Capitolo 2
Livello di applicazione

Reti di calcolatori e Internet: Un approccio top-down

4ª edizione
Jim Kurose, Keith Ross

Pearson Paravia Bruno Mondadori Spa
©2008

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 un adattamento (o di una copia) 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 Applicazioni P2P
- 2.7 Programmazione delle socket con TCP
- 2.8 Programmazione delle socket con UDP

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
- Condivisione di file P2P
- Giochi multiutente via rete
- Streaming di video-clip memorizzati
- Telefonia via Internet
- Videoconferenza in tempo reale
- Grid computing

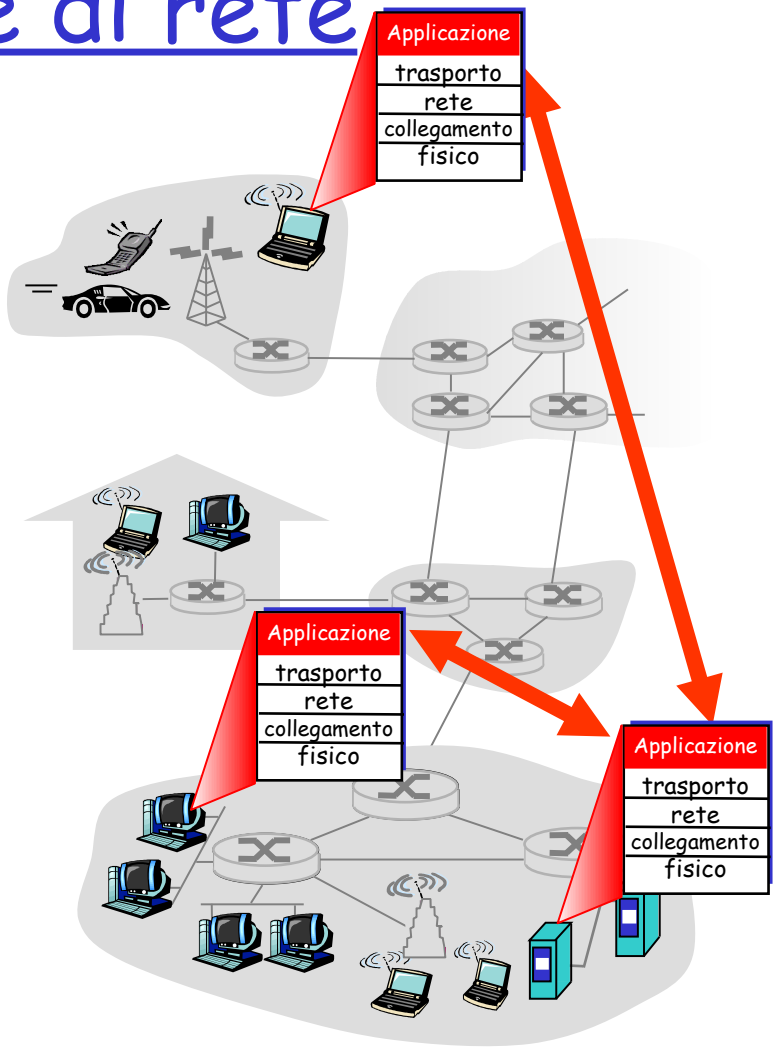
Creare un'applicazione di rete

Scrivere programmi che

- girano su sistemi terminali diversi
- comunicano attraverso la rete
- Ad es. 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



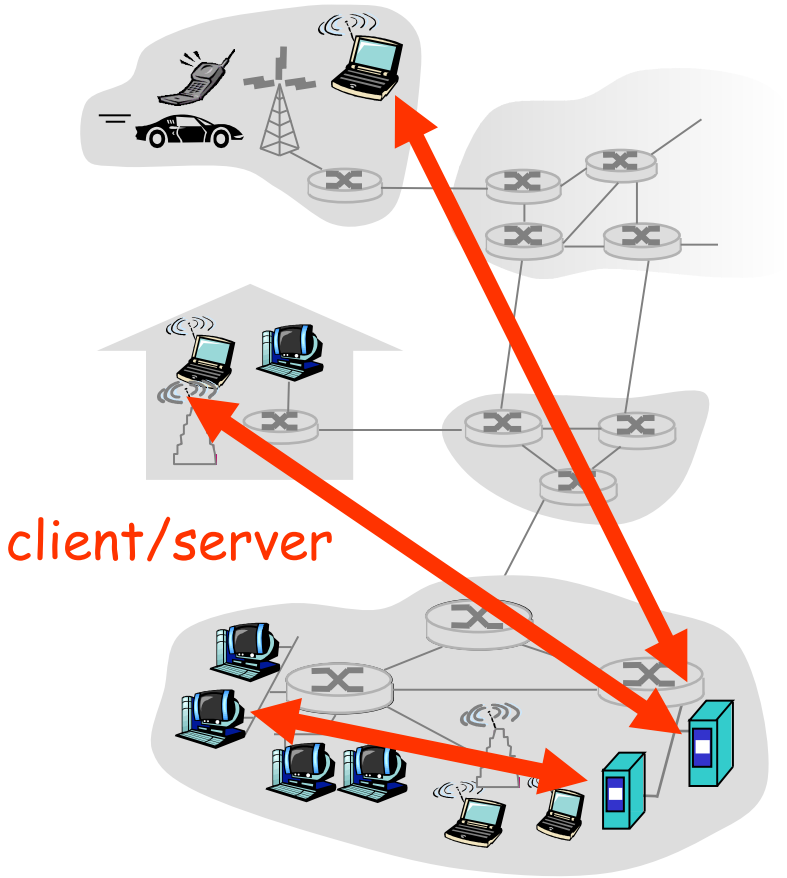
Capitolo 2: Livello di applicazione

- 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 Applicazioni P2P
- 2.7 Programmazione delle socket con TCP
- 2.8 Programmazione delle socket con UDP

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:

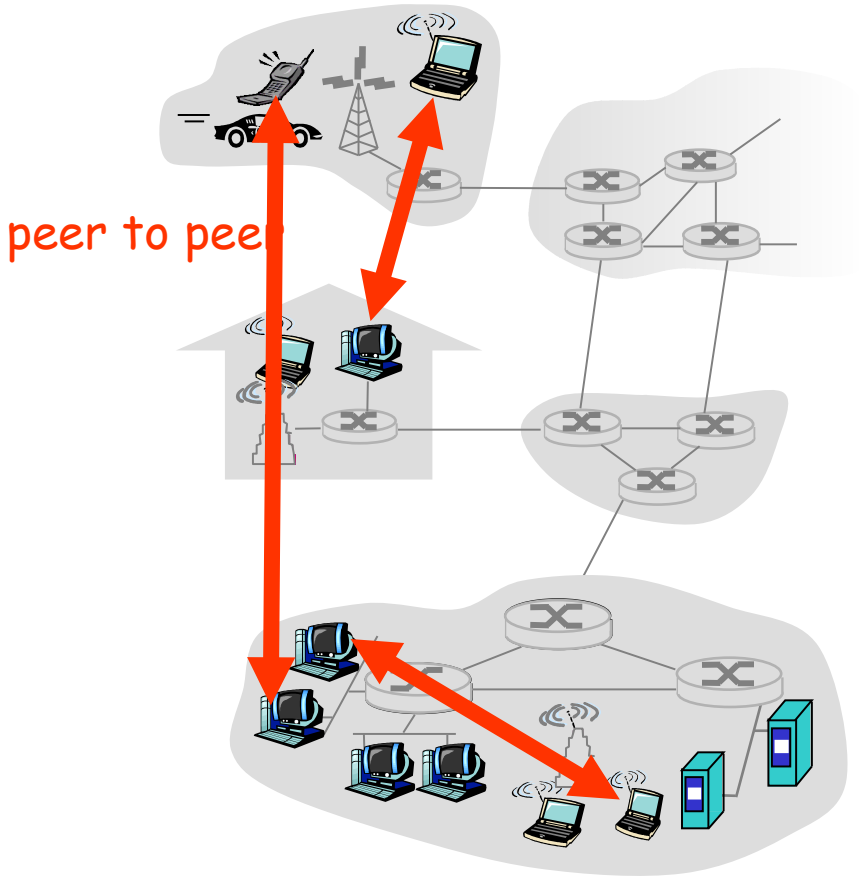
- comunica con il server
- può contattare il server in qualunque momento
- può avere indirizzi IP dinamici
- non comunica direttamente con gli altri client

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

Facilmente scalabile

Difficile da gestire



Ibridi (client-server e P2P)

Skype

- Applicazione P2P di Voice over IP
- Server centralizzato: ricerca indirizzi della parte remota
- Connessione client-client: diretta (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

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

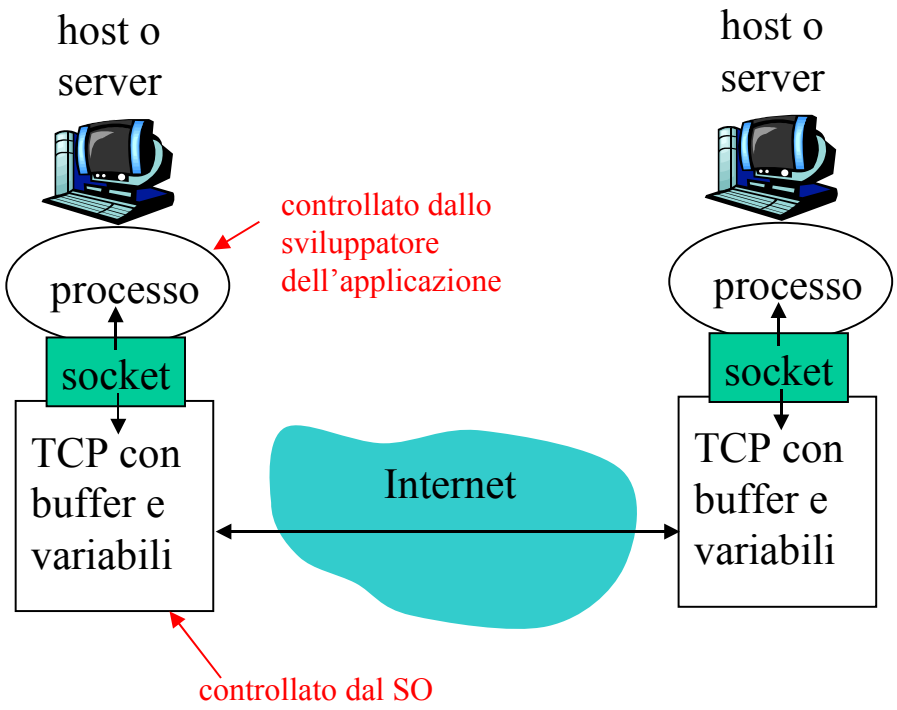
Processo client: processo che dà inizio alla comunicazione

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 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
- Per inviare un messaggio HTTP al server gaia.cs.umass.edu:
  - Indirizzo IP: 128.119.245.12
  - Numero di porta: 80

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

Protocolli proprietari:

- Ad esempio, Skype

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

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
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, 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

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	HTTP (es. YouTube) RTP [RFC 1889]	TCP o UDP
Telefonia Internet	SIP, RTP, proprietario (es. Skype)	Tipicamente UDP

Capitolo 2: Livello di applicazione

- 2.1 Principi delle applicazioni di rete
  - Architetture delle applicazioni
  - Servizi richiesti dalle applicazioni
- 2.2 Web e HTTP
- 2.3 FTP
- 2.4 Posta elettronica
  - SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 Applicazioni P2P
- 2.7 Programmazione delle socket con TCP
- 2.8 Programmazione delle socket con UDP

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
- Esempio di URL:

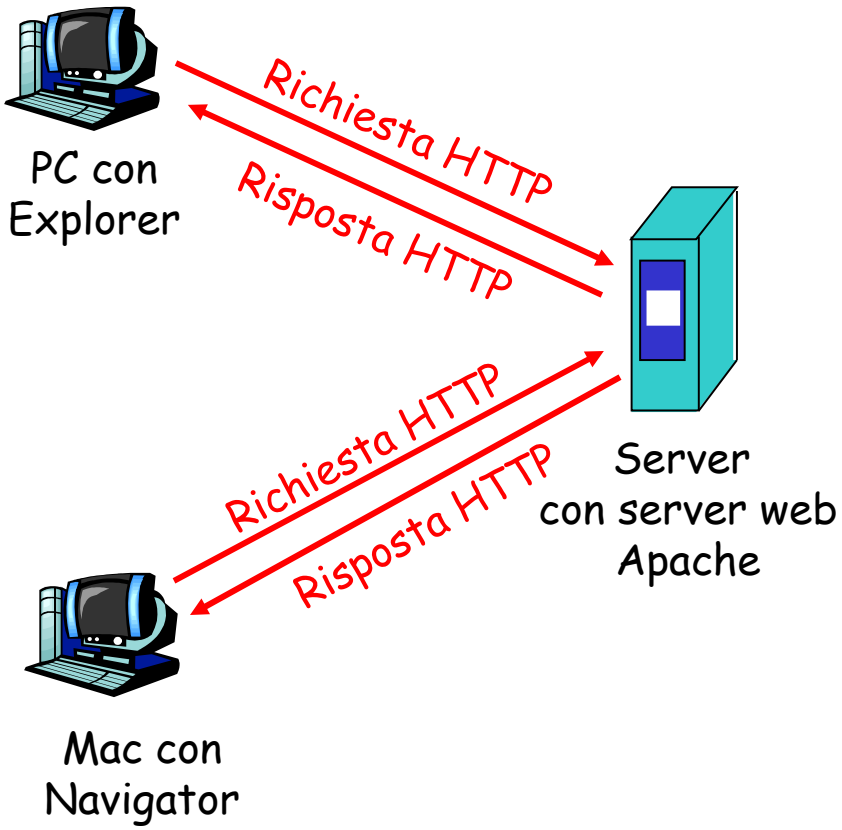
`www.someschool.edu/someDept/pic.gif`

nome dell'host — nome del percorso

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



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

Connessioni persistenti

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

Connessioni non persistenti

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

(contiene testo, riferimenti a 10 immagini jpeg)

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

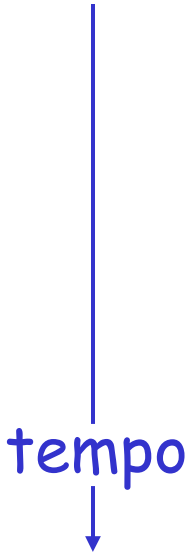
1b. Il server HTTP all'host www.someSchool.edu in attesa di una connessione TCP alla porta 80 "accetta" la connessione e avvisa il client

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

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

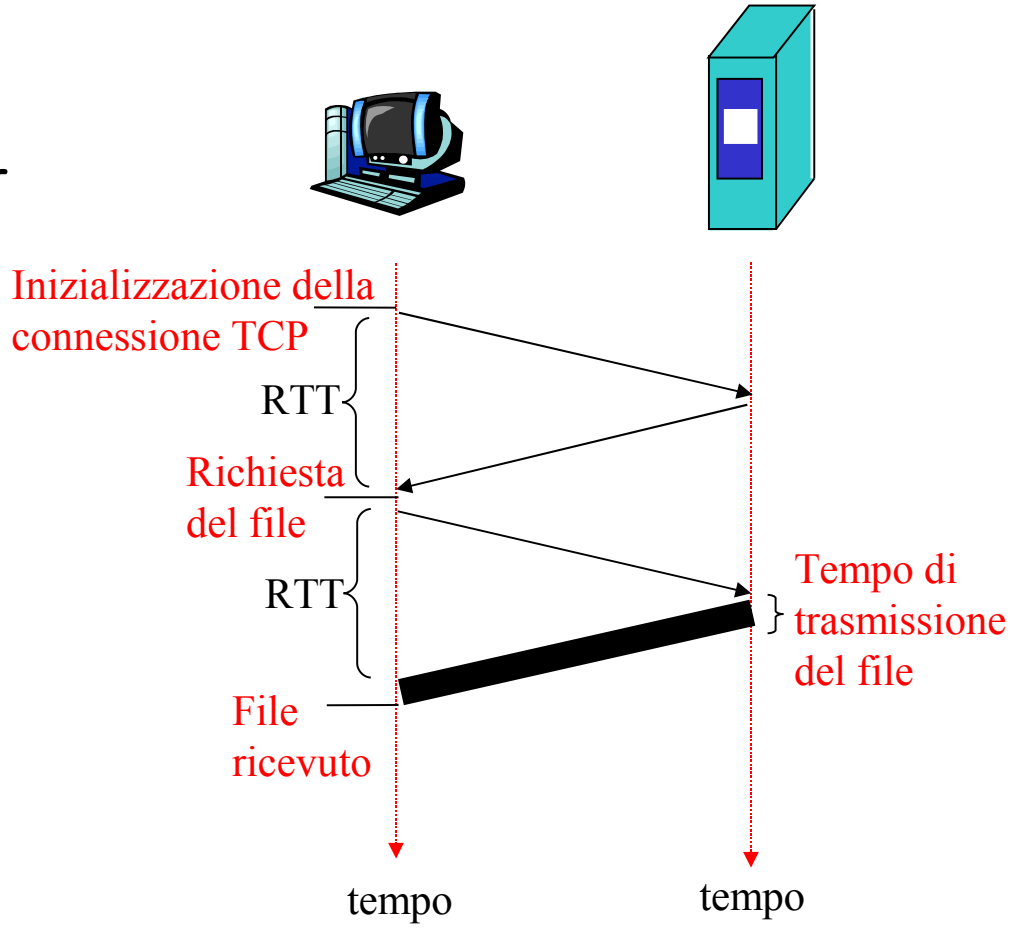
6. I passi 1-5 sono ripetuti per ciascuno dei 10 oggetti jpeg

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:

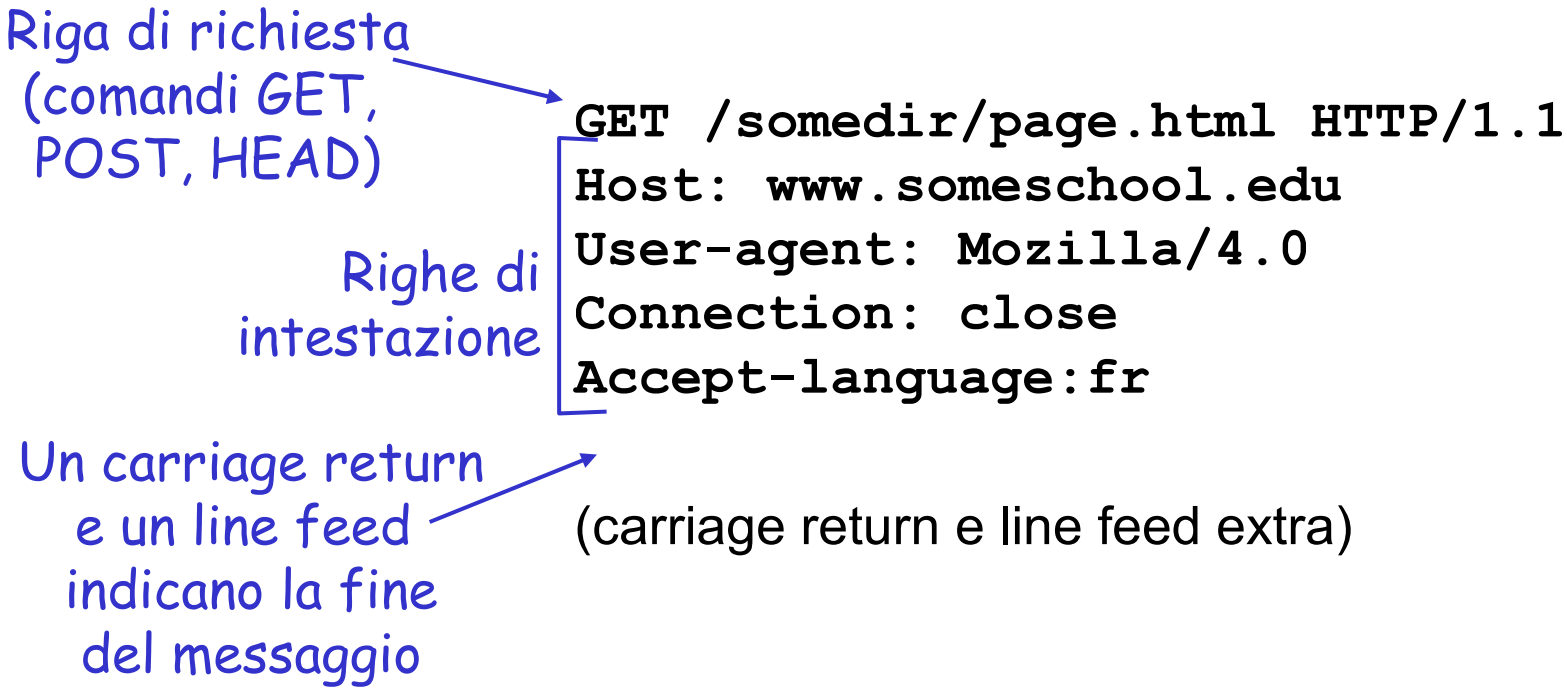
- richiedono 2 RTT per oggetto
- overhead del sistema operativo per *ogni* connessione TCP
- i browser spesso aprono connessioni TCP parallele per caricare gli oggetti referenziati

Connessioni persistenti

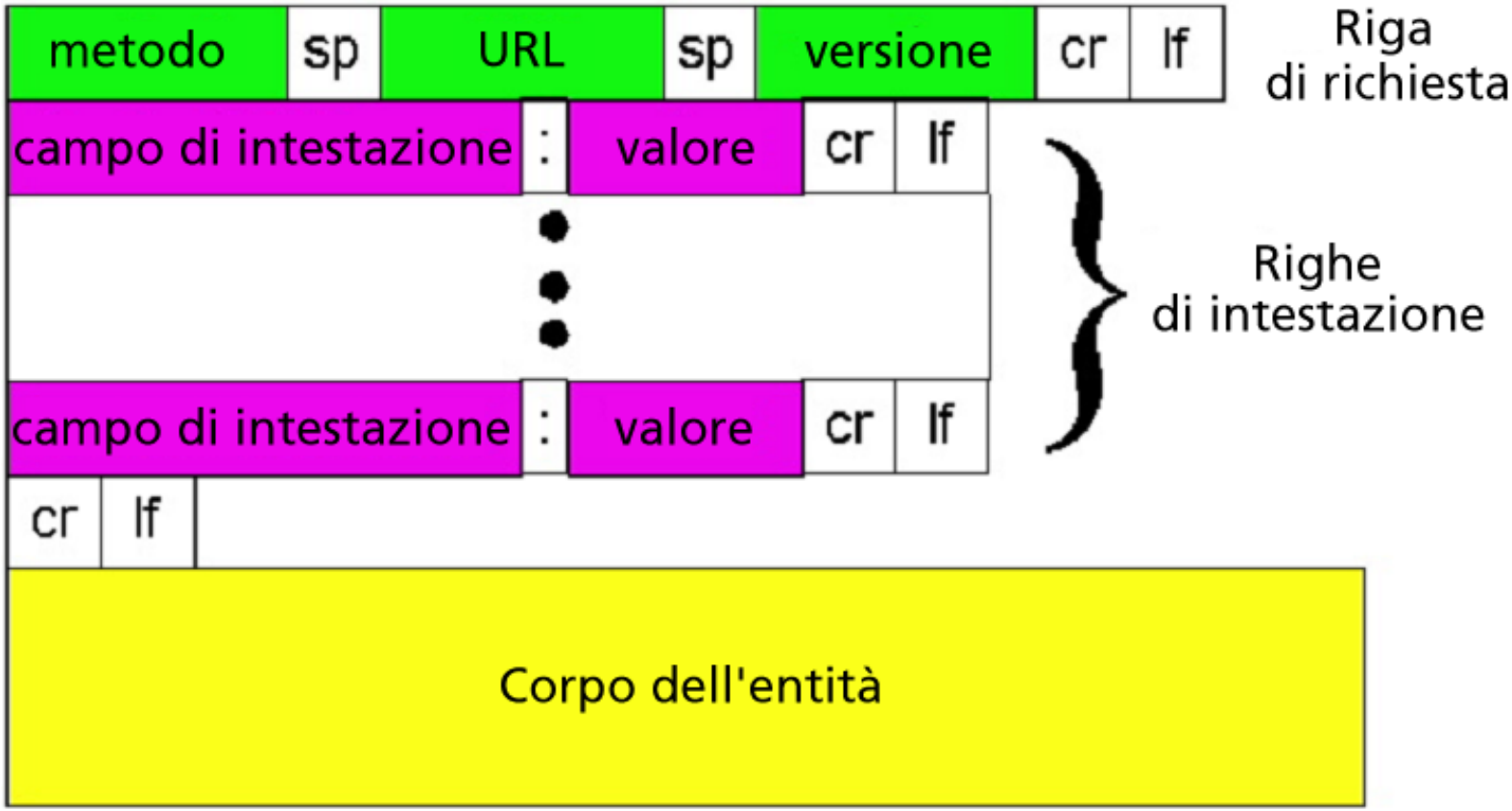
- il server lascia la connessione TCP aperta dopo l'invio di una risposta
- i successivi messaggi tra gli stessi client/server vengono trasmessi sulla connessione aperta
- il client invia le richieste non appena incontra un oggetto referenziato
- un solo RTT per tutti gli oggetti referenziati

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

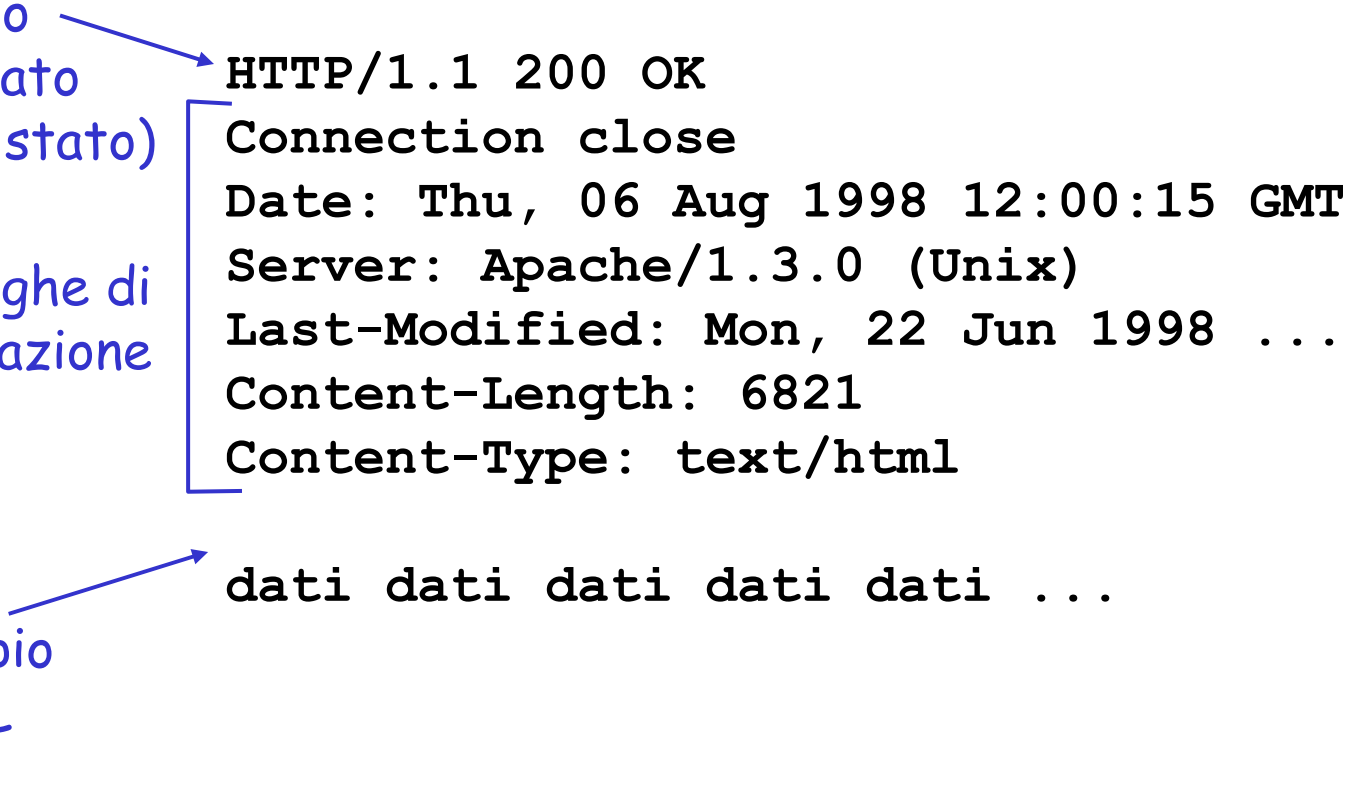
- GET, POST, HEAD
- PUT
  - include il file nel corpo dell'entità e lo invia al percorso specificato nel campo URL
- DELETE
  - cancella il file specificato nel campo URL

Messaggio di risposta HTTP

Riga di stato (protocollo codice di stato espressione di stato)

Righe di intestazione

dati, ad esempio il file HTML richiesto



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

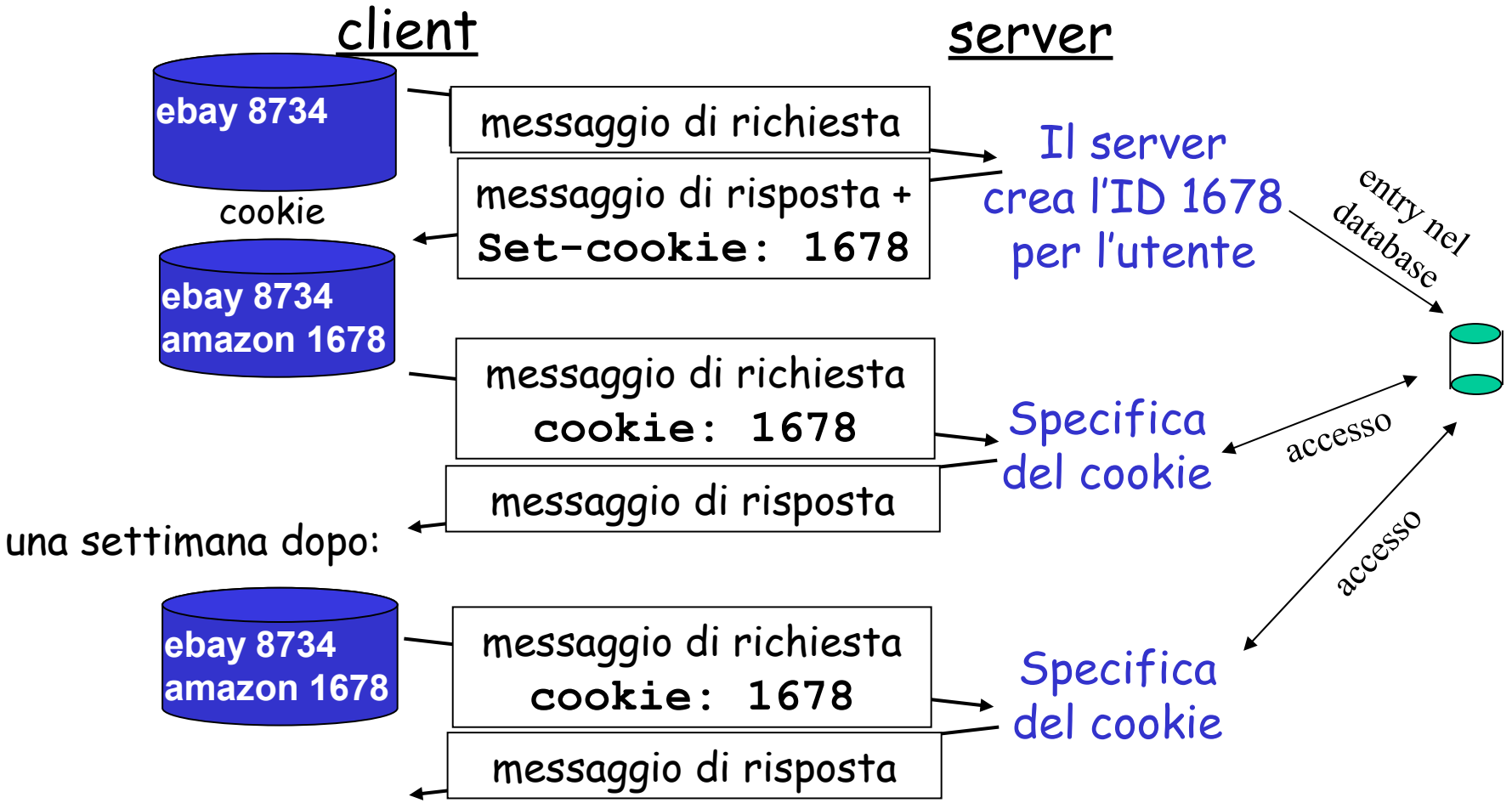
Quattro componenti:

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)

Lo "stato"

- Mantengono lo stato del mittente e del ricevente per più transazioni
- I messaggi http trasportano lo stato

nota

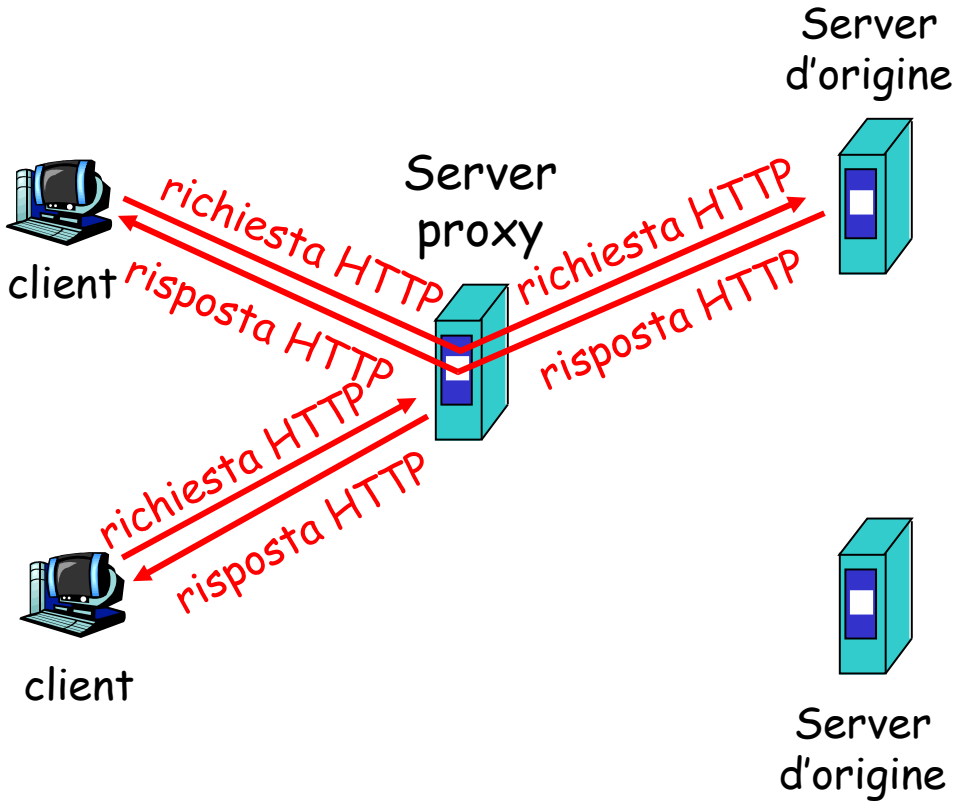
Cookie e privacy:

- i cookie permettono ai siti di imparare molte cose sugli utenti
- l'utente può fornire al sito il nome e l'indirizzo e-mail

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 anche la condivisione di file P2P)

Esempio di caching

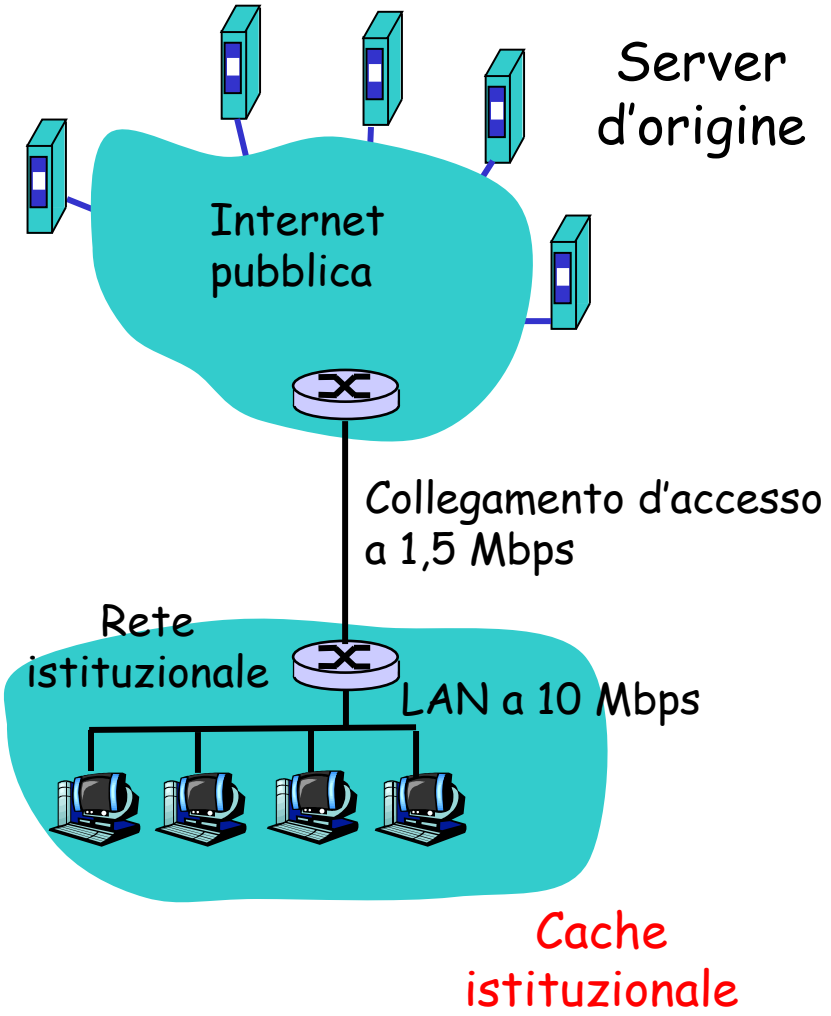
Ipotesi

- 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)

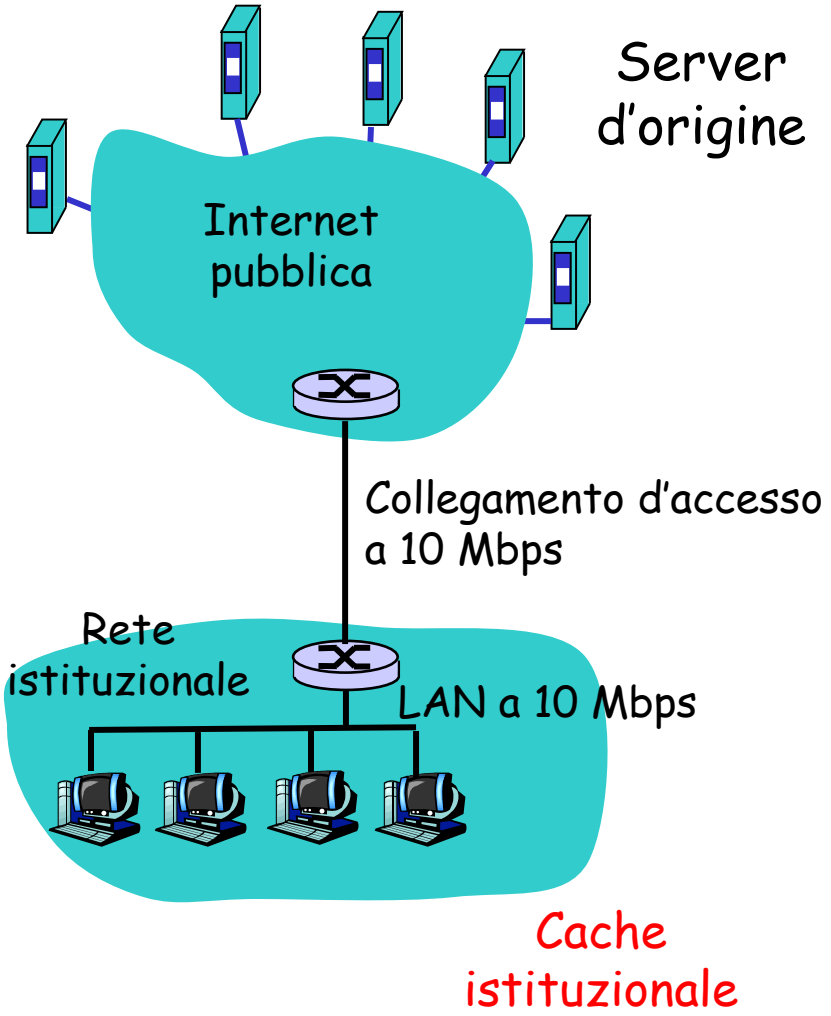
Soluzione possibile

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

Conseguenze

- utilizzo sulla LAN = 15%
- utilizzo 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



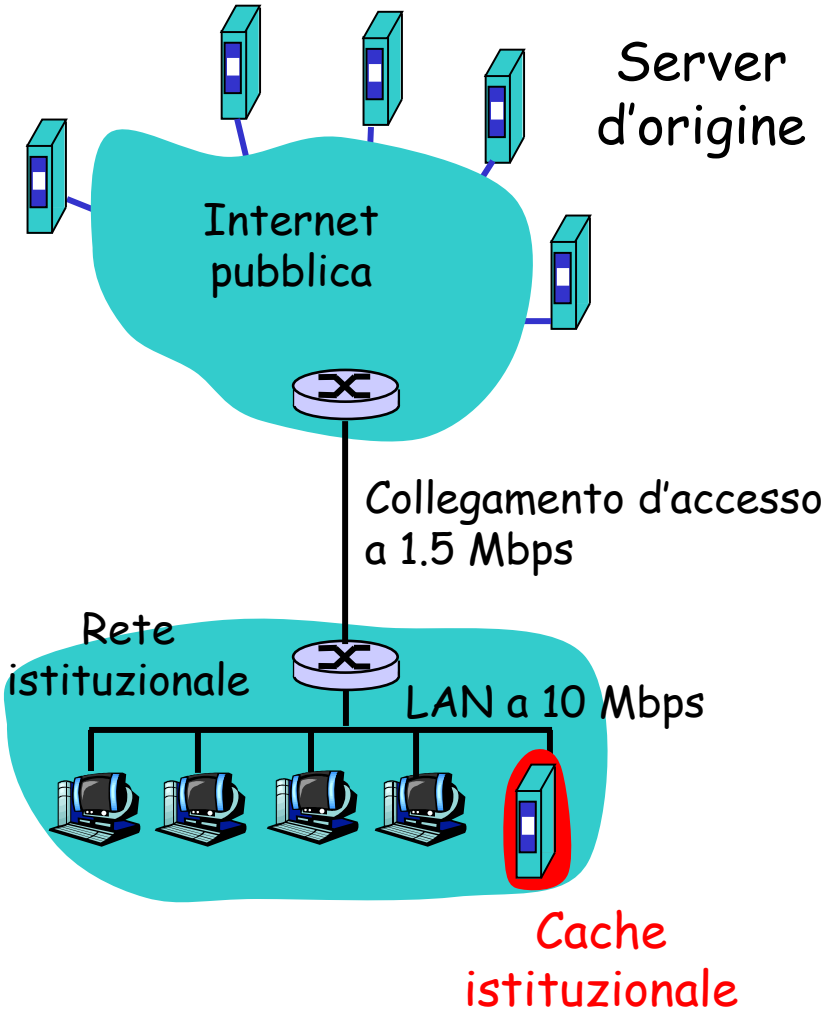
Esempio di caching (continua)

Soluzione possibile: installare la cache

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

Conseguenze

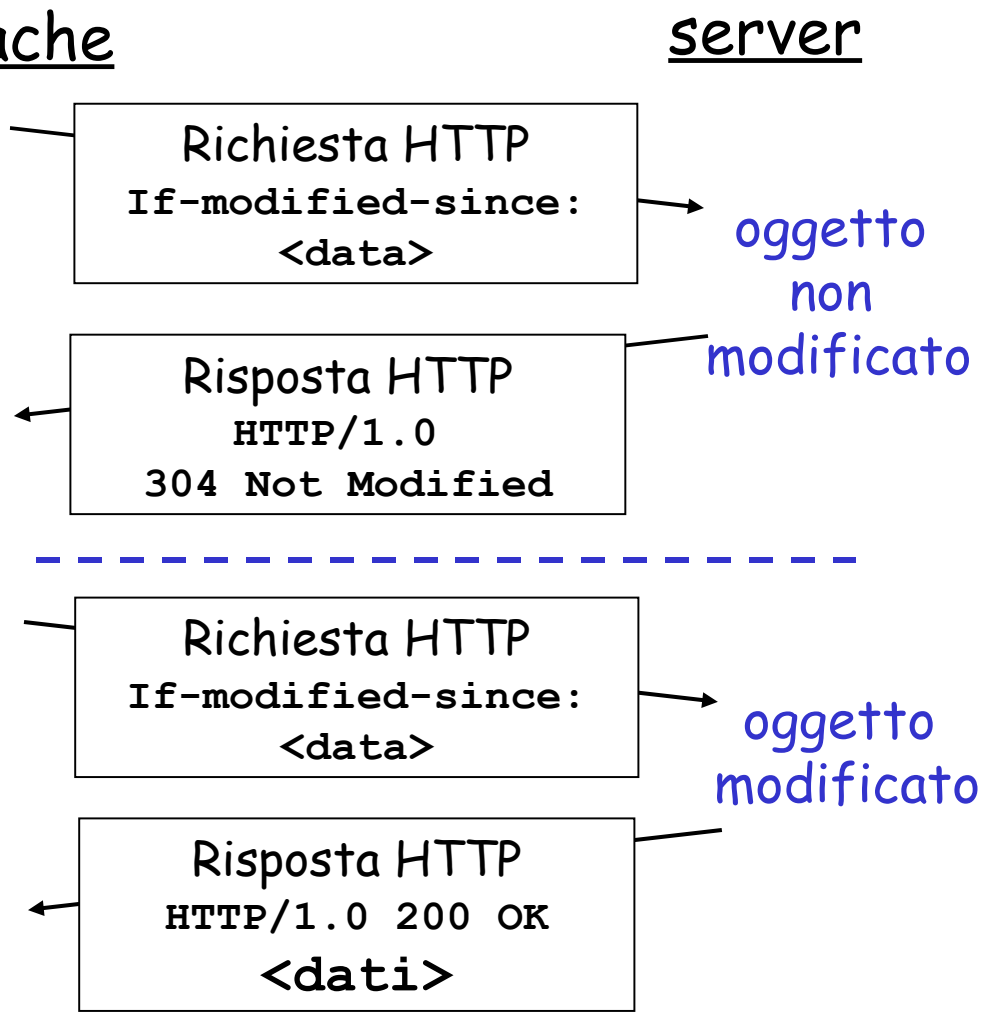
- il 40% delle richieste sarà soddisfatto quasi immediatamente
- il 60% delle richieste sarà soddisfatto dal server d'origine
- l'utilizzazione del collegamento d'accesso si è ridotta al 60%, determinando ritardi trascurabili (circa 10 msec)
- ritardo totale medio = ritardo di Internet + ritardo di accesso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



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

GET condizionale

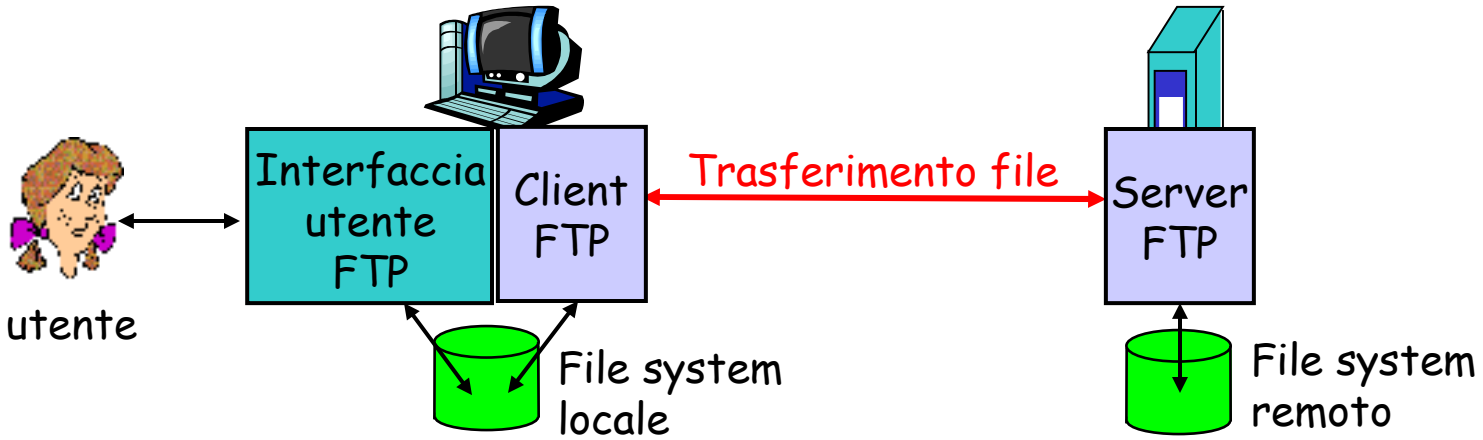
- Obiettivo: non inviare un oggetto se la cache ha una copia aggiornata dell'oggetto
- cache: specifica la data della copia dell'oggetto nella richiesta HTTP
  - `If-modified-since: <data>`
- server: la risposta non contiene l'oggetto se la copia nella cache è aggiornata:
  - `HTTP/1.0 304 Not Modified`



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 Applicazioni P2P
- 2.7 Programmazione delle socket con TCP
- 2.8 Programmazione delle socket con UDP

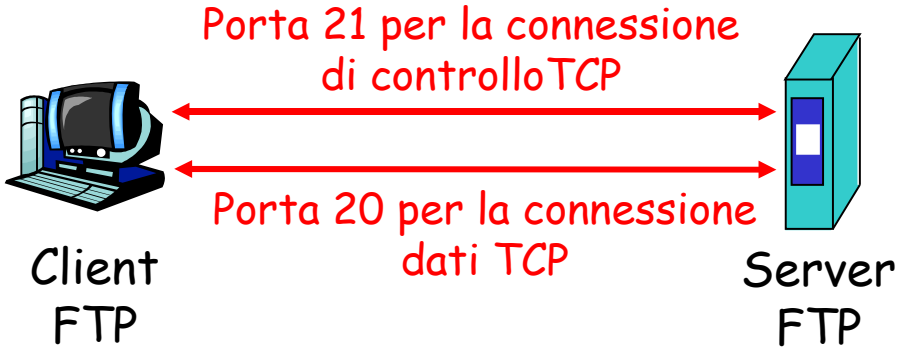
FTP: file transfer protocol



- 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 Applicazioni P2P
- 2.7 Programmazione delle socket con TCP
- 2.8 Programmazione delle socket con UDP

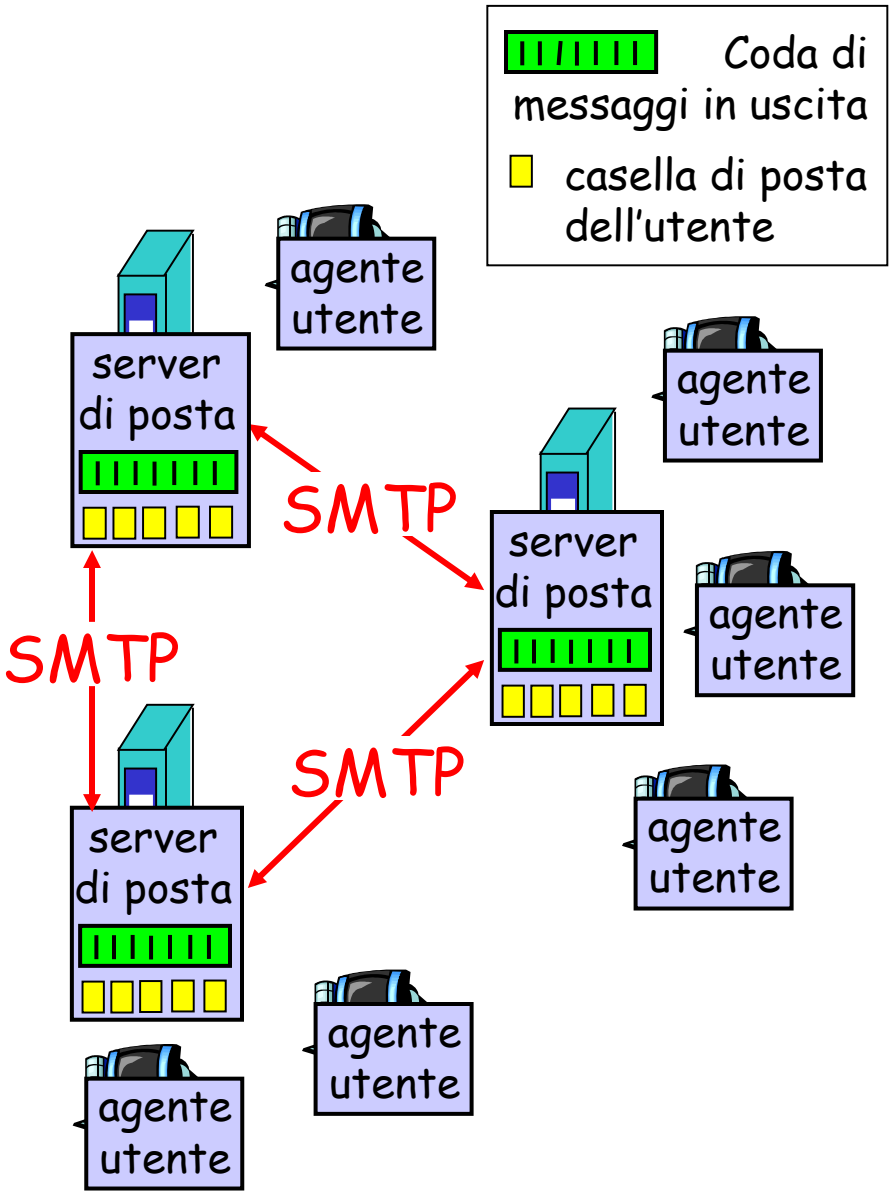
Posta elettronica

Tre componenti principali:

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

Agente utente

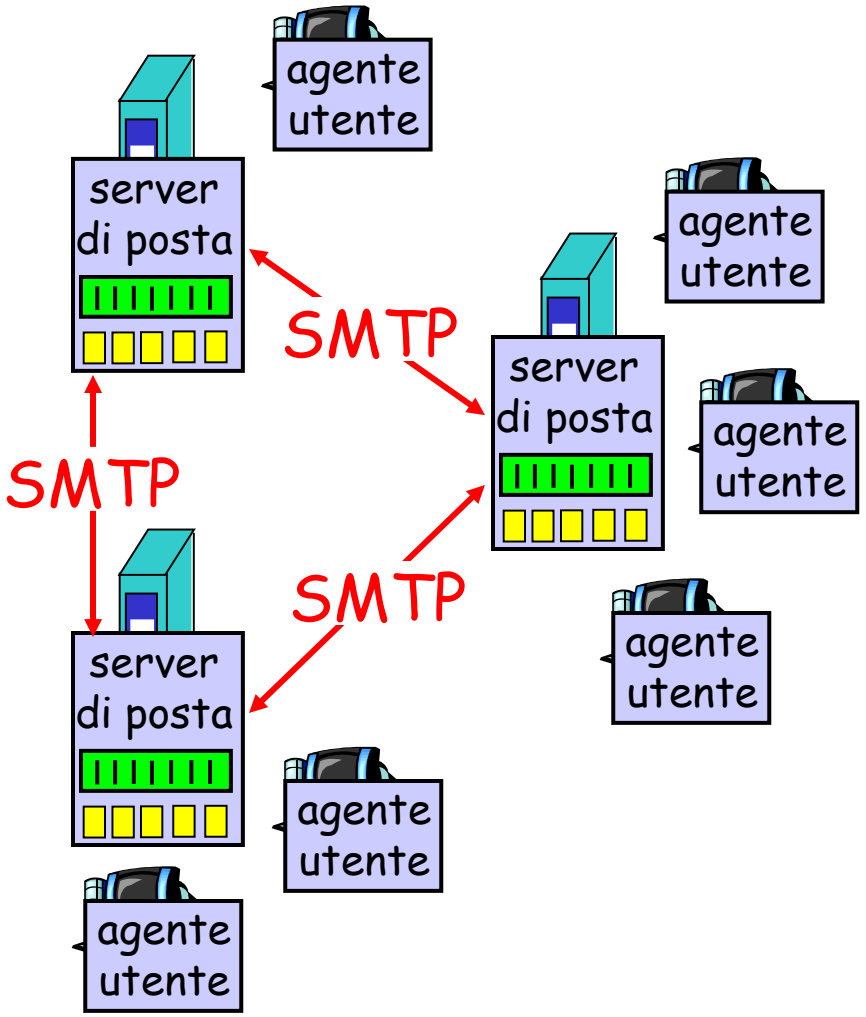
- detto anche "mail reader"
- composizione, editing, lettura dei messaggi di posta elettronica
- esempi: Eudora, Outlook, elm, Mozilla, Thunderbird
- i messaggi in uscita o in arrivo sono memorizzati sul server



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

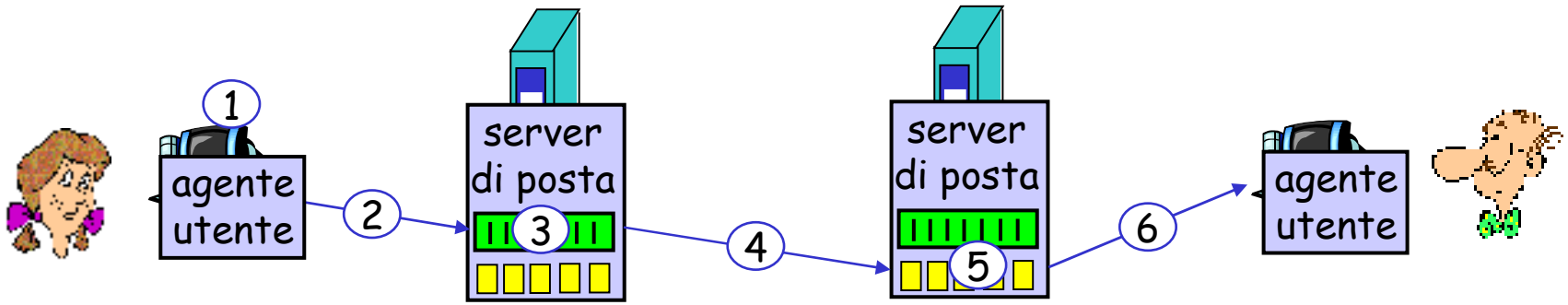


Posta elettronica: SMTP [RFC 2821]

- 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 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

Scenario: Alice invia un messaggio a Roberto

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



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: <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
```

Provate un'interazione SMTP:

- `telnet servername 25`
- Riceverete la risposta 220 dal server
- Digitate i comandi HELO, MAIL FROM, RCPT TO, DATA, QUIT

Questo vi consente di inviare messaggi di posta elettronica senza usare il client di posta (lettore)

SMTP: note finali

- 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

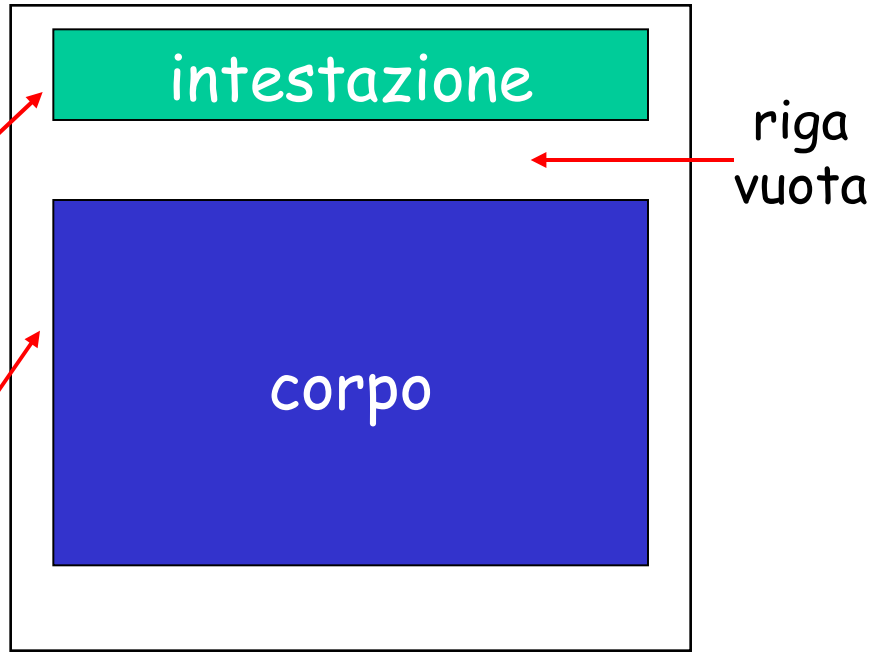
Formato dei messaggi di posta elettronica

SMTP: protocollo per scambiare messaggi di posta elettronica

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

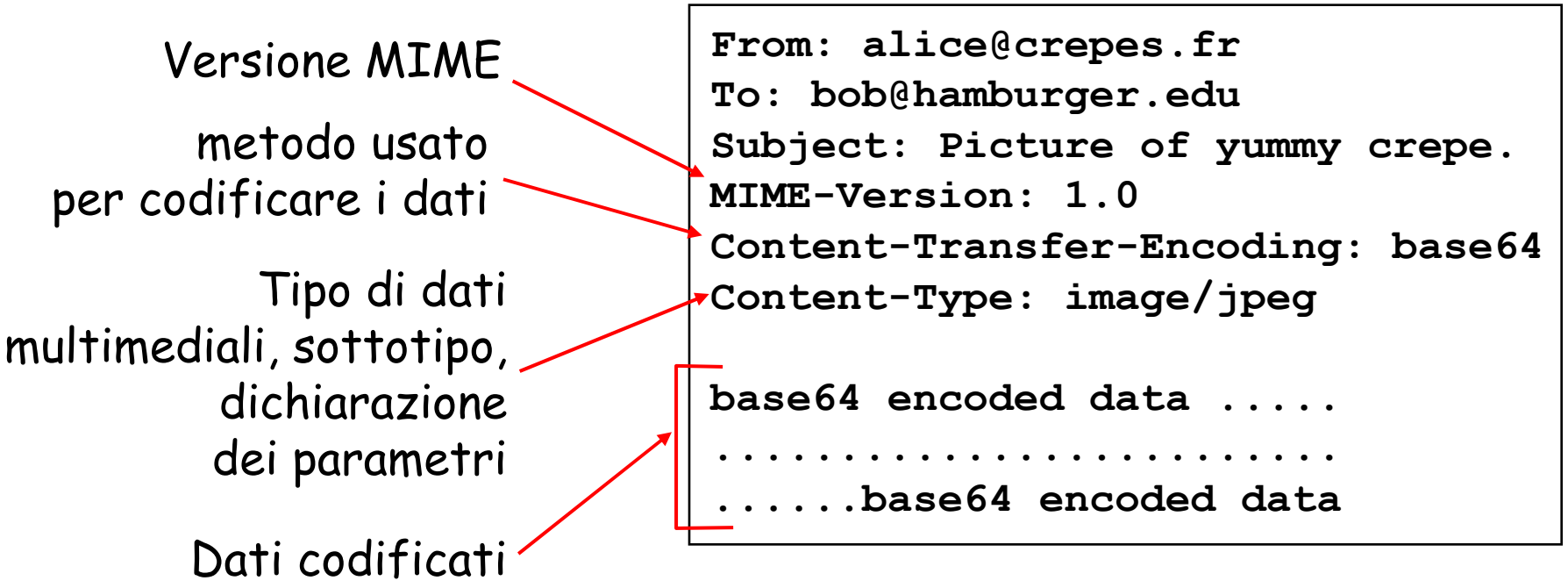
- Righe di intestazione, per esempio
  - To/A:
  - From/Da:
  - Subject/Oggetto:

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

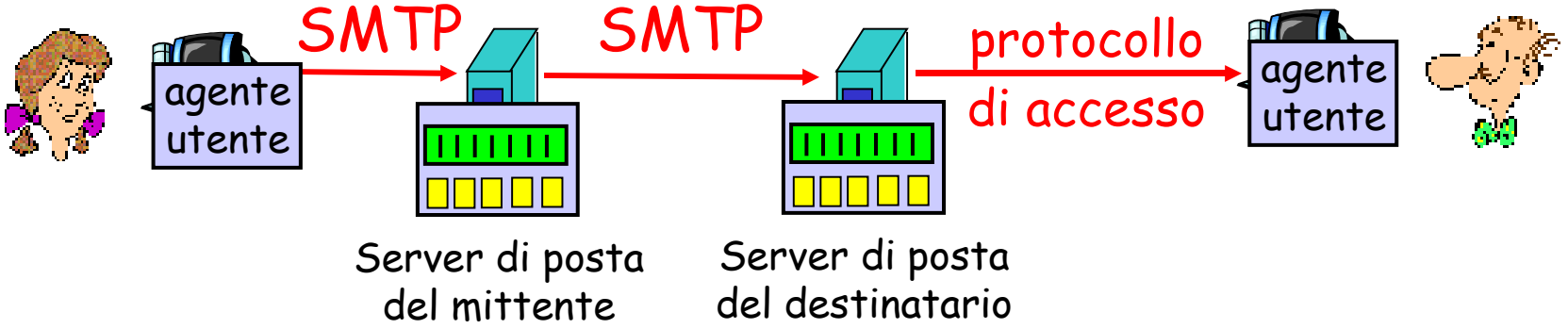


Formato del messaggio: estensioni di messaggi multimediali

- 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: gmail, 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 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
```

POP3 (altro) e IMAP

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 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 Applicazioni P2P
- 2.7 Programmazione delle socket con TCP
- 2.8 Programmazione delle socket con UDP

DNS: Domain Name System

Persone: molti identificatori:

- nome, codice fiscale, numero della 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)
  - Si noti: funzioni critiche di Internet implementate come protocollo a livello di applicazione
  - complessità nelle parti periferiche della rete

DNS

Servizi DNS

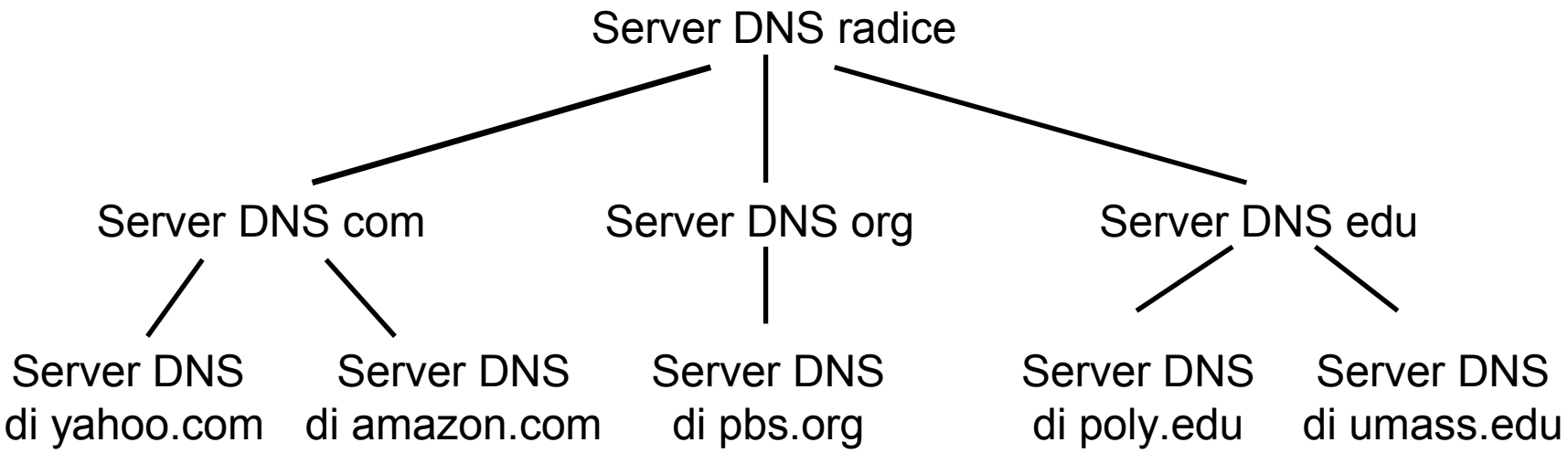
- Traduzione degli hostname in indirizzi IP
- Host aliasing
  - un host può avere più nomi
- Mail server aliasing
- Distribuzione locale
  - server web replicati: insieme di indirizzi IP per un nome canonico

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

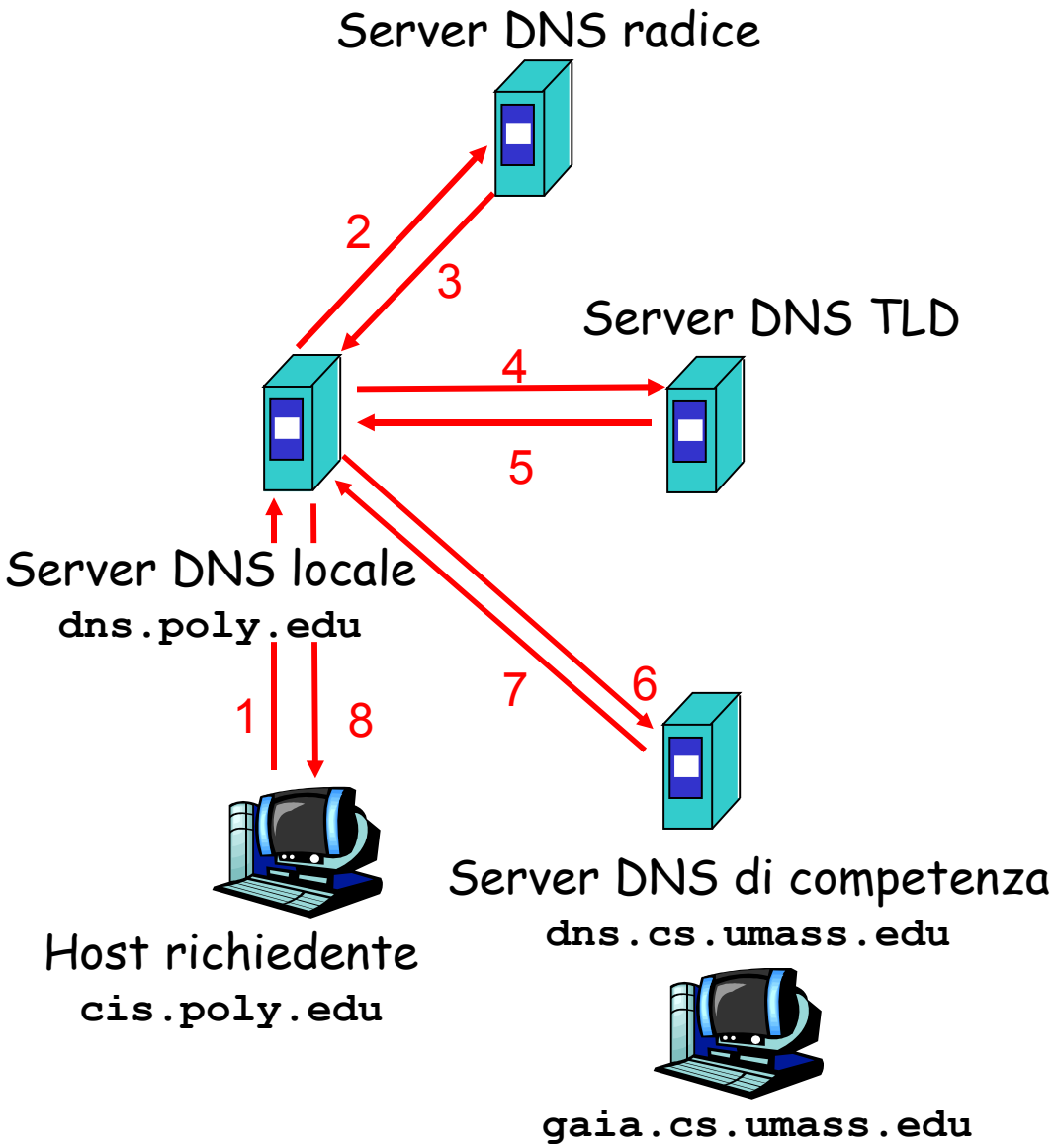
- Non appartiene strettamente alla gerarchia dei server
- Ciascun ISP (università, società, ISP residenziale) ha un server DNS locale.
  - detto anche "default name server"
- Quando un host effettua una richiesta DNS, la query viene inviata al suo server DNS locale
  - il server DNS locale opera da proxy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

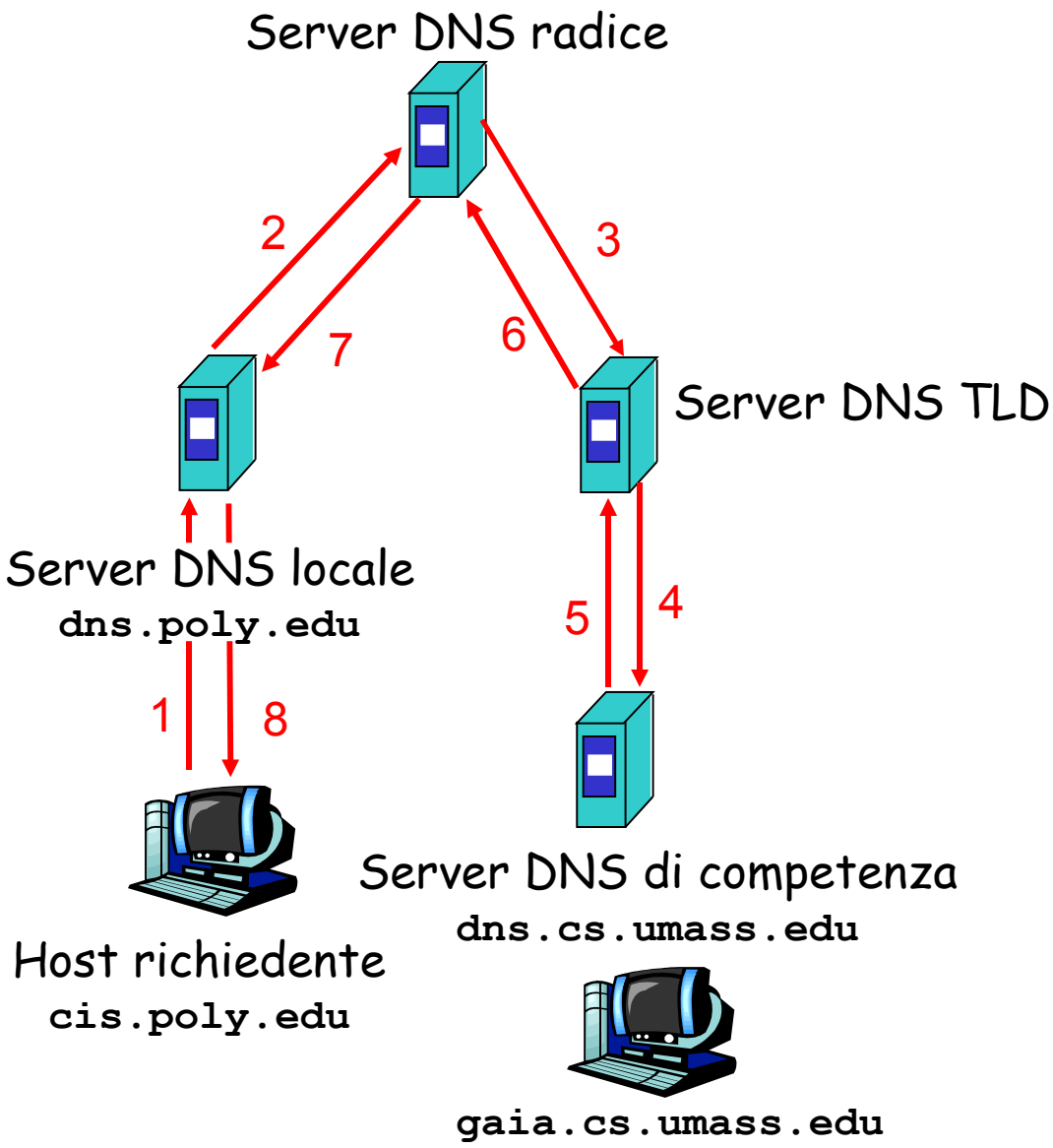
- Il server contattato risponde con il nome del server da contattare
- "Io non conosco questo nome, ma puoi chiederlo a questo server".



Esempio

Query ricorsiva:

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



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)

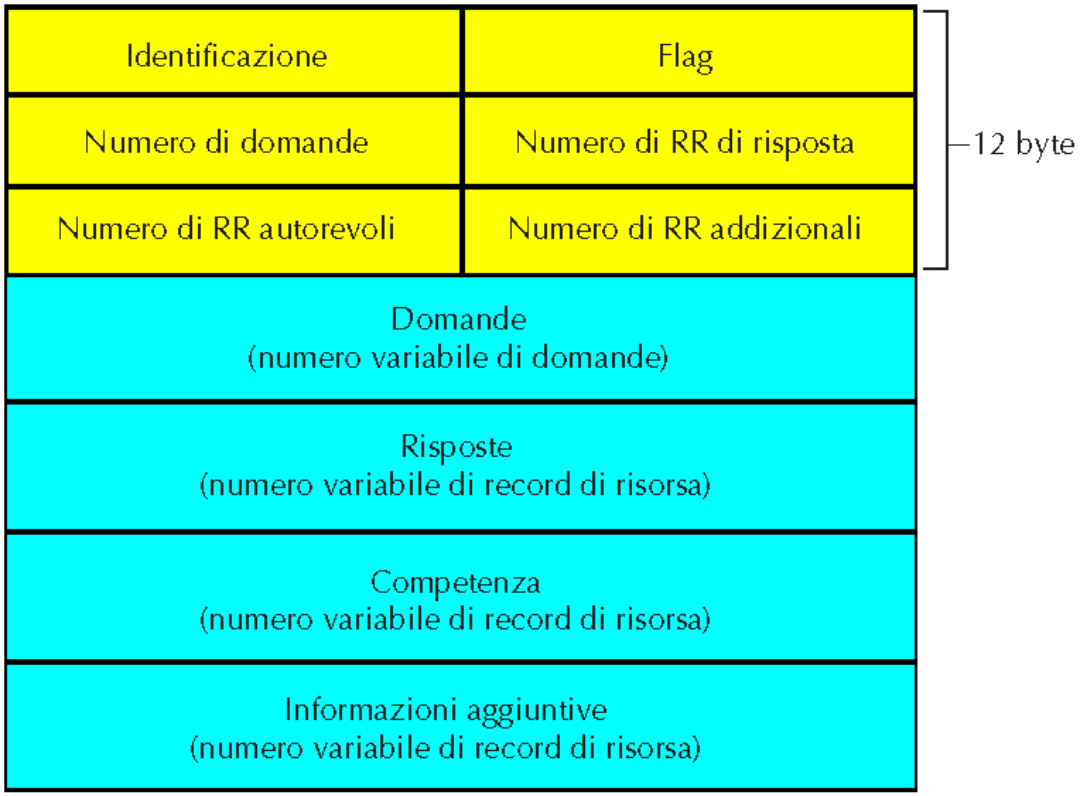
    `www.ibm.com` è in realtà `servereast.backup2.ibm.com`
  - **value** è il nome canonico
- Type=MX
  - **value** è il nome del server di posta associato a **name**

Messaggi DNS

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

Intestazione del messagggio

- 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



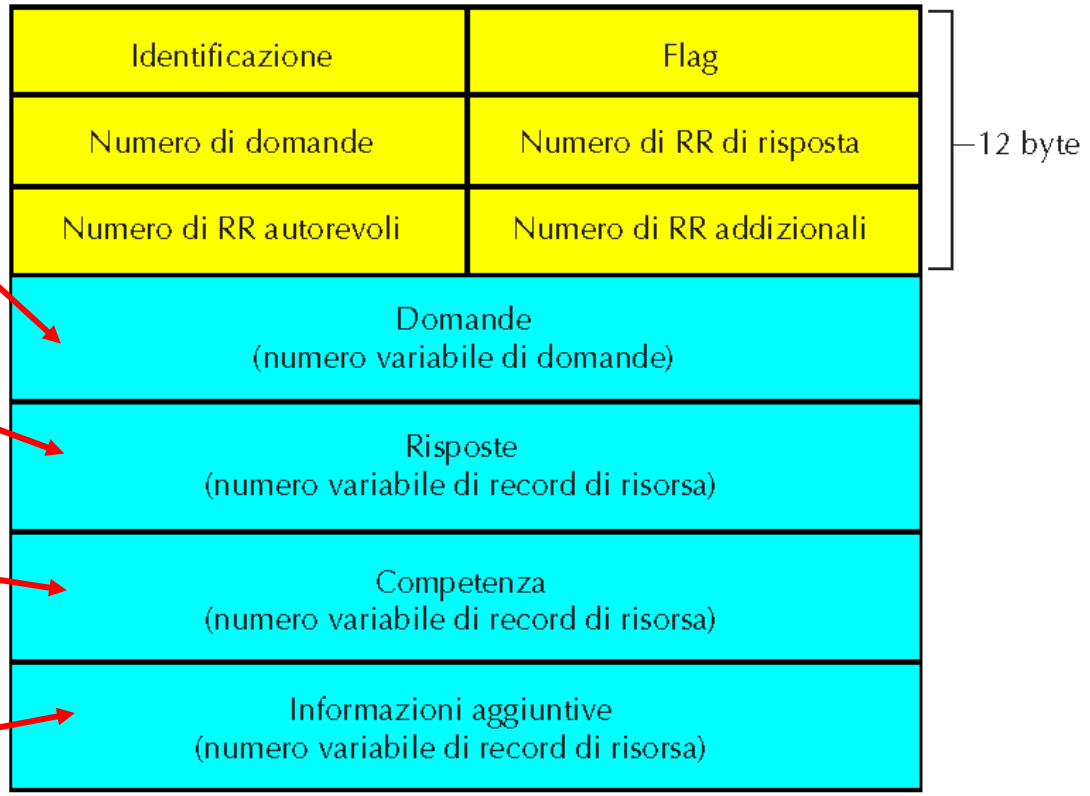
Messaggi DNS

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



Inserire record nel database DNS

- Esempio: abbiamo appena avviato la nuova società "Network Utopia"
- Registriamo il nome **networkuptopia.com** presso registrar (ad esempio, Network Solutions)
  - Forniamo a registrar i nomi e gli indirizzi IP dei server DNS di competenza (primario e secondario)
  - Registrar inserisce due RR nel server TLD com:

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

- Inseriamo nel server di competenza un record tipo A per www.networkuptopia.com e un record tipo MX per networkutopia.com
- In che modo gli utenti otterranno l'indirizzo IP del nostro sito web?