Web e HTTP

Terminologia

- Una pagina web consiste di oggetti
- Un oggetto può essere un file HTML, una immagine JPG, ecc.
- Una pagina web consiste di un file HTML *base* che fa riferimento a diversi oggetti al suo interno
- Ogni oggetto è riferito tramite un URL
- Esempio di URL:

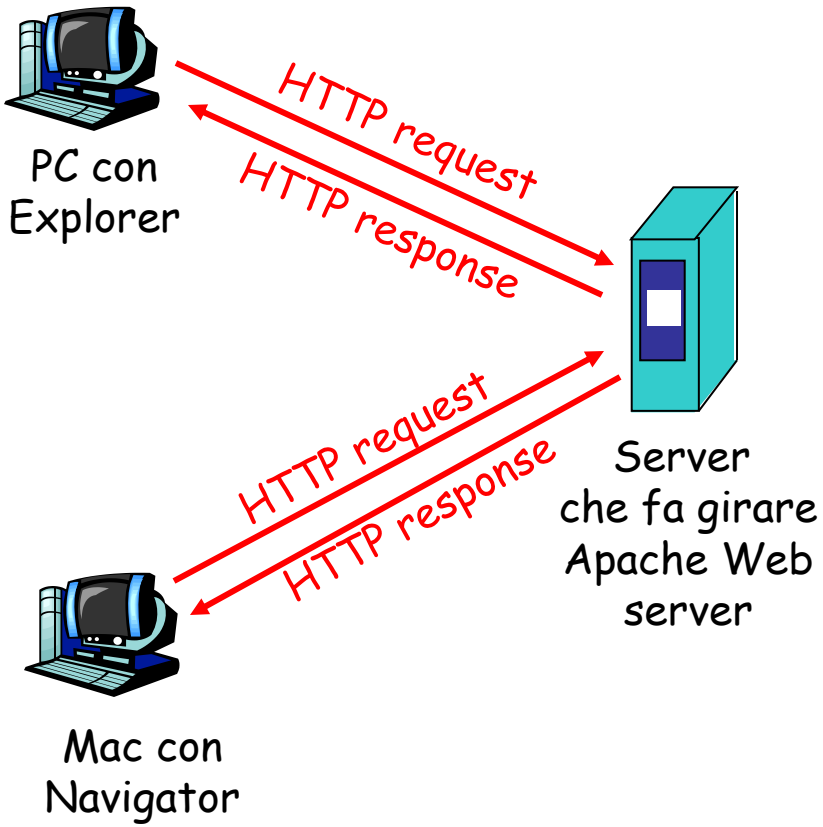
`www.someschool.edu/someDept/pic.gif`

host name | path name

Panoramica HTTP

HTTP: hypertext transfer protocol

- Protocollo applicazione per il web
- modello client/server
  - *client:* un programma browser che richiede e riceve oggetti web
  - *server:* un Web server che invia oggetti in risposta a richieste
- HTTP 1.0: RFC 1945
- HTTP 1.1: RFC 2068
- Porta 80 (a volte 8080)



Panoramica su HTTP (continua)

Usa TCP:

- Il client crea un socket verso il server, sulla porta 80
- Il server accetta la connessione
- i due interlocutori si scambiano messaggi espressi in HTTP
- La connessione

HTTP è “stateless”

- Non ci sono normalmente informazioni sulle precedenti connessioni
- Il concetto di ‘sessione’ è stato aggiunto in seguito

Le connessioni HTTP

Nonpersistenti

- Al più un oggetto è inviato su una connessione
- HTTP/1.0 è nonpersistente

Persistenti

- Si può usare la stessa connessione per inviare più oggetti in sequenza
- HTTP/1.1 usa di default le connessioni persistenti

HTTP Nonpersistente

Supponiamo l'utente richieda l'URL

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

(contiene testo, e riferimenti a 10 immagini JPG)

1a. Il client HTTP inizia una connessione TCP at www.someSchool.edu sulla porta 80

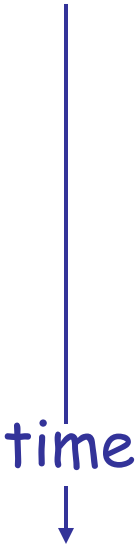
1b. Il server, in attesa di connessione, accetta la connessione dal client

2. Il client HTTP invia un HTTP *request message* (contiene l'URL voluto) attraverso il socket appena aperto.

3. Il server riceve la richiesta, predispone un *response message* contenente l'oggetto desiderato e lo invia attraverso il suo socket

time

HTTP Nonpersistente (2)



4. Il server chiude la connessione.

5. Il client HTTP riceve il messaggio di risposta. E' un testo html che viene visualizzato. Leggendo il file vengono incontrati i riferimenti a 10 diverse immagini JPG

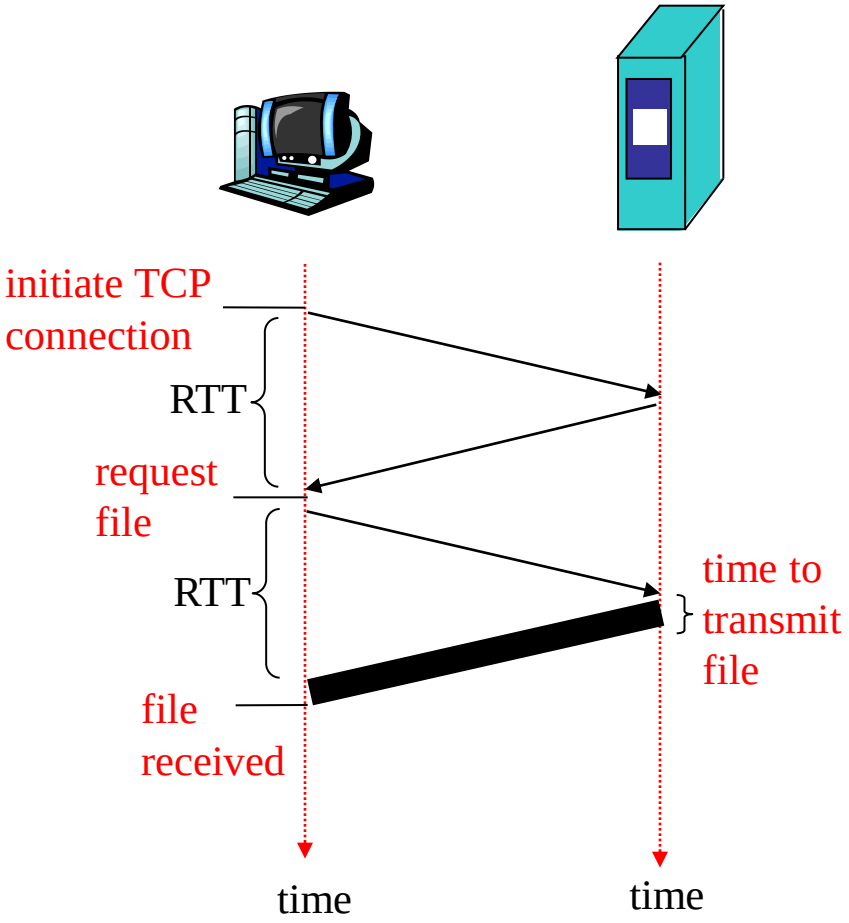
6. I passi da 1 a 5 vengono ripetuti per tutti i 10 oggetti

Tempi di risposta

Definition of RTT: tempo che ci mette un pacchetto ad arrivare al server e ritorno.

Tempo di risposta:

- 1 RTT per iniziare la connessione.
- 1 RTT per la HTTP request e l'arrivo dei primi byte di risposta
- Tempo totale di trasmissione



HTTP Persistente

Problemi con HTTP nonpers.:

- Ci vogliono 2 RTT per oggetto
- Ogni connessione richiede un overhead
- Spesso vengono aperte molte connessioni parallele

HTTP persistente

- Il server non chiude la connessione dopo l'invio del primo oggetto
- La connessione viene

Persistente ***senza pipeline***:

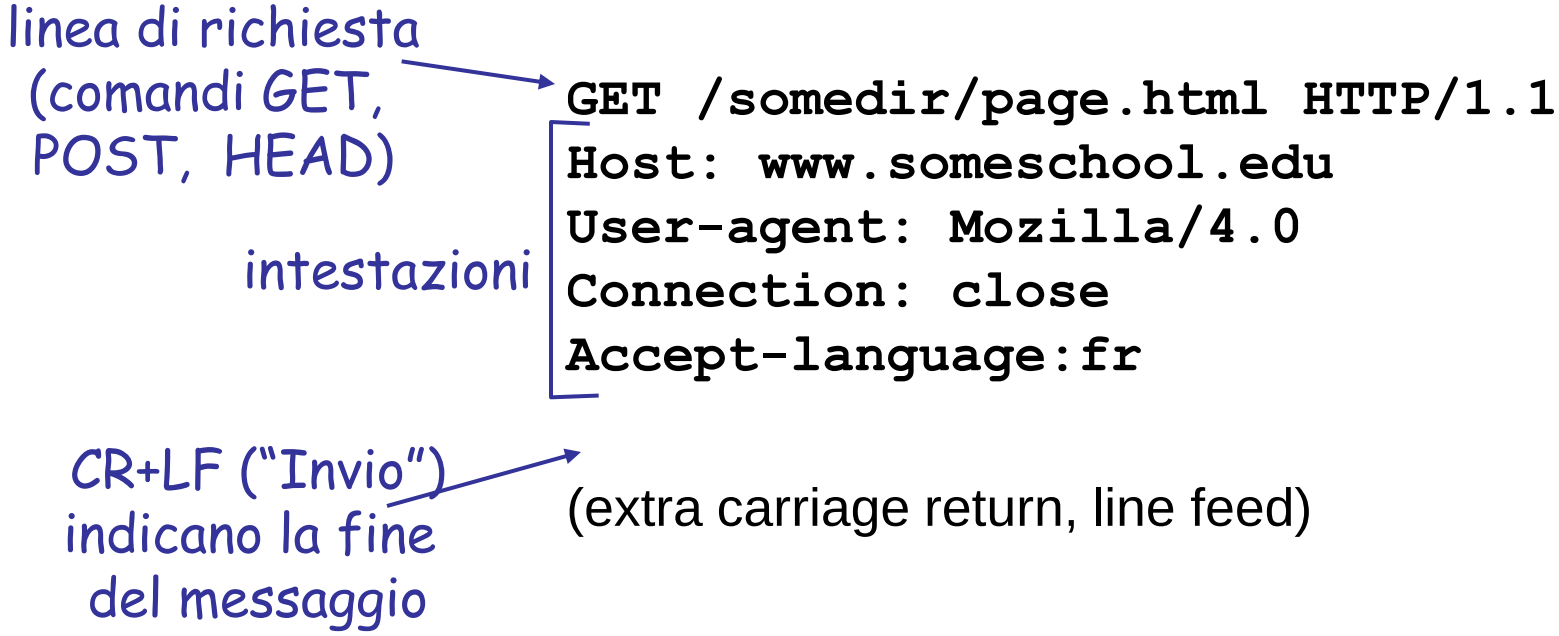
- Il client aspetta la risposta prima di inviare una ulteriore richiesta
- 1 RTT per ogni oggetto richiesto

Persistente ***con*** pipelining:

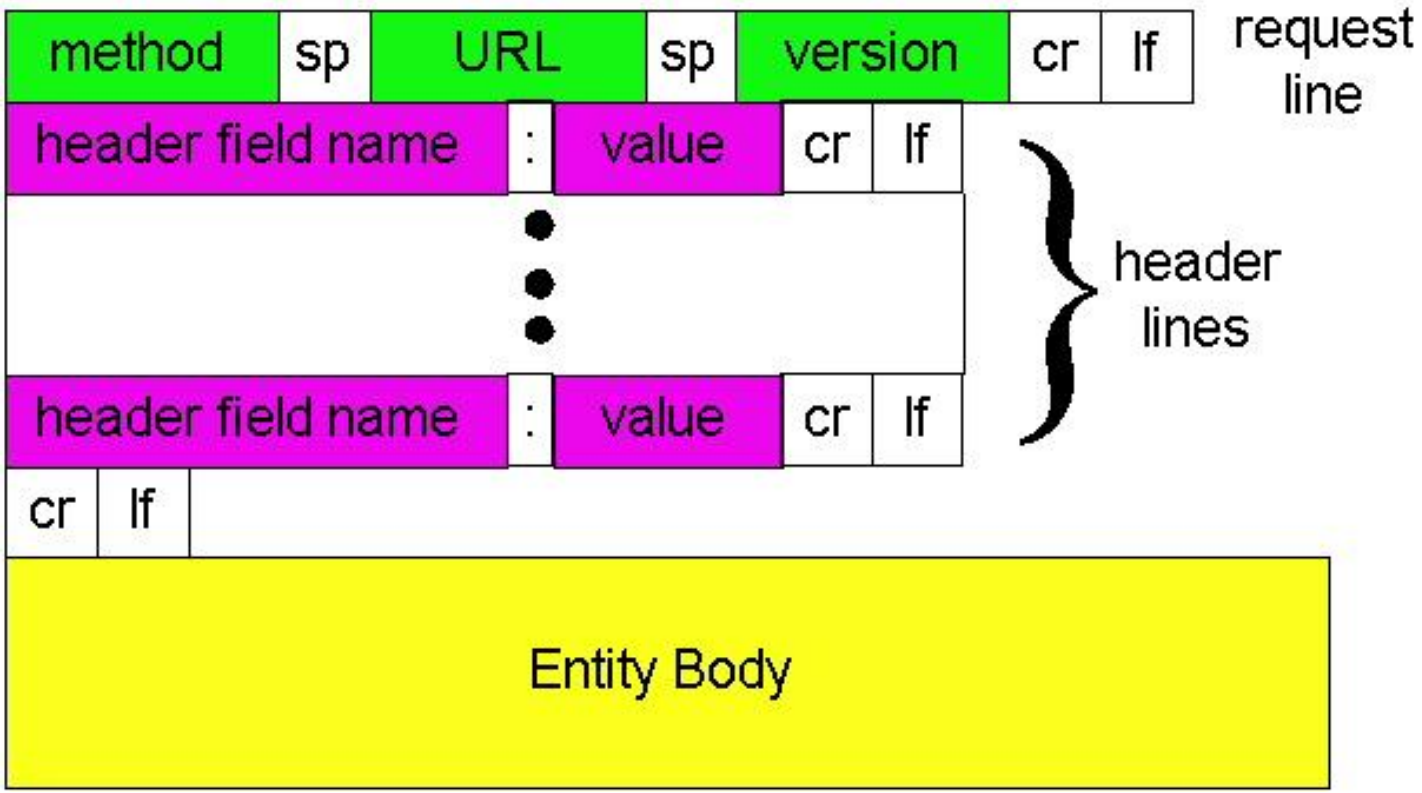
- default in HTTP/1.1
- Il client invia le richieste a raffica senza aspettare i precedenti oggetti
- Un solo RTT di attesa

Formato del messaggio di richiesta HTTP

- Due tipi di messaggi: *request*, *response*
- HTTP request message:
  - ASCII (leggibile, urrà)



Formato generale



Trasmissione di molti dati insieme alla richiesta

Metodo POST:

- Usato se nella pagina c'è una form con tanti dati
- L'input sta nel body del messaggio

Metodo URL:

- Usa il comando GET
- L'input fa parte dell'URL:

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

Tipologie di metodi

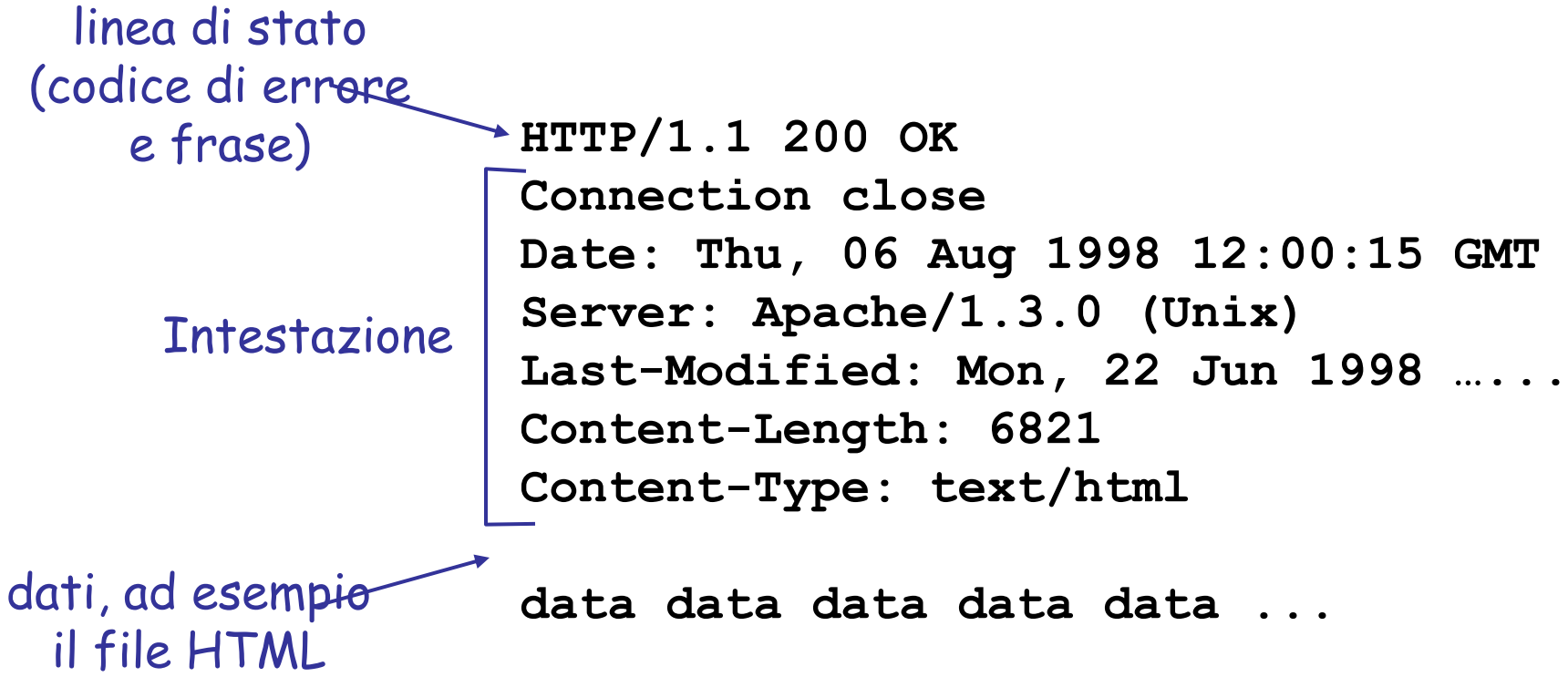
HTTP/1.0

- GET
- POST
- HEAD
  - per avere solo INFO sull'oggetto e non l'oggetto stesso (ad esempio sulla data di ultima modifica). Utile per il caching

HTTP/1.1

- GET, POST, HEAD
- PUT
  - upload un file
- DELETE
  - Cancella un certo file

Messaggio di risposta



Codici di errore

Stanno nella prima linea della risposta.

Alcuni codici:

200 OK

- richiesta OK, l'oggetto è in questo messaggio

301 Moved Permanently

- L'oggetto è stato spostato, questa è la nuova locazione (Location:)

400 Bad Request

- Che diavolo stai dicendo? Non ti capisco

404 Not Found

- Il documento richiesto non c'è qui.

505 HTTP Version Not Supported

Provate da soli

1. Telnet sul vostro Web server preferito

```
telnet
```

Assicuratevi di impostare il local echo
->set localecho
->set crlf
->open www.libero.it 80

2. Digitate una GET HTTP request:

```
GET /~ianni/ HTTP/1.1
Host: www.mat.unical.it
```

Lasciare un doppio invio alla fine!

3. Date un occhiata al messaggio di risposta

HTTP in azione

- Analizziamo i pacchetti con Ethereal!

I cookies: una forma di ‘stato’

Oramai irrinunciabili

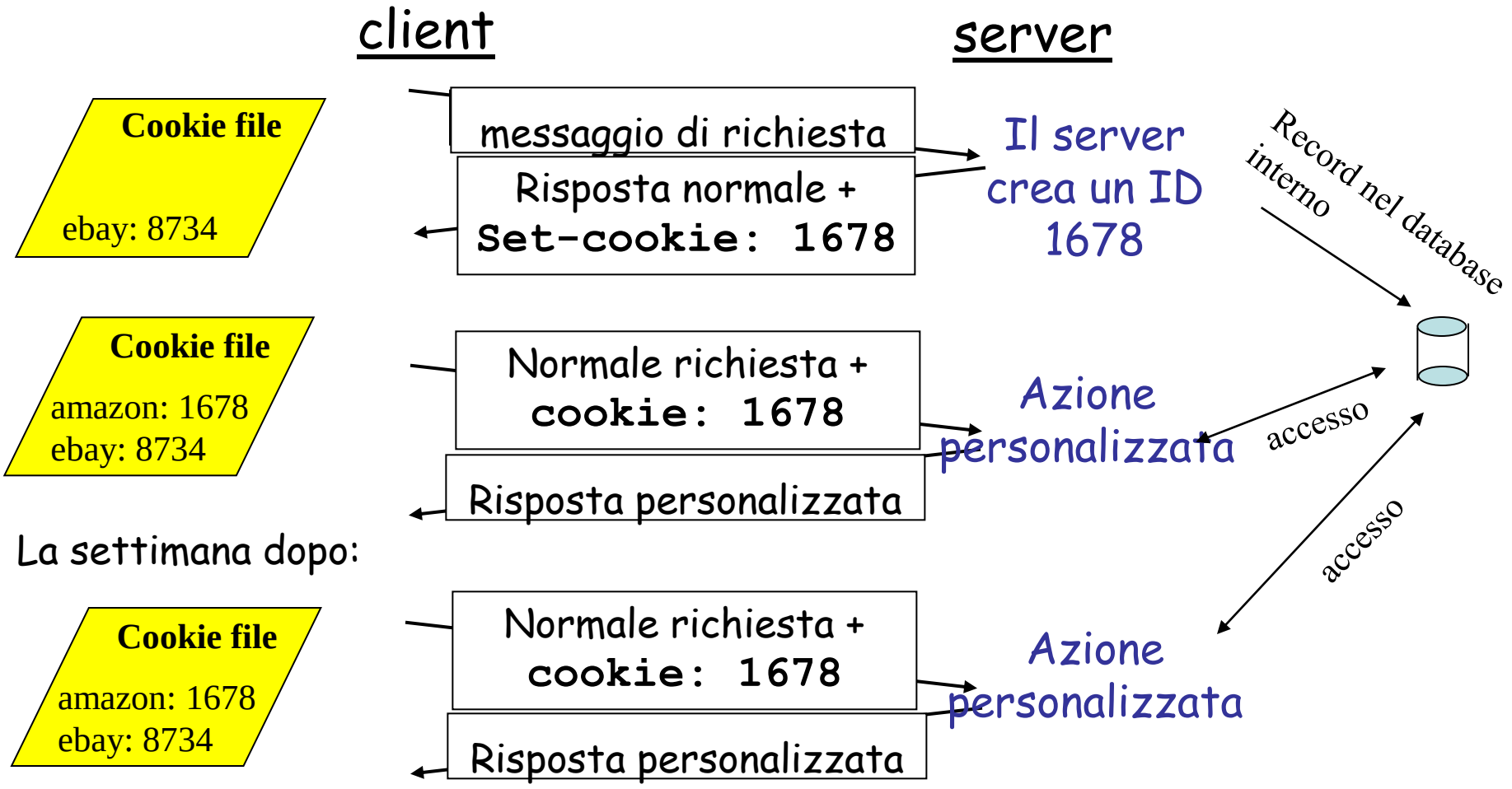
Quattro componenti:

1) Campo Cookie nei messaggi di risposta
2) Campo Cookie nei messaggi di richiesta
3) Il browser salva i cookies nei messaggi di risposta e li reinvia la volta successiva che chiede lo stesso oggetto
4) Il sito web contiene invece un suo database dei cookie inviati a tutti i client

Esempio:

- Susanna accede a Internet sempre dallo stesso PC
- Visita un certo sito di e-commerce
- Alla prima richiesta HTTP, il web server associa un ID all’IP di Susanna e lo salva nel database. Susanna verrà riconosciuta tramite il cookie di risposta e si potrà inviarle contenuti personalizzati

Cookies: come funzionano



Ancora cookies

Cosa possono trasportare:

- autorizzazioni
- carrelli della spesa
- consigli per gli acquisti
- stato della sessione (Web e-mail)

N.B.

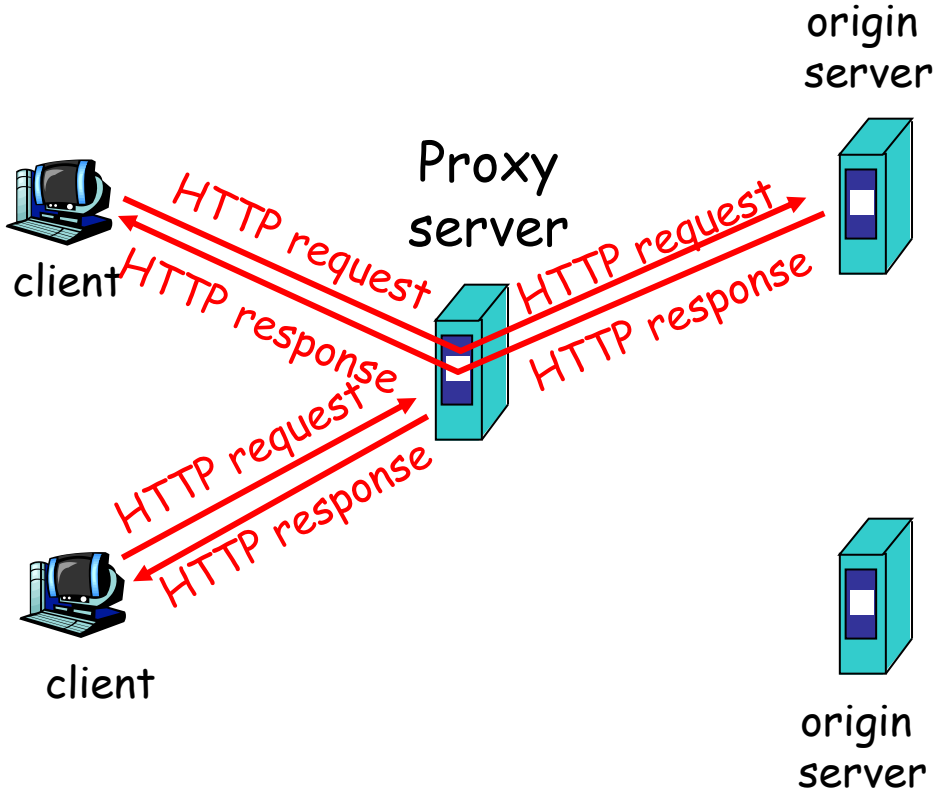
Cookies e privacy:

- I cookies consentono ai siti di web di scoprire tante cose di voi
- I motori di ricerca usano redirezione e cookies per tanti scopi
- In realtà i cookies si possono scambiare tra un sito e un altro

Web caches (proxy servers)

Goal: evitano di generare traffico se la stessa richiesta si ripete

- L'accesso al Web è fatto tramite un cache server (proxy)
- Il fa tutte le richieste al proxy
  - Se l'oggetto è in cache viene ritornato
  - Altrimenti il proxy si occupa di



Proxy (2)

- Il cache server fa sia da client che da server
- Di solito il proxy è installato nella sede della propria rete locale (dipartimento, azienda)

Perchè fare caching?

- Ridurre il tempo di risposta.
- Ridurre il traffico in uscita complessivo per una rete locale.

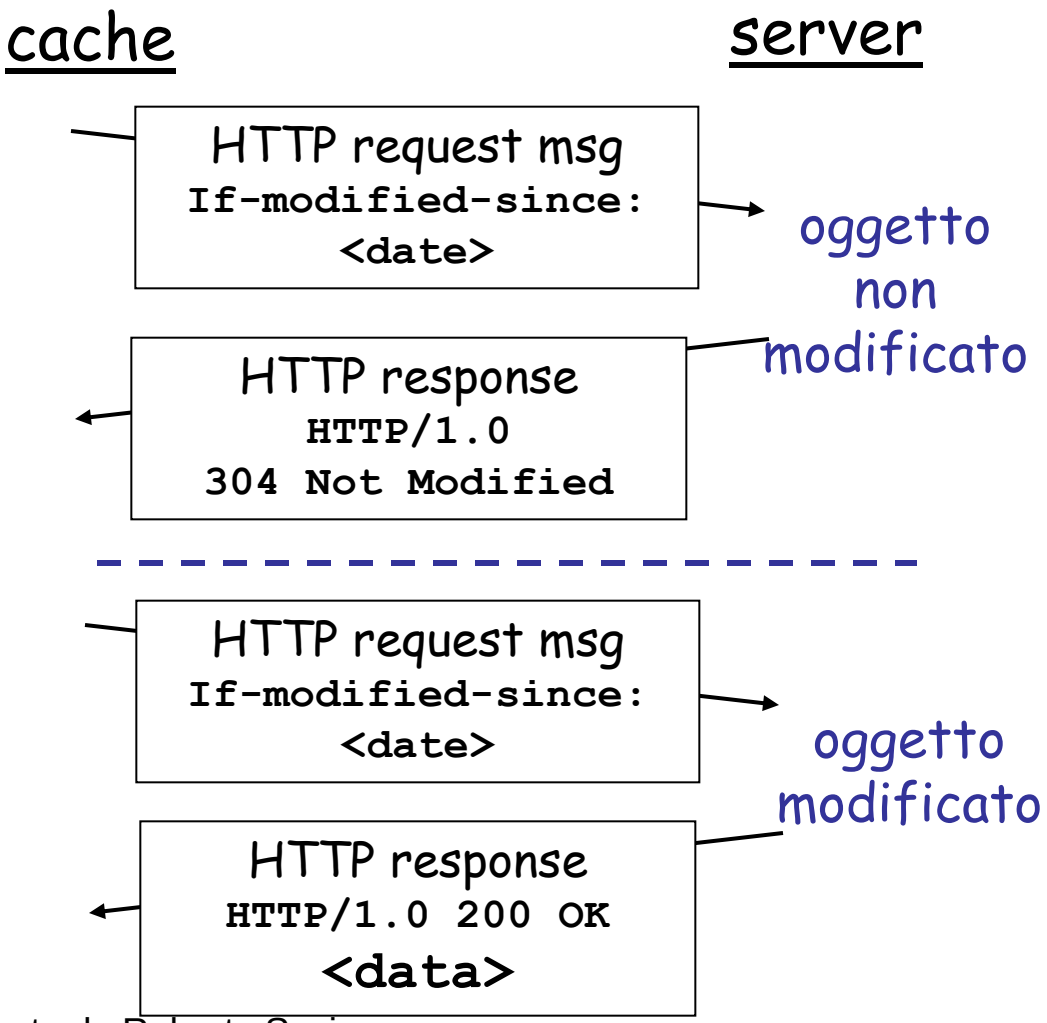
GET condizionale

- Scopo: Non mandare l'oggetto se non necessario
- client: quando si fa una richiesta si indica la data della propria copia

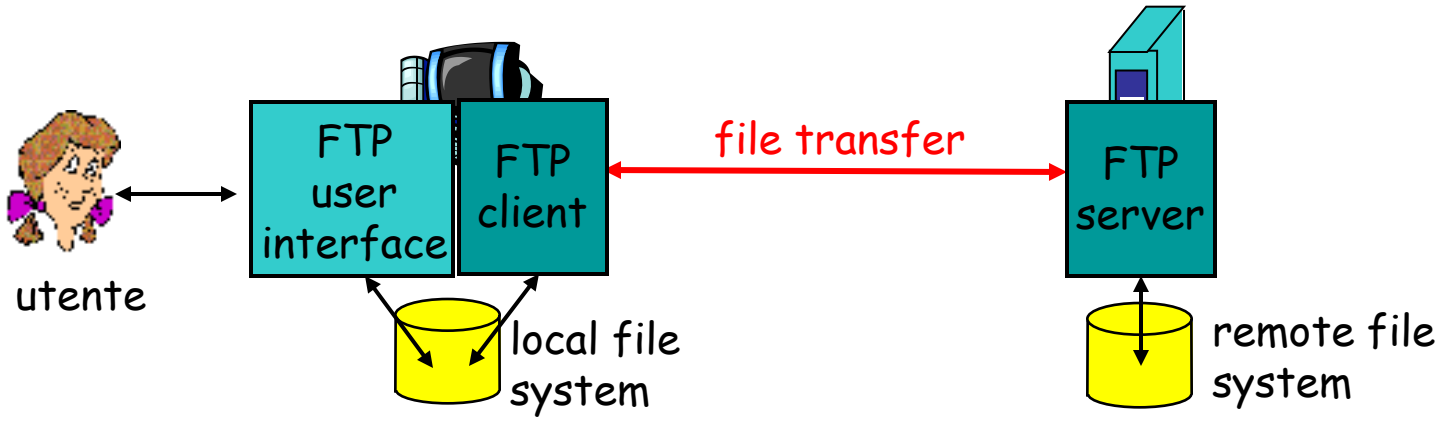
```
If-modified-since:
  <date>
```

- server: la risposta non contiene nulla se la copia del client è aggiornata:

```
HTTP/1.0 304 Not
  Modified
```



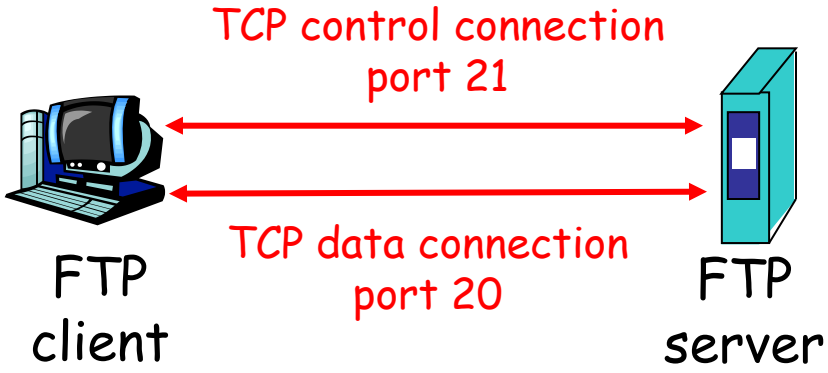
FTP: File Transfer Protocol



- Trasferisce file da e per un host remoto
- modello client server
  - *client:* lato che inizia la connessione
  - *server:* host remoto
- ftp: RFC 959
- ftp server: porta 21

FTP: meccanismo a doppia connessione

- Il client FTP contatta il server sulla porta 21 con protocollo TCP.
- La negoziazione avviene su questa connessione (connessione di controllo)
- Si possono navigare le directory sulla connessione
- Una connessione dati separata viene aperta per trasferire i file
- Dopo aver trasferito i file il server chiude la connessione dati



- Controllo *Fuoribanda*
- FTP è un protocollo con stato (utente, directory corrente)

Comandi e risposte FTP

Comandi di controllo:

- Inviati come ASCII
- **USER *username***
- **PASS *password***
- **LIST** lista i file
- **RETR filename** preleva un file (download)
- **STOR filename** fa upload di un file

Codici di ritorno

- Codice di ritorno e frase (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**