58.100.196 with HTTP; Tue, 12 Mar 2013 01:31:08 -0700 (PDT)

In-Reply-To: <513ED9EB.8060903@disi.unitn.it>

References: <CAC71bEQtwoM8QdiGTBXVe1euBg3KkzACbD3inSvQhd5xGnTP-A@mail.gmail.com>
<513ED9EB.8060903@disi.unitn.it>

Date: Tue, 12 Mar 2013 09:31:08 +0100

Message-ID: <CAC71bEQ8G1f5prSLzuvJ38jQcnSWNxoHcmagev0z3T_VV3Wt_w@mail.gmail.com>

Subject: Re: sabato

From: alberto fondriest <alberto.fondriest@gmail.com>

To: Renato Lo Cigno <locigno@disi.unitn.it>

Cc: Giorgio Ranzani <Giorgio.Ranzani@ascot.tn.it>



Content-Type: multipart/alternative; boundary=089e012954667edeb504d7b61c1c

X-Sophos-ESA: [mail2.unitn.it] 3.7.7.1, Antispam-Engine: 2.7.2.1390750, Antispam-Data: 2013.3.12.81526

--089e012954667edeb504d7b61c1c

Content-Type: text/plain; charset=ISO-8859-1

Content-Transfer-Encoding: quoted-printable

questo tratto lo rifaremo ...

....

....

--089e012954667edeb504d7b61c1c

Content-Type: text/html; charset=ISO-8859-1

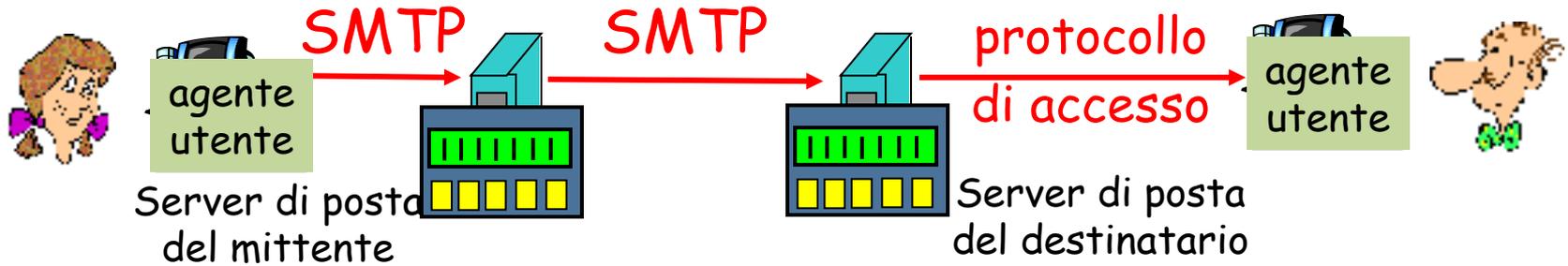
Content-Transfer-Encoding: quoted-printable

<div dir=3D"ltr">questo tratto lo rifaremo a breve con condizioni buone per avere un'idea dei tempi, credo che noi in cinque ore possiamo arrivarci, ...



telnet servername 25

- Server risponde con codice 220
- Utilizzando I comandi
 - HELO, MAIL FROM, RCPT TO, DATA, QUIT
- potete mandare e-mail “a mano”
- Anche questo in una rete ben gestita funziona solo fino a un certo punto (magari anche molto avanzato ...)
- Potete anche provare ad abilitare un server SMTP sul vostro PC per provarci ... ricordatevi di chiuderlo dopo!!!



- ❑ SMTP: consegna/memorizzazione sul server del destinatario
- ❑ Protocollo di accesso alla posta: ottenere i messaggi dal server
 - (agente utente direttamente sul server di posta)
 - accesso diretto tramite il file system
 - POP: Post Office Protocol [RFC 1939]
 - Porta 110
 - autorizzazione (agente <--> server) e download
 - IMAP: Internet Mail Access Protocol [RFC 1730]
 - più funzioni (più complesse)
 - manipolazione di messaggi memorizzati sul server
 - HTTP: gmail, Hotmail , Yahoo! Mail, webmail UNITN, ecc.

Fase di autorizzazione

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

Fase di transazione

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

```
S: +OK POP3 server ready
C: user rob
S: +OK
C: pass hungry
S: +OK user successfully logged on
```

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```



Ancora su POP3

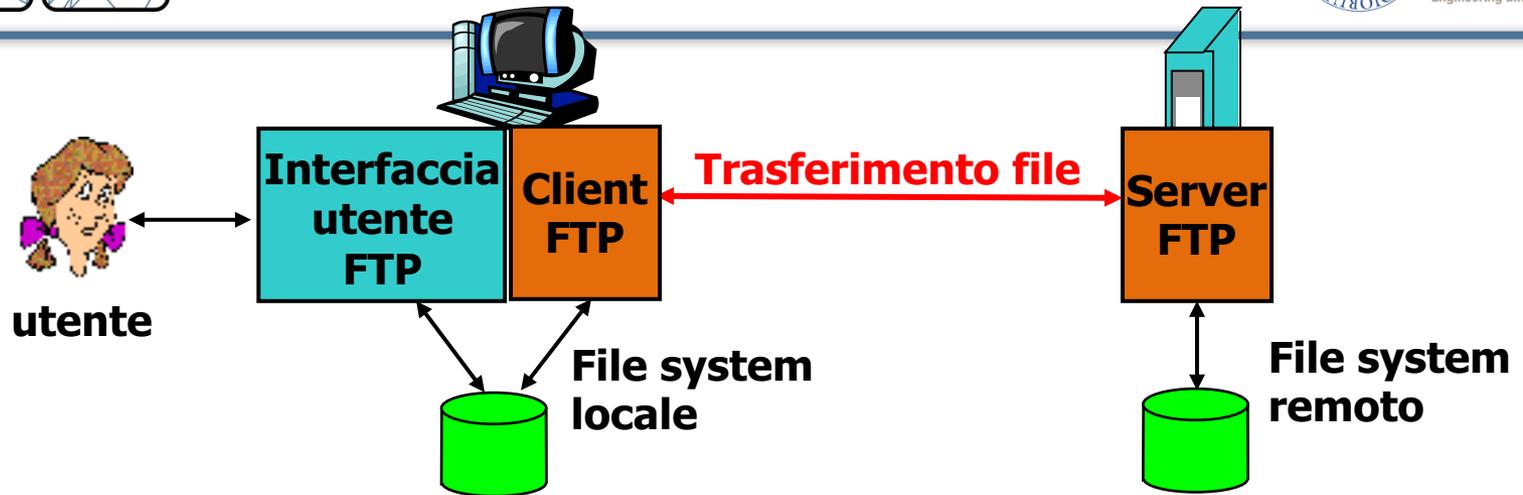
- ❑ Il precedente esempio usa la modalità "scarica e cancella"
- ❑ Roberto non può rileggere le e-mail se cambia client
- ❑ Modalità "scarica e mantieni": copia i messaggi sul client e ne mantiene una copia sul server
- ❑ POP3 è un protocollo senza stato tra le varie sessioni

IMAP

- ❑ Mantiene tutti i messaggi anche sul server
- ❑ Consente all'utente di organizzare i messaggi in cartelle anche sul server
- ❑ IMAP conserva lo stato dell'utente tra le varie sessioni:
 - I nomi delle cartelle e l'associazione tra identificatori dei messaggi e nomi delle cartelle
 - Sincronizzazione tra sessioni



- ❑ 2.1 Principi delle applicazioni di rete
- ❑ 2.2 Web e HTTP
- ❑ **2.3 FTP**
- ❑ 2.4 Posta Elettronica
 - ❑ SMTP, POP3, IMAP
- ❑ 2.5 DNS
- ❑ Protocolli per applicazioni multimediali



- ❑ Trasferimento file a/da un host remoto
- ❑ Modello client/server
 - *client*: il lato che inizia il trasferimento (a/da un host remoto)
 - *server*: host remoto
- ❑ ftp: RFC 959
- ❑ server ftp: porta 21



connessione di controllo connessione dati

- ❑ Il client FTP contatta il server FTP alla porta 21, specificando TCP come protocollo di trasporto
- ❑ Il client ottiene l'autorizzazione sulla connessione di controllo
- ❑ Il client cambia la directory remota inviando i comandi sulla connessione di controllo
- ❑ Quando il server riceve un comando per trasferire un file, apre una connessione dati TCP con il client
- ❑ Dopo il trasferimento di un file, il server chiude la connessione



- ❑ Il server apre una seconda connessione dati TCP per trasferire un altro file.
- ❑ Connessione di controllo: “fuori banda” (*out of band*)
- ❑ Il server FTP mantiene lo “stato”: associare la connessione di controllo ad un utente e tenere traccia della directory corrente



Comandi comuni:

- Inviati come testo ASCII sulla connessione di controllo
- USER *username***
- PASS *password***
- LIST**
elenca i file della directory corrente
- RETR *filename***
recupera (*get*) un file dalla directory corrente
- STOR *filename*** memorizza (*put*) un file nell'host remoto

Codici di ritorno comuni:

- Codice di stato ed espressione (come in HTTP)
- 331 Username OK, password required
- 125 data connection already open; transfer starting
- 425 Can't open data connection
- 452 Error writing file



- ❑ 2.1 Principi delle applicazioni di rete
- ❑ 2.2 Web e HTTP
- ❑ 2.3 FTP
- ❑ 2.4 Posta Elettronica
 - ❑ SMTP, POP3, IMAP
- ❑ **2.5 DNS**
- ❑ Protocolli per applicazioni multimediali



Persone:

- molti identificatori:
 - nome, codice fiscale, numero della carta d'identità, ecc.

Host e router di Internet:

- indirizzo IP (32 bit) - usato per indirizzare i datagrammi
 - Esempio: 193.205.194.4
 - Utilizzato a livello IP (anche a livello trasporto)
 - <http://193.205.194.4/>
 - kiraly@193.205.194.4
- “nome”, ad esempio, disi.unitn.it, www.yahoo.com
 - Usato dagli esseri umani



Domain Name System:

- ❑ *Servizio: traduzione "nome" -> indirizzo IP*

- ❑ *Architettura: Database distribuito implementato in una gerarchia di server DNS*

- ❑ *Protocollo: protocollo a livello di applicazione che consente agli host e ai server DNS di comunicare per risolvere i nomi (tradurre indirizzi/nomi)*
 - *funzione vitale(?) di Internet implementata come protocollo a livello di applicazione*

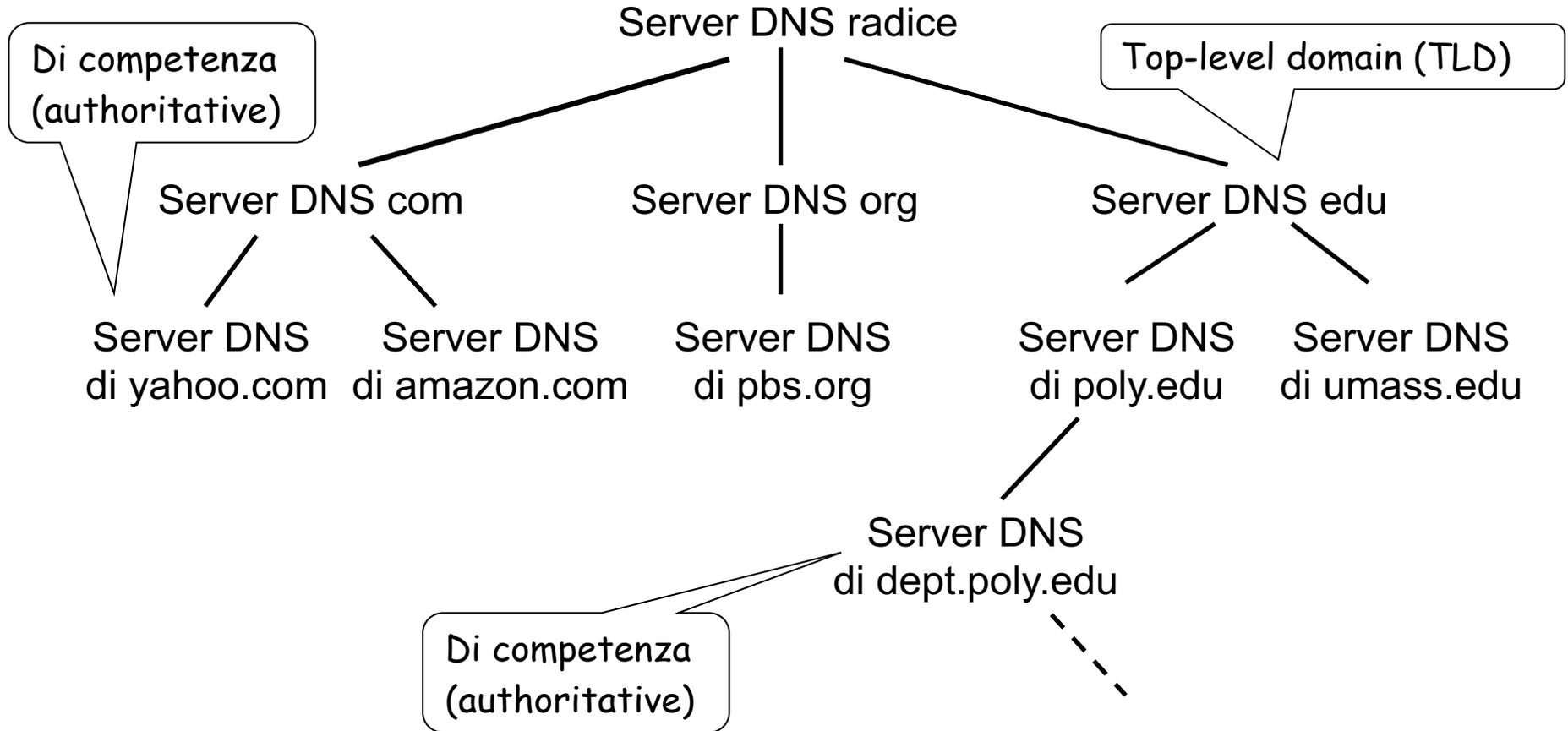


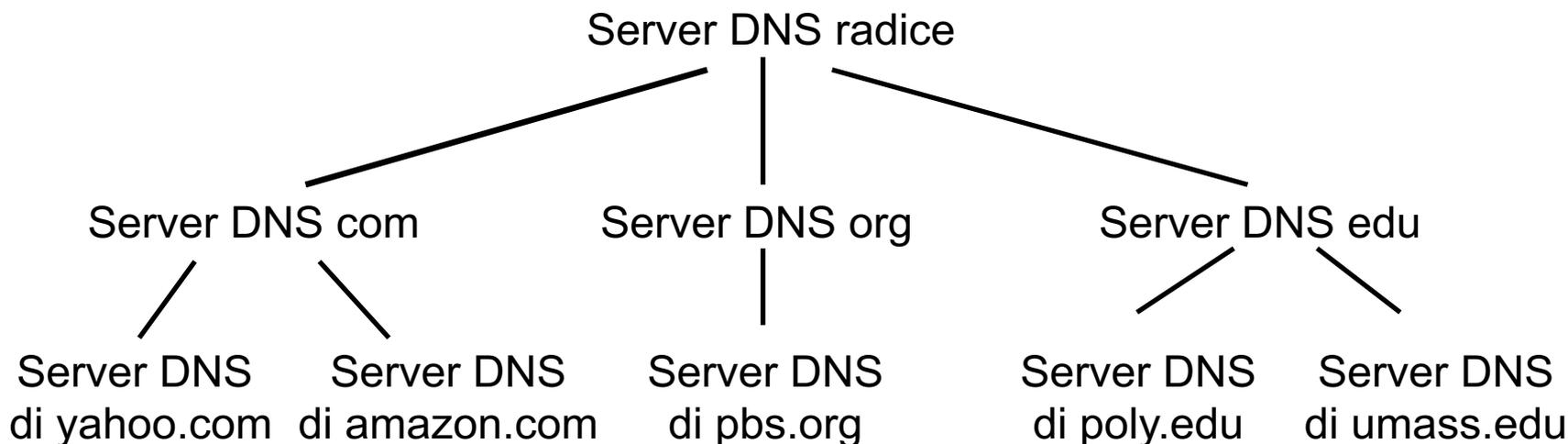
FQDN: Fully Qualified Domain Name

- Esempio: alpha.science.unitn.it
- Gerarchico:
 - it: Top Level Domain (TLD)
 - Gestito da un ente nazionale
 - unitn.it
 - Gestito da UNITN
 - science.unitn.it
 - Gestito da CISCA, Presidio I.T. del Polo di Collina
 - alpha.science.unitn.it
 - FQDN di un host gestito da CISCA



Database distribuiti e gerarchici





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

Ogni ISP/Organizzazione ha un proprio server DNS locale (v. esempi dopo)



Servizi DNS

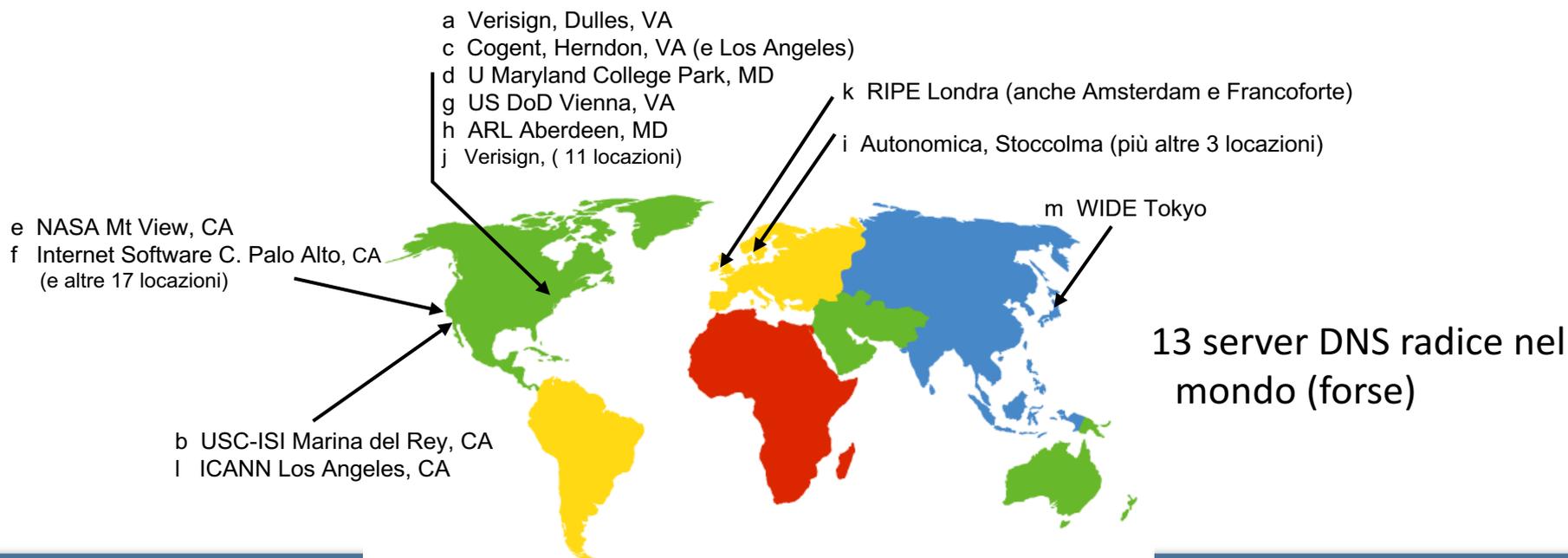
- Traduzione degli hostname in indirizzi IP
- Host aliasing
 - un host può avere più nomi
- Mail server aliasing
 - indica il server di posta elettronica per un certo dominio
- Distribuzione locale
 - server web replicati: insieme di indirizzi IP per un unico nome canonico

Perché non centralizzare DNS?

- singolo punto di guasto
- volume di traffico
- database centralizzato distante
- aggiornamento frequente

Un database centralizzato su un singolo server DNS non è *scalabile* !

- ❑ <https://www.iana.org/domains/root/servers>
- ❑ contattato da un server DNS locale che non può tradurre il nome
- ❑ server DNS radice:
 - contatta un server DNS autorizzato se non conosce la mappatura
 - ottiene la mappatura
 - restituisce la mappatura al server DNS locale





- **Server TLD (top-level domain):** si occupano dei domini com, org, net, edu, ecc. e di tutti i domini locali di alto livello, quali it, uk, fr, ca e jp
 - Network Solutions gestisce i server TLD per il dominio com
 - Educause gestisce quelli per il dominio edu

- **Server di competenza (*authoritative server*):** ogni organizzazione dotata di host Internet pubblicamente accessibili (quali i server web e i server di posta) deve fornire i record DNS di pubblico dominio che mappano i nomi di tali host in indirizzi IP
 - possono essere mantenuti dall'organizzazione o dal service provider

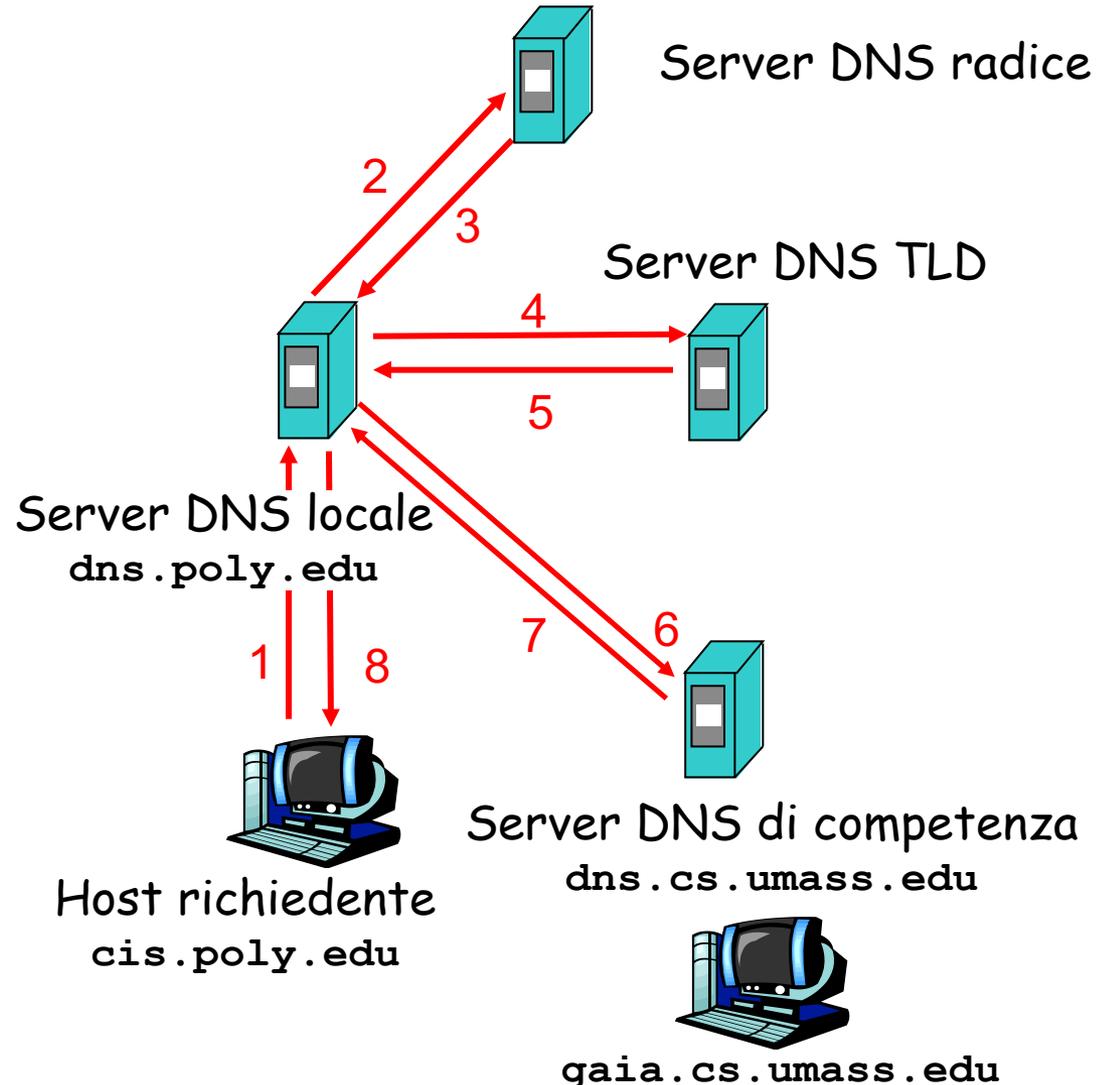


- ❑ Non appartiene strettamente alla gerarchia dei server
- ❑ Ciascun ISP (università, società, ISP residenziale) ha un server DNS locale.
 - detto anche “default name server”
- ❑ Quando un host effettua una richiesta DNS, la query viene inviata al suo server DNS locale
 - il server DNS locale opera da proxy e inoltra la query in una gerarchia di server DNS

- L'host `cis.poly.edu` vuole l'indirizzo IP di `gaia.cs.umass.edu`

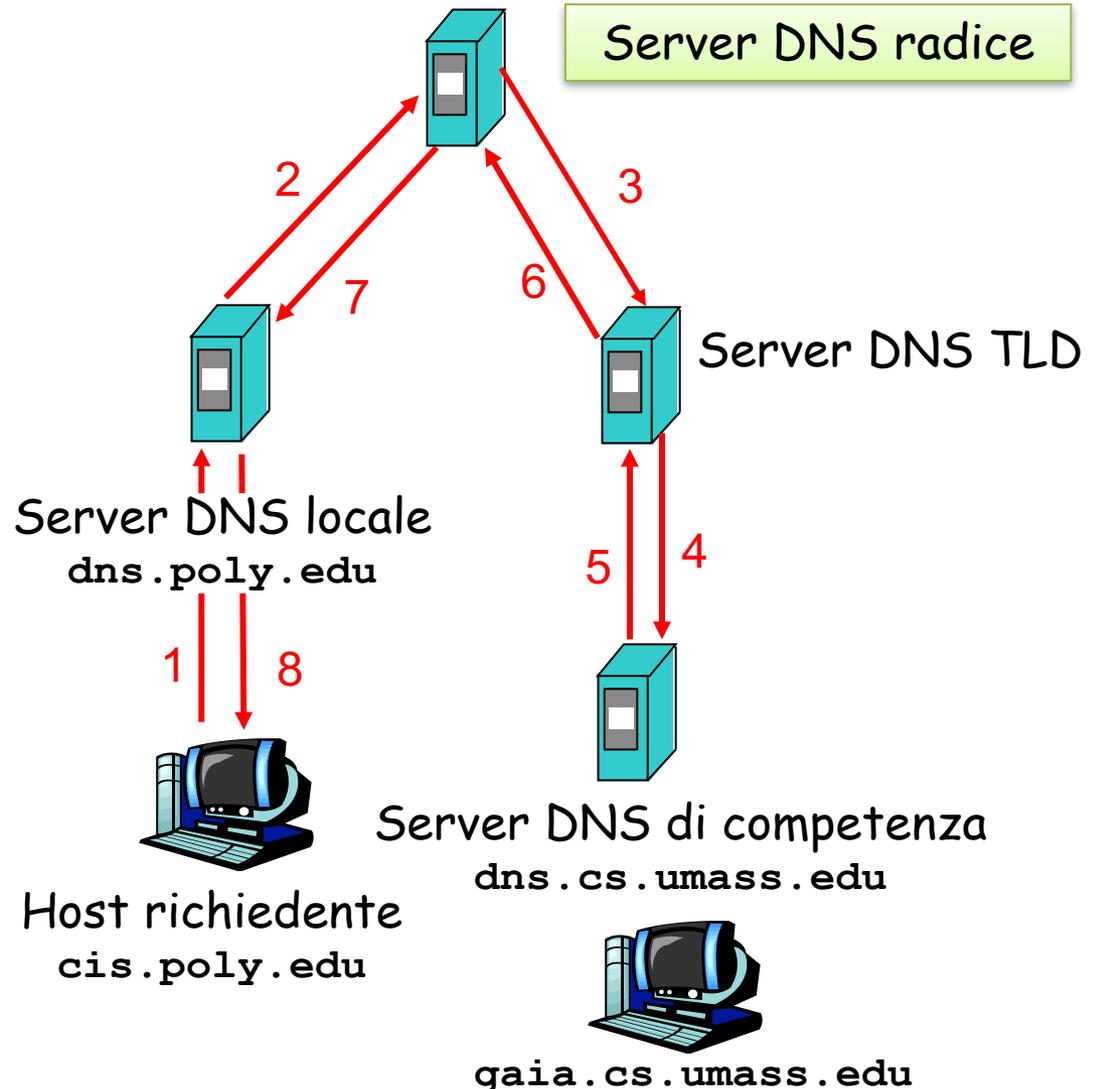
Query iterativa:

- Il server contattato risponde con il nome del server da contattare
- “Io non conosco questo nome, ma puoi chiederlo a questo server”



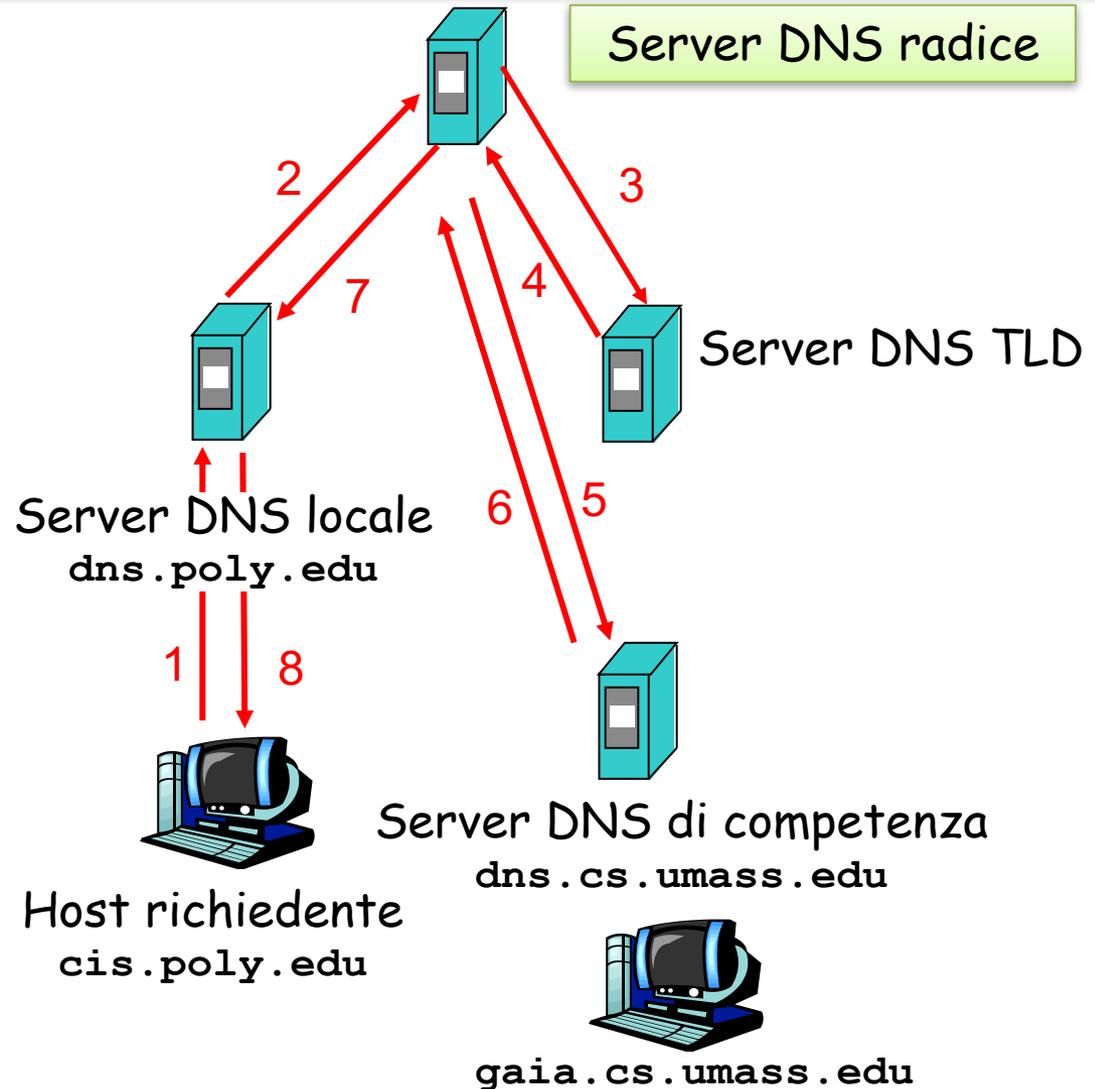
Query ricorsiva:

- ❑ Affida il compito di tradurre il nome al server DNS contattato
- ❑ Il server contattato risponde con l'indirizzo IP di `gaia.cs.umass.edu`
- ❑ “lo non conosco questo nome, ma posso procurarlo”



Query:

- ❑ Ricorsiva per il server DNS locale
- ❑ Iterativa per il server DNS radice





- Una volta che un server DNS impara la mappatura (=associazione nome-indirizzo IP), la mette nella *cache*
 - le informazioni nella cache vengono invalidate (sariscono) dopo un certo periodo di tempo
 - tipicamente un server DNS locale memorizza nella cache gli indirizzi IP dei server TLD
 - quindi i server DNS radice non vengono visitati spesso
- I meccanismi di aggiornamento/notifica sono progettati da IETF
 - RFC 2136
 - <http://www.ietf.org/html.charters/dnsind-charter.html>

DNS: database distribuito che memorizza i record di risorsa (**Resource Record - RR**)

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

- Type=A
 - name è il nome dell' host
 - value è l' indirizzo IP
- Type=NS
 - name è il dominio (ad esempio foo.com)
 - value è il nome dell'host del server di competenza di questo dominio
- Type=CNAME
 - name è il nome alias di qualche nome “canonico” (nome vero)
www.ibm.com è in realtà servereast.backup2.ibm.com
 - value è il nome canonico
- Type=MX
 - value è il nome del server di posta associato a name



La società Barsport possiede due calcolatori:

1. *hobbes.barsport.com*

2. *calvin.barsport.com*

Decide di installare un server web su *hobbes.barsport.com* e di assegnargli il nome *www.barsport.com*

Soluzioni:

- cambiare il nome di *hobbes.barsport.com*
- aggiungere un record CNAME corrispondente a *www.barsport.com* che punta a *hobbes.barsport.com*:

(*www.barsport.com*, *hobbes.barsport.com*, CNAME)

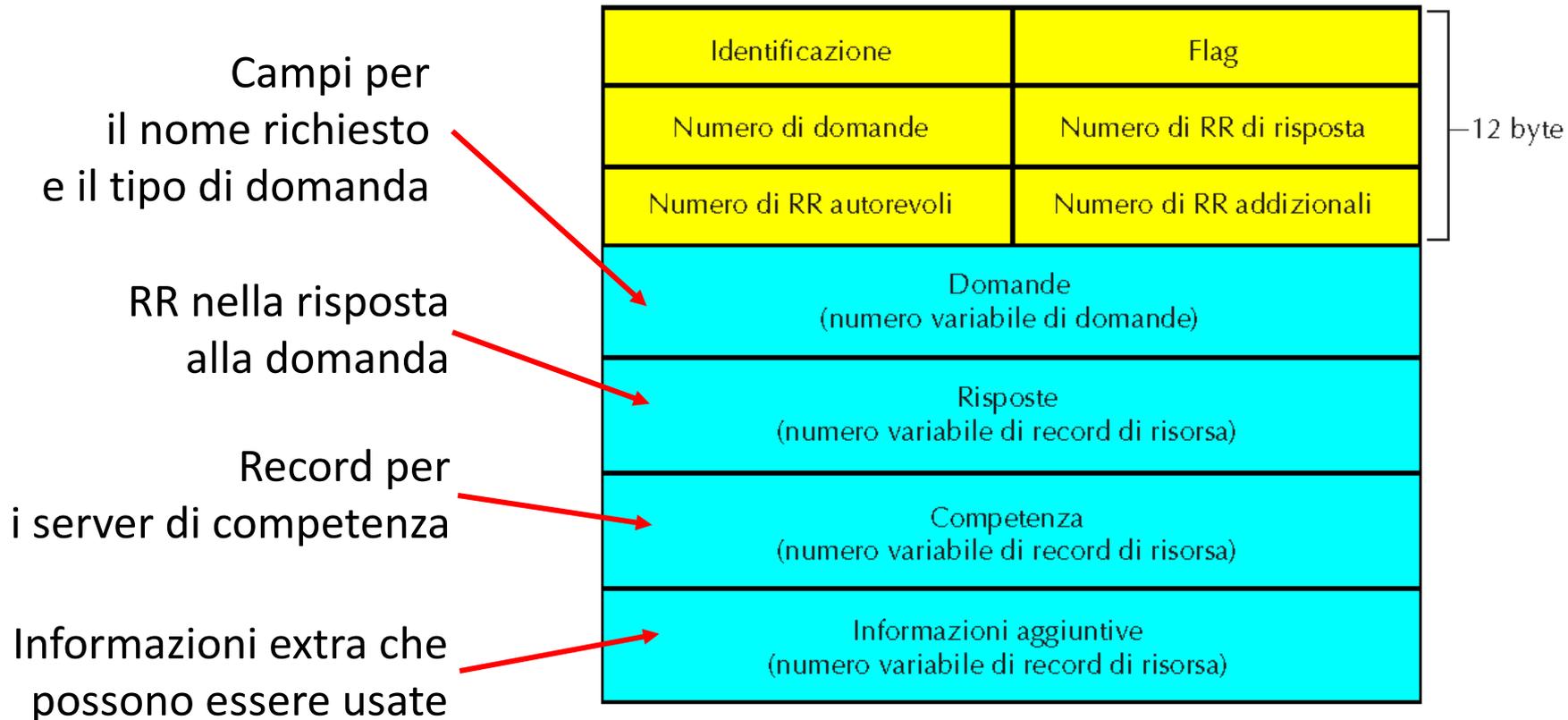
(*hobbes.barsport.com*, 123.144.134.211, A) **record preesistente**

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

Intestazione del messaggio

- **Identificazione**: numero di 16 bit per la domanda;
la risposta alla domanda usa lo stesso numero
- **Flag**:
 - domanda(0) / risposta(1)
 - richiesta di ricorsione
 - ricorsione disponibile
 - risposta di competenza

| | | |
|--|--------------------------|-----------|
| Identificazione | Flag | } 12 byte |
| Numero di domande | Numero di RR di risposta | |
| Numero di RR autorevoli | Numero di RR aggiuntivi | |
| Domande (numero variabile di domande) | | |
| Risposte (numero variabile di record di risorsa) | | |
| Competenza (numero variabile di record di risorsa) | | |
| Informazioni aggiuntive (numero variabile di record di risorsa) | | |





- ❑ Esempio: abbiamo appena avviato la nuova società “Network Utopia”
- ❑ Registriamo il nome `networkutopia.com` presso registrar (ad esempio, Network Solutions)
- ❑ Inseriamo nel server di competenza `dns1.networkutopia.com` un record tipo A per `www.networkutopia.com` e un record tipo MX per `networkutopia.com`
- ❑ Forniamo a registrar i nomi e gli indirizzi IP dei server DNS di competenza
 - Registrar inserisce due RR nel server TLD com:

```
(networkutopia.com, dns1.networkutopia.com, NS)  
(dns1.networkutopia.com, 212.212.212.1, A)
```

TLD com

(networkutopia.com, dns1.networkutopia.com, NS)

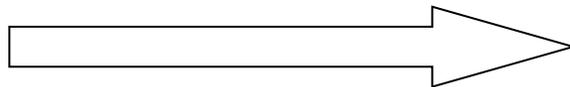
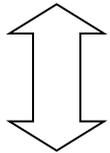
(dns1.networkutopia.com, 212.212.212.1, A)

dns1.networkutopia.com

(server di competenza per networkutopia.com)

(www.networkutopia.com, 1111.1111.1111.1111, A)

(networkutopia.com, 2222.2222.2222.2222, MX)



www.networkutopia.com



- ❑ 2.1 Principi delle applicazioni di rete
- ❑ 2.2 Web e HTTP
- ❑ 2.3 FTP
- ❑ 2.4 Posta Elettronica
 - ❑ SMTP, POP3, IMAP
- ❑ 2.5 DNS
- ❑ **Protocolli per applicazioni multimediali**



- Multimediale è un servizio che usa più di un “senso” (vista, udito, ... tatto, gusto, olfatto? ... dati??) per la comunicazione
- In pratica si parla di “multimedia” tutte le volte che la comunicazione non è strettamente “dati” (qualsiasi cosa voglia dire)
- Spesso le comunicazioni multimediali sono anche conversazionali
- In pratica la comunicazione multimediale su TCP/IP vuole dire telefonia e video-conferenza / telelavoro
- Tutto il resto viene classificato come “dati” e di recente “streaming”



- Nell'ottica IP la telefonia costituisce un “normale” servizio applicativo
- Realizzato con protocolli applicativi (end-to-end)
- La connettività è tramite protocollo IP (fornisce il servizio end-to-end)
- L'intelligenza è ai bordi della rete (nei terminali) e non nascosta nella rete
- Protocolli e procedure piccoli e mono-funzionali
 - Modularità per realizzare applicazioni differenti



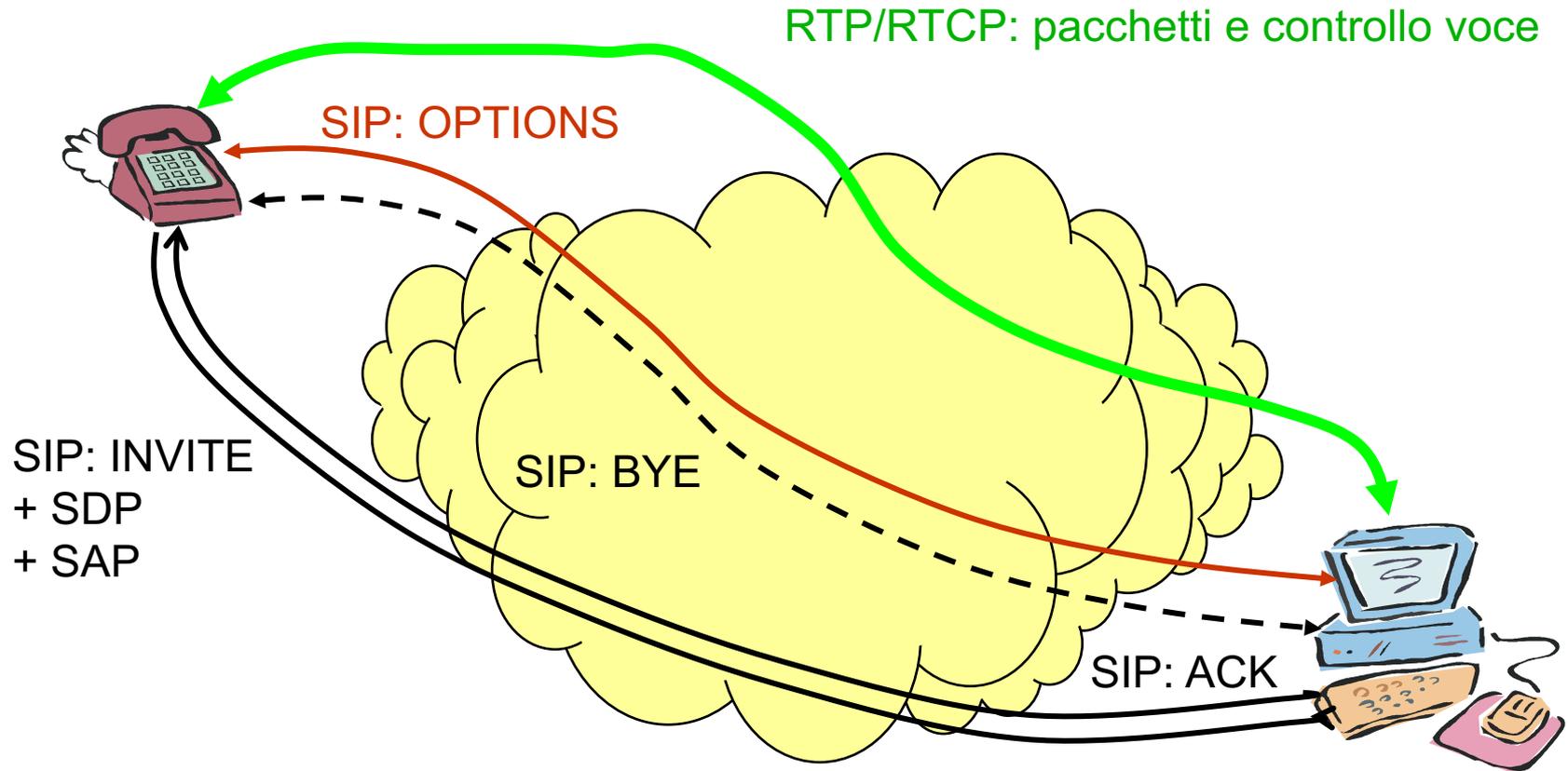
- **Concetti principali:**
 - Sessione:
 - Chiamata con 2 (o più) utenti
 - Diversi “media stream”
 - Generati da diversi utenti
 - Audio + video
 - Segnalazione “out-of-band”
 - Separazione della gestione di una sessione da trasferimento media
- **Protocolli:**
 - Segnalazione:
 - SIP: Session Initiation Protocol (RFC 2543, marzo 1999)
 - SDP: Session Description Protocol
 - SAP: Session Announcement Protocol
 - Trasporto voce:
 - RTP/RTCP



- User Agent (o end system)
 - Client: Invia le richieste SIP
 - Server: Soddisfa le richieste di chiamata entranti
- SIP Redirect Server
 - Redirige una chiamata su un altro server
- SIP Proxy Server
 - Invia la richiesta ad un altro server
- SIP Registrar
 - accetta la registrazione degli utenti su un server
 - mantiene la locazione logica dell'utente



- **Gli indirizzi sono URI (Universal Resource Identifier):**
 - sip:jdrosen@bell-labs.com:5067
 - sip:ann:passwd@lucent.com
- 6 metodi:
 - INVITE: Inizia o invita ad una conferenza
 - BYE: Termina la partecipazione ad una conferenza
 - CANCEL: Termina una ricerca
 - OPTIONS: Interroga un client sulle sue “capabilities”
 - ACK: Accetta la chiamata (invito)
 - REGISTER: Informa un SIP server sulla posizione di un utente





- La sintassi è ripresa da **HTTP:**

```
INVITE gerla@cs.ucla.edu SIP/2.0
From: locigno@disi.unitn.it (Renato Lo Cigno)
Subject: Next visit to L.A.
To: gerla@cs.ucla.edu (Mario Gerla)
Call-ID: 1999284605.56.86@
Content-type: application/sdp
CSeq: 4711
Content-Length: 187
```



- Sintassi testuale per descrivere sessioni multimediali unicast e multicast
- Caratteristiche base
 - Descrive i flussi Audio/Video che formano la sessione ed i relativi parametri
 - Contiene gli indirizzi di destinazione dei diversi stream
 - “Governa” i tempi di inizio e fine di ogni sessione
- È il “payload” di SIP



- Ha la funzione di “rendezvous point”, ovvero di locazione (virtuale) dove un dato chiamato è sempre (nello spazio e nel tempo) logicamente reperibile
- Ha funzioni di instradamento dell’applicazione, cioè definisce dove (a quale UA oppure proxy/redirect server) deve essere inoltrata una chiamata entrante
- Questa funzione è dinamica e programmabile
- Forking: Consente di “tentare” diverse terminazioni/destinazioni in parallelo o in sequenza (es. group calls)
- È in genere il punto in cui vengono svolte le operazioni di AAA



Request Method

INVITE sip:UserB@there.com SIP/2.0

Via: SIP/2.0/UDP here.com:5060
From: BigGuy <sip:UserA@here.com>
To: LittleGuy <sip:UserB@there.com>
Call-ID: 12345600@here.com
CSeq: 1 INVITE
Subject: Happy Christmas
Contact: BigGuy <sip:UserA@here.com>
Content-Type: application/sdp
Content-Length: 147

Message Header Fields

Response Status

SIP/2.0 200 OK

Via: SIP/2.0/UDP here.com:5060
From: BigGuy <sip:UserA@here.com>
To: LittleGuy <sip:UserB@there.com>;tag=65a35
Call-ID: 12345601@here.com
CSeq: 1 INVITE
Subject: Happy Christmas
Contact: LittleGuy <sip:UserB@there.com>
Content-Type: application/sdp
Content-Length: 134

v=0
o=UserA 2890844526 2890844526 IN IP4 here.com
s=Session SDP
c=IN IP4 100.101.102.103
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000

Payload

v=0
o=UserB 2890844527 2890844527 IN IP4 there.com
s=Session SDP
c=IN IP4 110.111.112.113
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000

“receive RTP G.711-encoded audio at
100.101.102.103:49172”