Reti di calcolatori e Internet: Un approccio top-down

7ª edizione
Jim Kurose, Keith Ross

Pearson Paravia Bruno Mondadori Spa

Capitolo 2: Livello di applicazione

- 2.1 Principi delle applicazioni di rete
- 2.2 Web e HTTP
- 2.3 Posta elettronica
  - SMTP, POP3, IMAP
- 2.4 DNS
- 2.5 Applicazioni 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

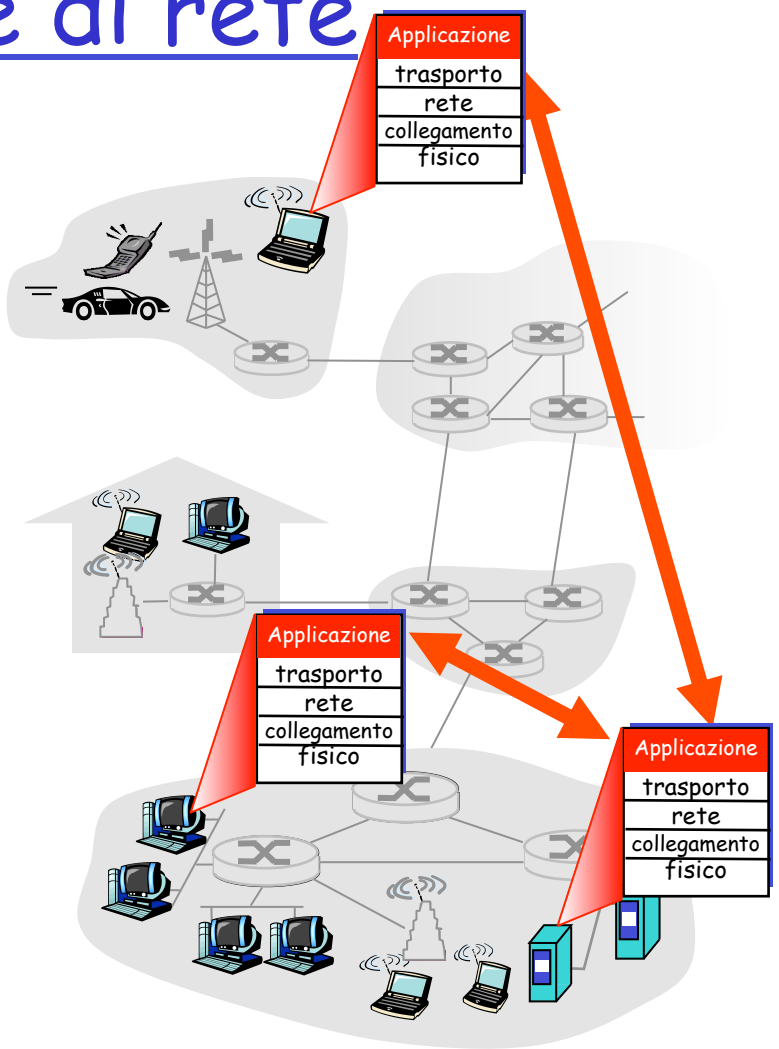
Creare un'applicazione di rete

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



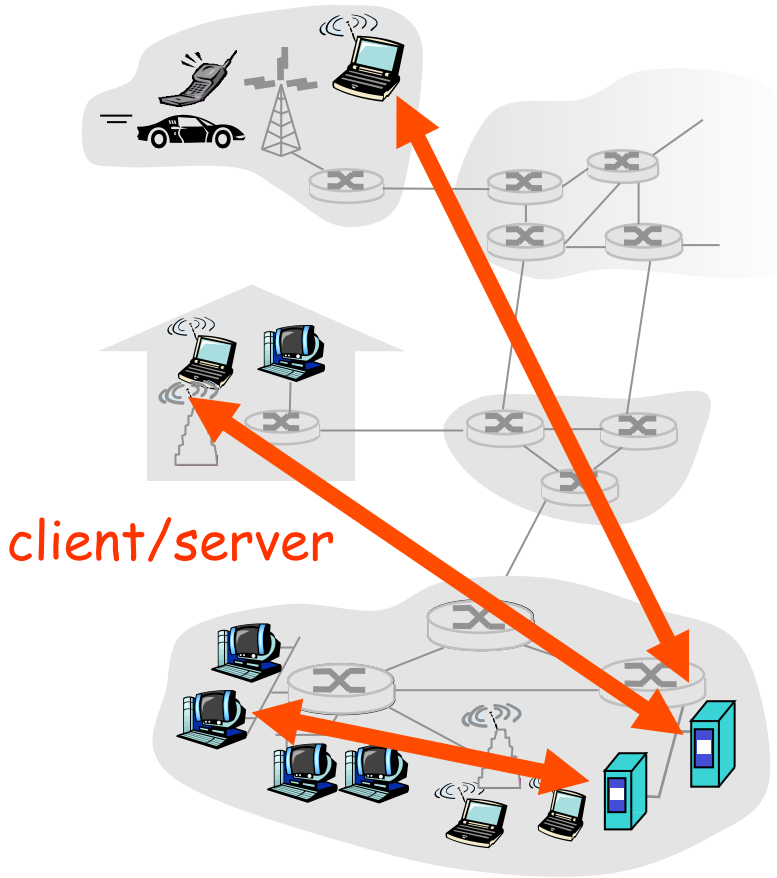
Capitolo 2: Livello di applicazione

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

Architettura delle applicazioni di rete

- Stabilisce l'organizzazione dell'applicazione sui diversi sistemi terminali
- 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

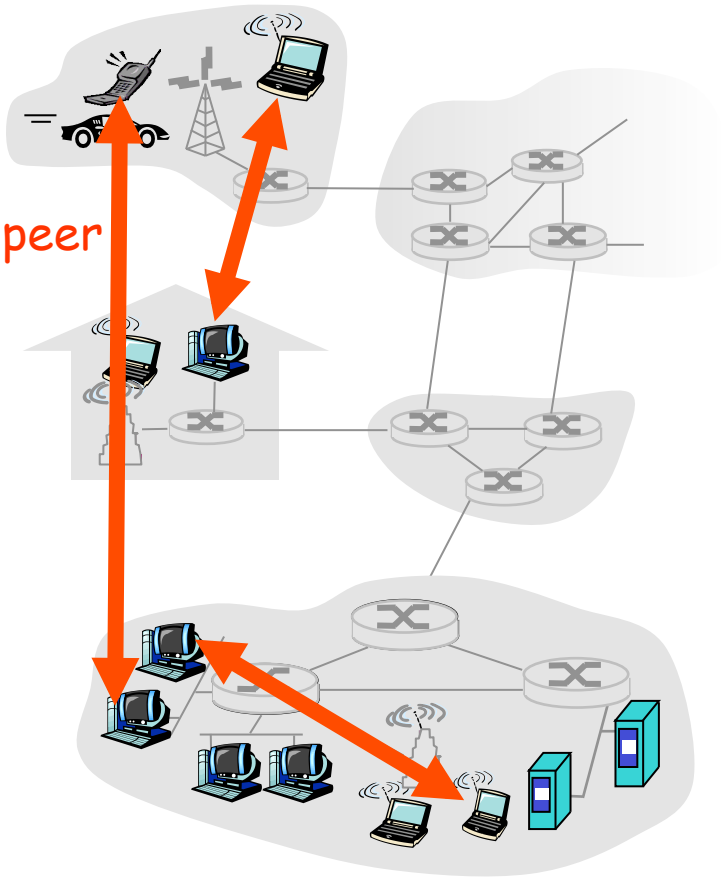
client:

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

- Uso intensivo dell'infrastruttura
  - acquisto, installazione, manutenzione, costi di banda

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
- Non richiede una infrastruttura di server significativa
- Difficile da gestire (sicurezza)



Ibridi (client-server e P2P)

Infrastruttura minima di server

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

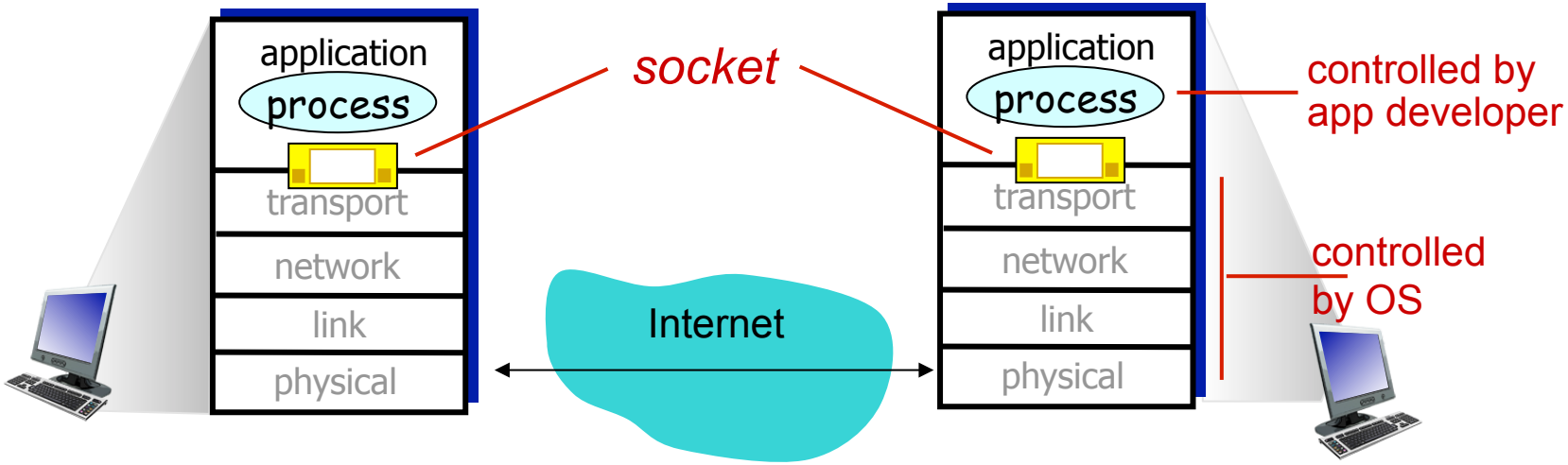
Processo client: processo che dà inizio alla comunicazione

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 (interfaccia sw)
- 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



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
- Numeri di porta noti
  - Internet Assigned Numbers Authority
  - www.iana.org

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?

Trasferimento dati affidabile (no 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
- Vincoli sui ritardi end-to-end

Throughput

- alcune applicazioni (ad esempio, quelle multimediali) per essere "efficaci" richiedono un'ampiezza di banda minima (sensibili alla banda)
- 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 (handshaking-connessione-chiusura)
- *trasporto affidabile* fra i processi d'invio e di ricezione (no errori e perdite, garantito l'ordine)
- *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** → Secure Socket Layer (SSL)

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 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
- 2.2 Web e HTTP
- 2.3 Posta elettronica
  - SMTP, POP3, IMAP
- 2.4 DNS
- 2.5 Applicazioni P2P

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)*
- Esempio di URL:

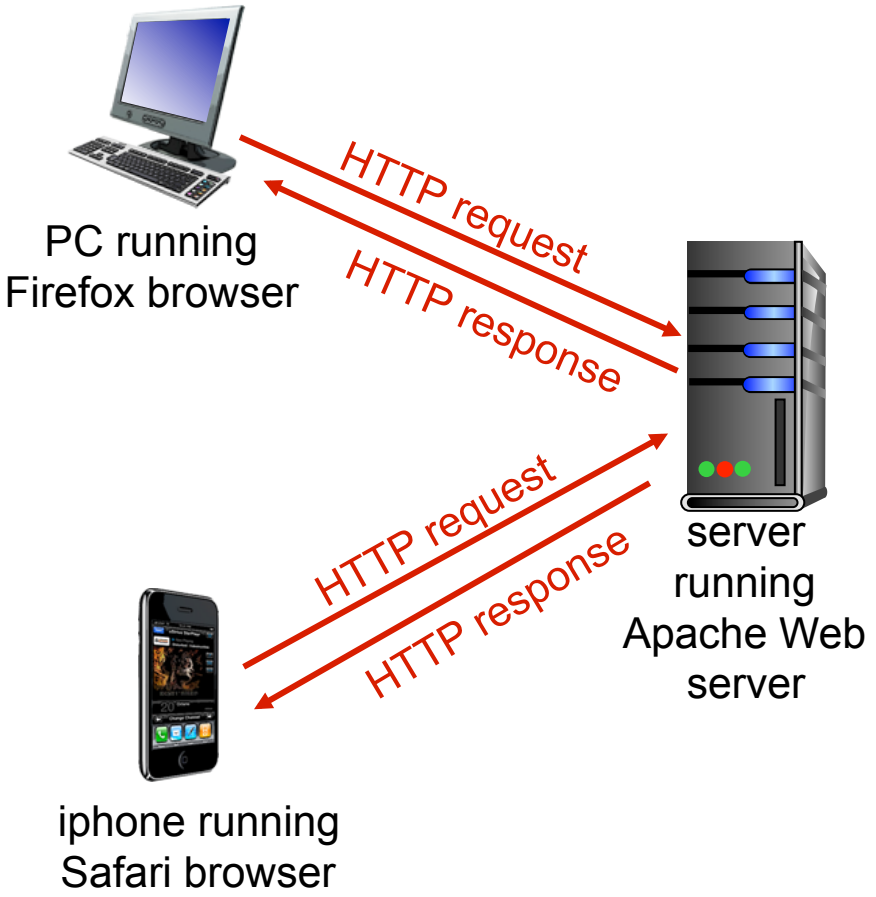
`www.someschool.edu` / `someDept/pic.gif`

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

- Solo un oggetto viene trasmesso su una connessione TCP
- Per richiedere più oggetti è necessario fare più richieste

Connessioni persistenti

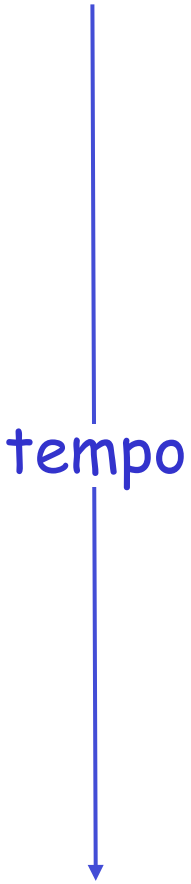
- Più oggetti possono essere trasmessi su una singola connessione TCP tra client e server

Connessioni non persistenti

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

(contiene testo, riferimenti a 10 immagini jpeg)

Connessioni non persistenti (continua)



4. Il server HTTP chiude la connessione TCP (quando è certo che il client ha ricevuto correttamente la risposta)

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

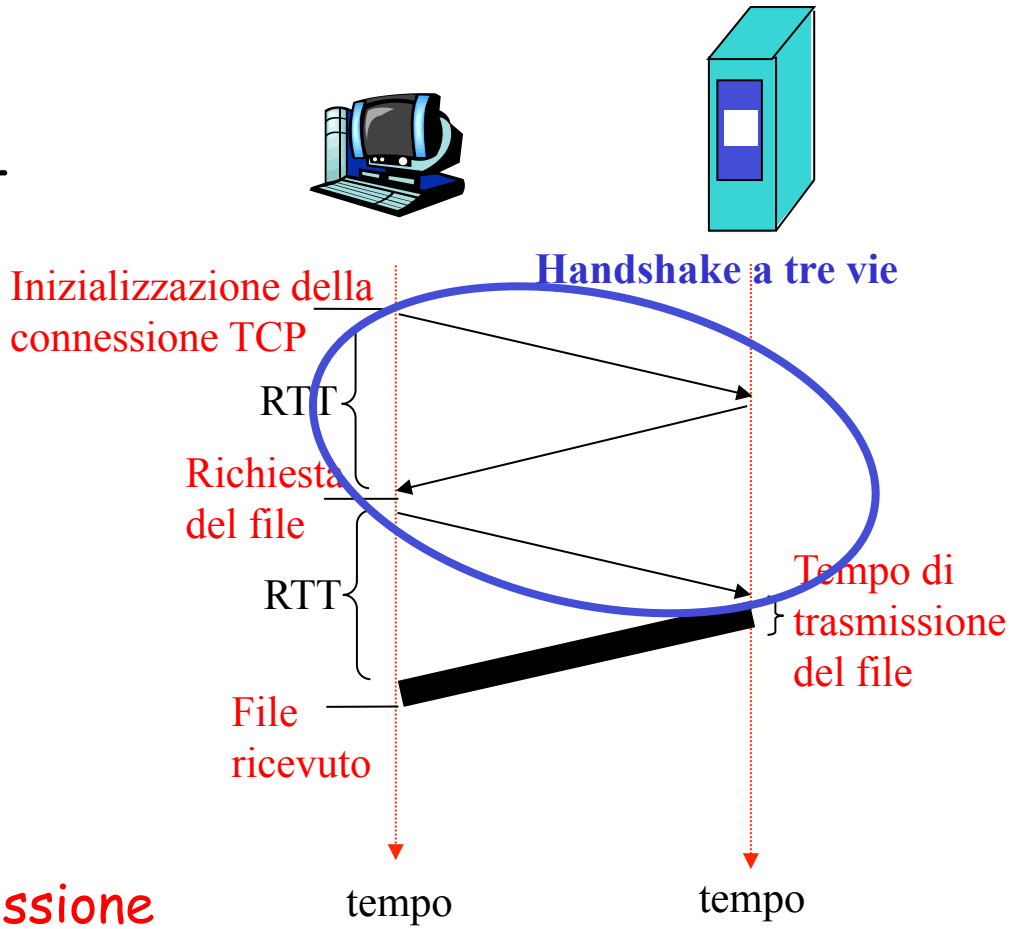
6. I passi 1-5 sono ripetuti per ciascuno dei 10 oggetti jpeg

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

N.B. Il RTT include i ritardi di propagazione, di accodamento e di elaborazione

Connessioni persistenti

Svantaggi delle connessioni non persistenti:

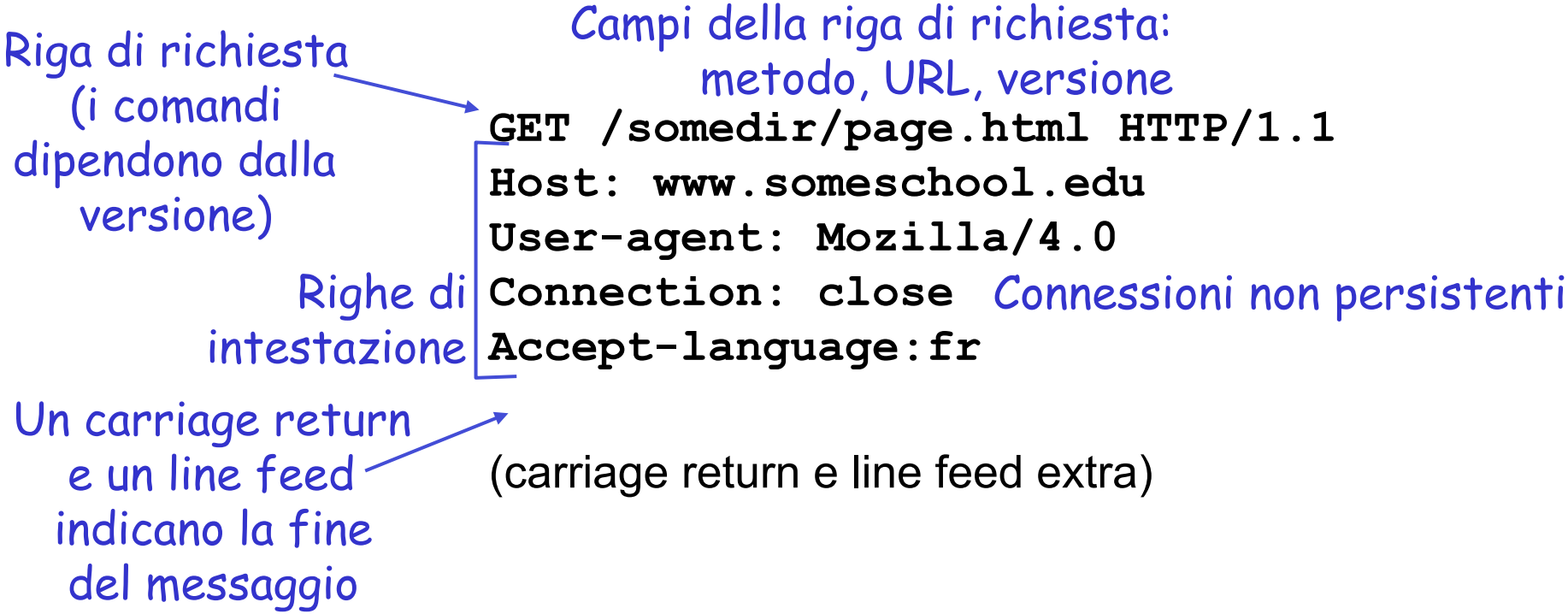
- richiedono 2 RTT per oggetto
- overhead del sistema operativo per *ogni* connessione TCP (lato server)
- i browser spesso aprono connessioni TCP parallele per caricare gli oggetti referenziati

Connessioni persistenti

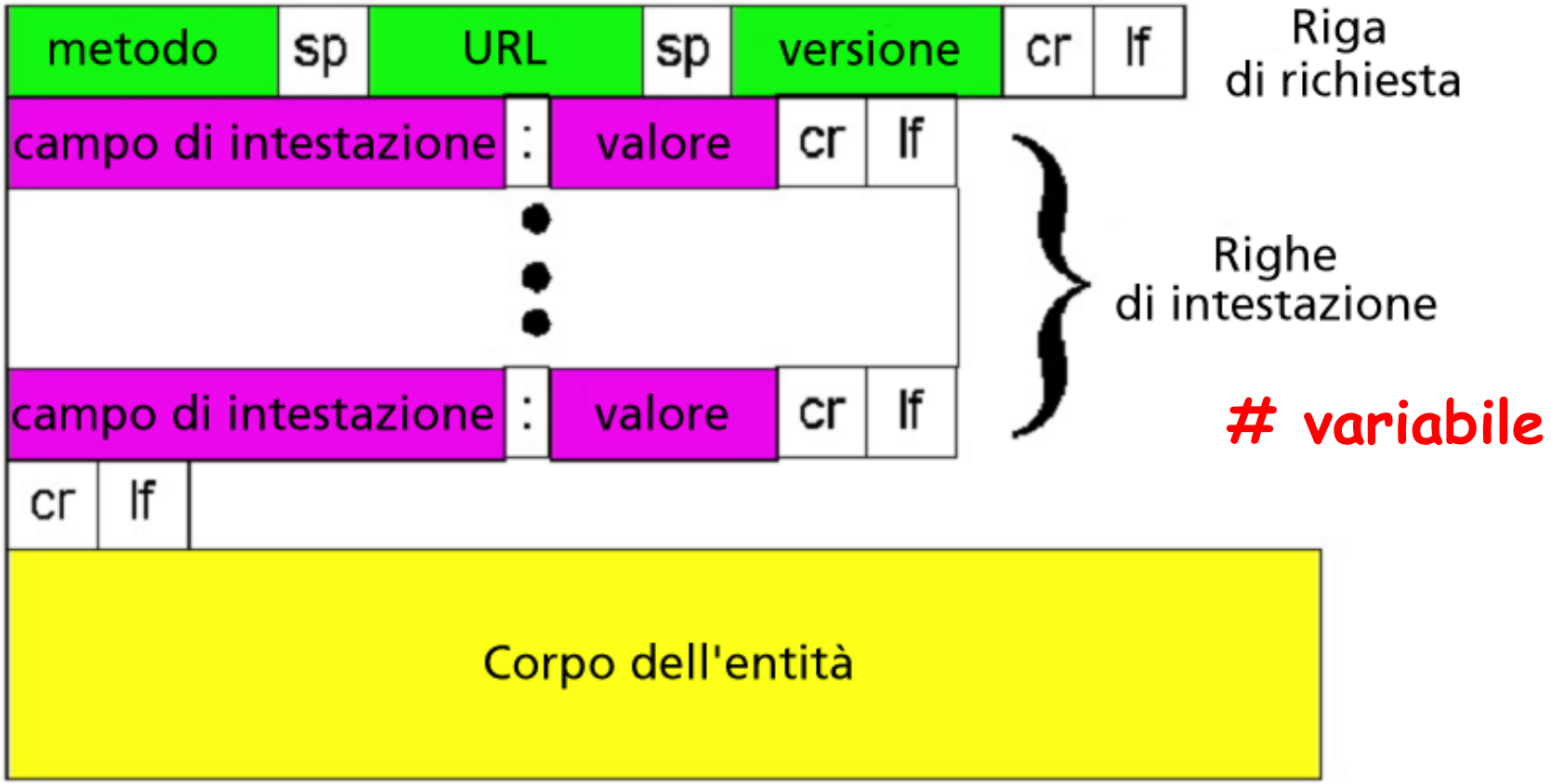
- il server lascia la connessione TCP aperta dopo l'invio di una risposta
- i successivi messaggi tra gli stessi client/server vengono trasmessi sulla connessione aperta
- il client invia le richieste non appena incontra un oggetto referenziato
- un solo RTT per tutti gli oggetti referenziati
- è possibile inviare le richieste *back-to-back (pipelining)*
- La connessione viene chiusa quando rimane inattiva oltre un timeout

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
- Si tratta sempre di una richiesta, ma dipende dall'input

Campo 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

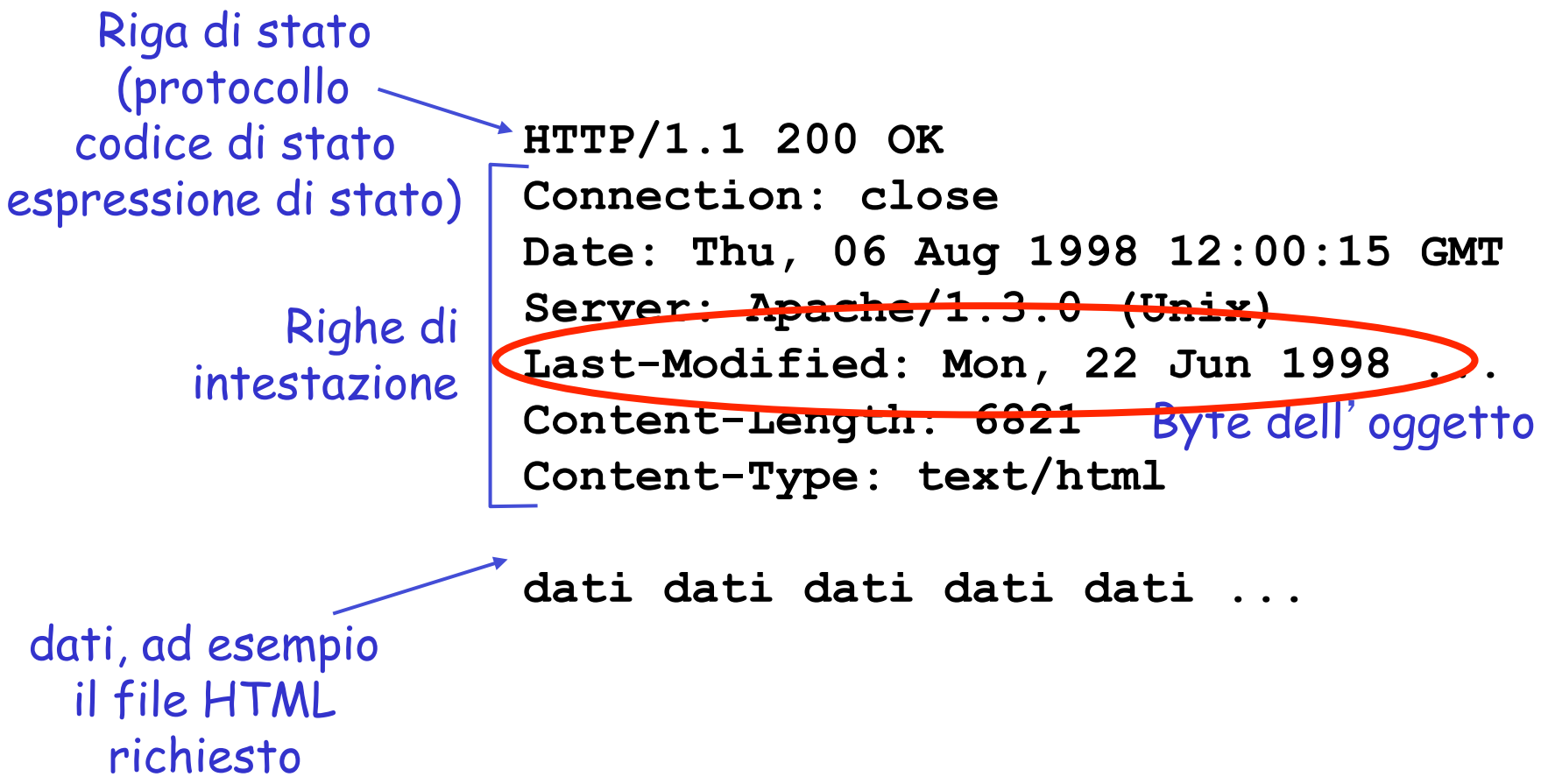
HTTP/1.0

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

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

Interazione utente-server: i cookie

Molti dei più importanti siti web usano i cookie

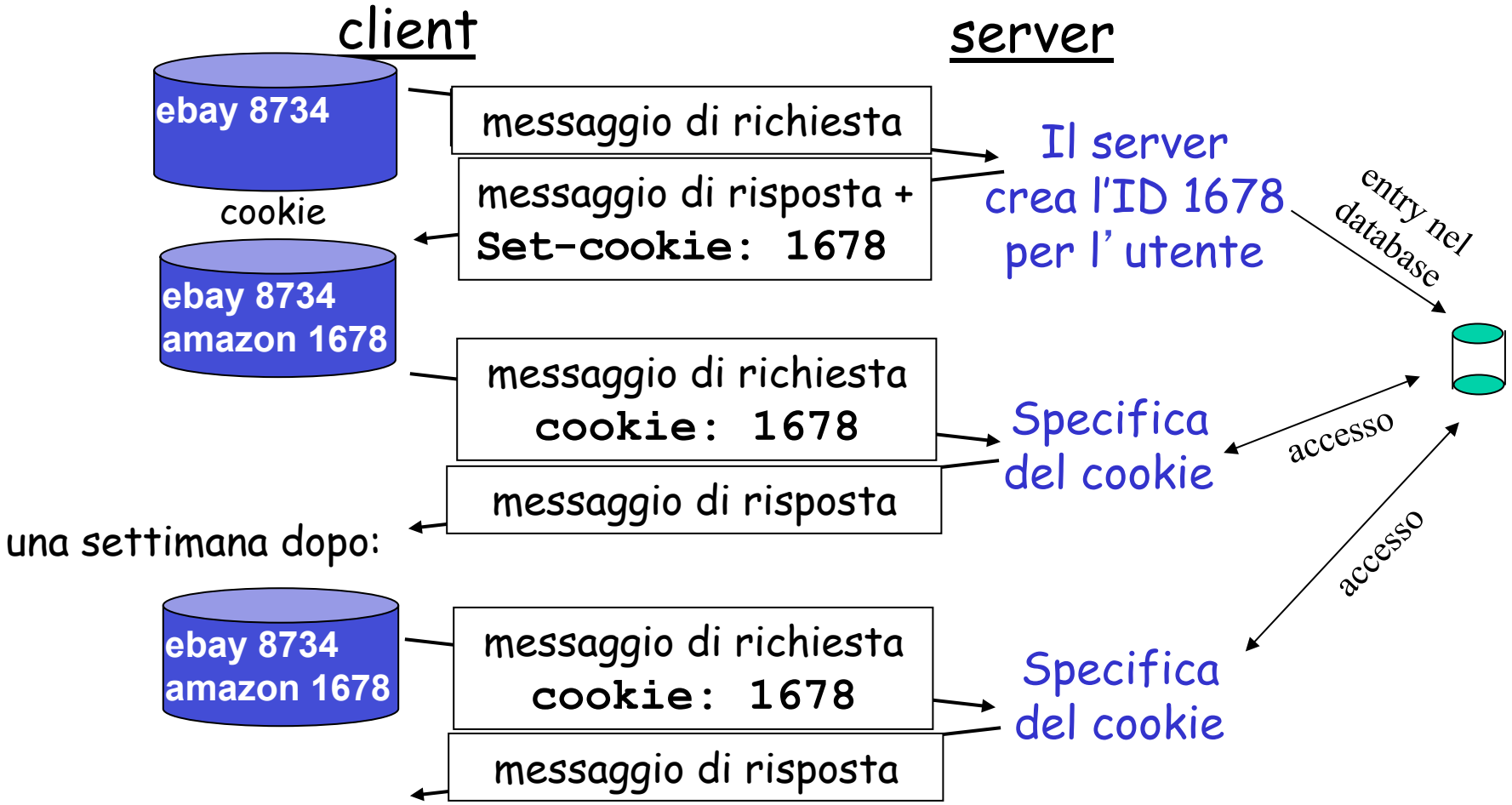
Quattro componenti:

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
- stato della sessione dell'utente (al di sopra dell' HTTP)
- ....

Lo "stato"

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

nota

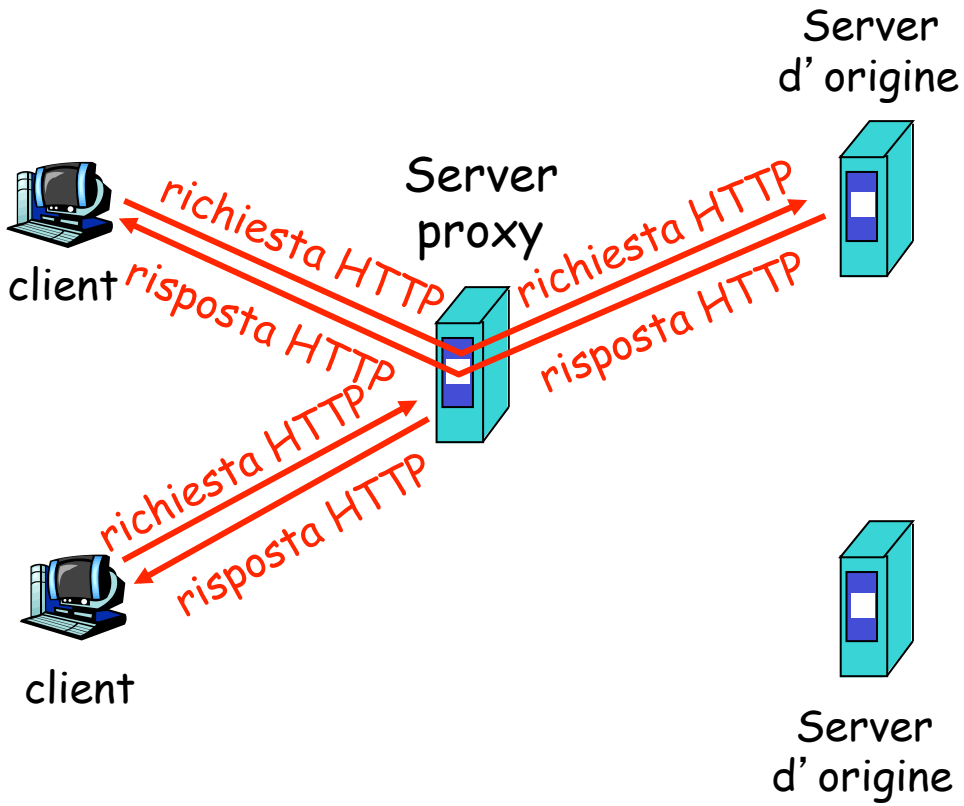
Cookie e privacy:

- 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
  - es. collo di bottiglia tra client e server minore di quello tra client e cache
- Riduce il traffico sul collegamento di accesso a Internet
- Internet arricchita di cache consente ai provider "scadenti" di fornire dati con efficacia

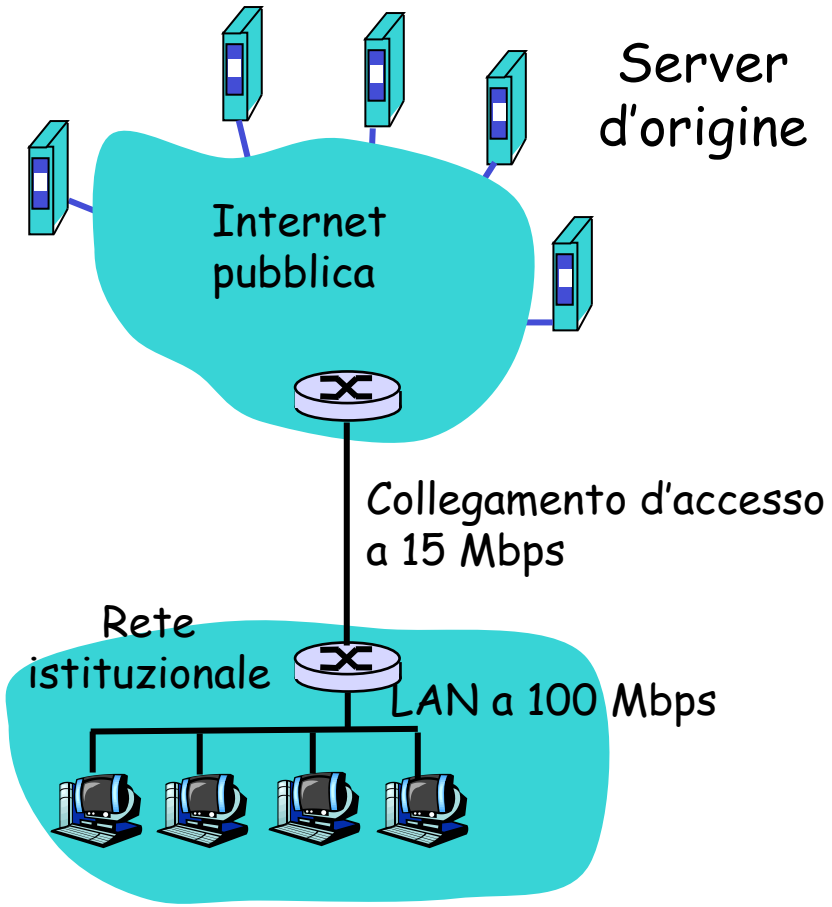
Esempio di caching

Ipotesi

- Dimensione media di un oggetto = 1Mbit = 1.000.000 bit
- Frequenza media di richieste dai browser istituzionali ai server d'origine = 15/sec
- Ritardo dal router lato Internet a qualsiasi server d'origine e ritorno al router = 2 sec (*ritardo Internet*)

Conseguenze

- utilizzazione sulla LAN = (15 req/sec) * (1Mbit/req) / 100Mbps = 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)

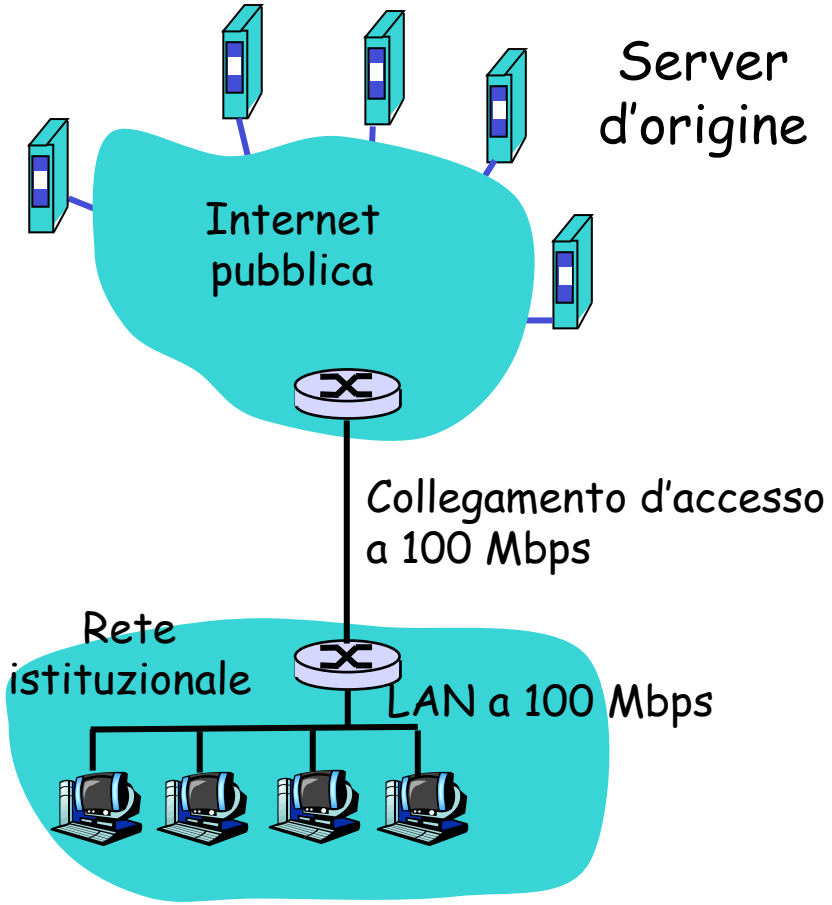
Soluzione possibile

- 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



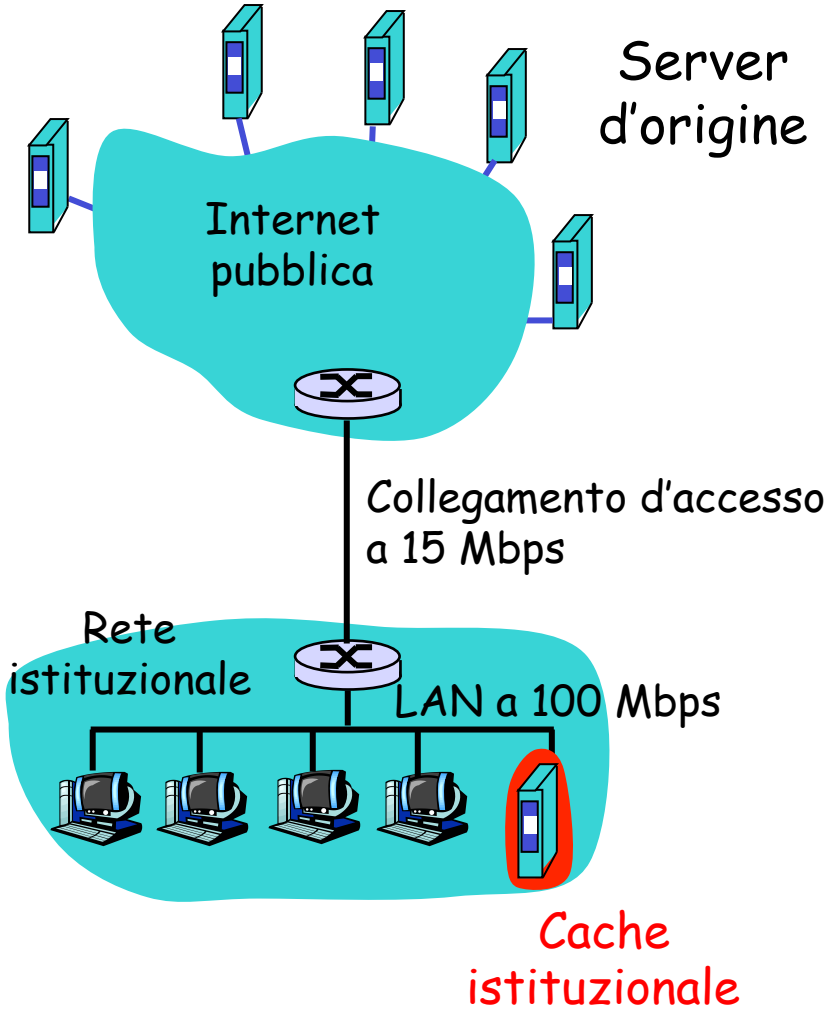
Esempio di caching (continua)

Soluzione possibile: installare la cache

- supponiamo una percentuale di successo (*hit rate*) pari a 0,4
  - (di solito tra 0,2 e 0,7)

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) + 0,4*(0,01 sec) < 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`