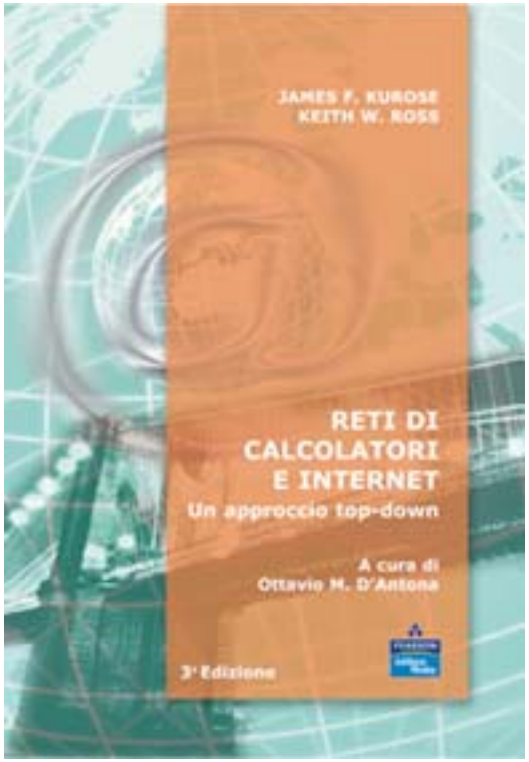
Capitolo 2
Livello di applicazione



Reti di calcolatori e Internet: Un approccio top-down

3ª edizione
Jim Kurose, Keith Ross
Pearson Education Italia
©2005

Nota per l'utilizzo:

Abbiamo preparato queste slide con l'intenzione di renderle disponibili a tutti (professori, studenti, lettori). Sono in formato PowerPoint in modo che voi possiate aggiungere e cancellare slide (compresa questa) o modificarne il contenuto in base alle vostre esigenze.
Come potete facilmente immaginare, da parte nostra abbiamo fatto *un sacco* di lavoro. In cambio, vi chiediamo solo di rispettare le seguenti condizioni:

- se utilizzate queste slide (ad esempio, in aula) in una forma sostanzialmente inalterata, fate riferimento alla fonte (dopo tutto, ci piacerebbe che la gente usasse il nostro libro!)
- se rendete disponibili queste slide in una forma sostanzialmente inalterata su un sito web, indicate che si tratta di una adattamento (o che sono identiche) delle nostre slide, e inserite la nota relativa al copyright.

Thanks and enjoy! JFK/KWR

Capitolo 2: Livello di applicazione

- 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
- 2.6 Condivisione di file P2P

Capitolo 2: Livello di applicazione

Obiettivi:

- Fornire i concetti base e gli aspetti implementativi dei protocolli delle applicazioni di rete
  - modelli di servizio del livello di trasporto
  - paradigma client-server
  - paradigma peer-to-peer
- Apprendere informazioni sui protocolli esaminando quelli delle più diffuse applicazioni di rete
  - HTTP
  - FTP
  - SMTP / POP3 / IMAP
  - DNS
- Programmare le applicazioni di rete
  - socket API

Alcune diffuse applicazioni di rete

- Posta elettronica
- Web
- Messaggistica istantanea
- Autenticazione in un calcolatore remoto (Telnet e SSH)
- Condivisione di file P2P
- Giochi multiutente via rete
- Streaming di video-clip memorizzati
- Telefonia via Internet
- Videoconferenza in tempo reale

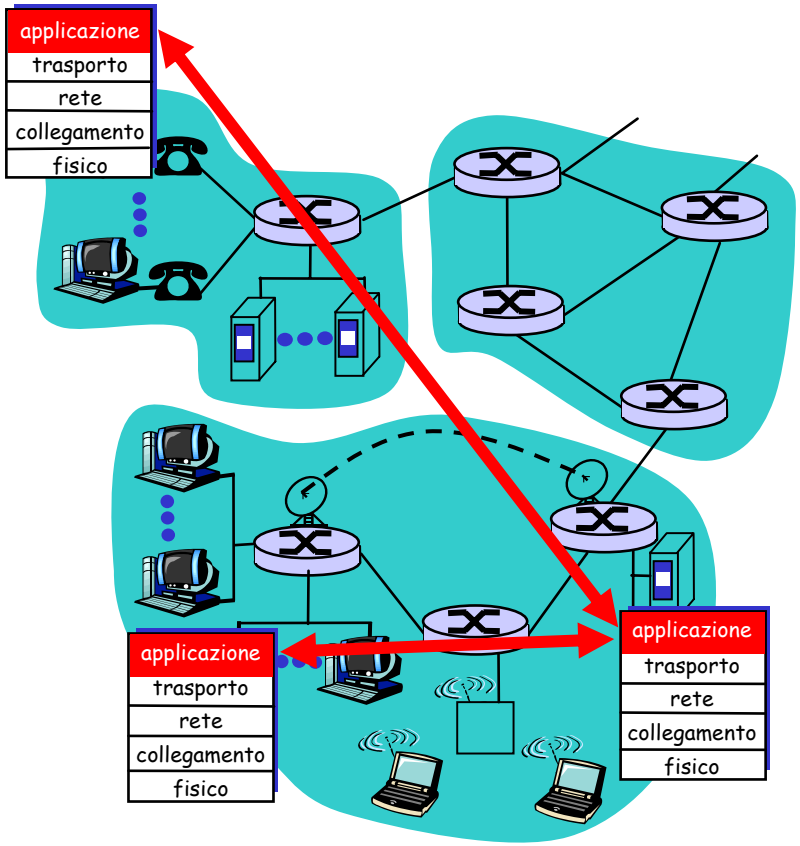
Creare un'applicazione di rete

Scrivere programmi che

- girano su sistemi terminali diversi
- comunicano attraverso la rete
- Ad es. il Web: il software di un server Web comunica con il software di un browser

software in grado di funzionare su più macchine

- non occorre predisporre programmi per i dispositivi del nucleo della rete, quali router o commutatori Ethernet



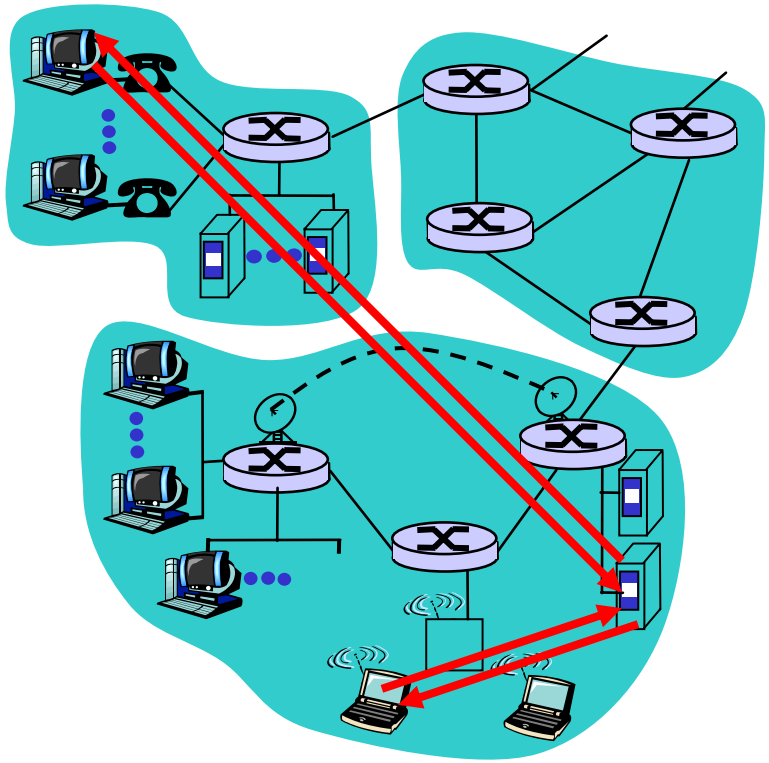
Capitolo 2: Livello di applicazione

- 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
- 2.6 Condivisione di file P2P

Architetture delle applicazioni di rete

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

Architettura client-server



server:

- host sempre attivo
- indirizzo IP fisso
- server farm per creare un potente server virtuale

client:

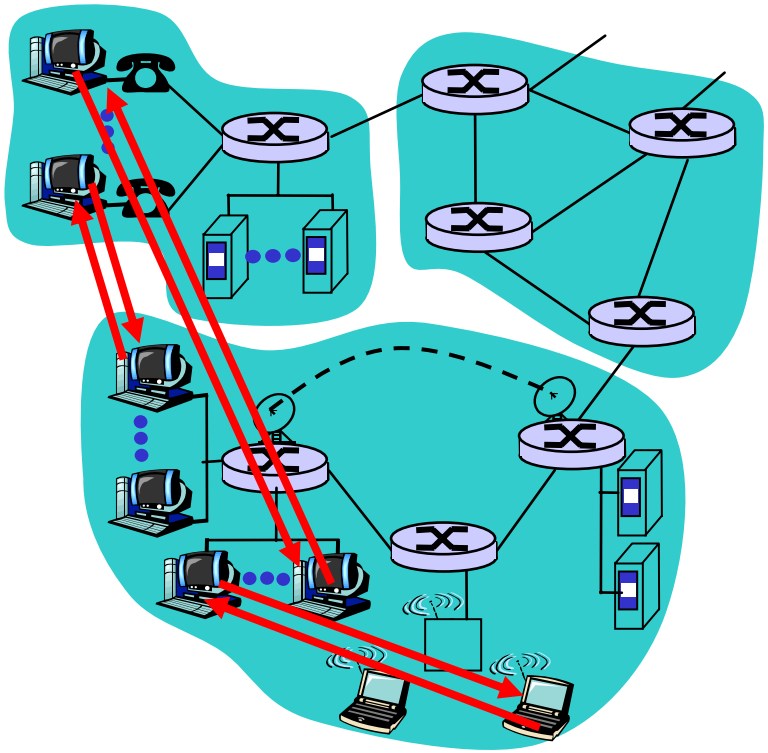
- comunica con il server
- può contattare il server in qualunque momento
- può avere indirizzi IP dinamici
- non comunica direttamente con gli altri client

Architettura P2P pura

- 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
- Un esempio: Gnutella

  Facilmente scalabile

  Difficile da gestire



Ibridi (client-server e P2P)

Napster

- Scambio di file secondo la logica P2P
- Ricerca di file centralizzata:
  - i peer registrano il loro contenuto presso un server centrale
  - i peer chiedono allo stesso server centrale di localizzare il contenuto

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

Processi comunicanti

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

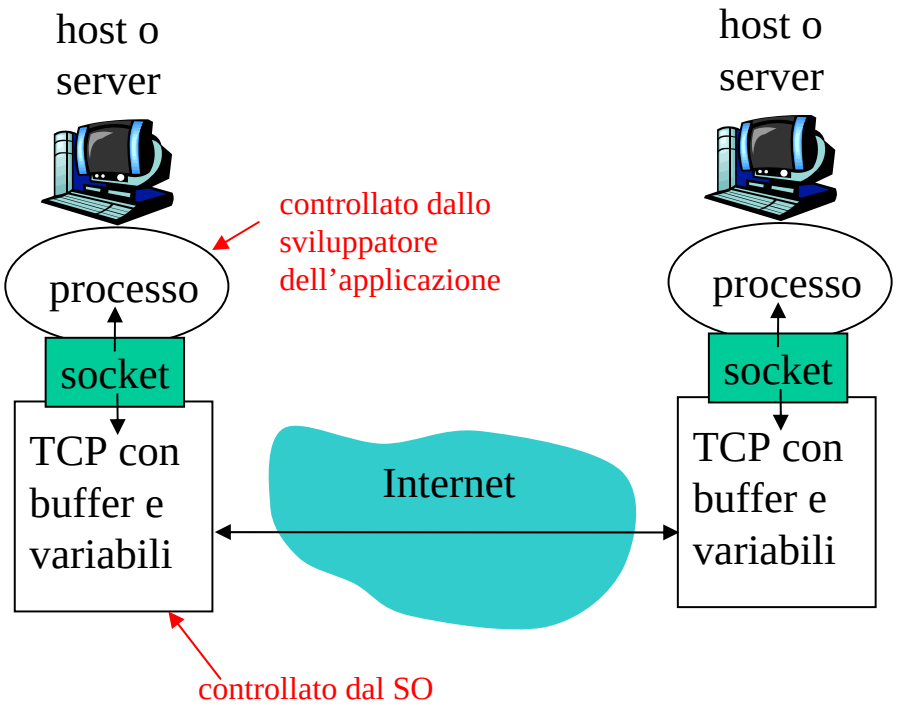
Processo client: processo che dà inizio alla comunicazione

Processo server : processo che attende di essere contattato

- Nota: le applicazioni con architetture P2P hanno processi client e processi server

Socket

- un processo invia/riceve messaggi a/da la sua socket
- una socket è analoga a una porta
  - un processo che vuole inviare un messaggio, lo fa uscire dalla propria "porta" (socket)
  - il processo presuppone l'esistenza di un'infrastruttura esterna che trasporterà il messaggio attraverso la rete fino alla "porta" del processo di destinazione



- API: (1) scelta del protocollo di trasporto; (2) capacità di determinare alcuni parametri (approfondiremo questo aspetto più avanti)

Processi di indirizzamento

- 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 A ha un indirizzo IP univoco a 32 bit
- D: 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
- Approfondiremo questi argomenti più avanti

Protocollo a livello di applicazione

- 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
- Consente l'interoperabilità
- Ad esempio, HTTP, SMTP

Protocolli proprietari:

- Ad esempio, KaZaA

Quale 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

Ampiezza di banda

- 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

Requisiti del servizio di trasporto di alcune applicazioni comuni

Applicazione	Tolleranza alla perdita di dati	Ampiezza di banda	Sensibilità al tempo
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	Sì	Fino a pochi Kbps	Sì, centinaia di ms
Messaggistica istantanea	No	Variabile	Sì e no

Servizi dei protocolli di trasporto Internet

Servizio di TCP:

- *orientato alla connessione:* è richiesto un setup fra i processi client e server
- *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 le rete è sovraccaricata
- *non offre:* temporizzazione, ampiezza di banda minima

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

D: Perché preoccuparsi? Perché esiste UDP?

Applicazioni Internet: protocollo a livello applicazione e protocollo di trasporto

Applicazione	Protocollo a livello applicazione	Protocollo di trasporto sottostante
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	Proprietario (ad esempio, RealNetworks)	TCP o UDP
Telefonia Internet	Proprietario (ad esempio, Vonage, Dialpad)	Tipicamente UDP

Capitolo 2: Livello di applicazione

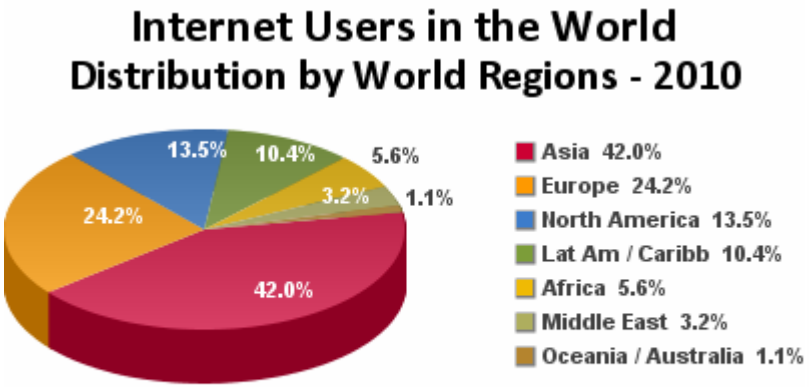
- 2.1 Principi delle applicazioni di rete
- 2.2 Web e HTTP
- 2.3 FTP
- 2.4 Posta elettronica
  - SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 Condivisione di file P2P

World Wide Web

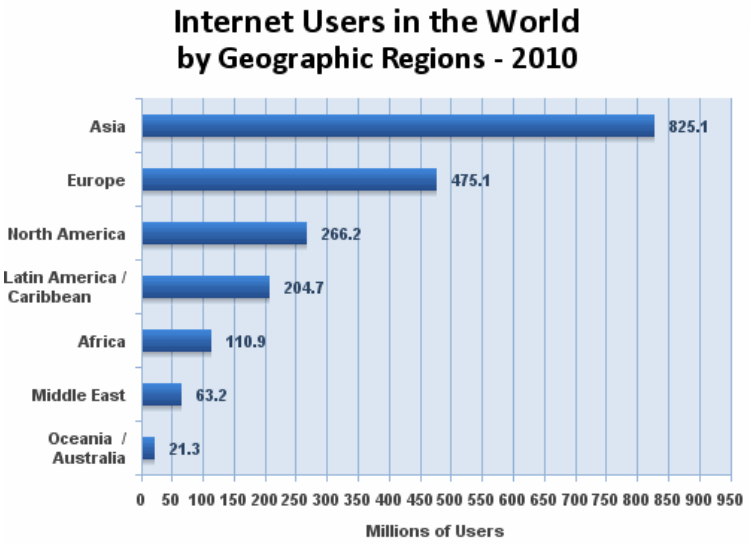
- Marzo 1989: nasce l'dea al CERN
- Dicembre 1991: presentazione del primo prototipo
- Febbraio 1993: rilascio del primo browser grafico (Mosaic)
- 1994: nasce Netscape Corporation
- 1995: Netscape va in borsa
- 1998: Netscape viene acquisita da America Online

- 1994: CERN e MIT creano il World Wide Web Consortium (W3C)

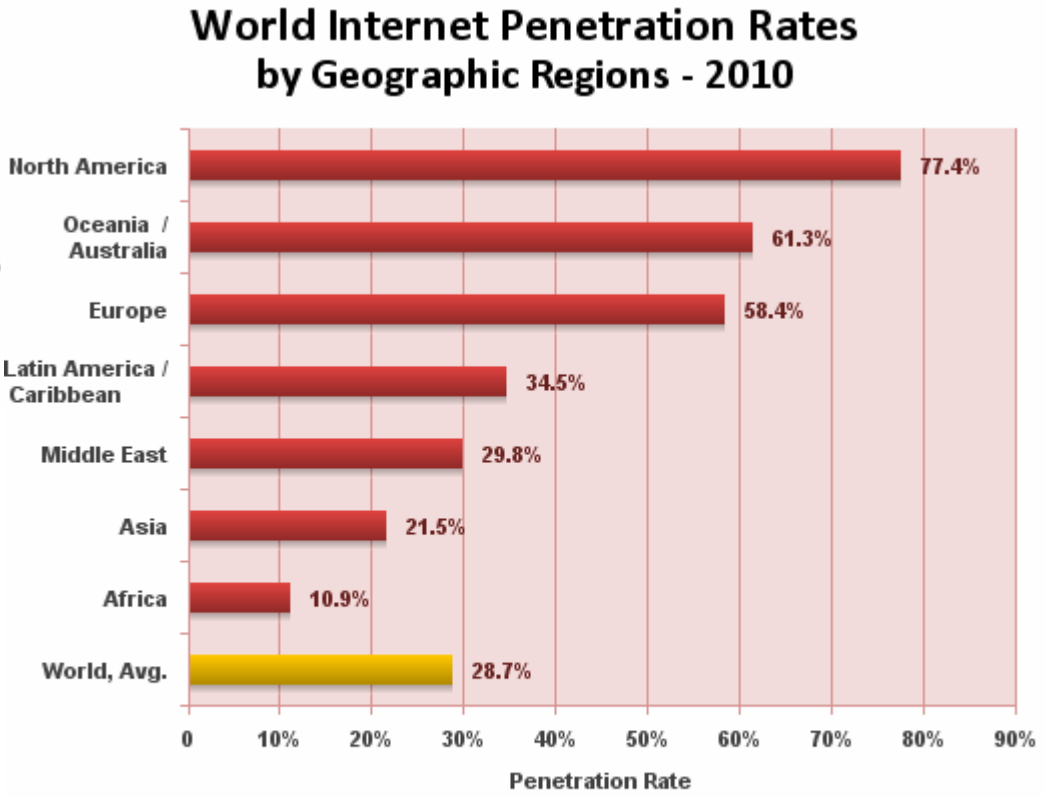
The Internet Big Picture



Source: Internet World Stats - www.internetworldstats.com/stats.htm
Basis: 1,966,514,816 Internet users on June 30, 2010

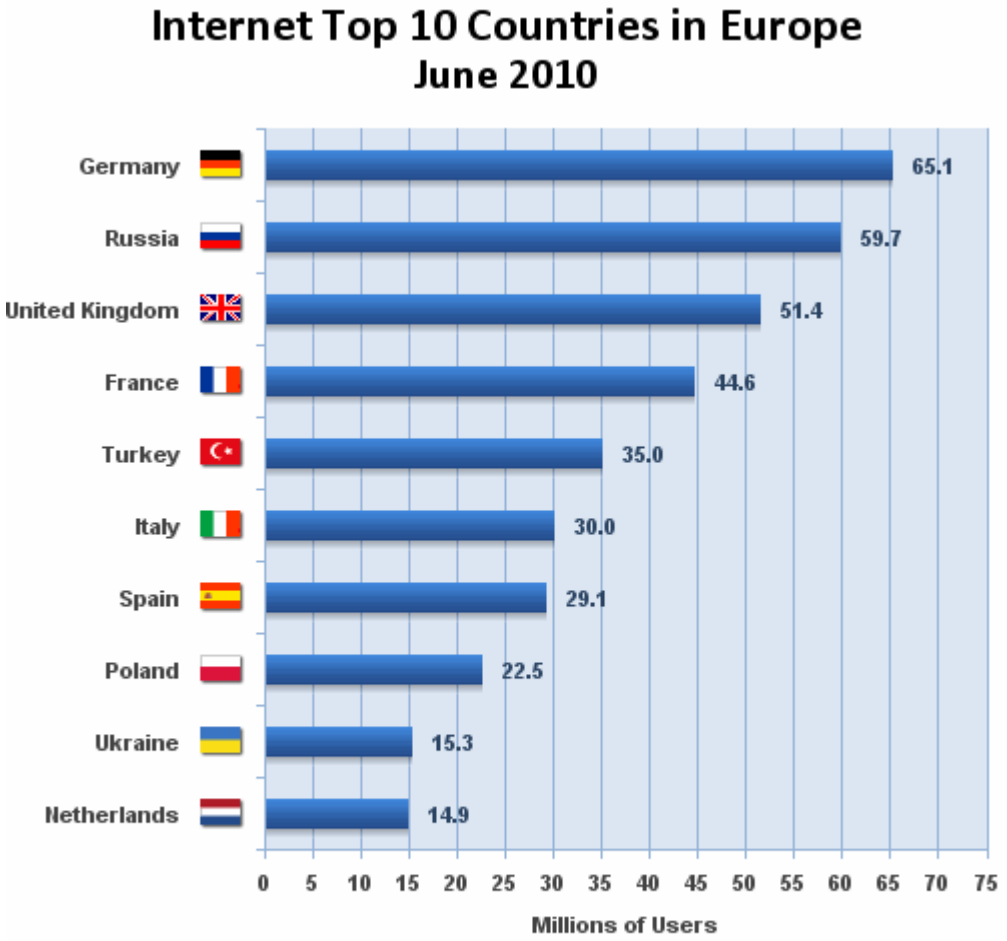


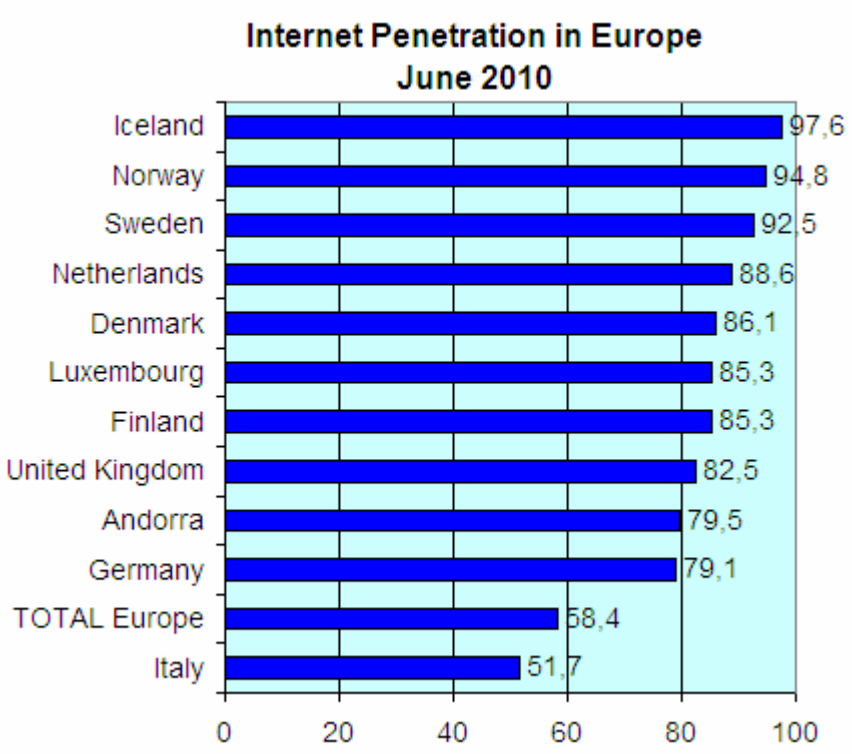
Source: Internet World Stats - www.internetworldstats.com/stats.htm
Estimated Internet users are 1,966,514,816 on June 31, 2010



Source: Internet World Stats - www.internetworldststs.com/stats.htm
Penetration Rates are based on a world population of 6,845,609,960
and 1,966,514,816 estimated Internet users on June 30, 2010.

Internet in Europa





Source: Internet World Stats - www.internetworldstats.com/stats4.htm
Basis: 475,069,448 estimated Internet Users in Europe on June 30, 2010

World Wide Web: Definizione

- Il www è l'insieme dei documenti ipertestuali (detti hypertext documents o web pages) raggiungibili in Internet.
- Il **programma** utilizzato per la lettura/visualizzazione di questi documenti è il "Web Browser"
- Il contenuto delle pagine web è formato da diversi elementi: immagini, testo, video clip, audio clip, link ad altri documenti etc...)
- I **linguaggi** utilizzati per la definizione delle pagine web sono l'HTML e le sue evoluzioni XML, e linguaggi di programmazione specifici (javaScript, php etc...)
- La pagine web sono identificate da un indirizzo detto "URI" Uniform Resource Identificator
- Le pagine web si trasferiscono attraverso la rete tramite un **protocollo** ad hoc HTTP/HTTPS (Hypertext transfer protocol).

WWW Consortium (W3C)

- Il W3C (www.w3c.org) Consorzio comprendente circa 300 membri tra aziende informatiche (Microsoft, IBM, Sun) telefoniche, università e istituzioni per la ricerca, associazioni come Mozilla Foundation etc...)

- Ha lo scopo di promuovere e sviluppare linee guida e standard per il world wide web, coinvolto anche nel training, pubblicazione di tutorial, sviluppo di software e di discussioni sul web in generale.

- Obiettivo del W3C è la "Web interoperability" ovvero la compatibilità tra le varie tecnologie usate nel web. A cura del W3C sono state proposte, discusse, definite ed ufficializzate diverse specifiche tecniche tra cui il protocollo http, linguaggi come html, xml, css, le linee guida per l'accessibilità.

Web e HTTP

Terminologia

- 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

Uniform Resource Locator

- URL è una sequenza di caratteri utilizzati per identificare una risorsa e permettere interazioni con essa.

Nell'ambito del world wide web l'URL identifica

- La locazione della risorsa
- Il protocollo di comunicazione (http, https, ftp, ftps, file, mailto, etc...)
- Eventualmente la porta di comunicazione (ad es. porta 80 per http)
- L'elemento specifico all'interno della risorsa

Uniform Resource Locator

Esempi di URL

www.someschool.edu/someDept/pic.gif

nome dell'host nome del percorso

file:///C:/MyDocuments/paper.pdf

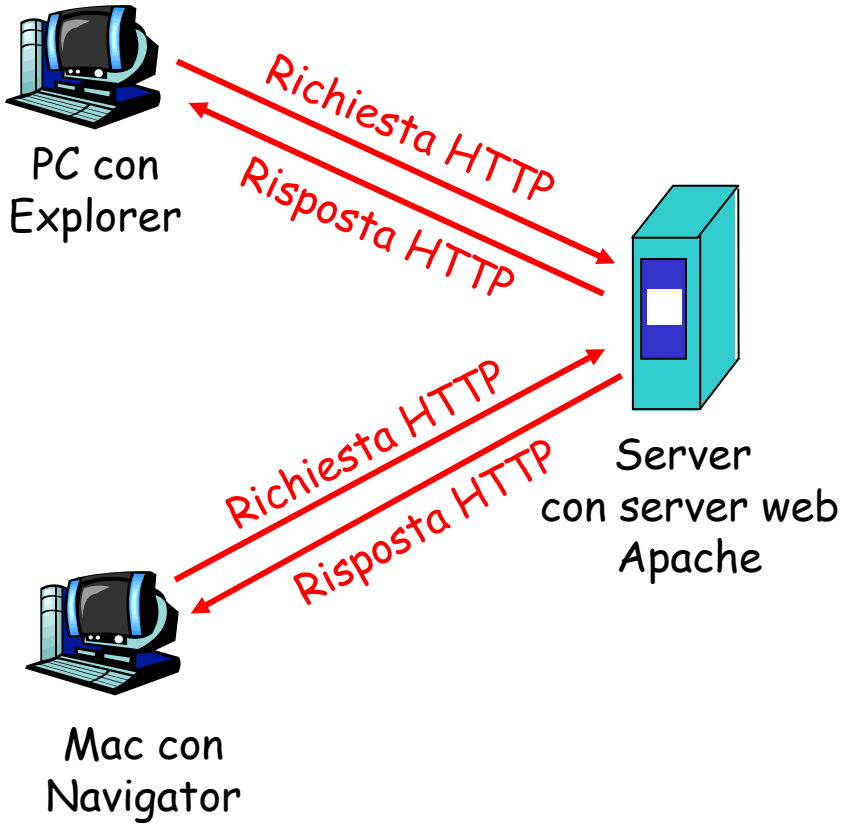
http://www.unipv.eu/on-line/Home/Navigaper/Studenti.html

https://esami.unipv.it:8443/uniwex/

Panoramica su HTTP

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
- HTTP 1.0: RFC 1945
- HTTP 1.1: RFC 2068



Panoramica su HTTP (continua)

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 HTTP

Connessioni non persistenti

- Almeno un oggetto viene trasmesso su una connessione TCP
- HTTP/1.0 usa connessioni non persistenti

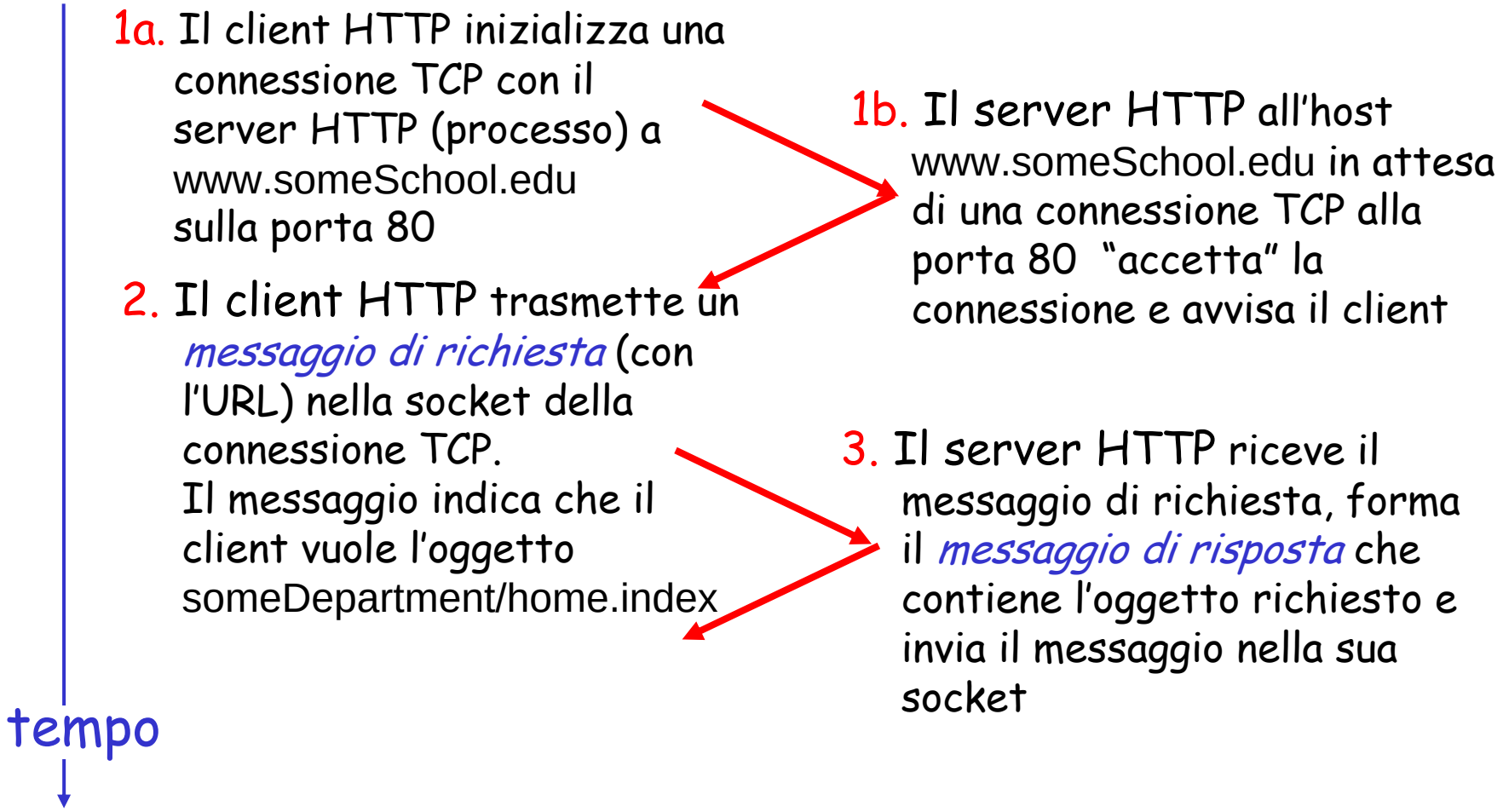
Connessioni persistenti

- Più oggetti possono essere trasmessi su una singola connessione TCP tra client e server
- HTTP/1.1 usa connessioni persistenti nella modalità di default

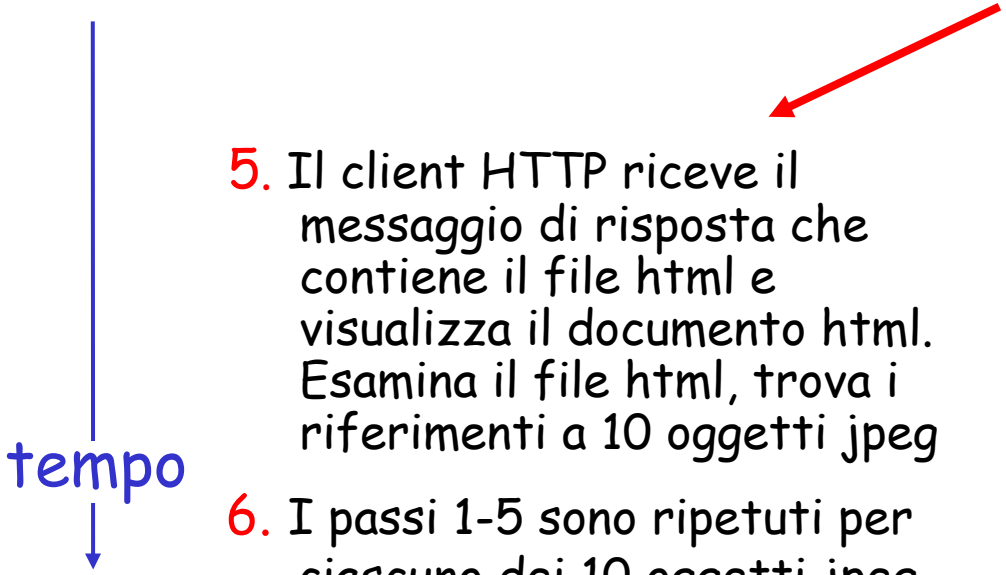
Connessioni non persistenti

Supponiamo che l'utente immetta l'URL
`www.someSchool.edu/someDepartment/home.index`

(contiene testo, riferimenti a 10 immagini jpeg)



Connessioni non persistenti (continua)



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

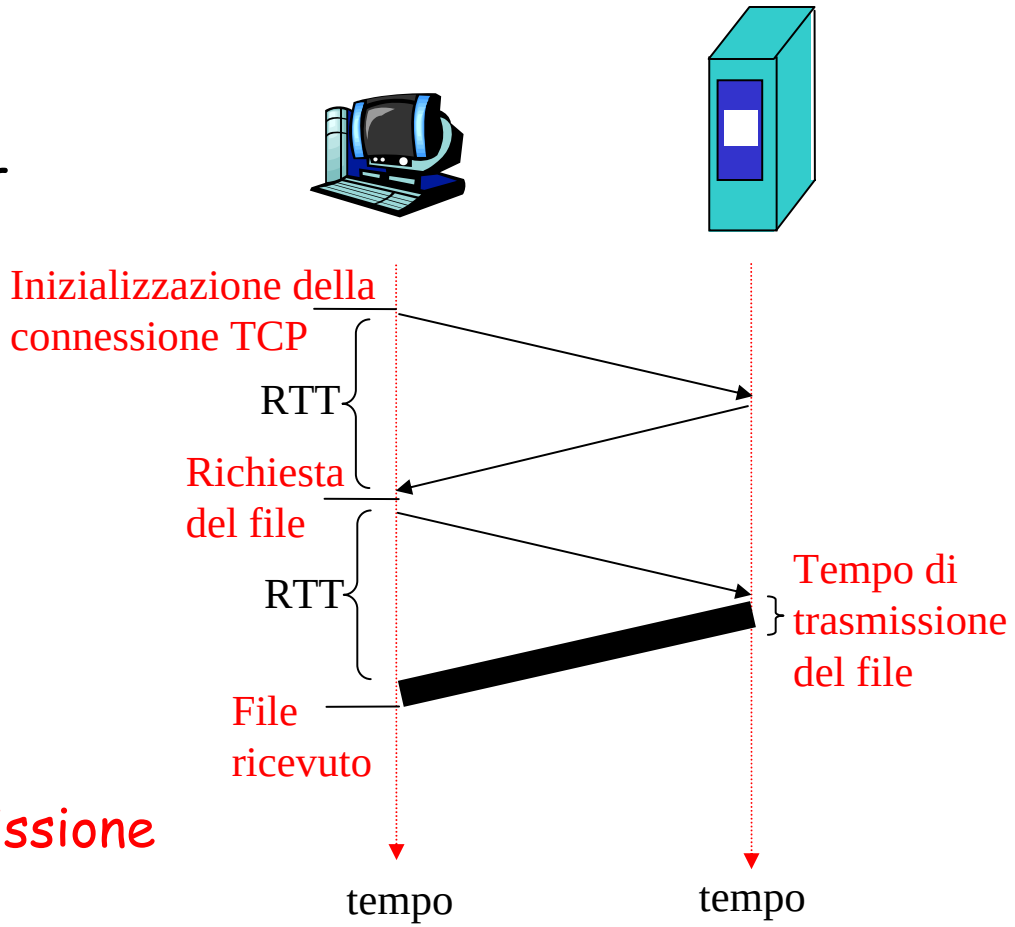
Schema del tempo di risposta

Definizione di RTT: tempo impiegato da un piccolo pacchetto per andare dal client al server e 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 persistenti

Svantaggi delle connessioni non persistenti:

- richiede 2 RTT per oggetto
- overhead del sistema operativo per *ogni* connessione TCP
- i browser spesso aprono connessioni TCP parallele per caricare gli oggetti referenziati

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

Connessione persistente ***senza*** pipelining:

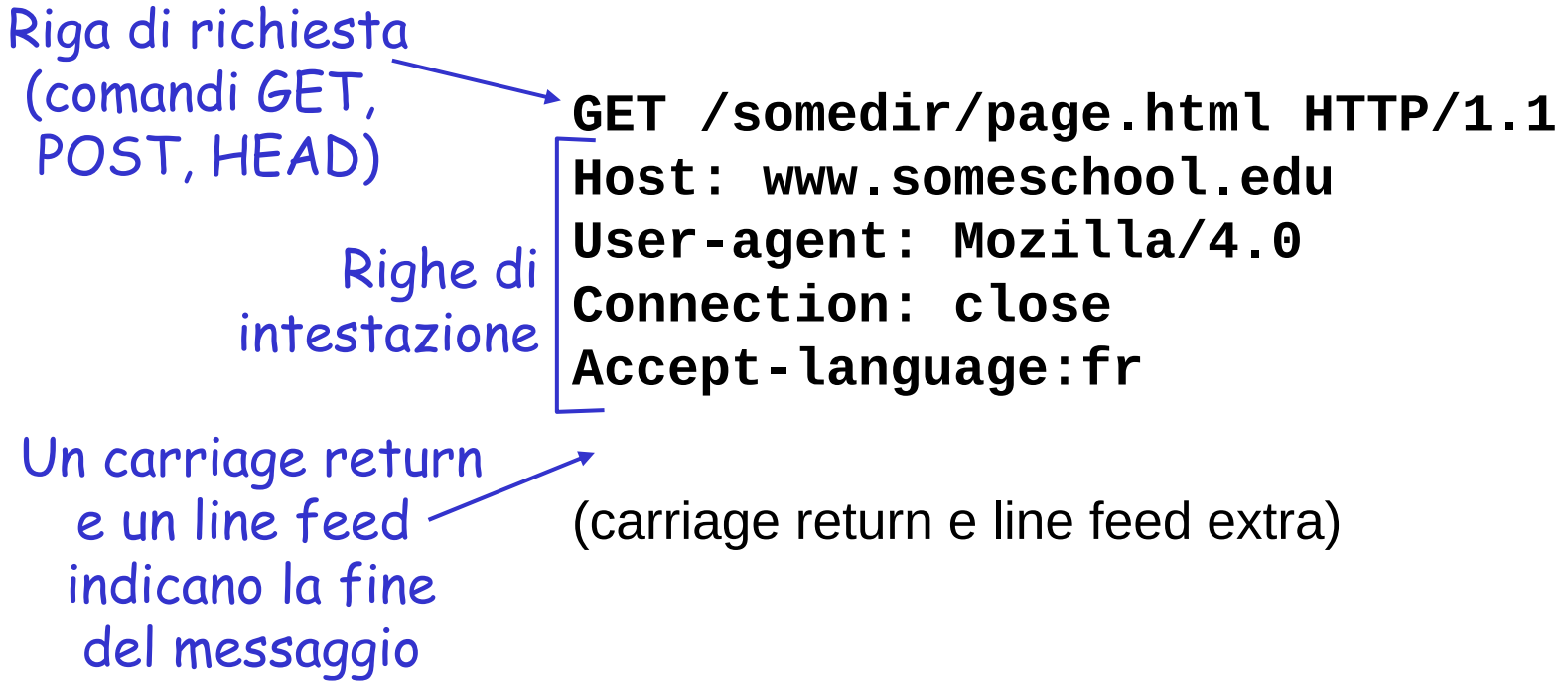
- il client invia una nuova richiesta solo quando ha ricevuto la risposta precedente
- un RTT per ogni oggetto referenziato

Connessione persistente ***con*** pipelining:

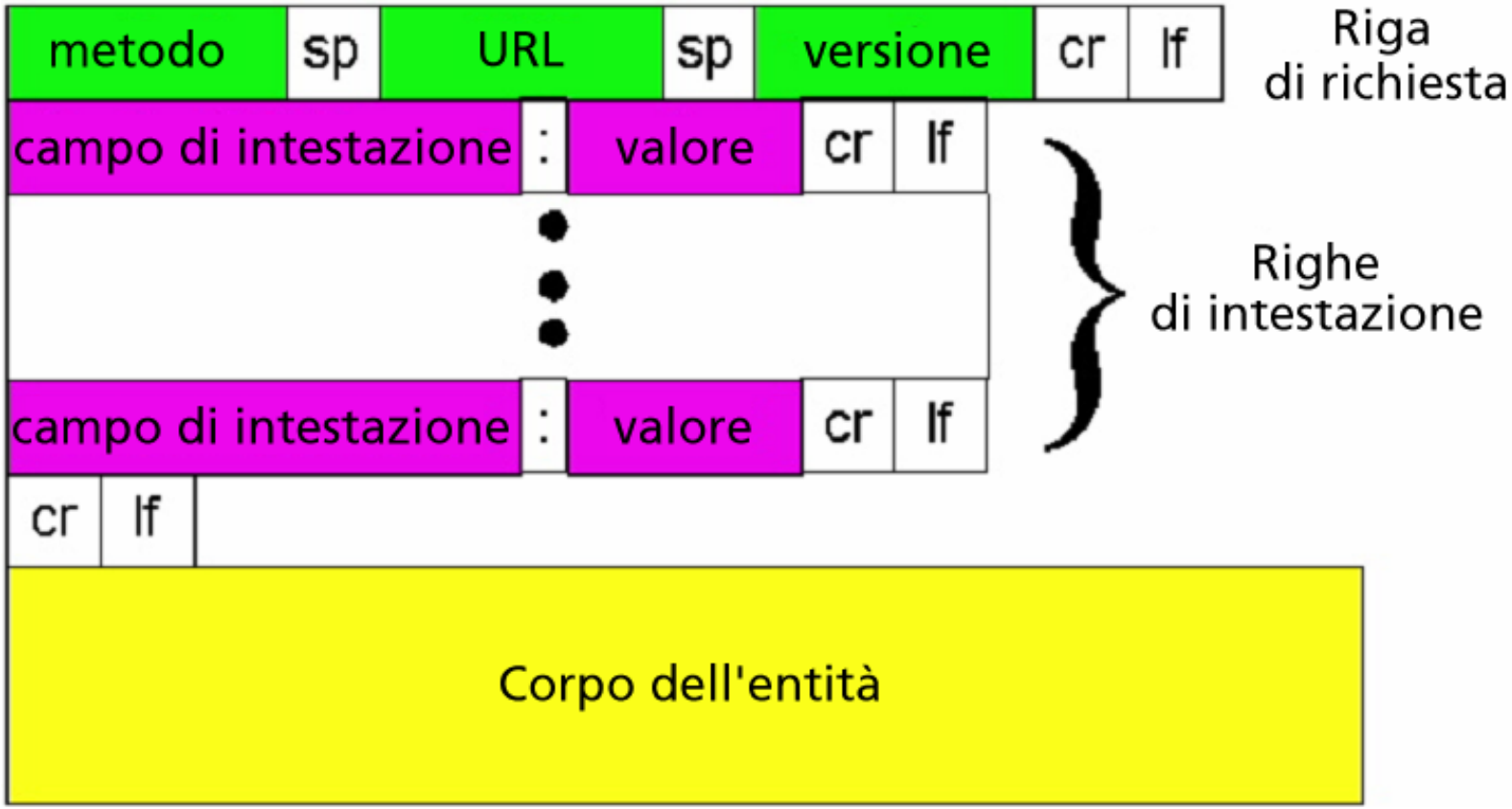
- è la modalità di default in HTTP/1.1
- il client invia le richieste non appena incontra un oggetto referenziato
- un solo RTT per tutti gli oggetti referenziati

Messaggi HTTP

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



Messaggio di richiesta HTTP: formato generale



Upload dell'input di un form

Metodo Post:

- La pagina web spesso include un form per l'input dell'utente
- L'input arriva al server nel corpo dell'entità

Metodo URL:

- Usa il metodo GET
- L'input arriva al server nel campo URL della riga di richiesta:

```
www.somesite.com/animalsearch?monkeys&banana
```

Tipi di metodi

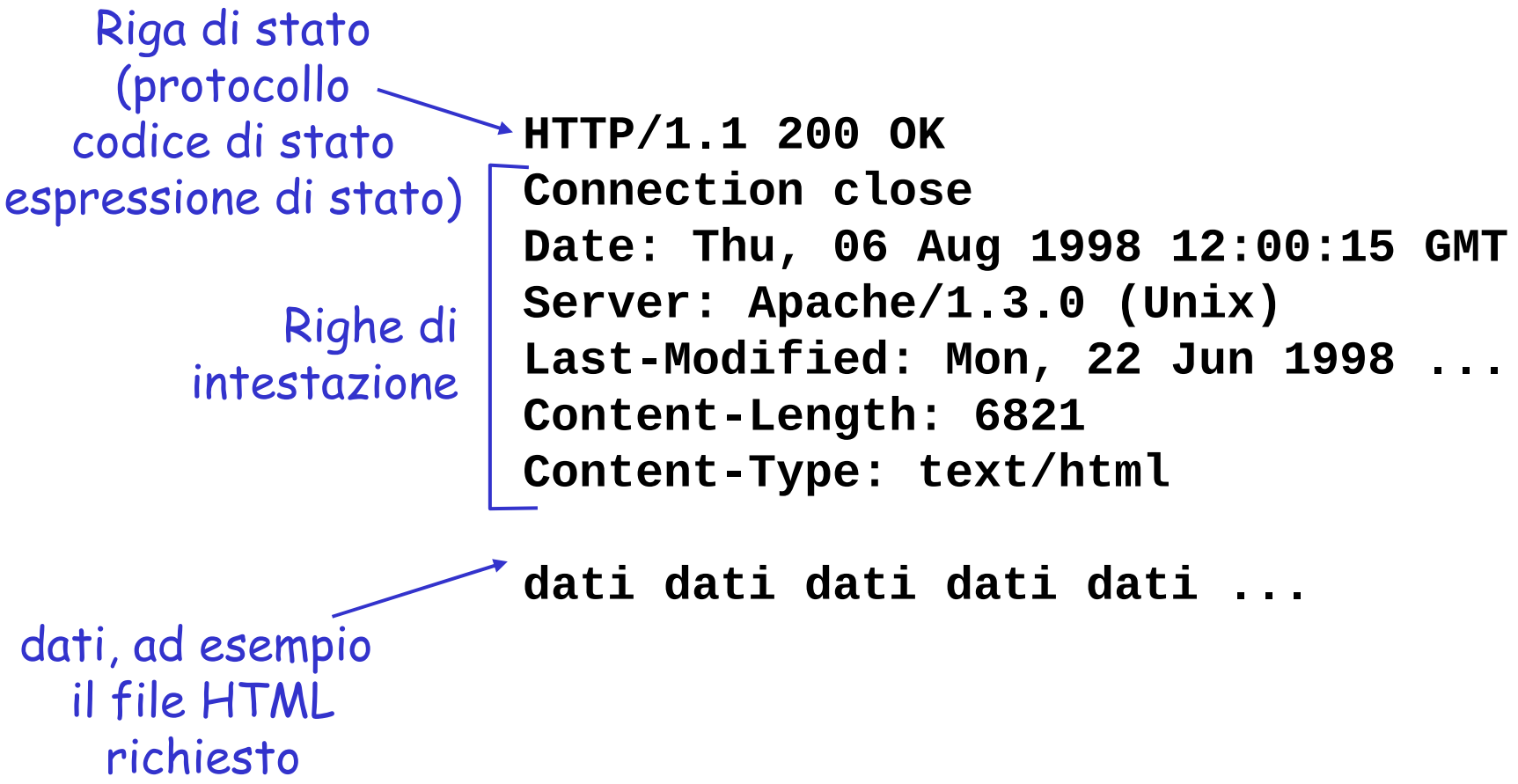
HTTP/1.0

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

HTTP/1.1

- GET, POST, HEAD
- PUT
  - include il file nel corpo dell'entità e lo invia al percorso specificato nel campo URL
- DELETE
  - cancella il file specificato nel campo URL

Messaggio di risposta HTTP



Codici di stato della risposta HTTP

Nella prima riga nel messaggio di risposta server->client.

Alcuni codici di stato e relative espressioni:

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

Provate HTTP (lato client)

1. Collegatevi via Telnet al vostro server web preferito:

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

Apre 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!

Interazione utente-server: i cookie

Molti dei più importanti siti web usano i cookie

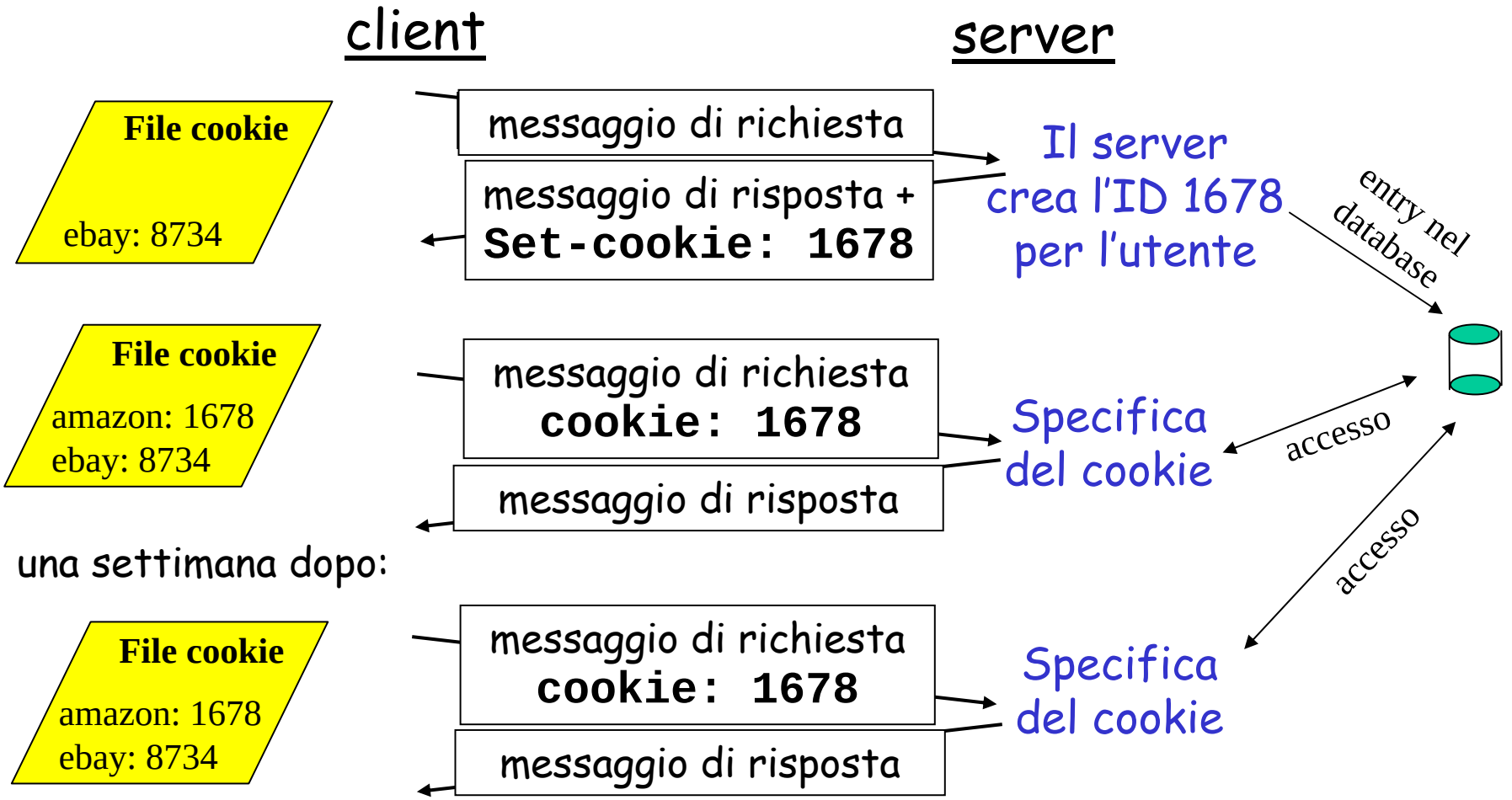
Quattro componenti:

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

Cookie (continua)



Cookie (continua)

Cosa possono contenere i cookie:

- autorizzazione
- carta per acquisti
- raccomandazioni
- stato della sessione dell'utente (e-mail)

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
- i motori di ricerca usano il reindirizzamento e i cookie per sapere ancora di più
- le agenzie pubblicitarie ottengono informazioni dai siti

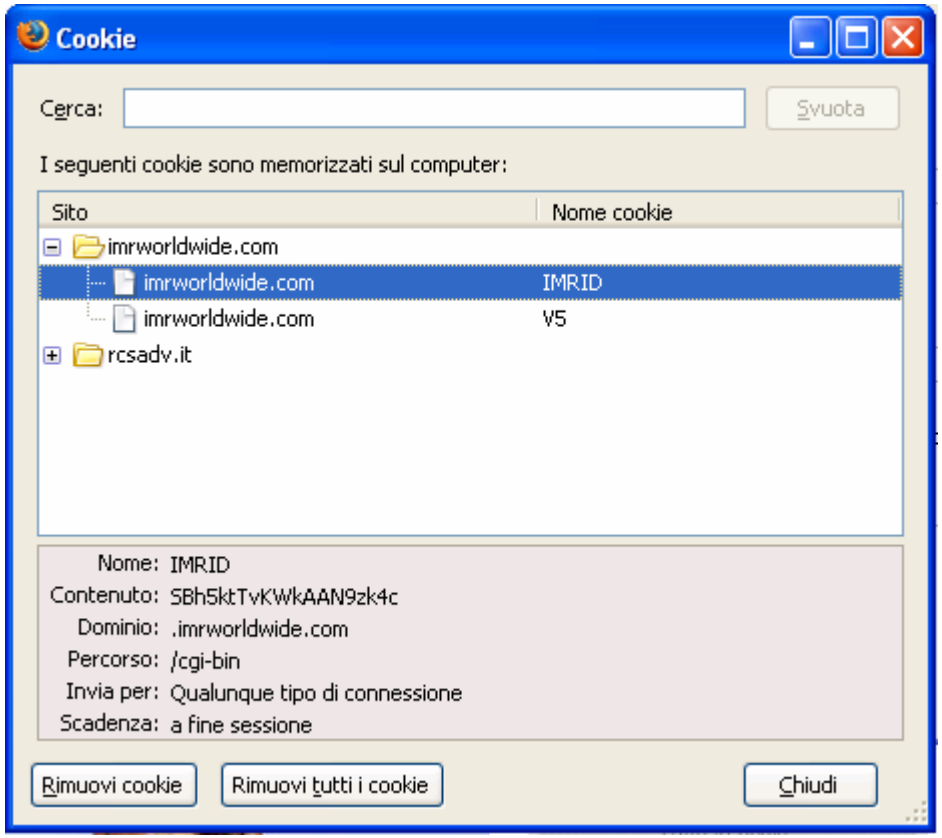
Cookies Proprietà

- I cookies permettono di monitorare l'interazione dell'utente con un determinato sito web e generare un profilo utente.
- I cookies sono inviati nei successivi collegamenti al server che li ha generati o a server che appartengono allo stesso dominio.
- Ciascun browser ha uno spazio su disco (cartella) all'interno del quale memorizza tutti i cookies.
- I cookies hanno durata variabile:
  - Sessione – cioè vengono cancellati quando si chiude il browser
  - Permanente – rimangono tra una sessione e l'altra fino alla scadenza

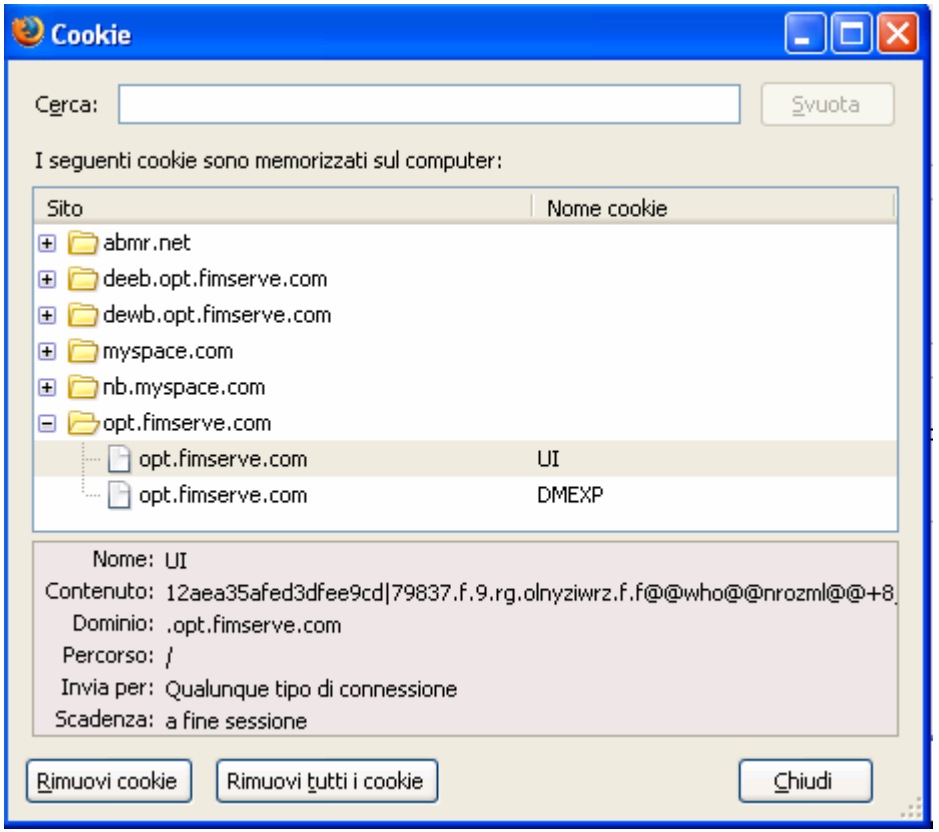
Cookies Proprietà

- I cookies sono file di testo che in genere hanno una dimensione limitata (4Kbite).
- I cookies non possono contenere ed eseguire codice.
- Essi contengono una serie di attributi (dati) alcuni opzionali tra cui:
  - Dominio di provenienza del cookie
  - Scadenza – validità del cookies
  - Modalità di accesso – ad es. HttpOnly rendo il cookie invisibile a javascript e altri linguaggi client side
  - Sicuro – se il cookie deve essere inviato in rete con protocolli https .
- I cookies identificano l'utente in base al browser utilizzato, indirizzo IP, ed ID dell'utente.

Cookies Esempi



Cookies installati dal sito del Corriere della Sera

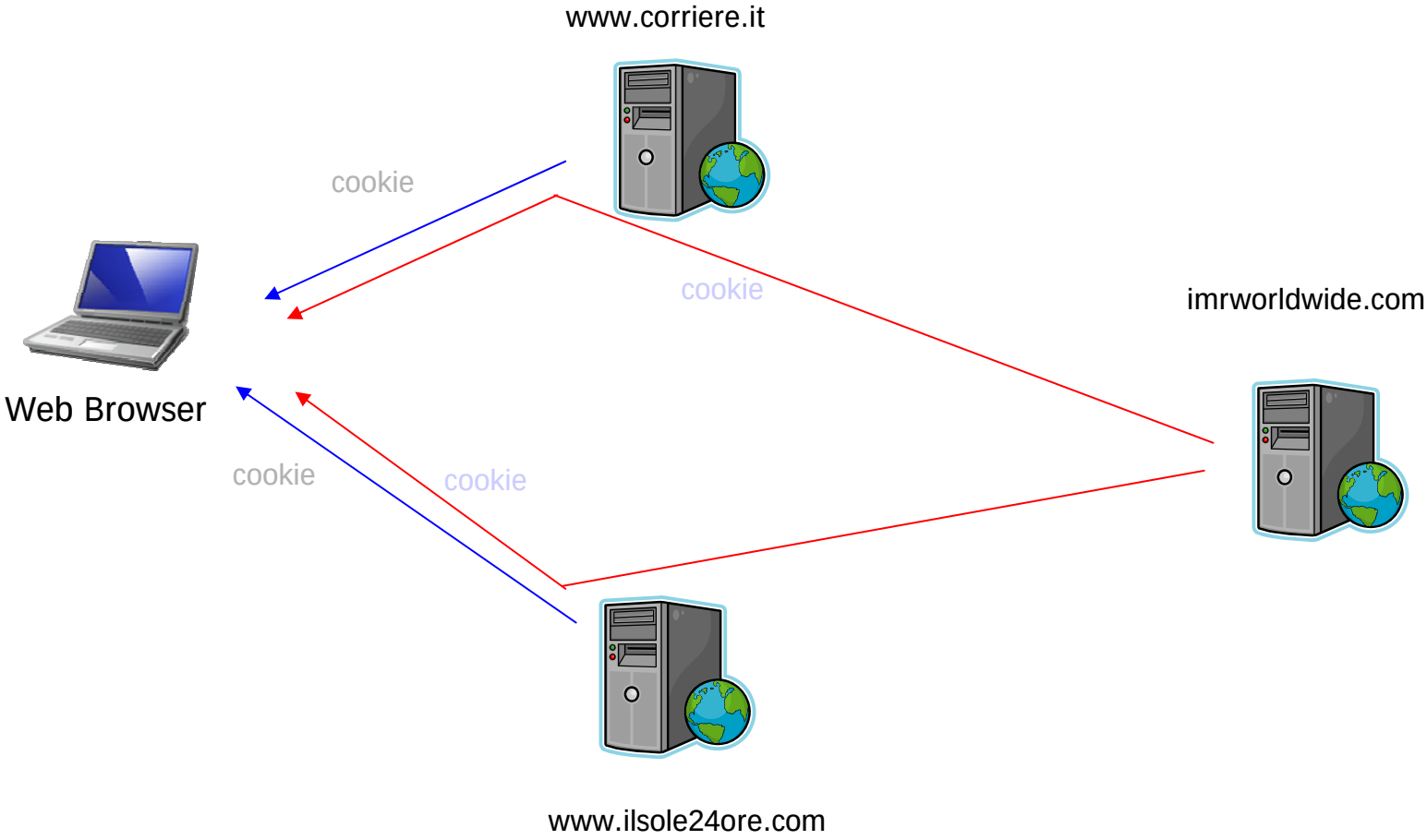


Cookies installati dal sito di MySpace

Third-party Cookies

- Negli esempi precedenti si vede come circa il 50% dei cookies installati dai vari siti non siano appartenenti al dominio del sito stesso ma di altri
- Questi ultimi sono i cosiddetti "third-party" cookies e sono impostati da domini di terze parti che hanno siglato accordi con il sito principale.
- Scopo dell'utilizzo di questa tipologia di cookies è la profilazione dell'utente. Le società che installano third-party cookies possono ricostruire i percorsi effettuati dall'utente nel web.
- La creazione del profilo utente è vista anche come una minaccia alla privacy dell'individuo e per questo diversi stati hanno regolamentato l'utilizzo dei cookies nei siti.

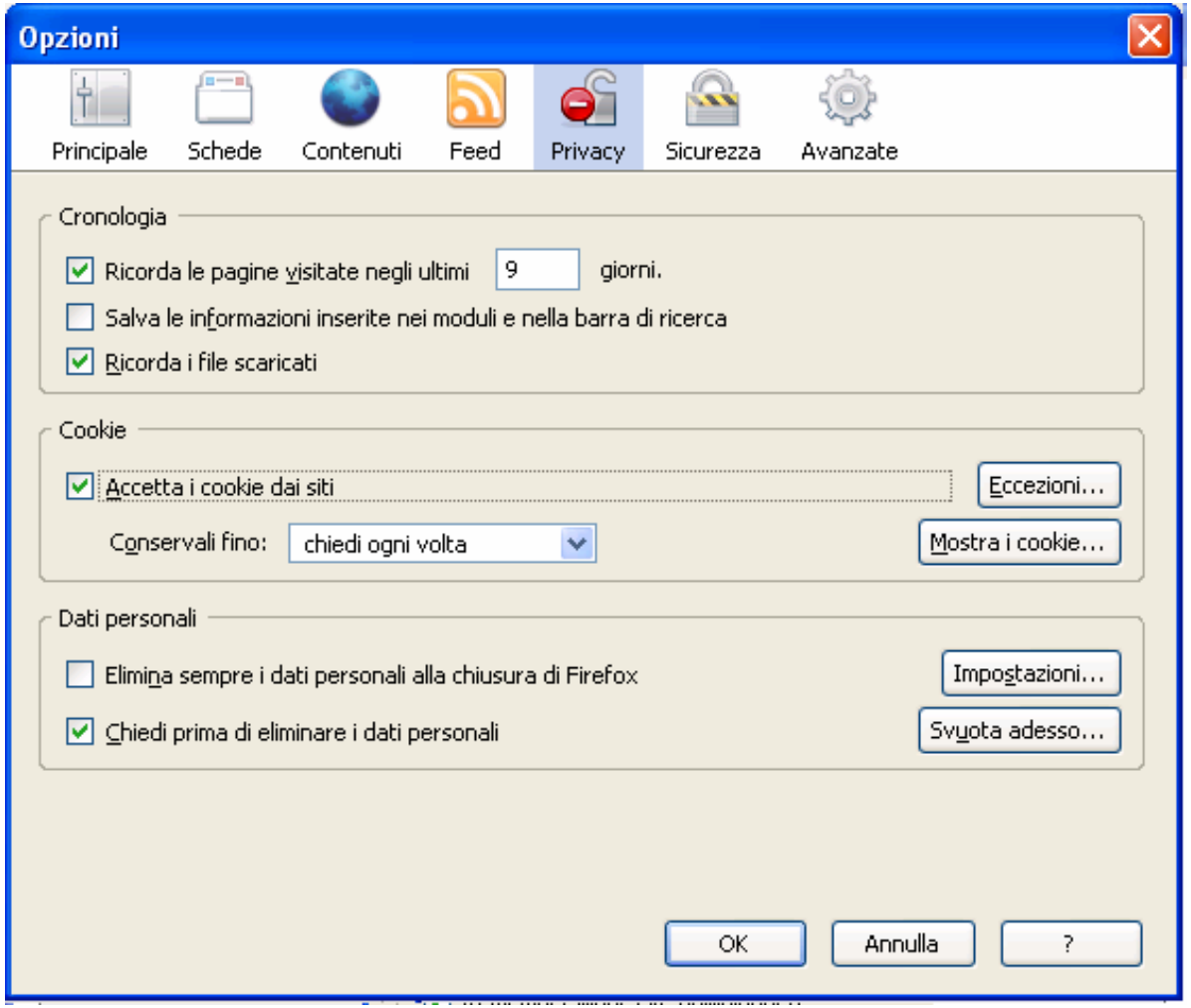
Cookies Schema di tracciamento



Cookies Configurazione

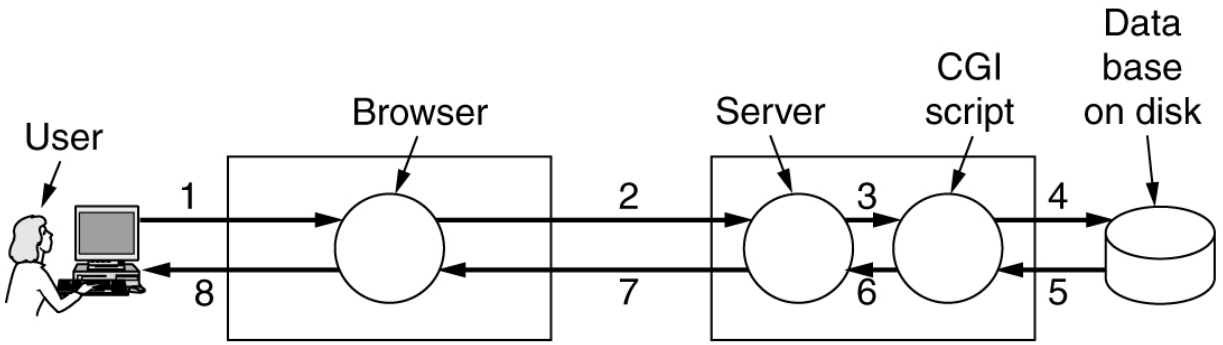
Gestione dei cookies

in Mozilla Firefox



Documenti Web Dinamici

- I passaggi per elaborare le informazioni di un form HTML.



1. User fills in form
2. Form sent back
3. Handed to CGI
4. CGI queries DB
5. Record found
6. CGI builds page
7. Page returned
8. Page displayed

Documenti Web Dinamici (2)

- Un esempio di pagina HTML con codice PHP.

```
<html>
<body>

<h2> This is what I know about you </h2>
<?php echo $HTTP_USER_AGENT ?>

</body>
</html>
```

Documenti Web Dinamici (3)

```
<html>
<head>
        <title> Una pagina contenente un form</title>
</head>
<body>
  <form action="action.php" method="post" >
        <p>Please enter your name: <input type="text" name="name"></p>
        <p>Please enter your age: <input type="text" name="age"></p>
        <input type="submit">
  </form>
</body>
</html>
```

Lo script action.php

```
<html>
<body>
<h1>Replay: </h1>
Hello <?php echo $name;  ?><br>
Prediction: next year you will be <?php echo $age + 1;  ?><br>
</body>
</html>
```

L'output dello script PHP
Valori di input "Barbara" e 24.

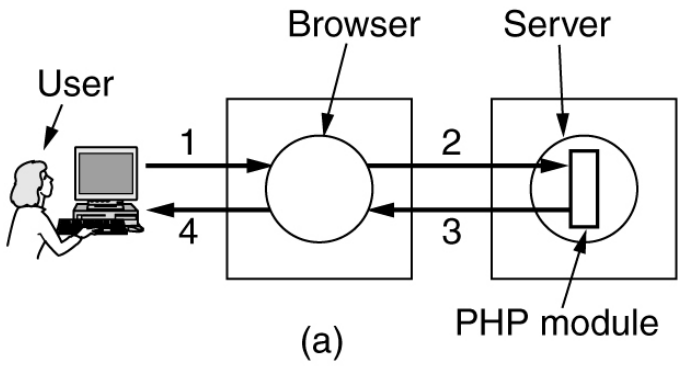
```
<html>
<body>
<h1>Replay: </h1>
Hello Barbara<br>
Prediction: next year you will be 25<br>
</body>
</html>
```

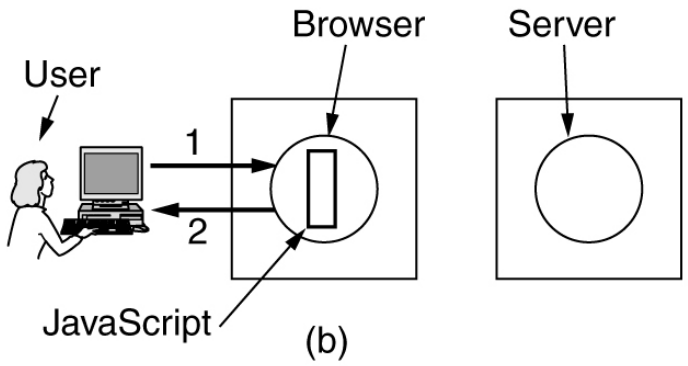
Generazione di pagine sul lato Client

```
<html>
<head>
        <title> Uso di JavaScript per il trattamento di un form </title>

        <script language="javascript" type="text/javascript">
          function response(myform)
          {
            var person = myform.name.value;
            var years = eval(myform.age.value) + 1;
            document.open();
            document.writeln("<html><head><title>Pagina generata</title></head><body>");
            document.writeln("Hello " + person + ".<br>");
            document.writeln("Prediction: next year you will be " + years + ".");
            document.writeln("</body></html>");
            document.close();
          }
        </script>
</head>

<body>
  <form>
        <p>Please enter your name: <input type="text" name="name"></p>
        <p>Please enter your age: <input type="text" name="age"></p>
        <input type="button" value="submit!" onClick="response(this.form)">
  </form>
</body>
</html>
```

Generazione di pagine sul lato Client (2)

(a) Server-side scripting con PHP.
(b) Client-side scripting con JavaScript.





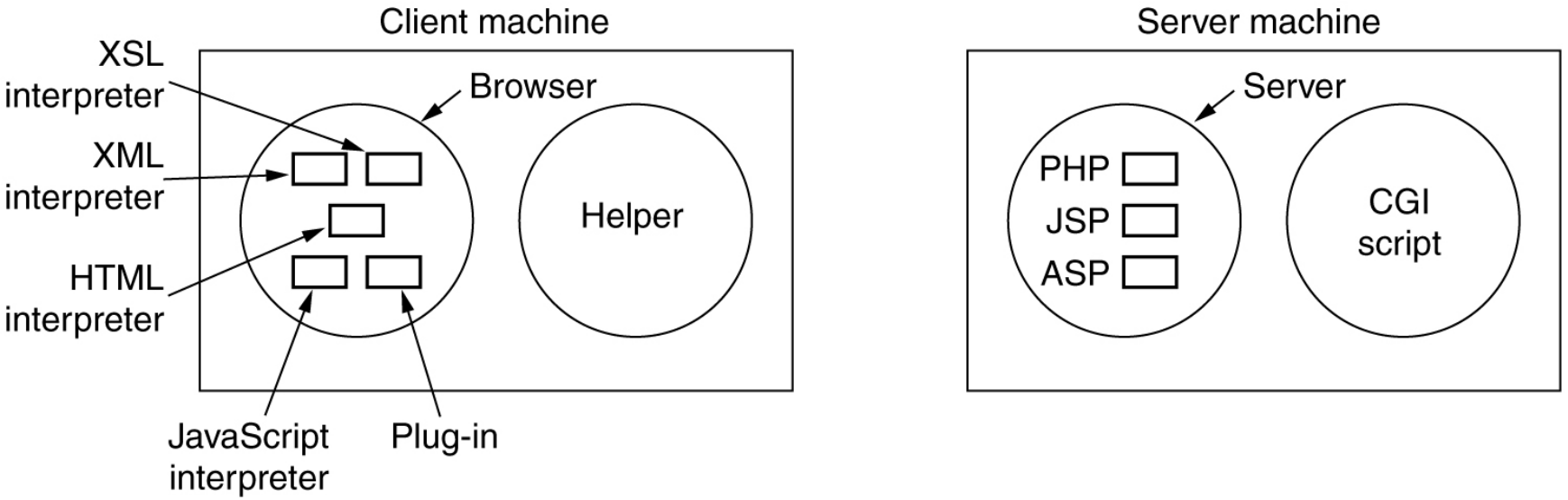
Generazione di pagine sul lato Client (3)

- Una pagina Web interattiva che risponde al movimento del mouse.

```
<html>
<head>
<script language="javascript" type="text/javascript">
if (!document.myurl) document.myurl = new Array();
document.myurl[0] = "http://www.cs.vu.nl/~ast/im/kitten.jpg";
document.myurl[1] = "http://www.cs.vu.nl/~ast/im/puppy.jpg";
document.myurl[2] = "http://www.cs.vu.nl/~ast/im/bunny.jpg";
function pop(m) {
    var urx = "http://www.cs.vu.nl/~ast/im/cat.jpg";
    popupwin = window.open(document.myurl[m],"mywind","width=250,height=250");
}
</script>
</head>

<body>
<p> <a href="#" onMouseover="pop(0); return false;" > Kitten  </a> </p>
<p> <a href="#" onMouseover="pop(1); return false;" > Puppy  </a> </p>
<p> <a href="#" onMouseover="pop(2); return false;" > Bunny  </a> </p>
</body>
</html>
```

Generazione di pagine dinamiche

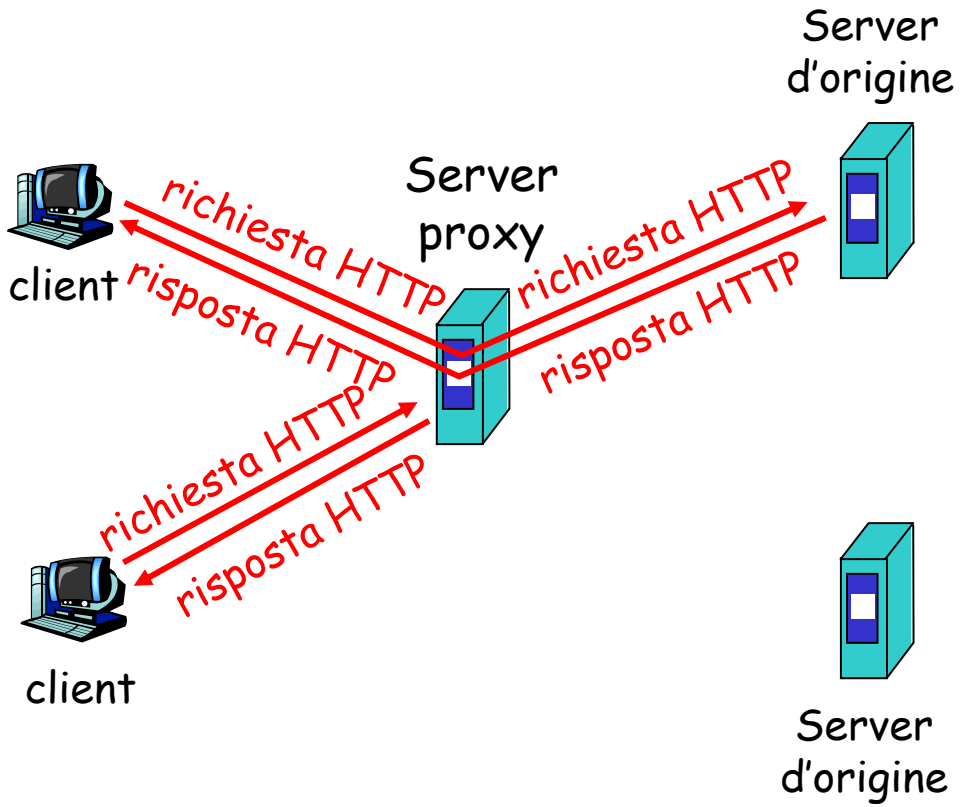
- I molti modi per generare e visualizzare il contenuto.



Cache web (server proxy)

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



Cache web (continua)

- La cache opera come client e come server
- Tipicamente la cache è installata da un ISP (università, aziende o ISP residenziali)

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 "scadenti" di fornire dati con efficacia (ma così fa la condivisione di file P2P)

Esempio di caching

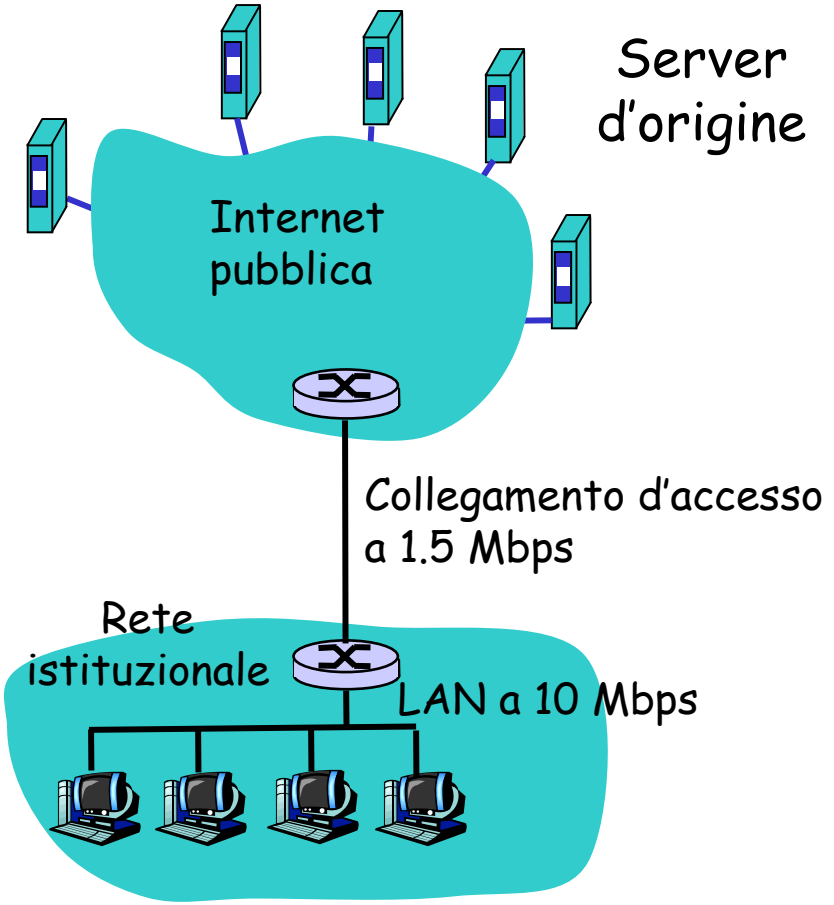
Ipotesi

- Dimensione media di un oggetto = 100.000 bit
- Frequenza media di richieste dai browser istituzionali ai server d'origine = 15/sec
- Ritardo dal router istituzionale a qualsiasi server d'origine e ritorno al router = 2 sec

Conseguenze

- utilizzazione sulla LAN = 15%
- utilizzazione sul collegamento d'accesso = 100%
- ritardo totale = ritardo di Internet + ritardo di accesso + ritardo della LAN

= 2 sec + minuti + millisecondi



Esempio di caching (continua)

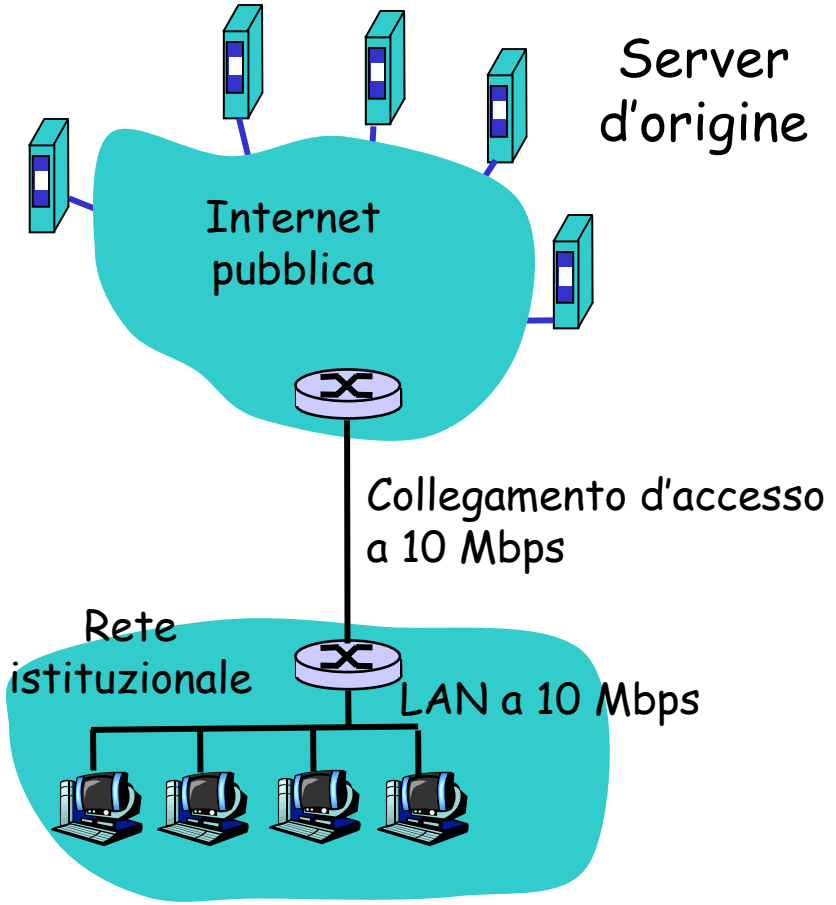
Soluzione possibile

- aumentare l'ampiezza di banda del collegamento d'accesso a 10 Mbps, per esempio

Conseguenze

- utilizzazione sulla LAN = 15%
- utilizzazione sul collegamento d'accesso = 15%
- ritardo totale = ritardo di Internet + ritardo di accesso + ritardo della LAN

  = 2 sec + msecs + msecs
- l'aggiornamento spesso è molto costoso



Esempio di caching (continua)

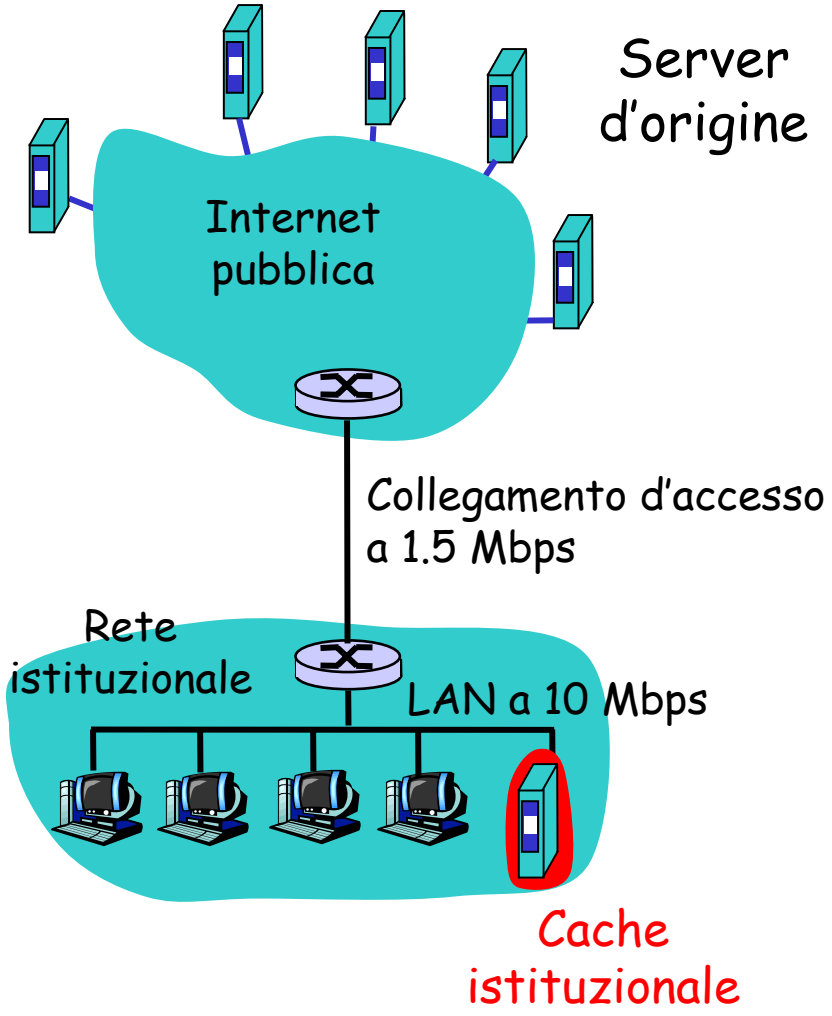
Installare la cache

- supponiamo una percentuale di successo (*hit rate*) pari a 0,4

Conseguenze

- il 40% delle richieste sarà soddisfatto quasi immediatamente
- il 60% delle richieste sarà soddisfatto dal server d'origine
- l'utilizzazione del collegamento d'accesso si è ridotta al 60%, determinando ritardi trascurabili (circa 10 msec)
- ritardo totale medio = ritardo di Internet + ritardo di accesso + ritardo della LAN =

  0,6*(2,01) sec + millisecondi < 1,4 sec

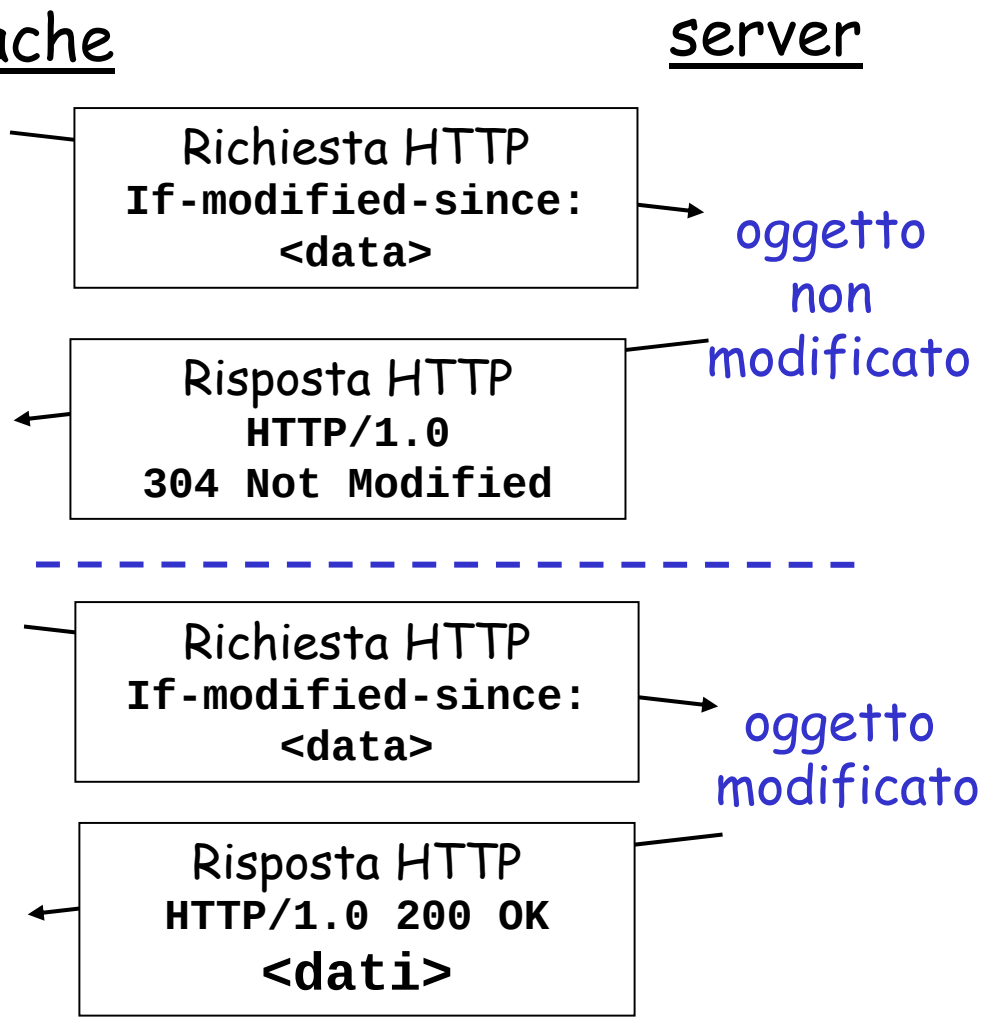


GET condizionale

- Obiettivo: non inviare un oggetto se la cache ha una copia aggiornata dell'oggetto
- cache: specifica la data della copia dell'oggetto nella richiesta HTTP

  If-modified-since: <data>
- server: la risposta non contiene l'oggetto se la copia nella cache è aggiornata:

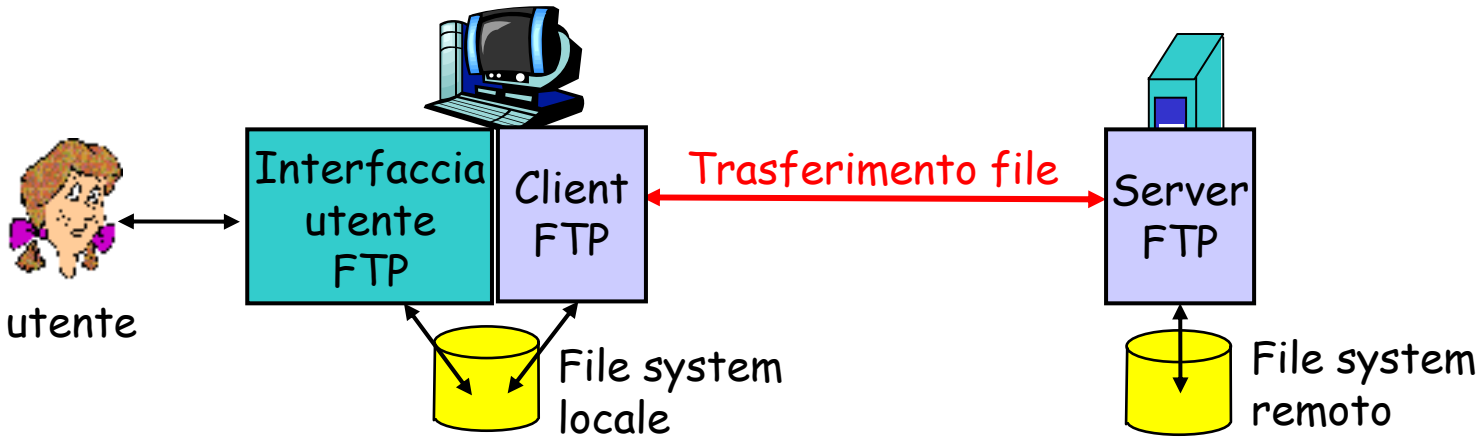
  HTTP/1.0 304 Not Modified



Capitolo 2: Livello di applicazione

- 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
- 2.6 Condivisione di file P2P

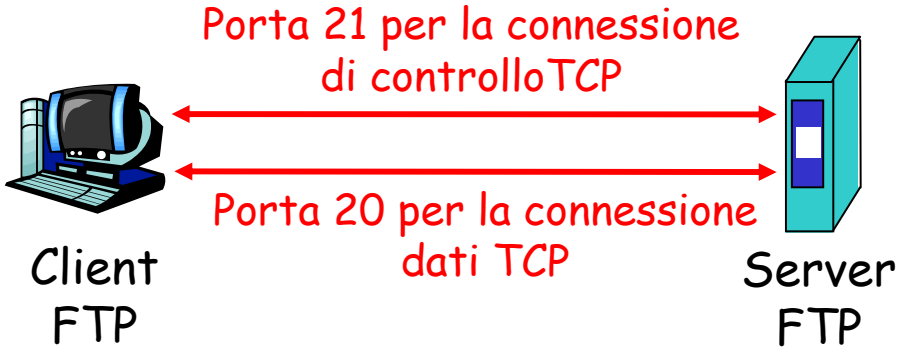
FTP: file transfer protocol



- 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

FTP: 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": directory corrente, autenticazione precedente

Comandi e risposte FTP

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**

Capitolo 2: Livello di applicazione

- 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
- 2.6 Condivisione di file P2P

DNS: Domain Name System

Persone: molti identificatori:

- nome, codice fiscale, carta d'identità

Host e router di Internet:

- indirizzo IP (32 bit) - usato per indirizzare i datagrammi
- "nome", ad esempio, www.yahoo.com – usato dagli esseri umani

D: Come associare un indirizzo IP a un nome?

Domain Name System:

- *Database distribuito* implementato in una gerarchia di *server DNS*
- *Protocollo a livello di applicazione* che consente agli host, ai router e ai server DNS di comunicare per *risolvere* i nomi (tradurre indirizzi/nomi)
  - nota: funzioni critiche di Internet implementate come protocollo a livello di applicazione
  - complessità nelle parti periferiche della rete

DNS

Servizi DNS

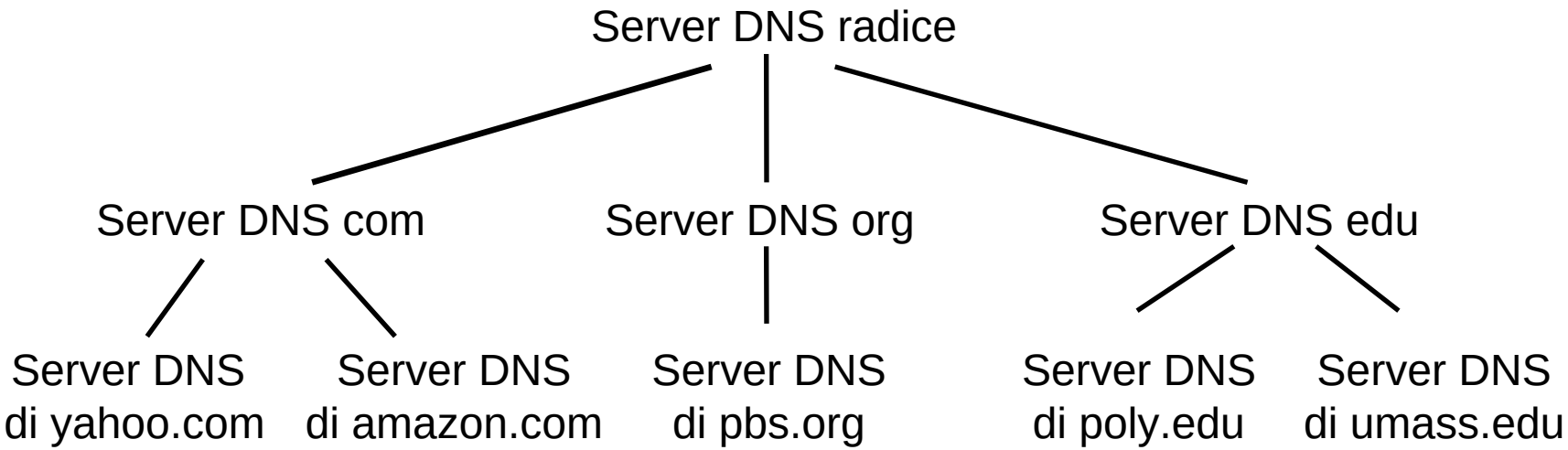
- Traduzione degli hostname in indirizzi IP
- Host aliasing
  - un host può avere più nomi
- Mail server aliasing
- Distribuzione locale
  - server web replicati: insieme di indirizzi IP per un nome canonico

Perché non centralizzare DNS?

- singolo punto di guasto
- volume di traffico
- database centralizzato distante
- manutenzione

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

Database distribuiti e gerarchici

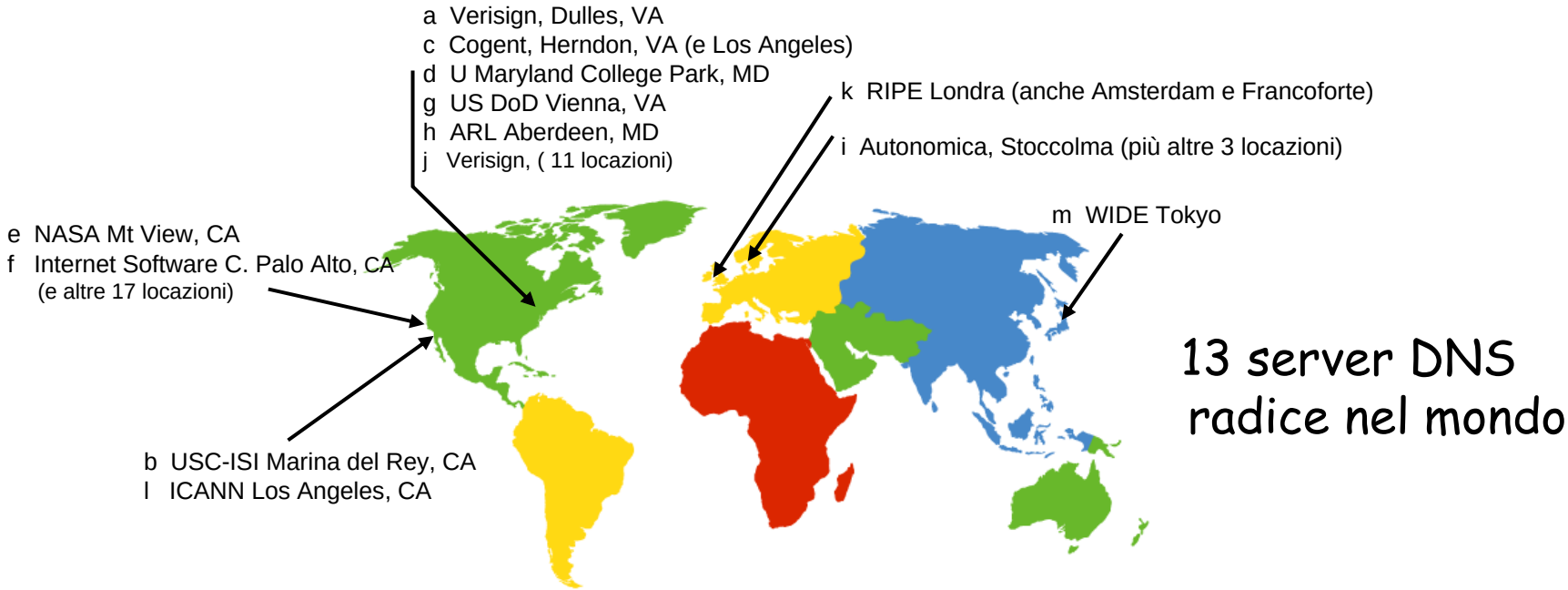


Il client vuole l'IP di www.amazon.com; 1ª approssimazione:

- 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

DNS: server DNS radice

- 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



13 server DNS radice nel mondo

Server TLD e server di competenza

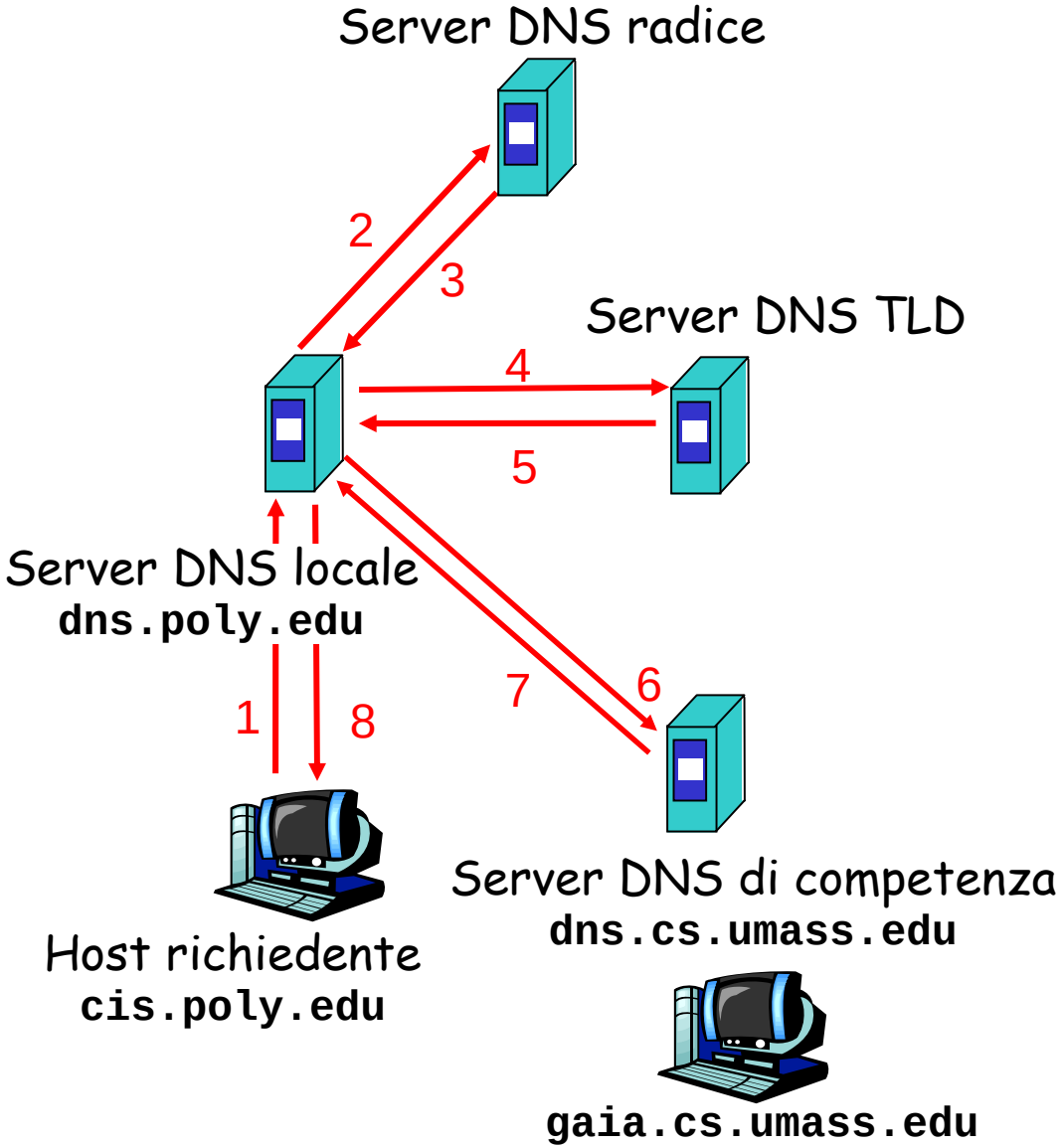
- Server TLD (top-level domain): si occupano dei domini com, org, net, edu, ecc. e di tutti i domini locali di alto livello, quali uk, fr, ca e jp.
  - Network Solutions gestisce i server TLD per il dominio com
  - Educause gestisce quelli per il dominio edu
- Server di competenza (*authoritative server*):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

Server DNS locale

- 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

Esempio

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



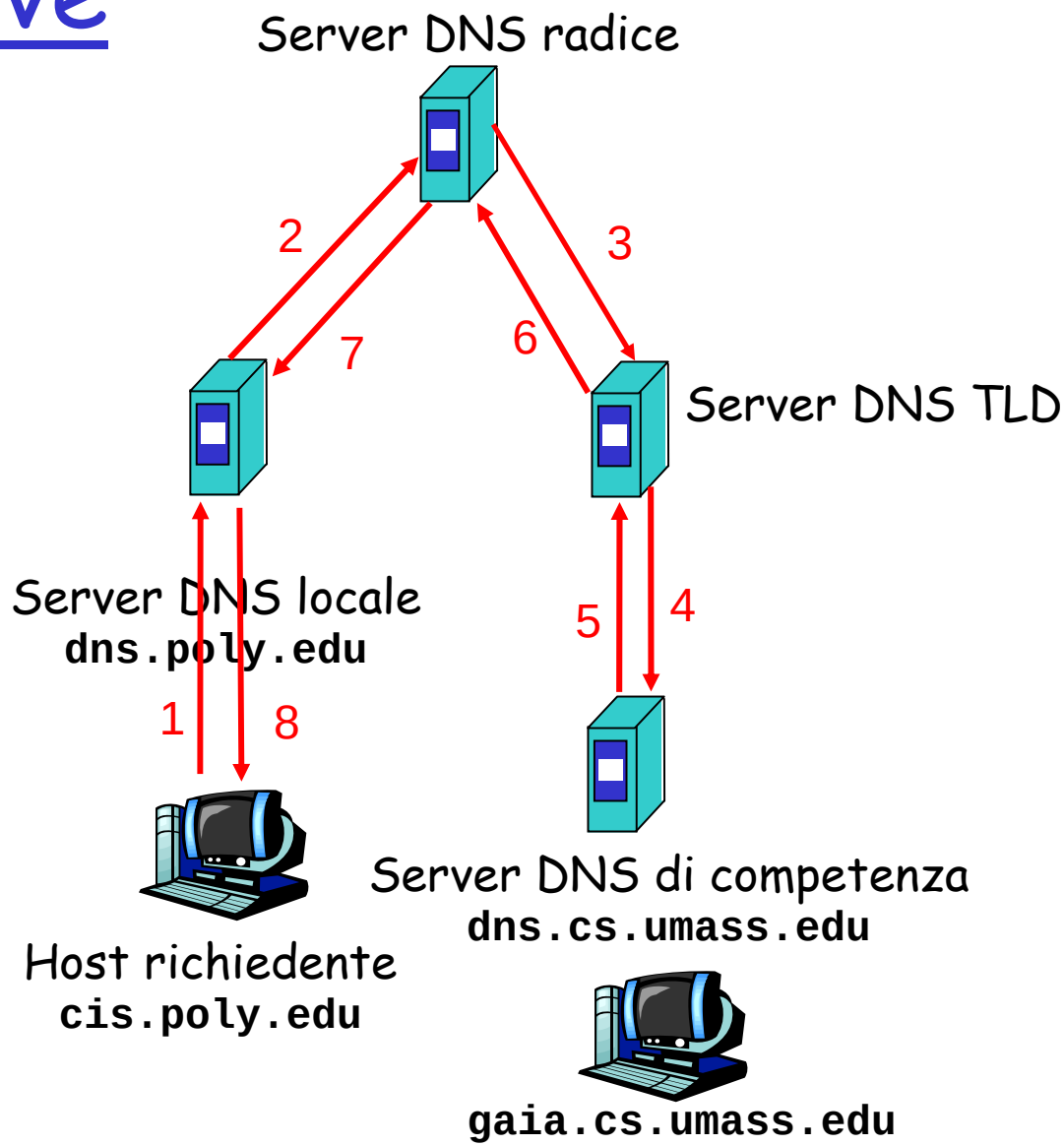
Query ricorsive

Query ricorsiva:

- Affida il compito di tradurre il nome al server DNS contattato
- Compito difficile?

Query iterativa:

- Il server contattato risponde con il nome del server da contattare
- "Non conosco questo nome, ma chiedi a questo server"



DNS: caching e aggiornamento dei record

- Una volta che un server DNS impara la mappatura, la mette nella *cache*
  - le informazioni nella cache vengono invalidate (spariscono) 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

Record DNS

DNS: database distribuito che memorizza i record di risorsa (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)

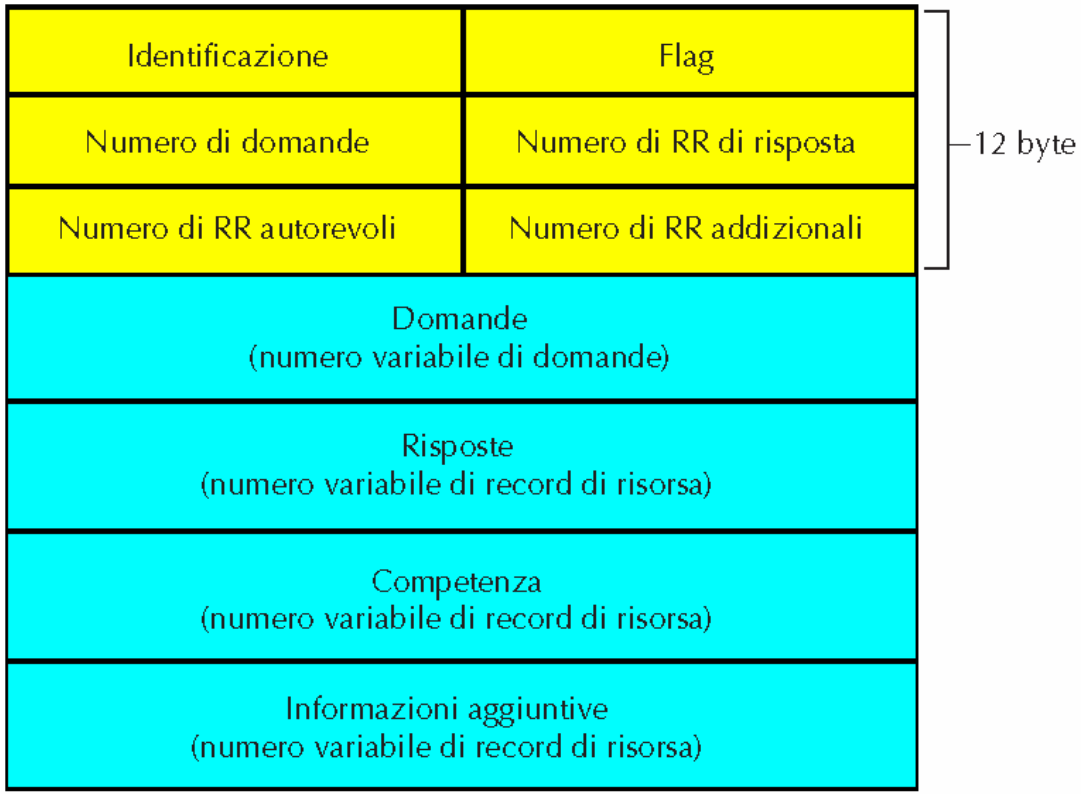
    `www.ibm.com` è in realtà `servereast.backup2.ibm.com`
  - **value** è il nome canonico
- Type=MX
  - **value** è il nome del server di posta associato a **name**

Messaggi DNS

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 o risposta
  - richiesta di ricorsione
  - ricorsione disponibile
  - risposta di competenza



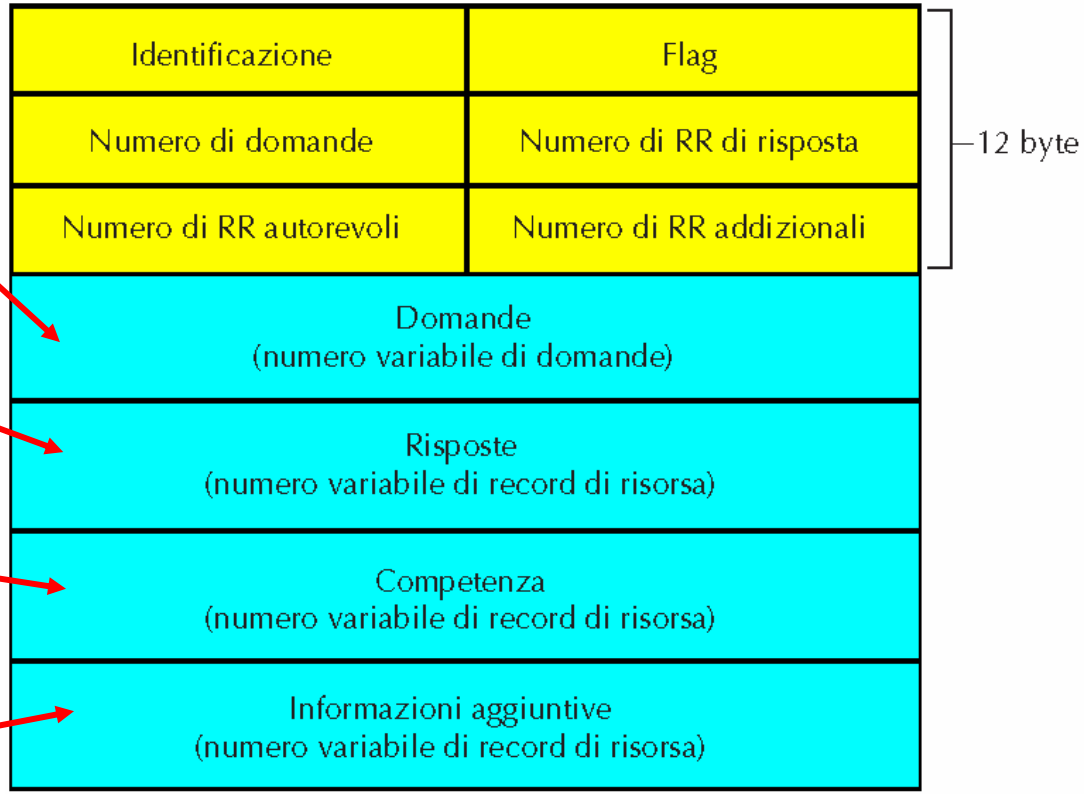
Messaggi DNS

Campi per
il nome richiesto
e il tipo di domanda

RR nella risposta
alla domanda

Record per
i server di competenza

Informazioni extra che
possono essere usate



Capitolo 2: Livello di applicazione

- 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
- 2.6 Condivisione di file P2P

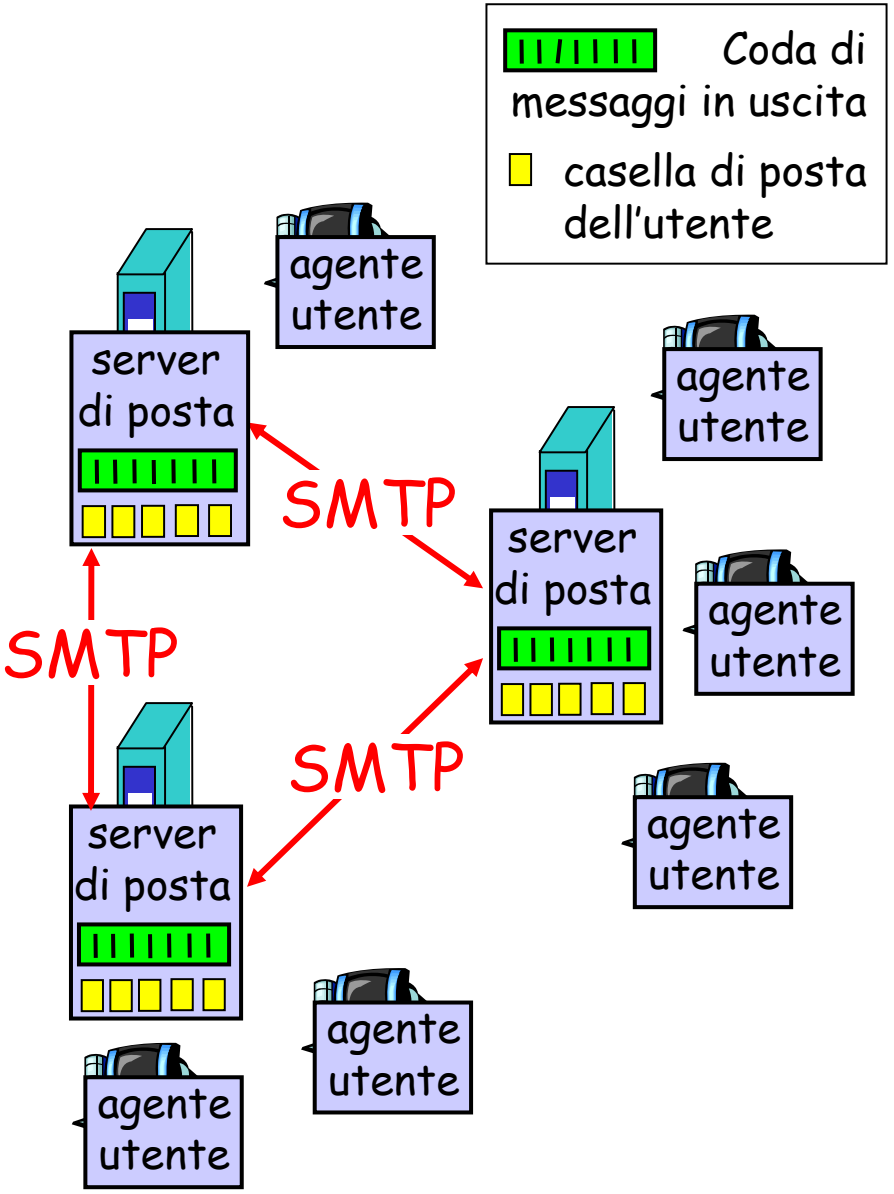
Posta elettronica

Tre componenti principali:

- agente utente
- server di posta
- simple mail transfer protocol: SMTP

Agente utente

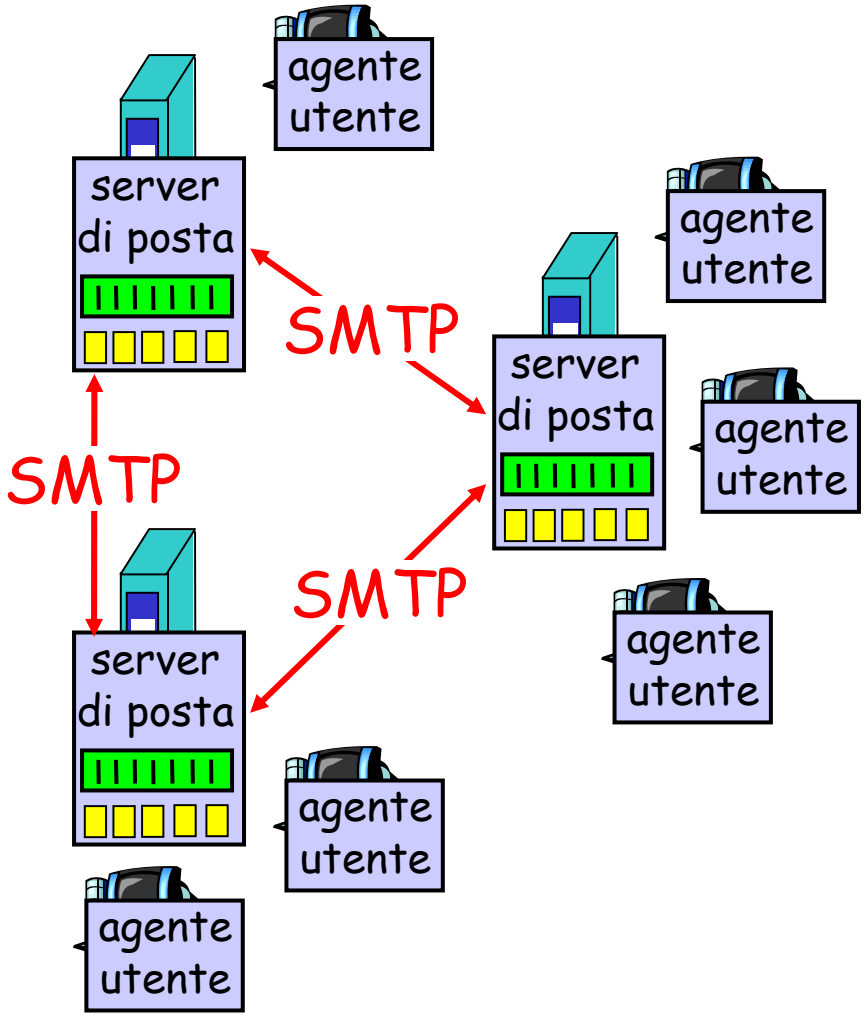
- detto anche "mail reader"
- composizione, editing, lettura dei messaggi di posta elettronica
- esempi: Eudora, Outlook, elm, Netscape Messenger
- i messaggi in uscita o in arrivo sono memorizzati sul server



Posta elettronica: server di posta

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

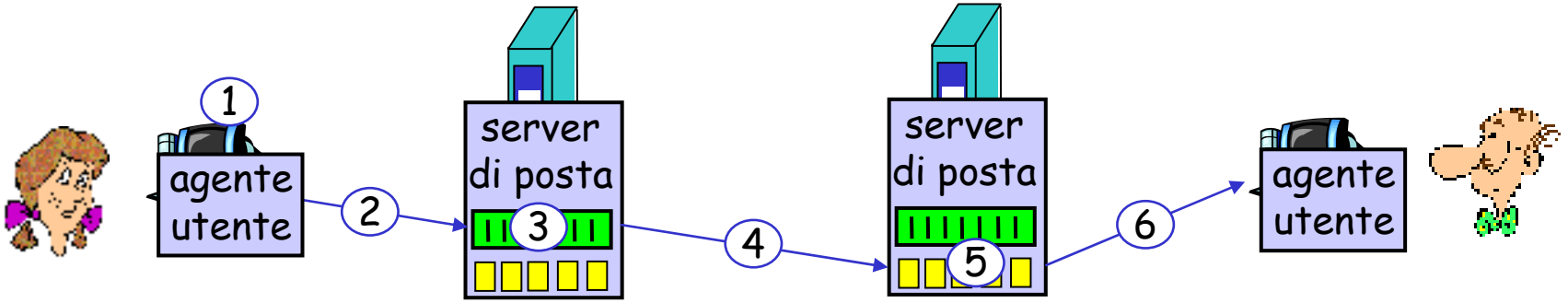


Posta elettronica: SMTP [RFC 2821]

- 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
- tre espressioni 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

Scenario: Alice invia un messaggio a Bob

1) Alice usa il suo agente utente per comporre il messaggio da inviare "a" bob@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 Bob
4) Il client SMTP invia il messaggio di Alice sulla connessione TCP
5) Il server di posta di Bob pone il messaggio nella casella di posta di Bob
6) Bob invoca il suo agente utente per leggere il messaggio



Esempio di interazione SMTP

```
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: <bob@hamburger.edu>
S: 250 bob@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: note finali

- SMTP usa connessioni persistenti
- SMTP richiede che il messaggio (intestazione e corpo) sia nel formato ASCCI 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: ogni oggetto è incapsulato nel suo messaggio di risposta
- SMTP: più oggetti vengono trasmessi in un unico messaggio

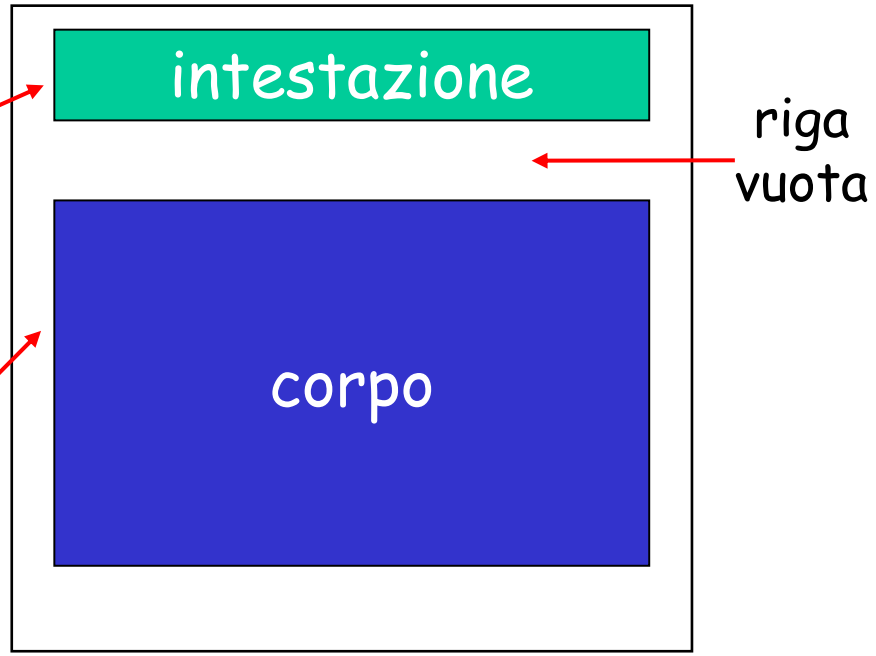
Formato dei messaggi di posta elettronica

SMTP: protocollo per scambiare messaggi di posta elettronica

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

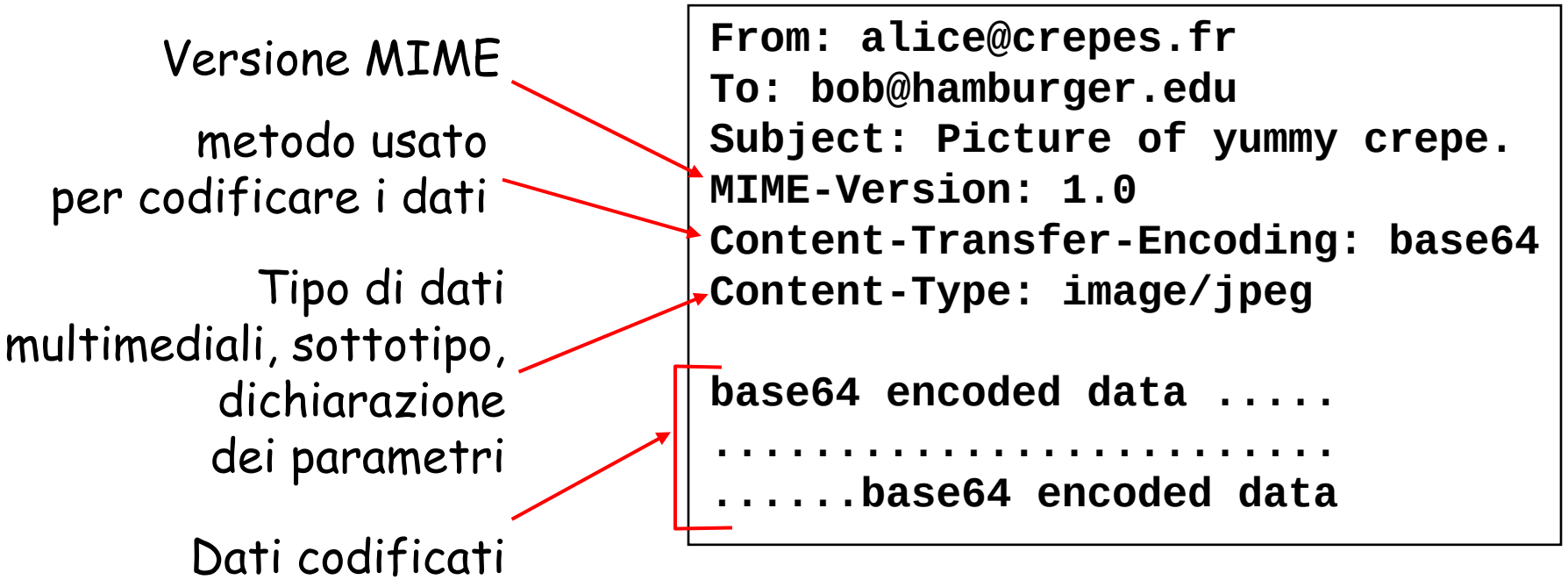
- Righe di intestazione, per esempio
  - To:
  - From:
  - Subject:

  differenti dai comandi SMTP!
- corpo
  - il "messaggio", soltanto caratteri ASCII

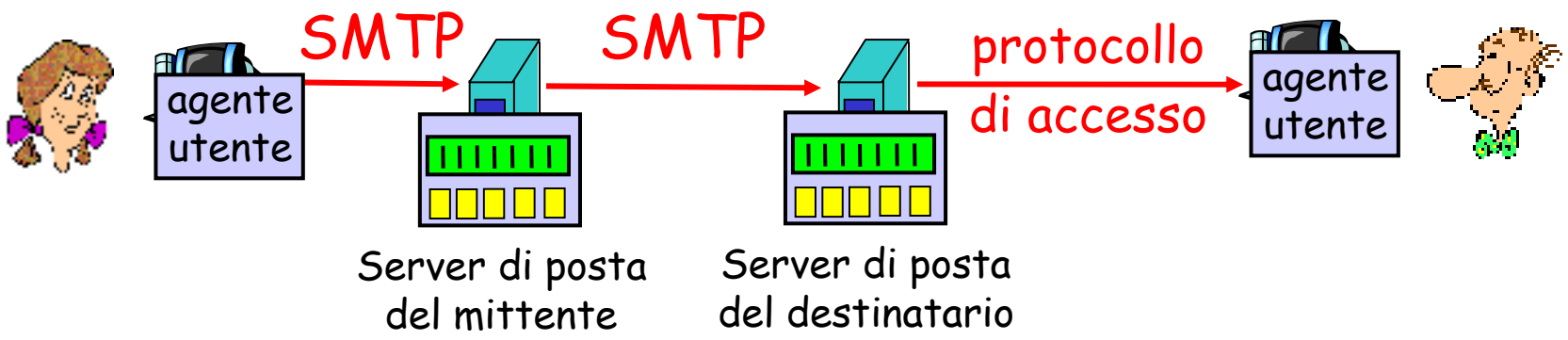


Formato del messaggio: estensioni di messaggi multimediali

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



Protocolli di accesso alla posta



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

Protocollo POP3

Fase di autorizzazione

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

Fase di transazione, client:

- **list:** elenca i numeri dei messaggi
- **retr:** ottiene i messaggi per numero
- **dele:** cancella
- **quit**

```
S: +OK POP3 server ready
C: user bob
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
```

POP3 (altro) e IMAP

Ancora su POP3

- Il precedente esempio usa la modalità "scarica e cancella"
- Bob non può rileggere le e-mail se cambia client
- Modalità "scarica e mantieni": copia i messaggi su più client
- POP3 è un protocollo senza stato tra le varie sessioni

IMAP

- Mantiene tutti i messaggi in un unico posto: il server
- Consente all'utente di organizzare i messaggi in cartelle
- 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

Capitolo 2: Livello di applicazione

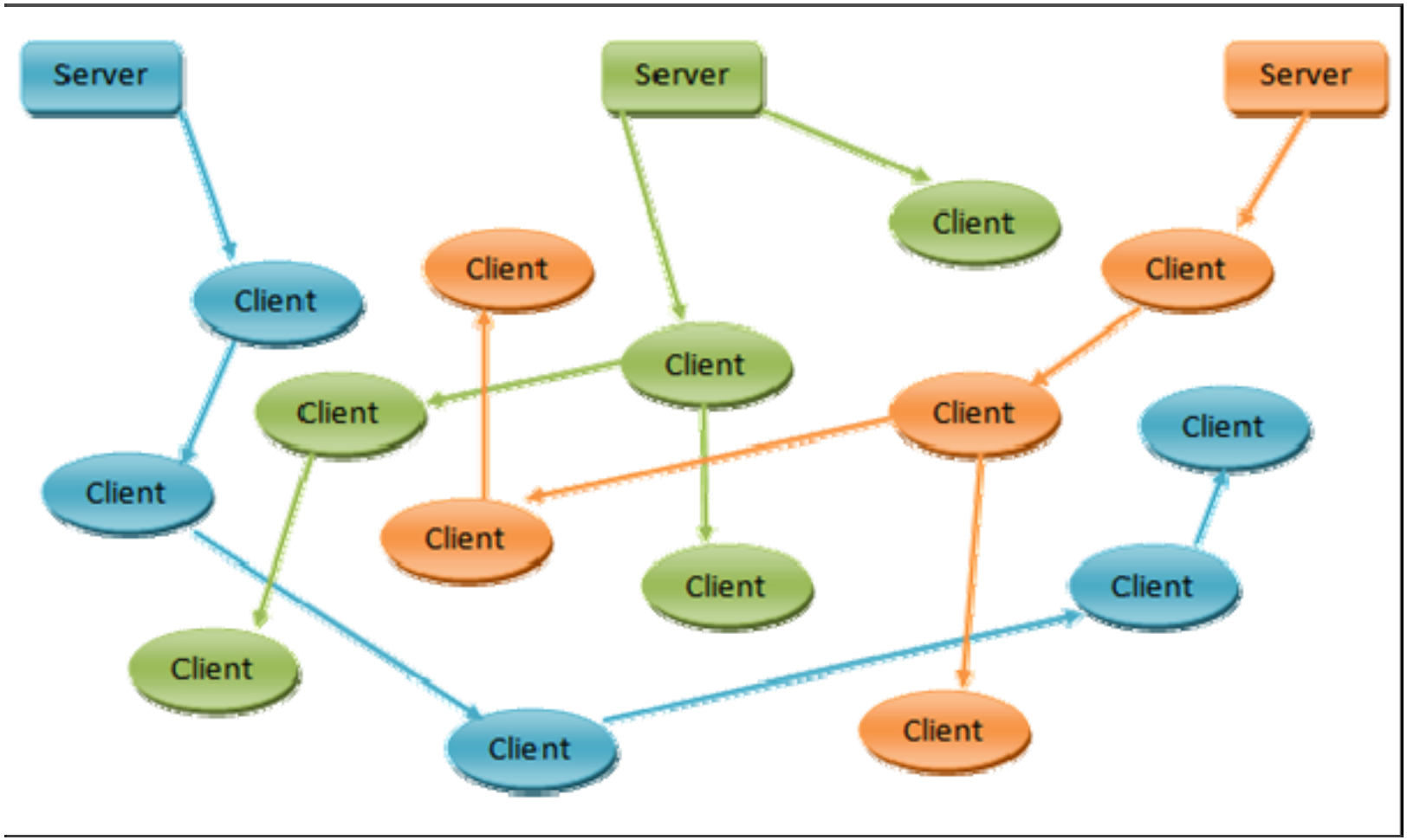
- 2.1 Principi delle applicazioni di rete
- 2.2 Web e HTTP
- 2.3 FTP
- 2.4 Posta elettronica
  - SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 Condivisione di file P2P

Cos'è il peer to peer?

- Una rete P2P ("da pari a pari") è una rete di computer (o qualsiasi rete) che non possiede client o server fissi, ma un numero di nodi equivalenti che fungono sia da client che da server verso gli altri nodi della rete

- Questo modello rappresenta l'antitesi alla tradizionale architettura client-server

Architettura tipica P2P



Caratteristiche principali

- non c'è un server sempre attivo

  →ogni nodo è indipendente da server centrale
- coppie arbitrarie di host possono comunicare tra loro in modo diretto

  →ogni nodo condivide le risorse con altri nodi
- i peer non devono essere sempre attivi

  →cambiano indirizzo IP in modo dinamico

Condivisione di file P2P

Esempio

- Alice esegue un'applicazione di condivisione file P2P sul suo notebook
- Si collega in modo intermittente a Internet; ottiene un nuovo indirizzo IP ogni volta che si collega
- Cerca la canzone intitolata "Hey Jude"
- L'applicazione visualizza altri peer che hanno una copia di "Hey Jude"
- Alice sceglie uno dei peer, Bob
- Il file viene inviato dal PC di Bob al notebook di Alice: HTTP
- Mentre Alice scarica il file, altri utenti potrebbero scaricare dei file da Alice
- Il peer di Alice è sia client web sia server web transitorio

Tutti i peer sono server = grande scalabilità!

Vantaggi delle reti P2P

- TECNOLOGICI
  - Decentramento: quando un nodo "cade" questo viene rimpiazzato da altri nodi
  - Scalabilità: più nodi ci sono nella rete e maggiore è l'efficienza

Vantaggi delle reti P2P

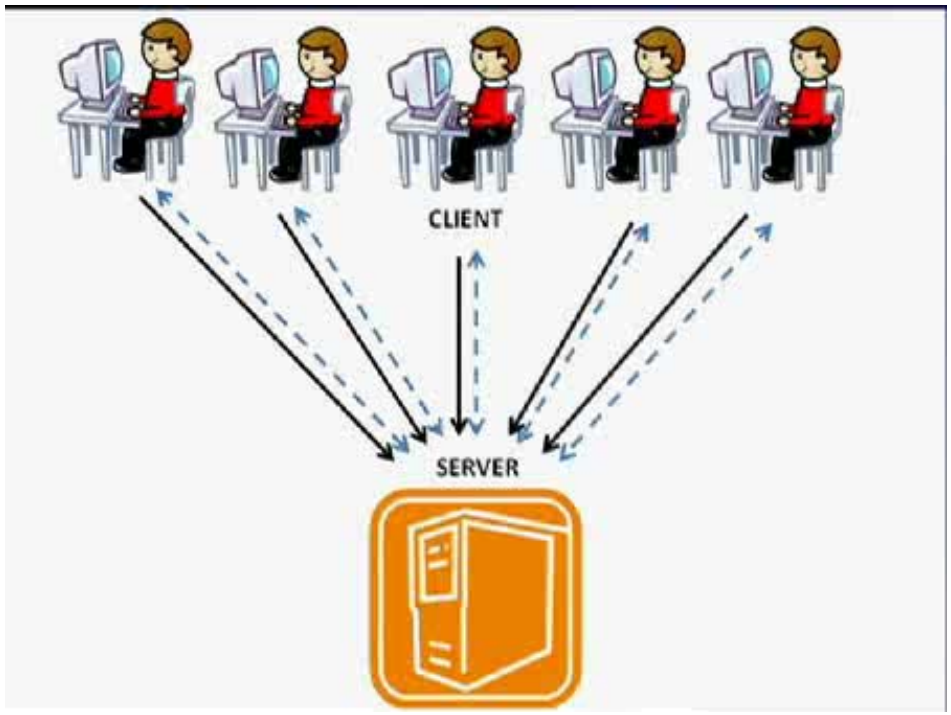
- SOCIALI
  - Organizzazione democratica:
    - ogni nodo ha lo stesso valore di quello vicino
    - tutti i nodi dal basso concorrono alla creazione di valore

Principali utilizzi delle reti P2P

1. FILESHARING
2. SISTEMI PER LA CONDIVISIONE DEL CALCOLO (es.SETI@HOME per ricerca intelligente extraterrestre)
3. SISTEMI PER LA PRIVACY E LA SICUREZZA (es. Freenet)
4. SISTEMI DI COMUNICAZIONE (es. Skype)

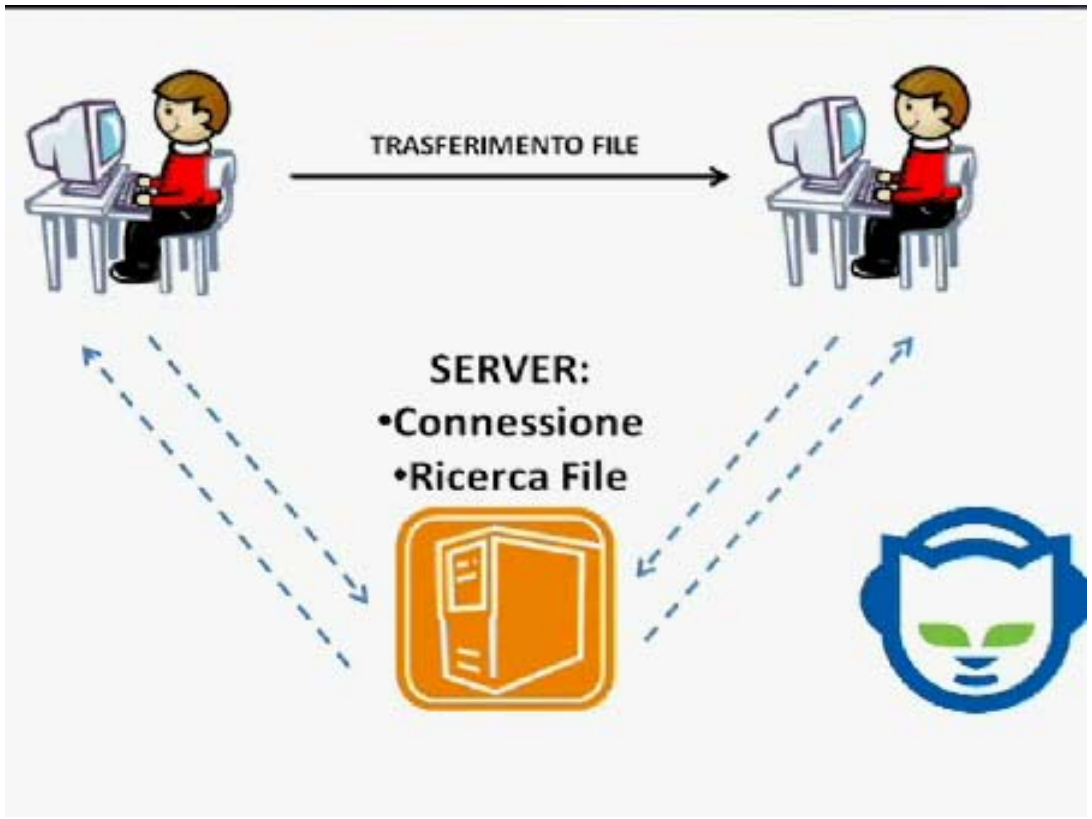
Evoluzione delle reti P2P

- Prima dell'architettura P2P il modello più diffuso era quello Client-Server:



Evoluzione delle reti P2P

- Reti P2P di prima generazione (Napster)



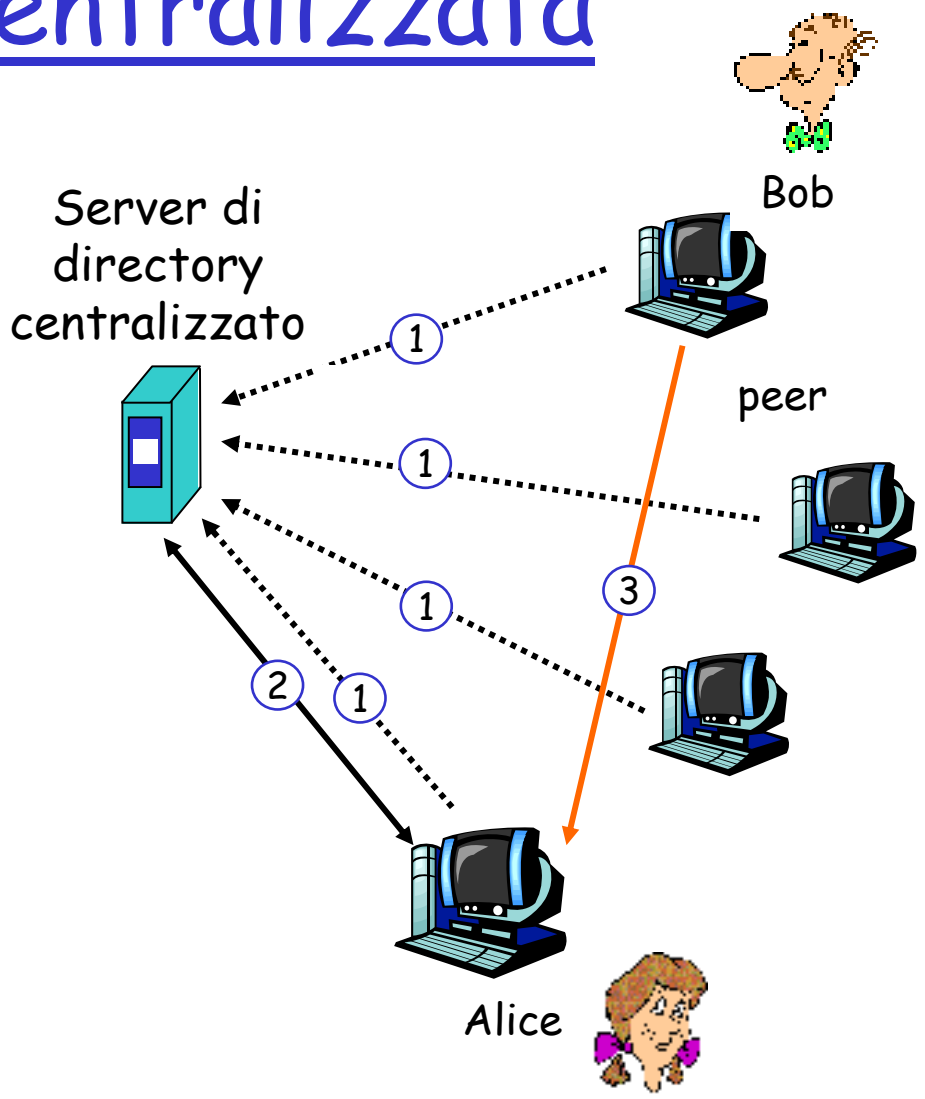
L'avvento di Napster

- Si tratta di un P2P di tipo ibrido, cioè nell'architettura è presente un server centrale che conserva informazioni sui peer e risponde a richieste riguardo quelle informazioni:
- P2P utilizzato solo per il trasferimento file
- Accesso alla rete gestito da un server
- Ricerca file gestita da un server

P2P: directory centralizzata

Progetto originale di "Napster"

1) quando il peer si collega, informa il server centrale:
   - indirizzo IP
   - contenuto
2) Alice cerca la canzone "Hey Jude"
3) Alice richiede il file a Bob



P2P: problemi con la directory centralizzata

- Unico punto di guasto
- Collo di bottiglia per le prestazioni
- Violazione del diritto d'autore

Il trasferimento dei file è distribuito, ma il processo di localizzazione è fortemente centralizzato

Query flooding: Gnutella

- Completamente distribuito
  - nessun server centrale
- Protocollo di pubblico dominio
- Molti client Gnutella implementano il protocollo

Rete di copertura: grafo

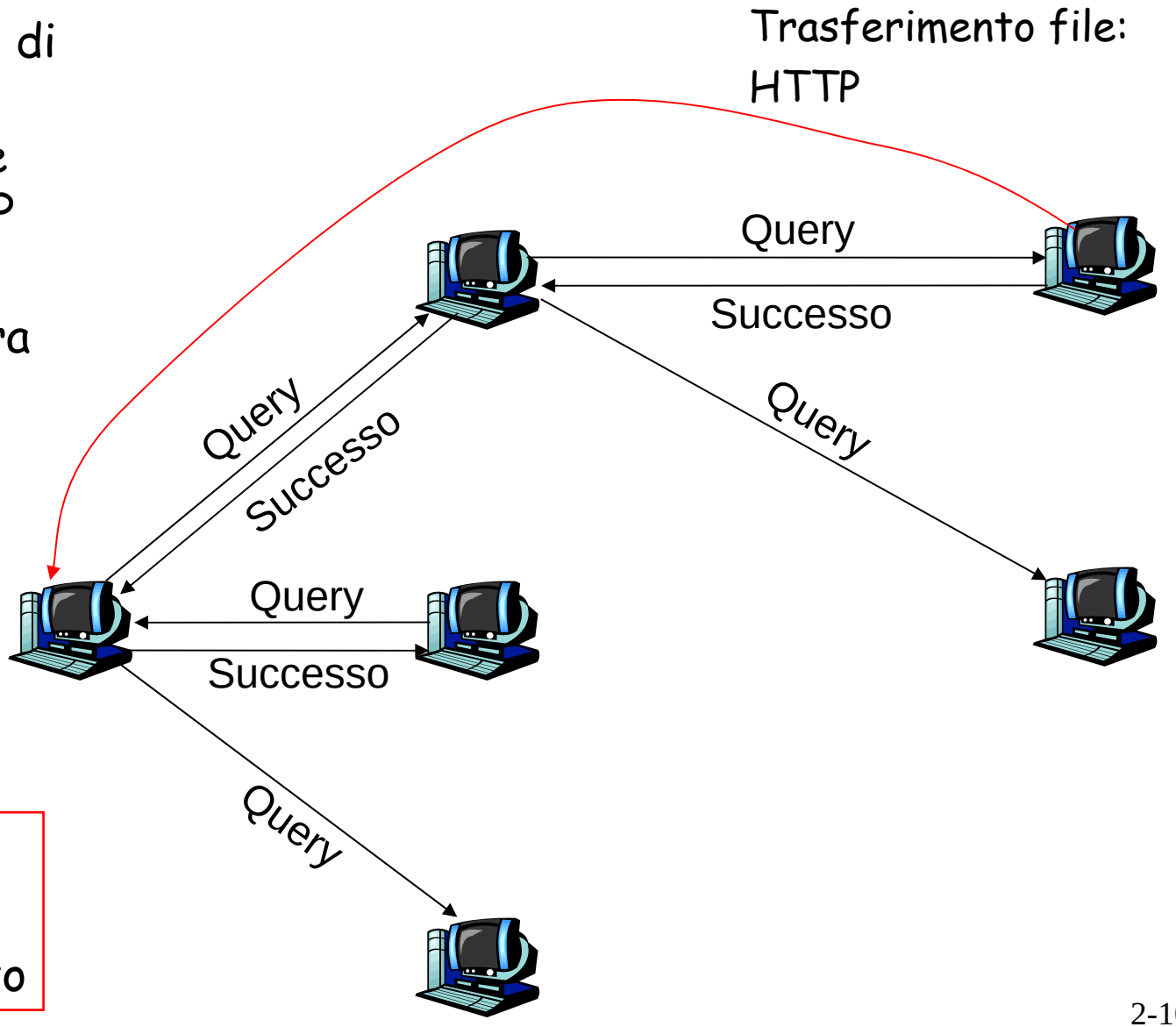
- Arco tra i peer X e Y se c'è una connessione TCP
- Tutti i peer attivi e gli archi formano la rete di copertura
- Un arco non è un collegamento fisico
- Un dato peer sarà solitamente connesso con meno di 10 peer vicini nella rete di copertura

Gnutella: protocollo

- Il messaggio di richiesta è trasmesso sulle connessioni TCP esistenti
- Il peer inoltra il messaggio di richiesta
- Il messaggio di successo è trasmesso sul percorso inverso

Scalabilità:

query flooding a raggio limitato



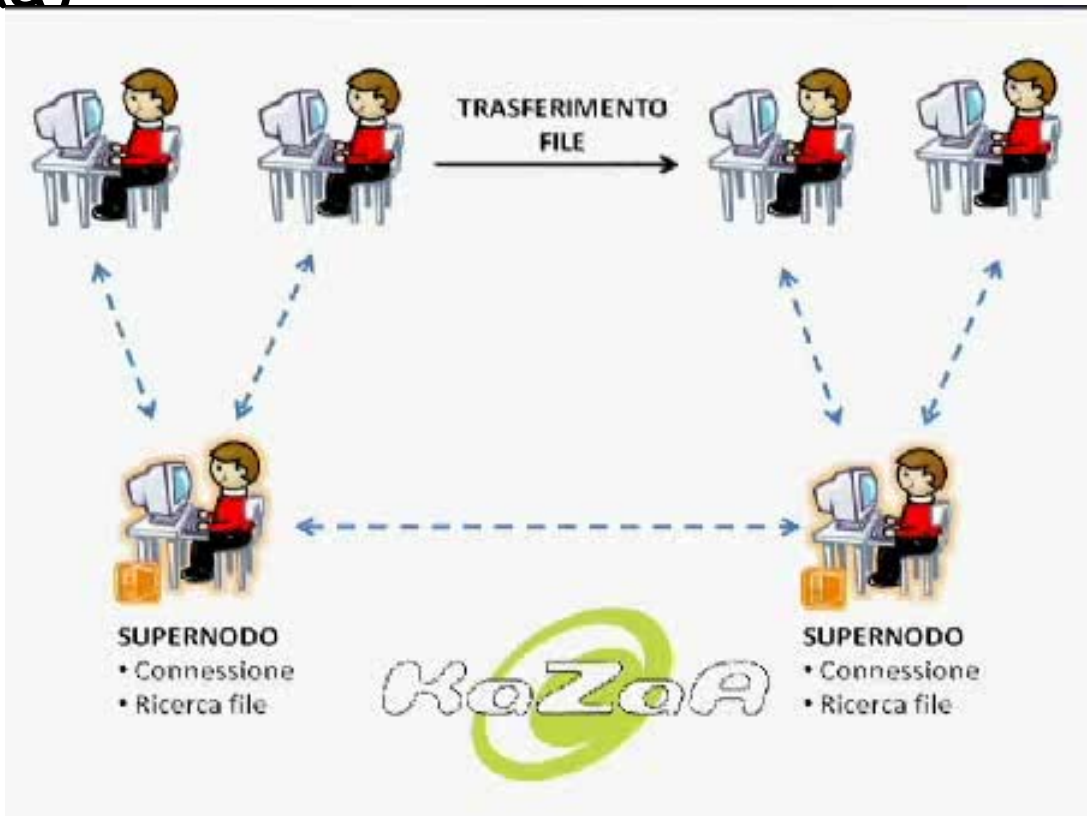
Gnutella: unione di peer

1. Per unire il peer X alla rete, bisogna trovare qualche altro peer della rete Gnutella: usate la lista dei peer candidati
2. X tenta in sequenza di impostare una connessione TCP con i peer della lista finché non stabilisce una connessione con Y
3. X invia un messaggio Ping a Y; Y inoltra il messaggio Ping
4. Tutti i peer che ricevono il messaggio Ping rispondono con un messaggio Pong
5. X riceve molti messaggi Pong. Quindi può impostare delle connessioni TCP addizionali

Distacco dei peer: consultate il problema alla fine del capitolo!

Evoluzione delle reti P2P

- Reti P2P di seconda generazione (es. Kazaa)



Verso un P2P puro: Kazaa

Si tratta di un P2P ancora di tipo ibrido ma con una fondamentale differenza:

- presenza di "supernodi", nodi più importanti di altri che assolvono funzioni da server

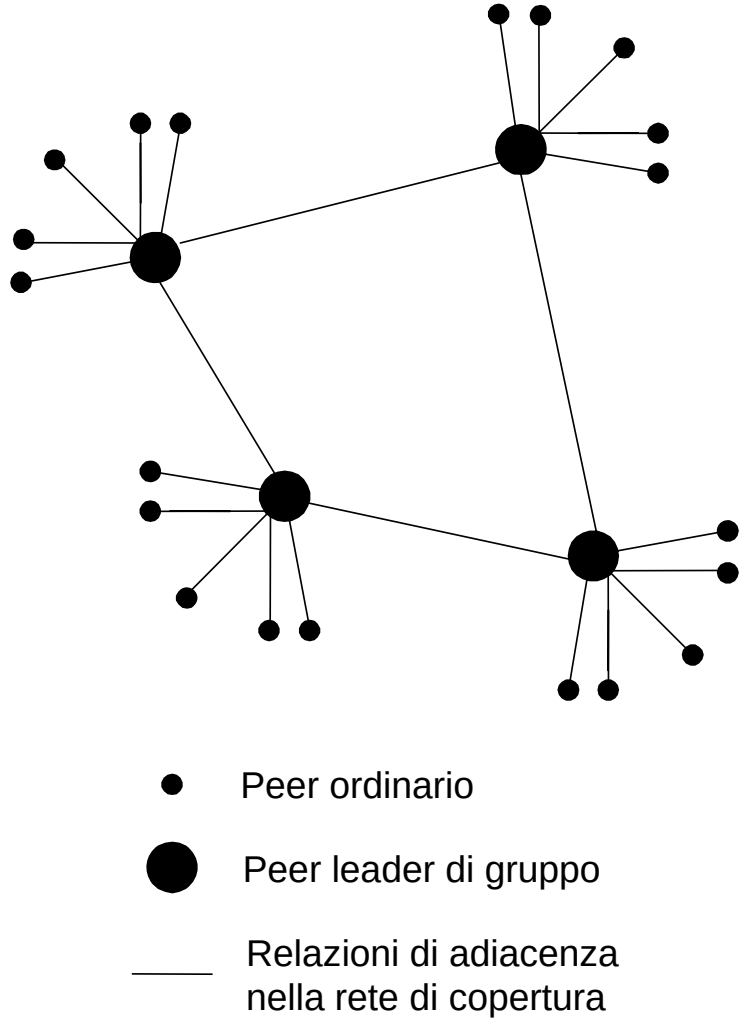
  →P2P utilizzato solo per il trasferimento file

  →accesso alla rete gestito da supernodi

  →ricerca file gestita da supernodi

Sfruttare l'eterogeneità: KaZaA

- Ogni peer è un leader di gruppo o è assegnato a un leader di gruppo
  - Connessione TCP tra peer e il suo leader di gruppo
  - Connessioni TCP tra qualche coppia di leader di gruppo
- Il leader di gruppo tiene traccia del contenuto di tutti i suoi figli.



KaZaA: query

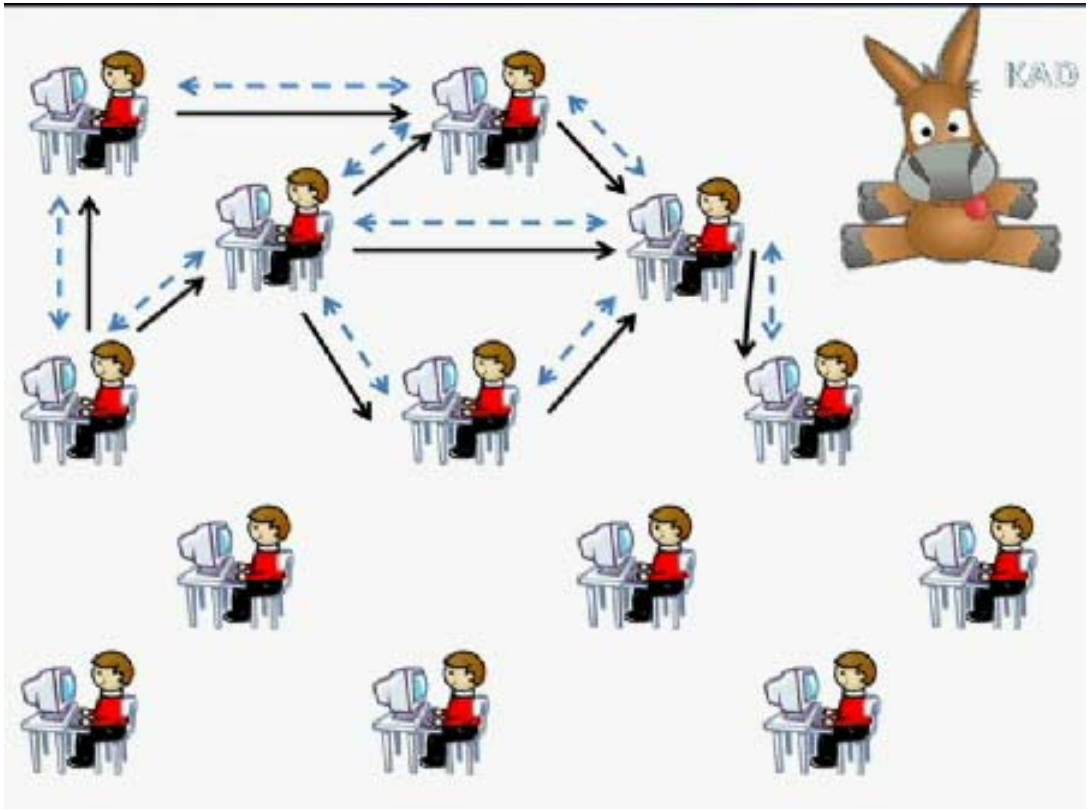
- Ogni file ha un identificatore hash e un descrittore
- Il client invia al suo leader di gruppo una query con una parola chiave
- Il leader di gruppo risponde con un elenco di peer che condividono i file i cui descrittori corrispondono alle parole chiave:
  - Per ogni corrispondenza: metadata, hash, indirizzo IP
- Se il leader di gruppo inoltra la query ad altri leader di gruppo, questi rispondono con le corrispondenze
- Il client quindi seleziona i file per il downloading
  - Le richieste HTTP che usano un identificatore hash sono trasmesse ai peer che hanno il file desiderato

Tecniche KaZaA

- Limitare il numero di upload simultanei
- Accodamento delle richieste
- Priorità di incentivo
- Downloading parallelo

Evoluzione delle reti P2P

- Reti P2P pure (es. Kad)



Approdo al P2P puro: rete Kad di Emule

Si tratta di un P2P puro:

- Non esistono più Server e Supernodi
- accesso alla rete gestite esclusivamente da pari a pari
- ricerca file gestita esclusivamente da pari a pari

→ se non si conosce l'indirizzo di un utente all'interno della rete non è possibile accedere alla rete stessa