World Wide Web

Introduzione

Caratteristiche fondamentali

URI

Architettura e funzioni base

HTTP:

struttura richiesta/risposta, metodi, codici di risposta, campi di intestazione, tipi MIME, passaggio dati di un form, gestione della sessione

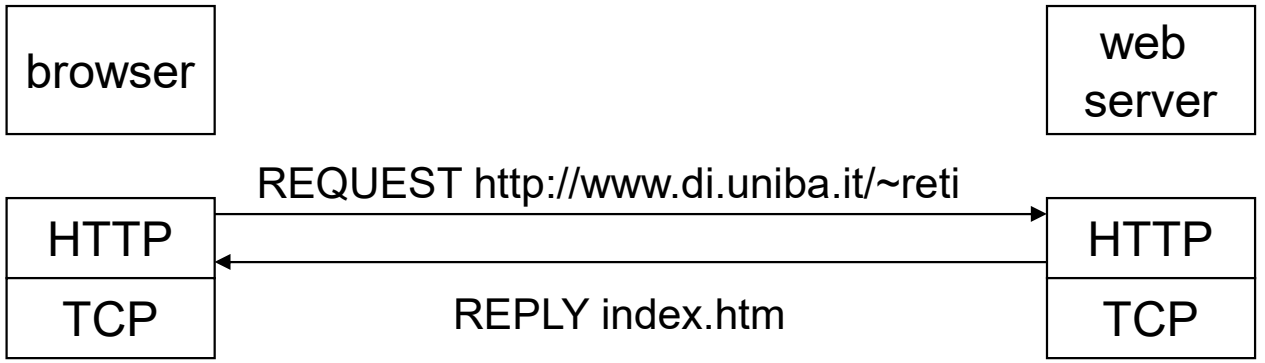
Prof. Filippo Lanubile

Introduzione

- Il Web nasce come sistema ipermediale distribuito
  - sistema ipermediale
    - permette l'accesso interattivo a collezioni di documenti di diverso tipo (testo, figure, immagini, animazioni, audio, video) collegati tra loro
  - sistema distribuito
    - documenti conservati su più server remoti
- Oggi il Web è una piattaforma per applicazioni distribuite
  - browser come client container

Architettura client/server

- l'utente seleziona un URL target
- il browser (web client) spedisce una richiesta HTTP al web server
- Il server elabora la richiesta e spedisce la risposta al browser
- Il browser interpreta i comandi HTML e mostra la pagina
- Ogni documento esterno referenziato richiede una richiesta separata



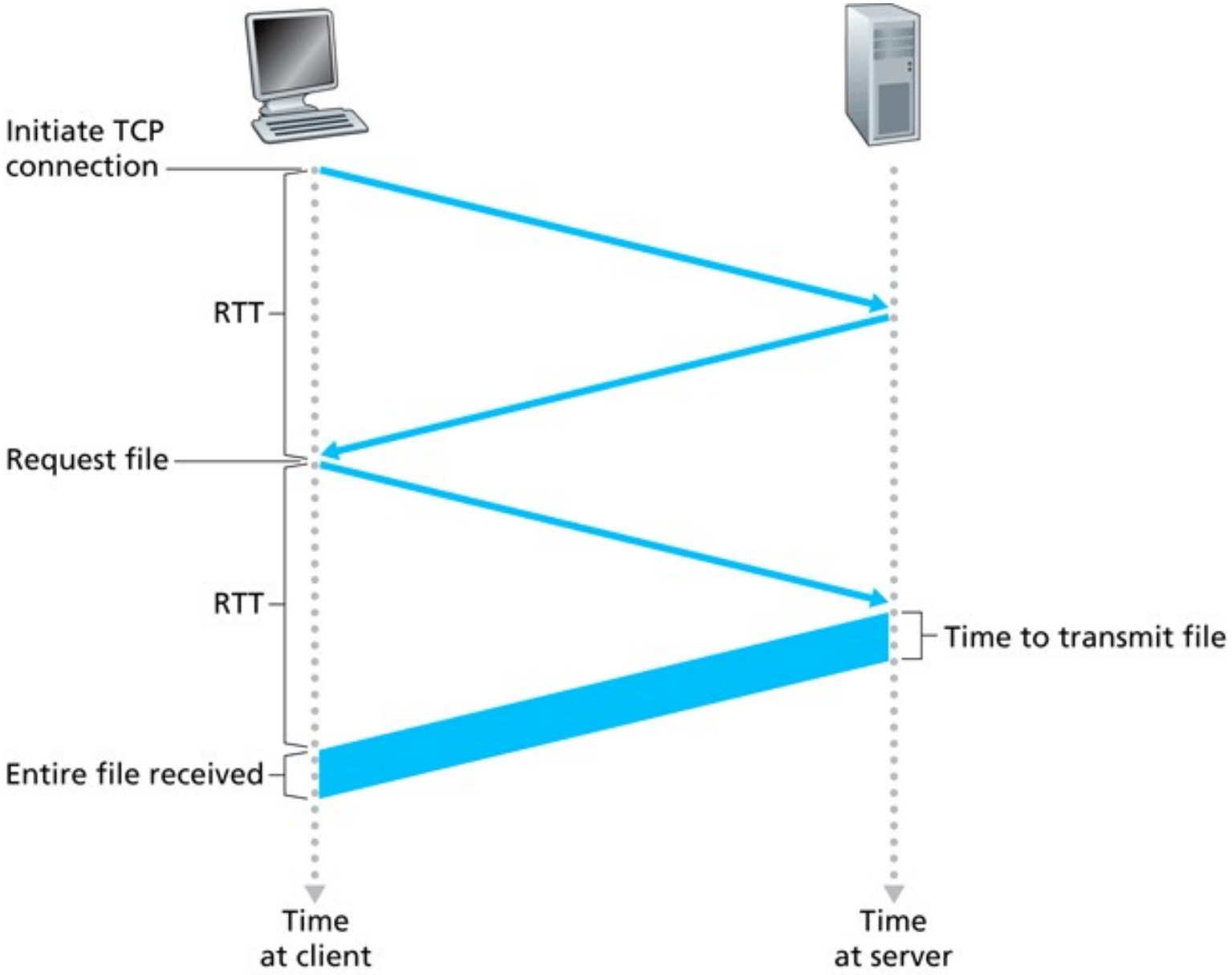
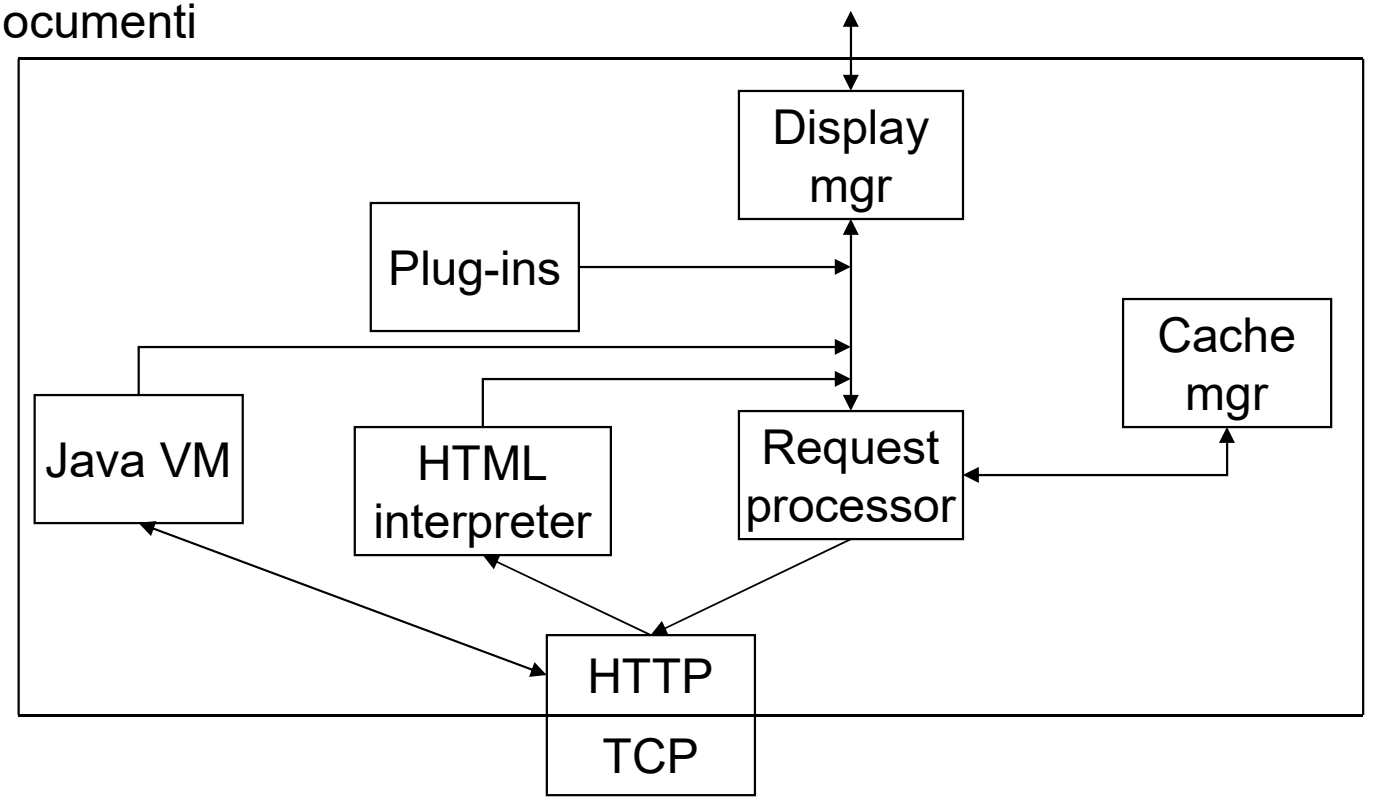


Figure 2.7 ♦ Back-of-the-envelope calculation for the time needed to request and receive an HTML file

Architettura del browser

Funzioni base

- Richiedere documenti
- Interpretare documenti
  - entita' testuali e non, applets, controls, script
- Mostrare documenti



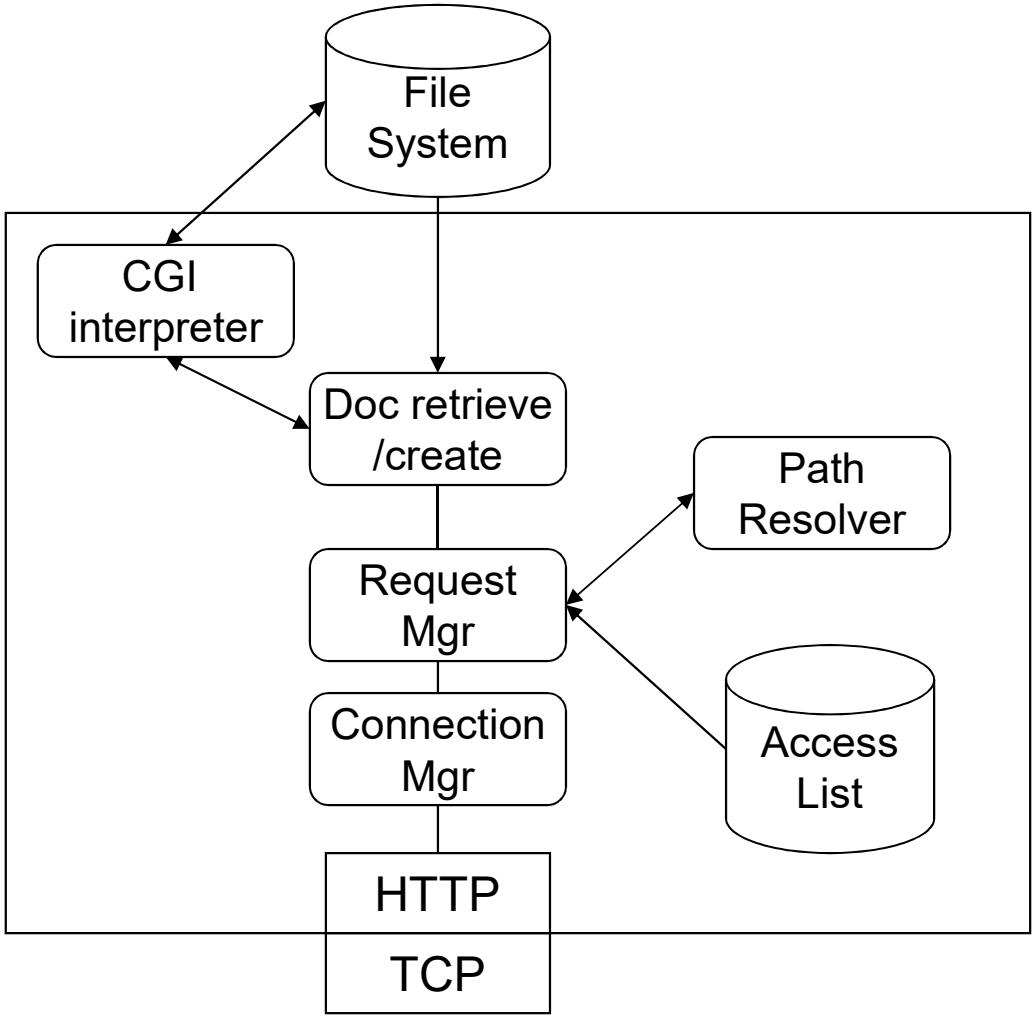
Architettura del web server (http daemon)

Funzioni base

- gestire connessioni con i browser
- gestire richieste e risposte HTTP
- ritrovare/creare documenti
- gestire sicurezza

Web server più diffusi

- Apache (open source)
- IIS (Microsoft)
  - Anche ftp, news e mail server

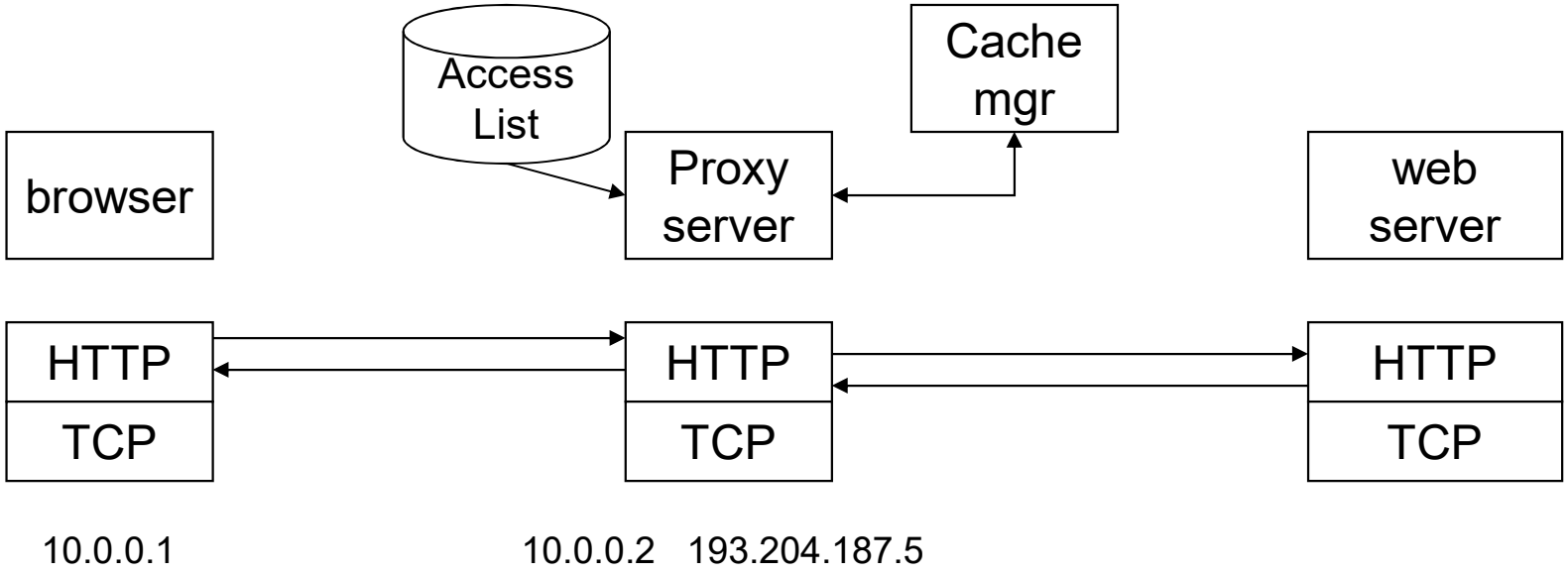


Proxy server

Proxy = entità locale che svolge azioni per conto di un un'entità remota

Funzioni base

- Richieste da client con indirizzi IP privati
- Controllo e filtro delle richieste in uscita
- Caching centralizzata



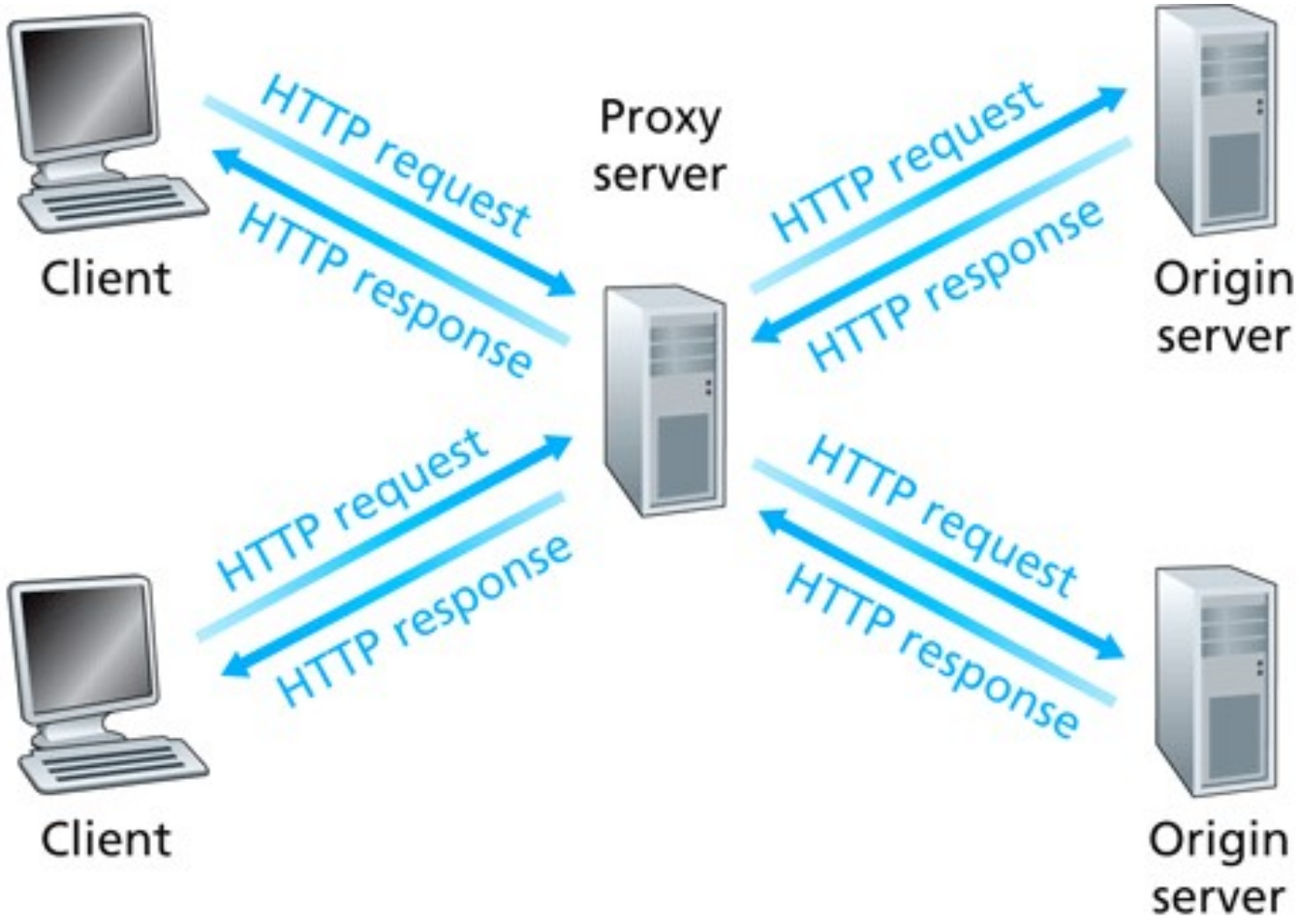


Figure 2.11 ♦ Clients requesting objects through a Web cache

Caratteristiche tecniche fondamentali

- Sistema di indirizzamento:URI
  - per rendere questo mondo possibile nonostante molti protocolli diversi
- Un protocollo richiesta-risposta: HTTP
  - per trasferire informazioni con l'efficienza necessaria al contesto ipermediale
- Un linguaggio di annotazione: HTML
  - che tutti i client capiscono, per presentare le informazioni e inserire gli iper-puntatori (hyperlinks)
  - Ma anche XML e altri linguaggi gestiti dal consorzio World Wide Web: W3C (http://www.w3.org/)

URI: Uniform Resource Identifier

- RFC 3986
- Una stringa di caratteri per identificare una risorsa astratta o fisica

  <scheme>:<scheme-specific-part>
- Tipi di URI
  - URN – Uniform Resource Name
    - RFC 2141
    - Associa un identificatore alla risorsa con caratteristiche di persistenza

      urn:namespace_identifier:namespace_specific_string
      - Esempio:

        <time xmlns='**urn:xmpp:time**'>

        <tzo>-06:00</tzo>

        <utc>2006-12-19T17:58:35Z</utc>
    - Il meccanismo di accesso alla risorsa non è specificato
  - URL - Uniform Resource Locator
    - RFC 1738
    - Specifica il metodo di accesso e la locazione della risorsa
      - Esempio: **http://www.w3c.org/index.html**
    - Se la locazione cambia la stringa non è più valida (broken link)

URL - Uniform Resource Locator

- scheme:["//"] [user [":"password] "@"] host [":"port] ["/"url-path]
  - scheme: protocollo o metodo di accesso usato per accedere alla risorsa richiesta (può essere http, https, ftp, mailto, file, news, telnet, gopher)
  - host: nome DNS o indirizzo IP (il browser richiede la risoluzione del nome DNS)
  - port: numero di porta (opzionale) del processo server a cui è diretta la richiesta (se method è http il default è 80)
  - url-path: percorso gerarchico + nome di un file (se method è http il default del filename può essere index.html) + “?" query (passaggio parametri)
- Più comunemente protocol://host/path/filename
- Esempi:
  - http://www.di.uniba.it/~reti/calc/index.htm
  - http://www.di.uniba.it/~reti/
  - http://seldi2.uniba.it:1025/
  - ftp://seldi.uniba.it/pub/papers/sew95.ps
  - file:/lucidi/web.zip
  - news:it.lavoro.offerte
  - mailto:lanubile@di.uniba.it

Documento HTML con riferimento a immagini

```
<html>
<head>
<title>Oracle Corporation</title> ...
</head>
<body bgcolor="#ffffff" link="#000000" vlink="#ff0000">
...
<a href="/ip/deploy/ias/">Application Server</a><br>
<a href="/ip/develop/ids/">Development Tools</a><br>
<img src="http://oracleimg.com/admin/images/stretch.gif"
   width="5" height="5" alt=""><br>
<a href="/ip/deploy/cs/">Collaboration Suite</a><br>
<img src="http://oracleimg.com/admin/images/stretch.gif"
   width="5" height="5" alt=""><br>
<a href="/applications/">E-Business Suite</a><br>
<a href="/outsourcing/sbs/">Oracle Small Business
   Suite</a><br>
...
</body>
</html>
```

HTTP - HyperText Tranfer Protocol (1)

- Protocollo di tipo request-reply
  - richiesta del client al server
  - risposta del server al client
- Protocollo stateless (senza stato)
  - Una nuova richiesta-risposta non ricorda la storia delle richieste-risposte precedenti (dati di sessione)
- Basato su TCP
- HTTP 1.0 (RFC1945)
  - Protocollo ASCII
  - ogni richiesta-risposta utilizza una connessione TCP diversa
  - Slow start di TCP
- HTTP 1.1 (RFC 2616)
  - Protocollo ASCII
  - Il server non chiude subito la connessione TCP
  - Nuove richieste al server sfruttano la connessione rimasta aperta
  - Pipeline delle immagini riferite in un documento HTML

HTTP - HyperText Tranfer Protocol (2)

- HTTP/2 (RFC 7540, 7541)
  - Stessa semantica di HTTP/1.1
  - Basato su protocollo SPDY di Google
  - Obiettivo: ridurre la latenza
  - Protocollo binario
  - Compressione dell'header
  - Risoluzione di problemi noti in HTTP 1.1 che danneggiano le prestazioni

Esempio di richiesta/risposta HTTP

```
St7a62% telnet www.w3.org 80

Trying 18.23.0.22...
Connected to www.w3.org.
Escape character is '^]'.

GET /Protocols/ HTTP/1.0

HTTP/1.1 200 OK
Date: Thu, 23 Apr 1998 12:47:22 GMT
Server: Apache/1.2.6
Last-Modified: Tue, 21 Apr 1998 17:57:50 GMT
Content-Length: 21259
Accept-Ranges: bytes
Connection: close
Content-Type: text/html; charset=ISO-8859-1

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
...
```

Tipi e sottotipi MIME

- Text
  - text/plain: testo non formattato
  - text/html: testo in HTML
  - text/xml: testo in XML
  - …
- Image
  - image/gif: immagine GIF
  - image/jpeg: immag. JPEG
  - …
- …
- Audio
  - audio/basic: suono udibile
  - …
- Video
  - video/mpeg: filmato MPEG
  - …
- Application
  - application/pdf: sequenza di byte in formato pdf
  - …

https://www.rfc-editor.org/rfc/rfc2046.txt
http://www.iana.org/go/rfc2046
http://www.iana.org/assignments/media-types/media-types.xhtml

Richiesta HTTP

- Request header line
  - method azione da intraprendere (GET, HEAD, POST, ...)
  - identifier indirizzo della risorsa target (solo url-path)
  - version versione http
- Request header fields
  - From: email address
  - User-Agent: quale browser (Mozilla/...)
  - Accept: lista di tipi/sottotipi MIME accettabili
  - If-Modified-Since: Restituisci il documento se piu' recente
  - …
- Entity Body
  - usato per passare al server informazioni non predefinite
- CRLFCRLF
  - Doppio <RETURN>

Richiesta HTTP

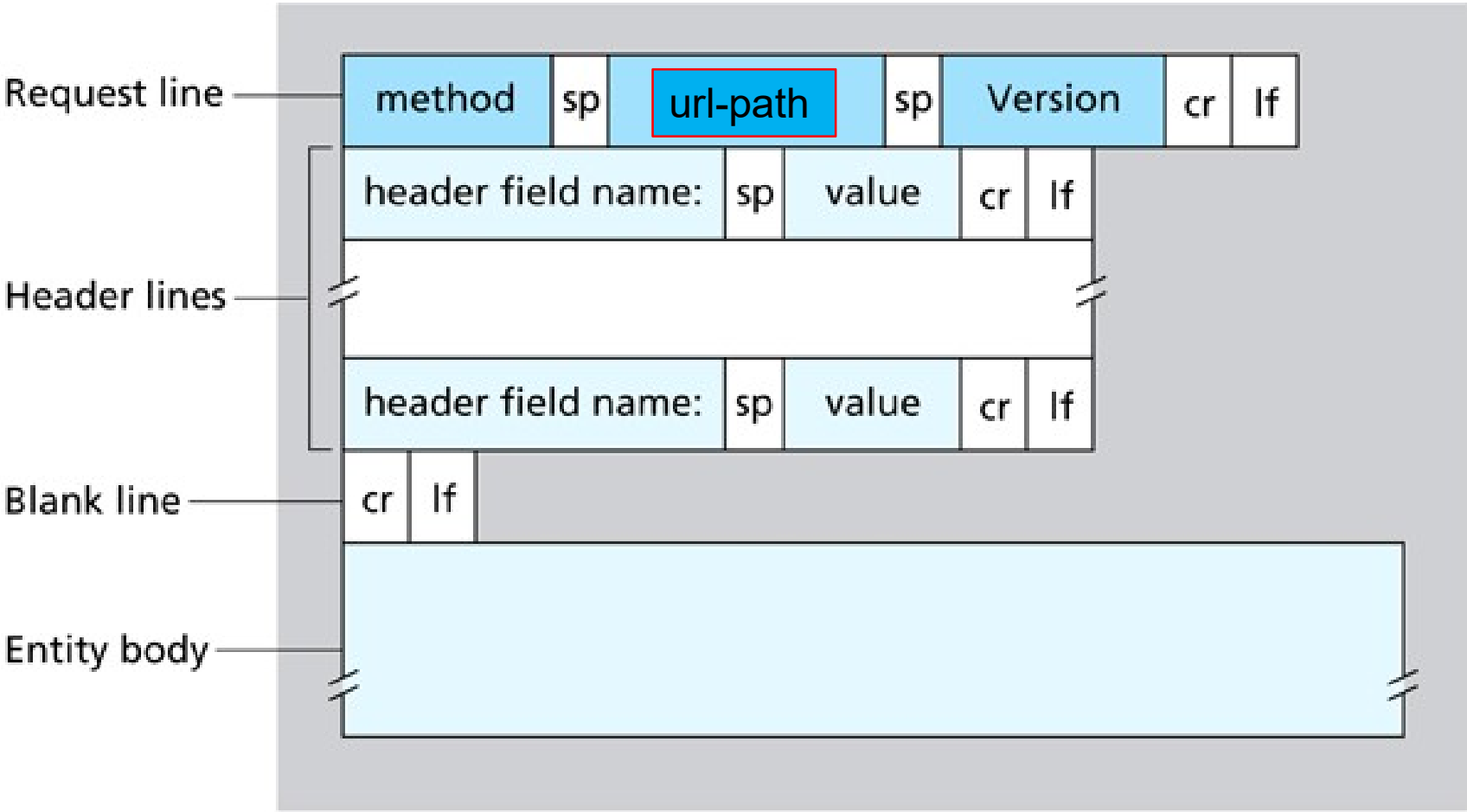


Figure 2.8 ♦ General format of an HTTP request message

Metodi HTTP

Method	Description
GET	Request to read a Web page
HEAD	Request to read a Web page's header
PUT	Request to store a Web page
POST	Append to a named resource (e.g., a Web page)
DELETE	Remove the Web page
TRACE	Echo the incoming request
CONNECT	Reserved for future use
OPTIONS	Query certain options

Risposta HTTP

- Response header line
  - version http version
  - code (200 OK, 400 bad request, 404 not found, …)
- Response header fields
  - MIME Version: versione MIME per codificare il messaggio
  - Server: software usato per il server
  - Date: data e tempo all'origine del messaggio
  - Content-type: tipo/sottotipo MIME e codifica usato per il corpo
  - Content-length: lunghezza in byte del corpo
  - …
- Entity body
  - generalmente il documento HTML

Risposta HTTP

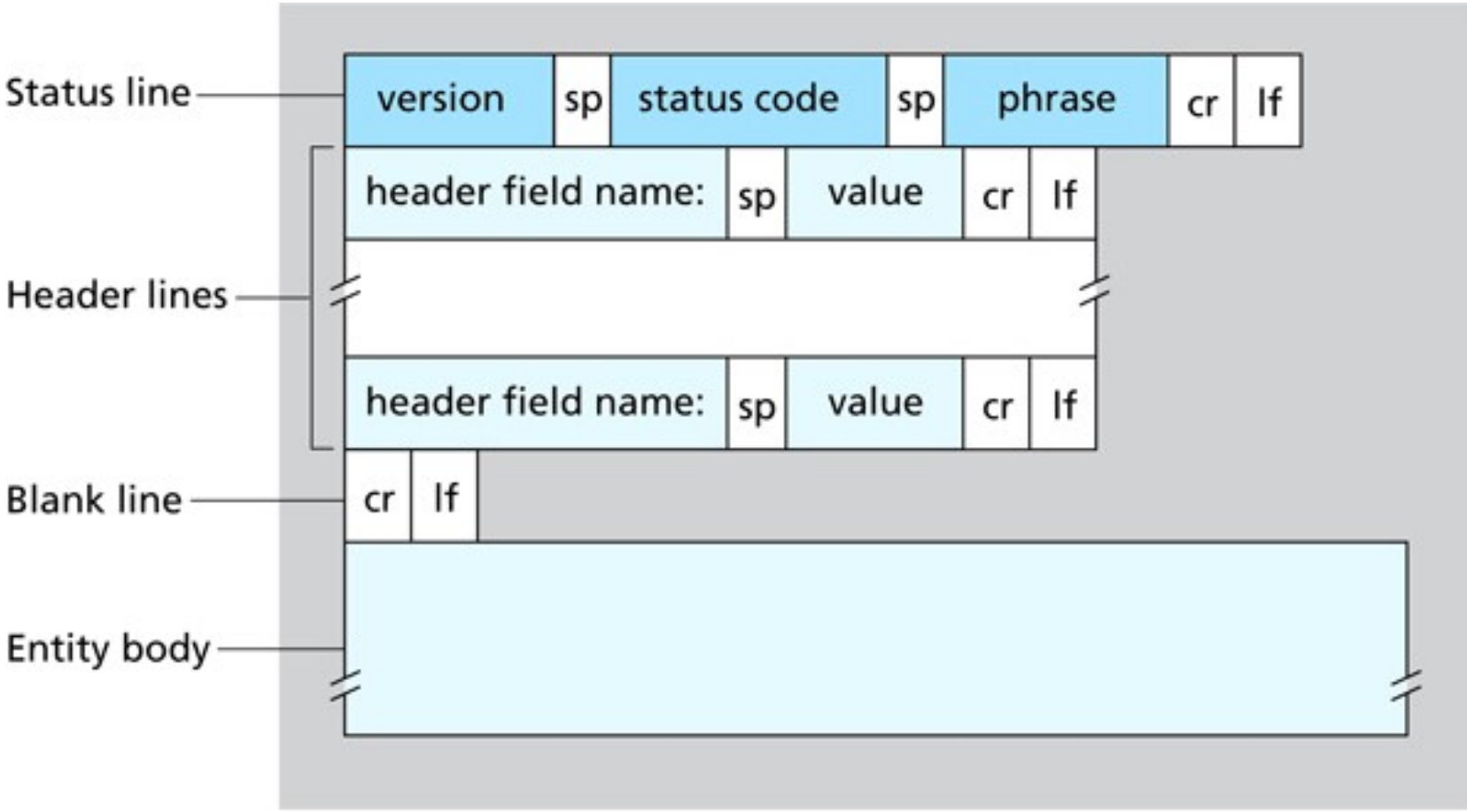


Figure 2.9 ♦ General format of an HTTP response message

Codici di risposta HTTP

Code	Meaning	Examples
1xx	Information	100 = server agrees to handle client's request
2xx	Success	200 = request succeeded; 204 = no content present
3xx	Redirection	301 = page moved; 304 = cached page still valid
4xx	Client error	403 = forbidden page; 404 = page not found
5xx	Server error	500 = internal server error; 503 = try again later

Campi di intestazione HTTP

Header	Type	Contents
User-Agent	Request	Information about the browser and its platform
Accept	Request	The type of pages the client can handle
Accept-Charset	Request	The character sets that are acceptable to the client
Accept-Encoding	Request	The page encodings the client can handle
Accept-Language	Request	The natural languages the client can handle
Host	Request	The server's DNS name
Authorization	Request	A list of the client's credentials
Cookie	Request	Sends a previously set cookie back to the server
Date	Both	Date and time the message was sent
Upgrade	Both	The protocol the sender wants to switch to
Server	Response	Information about the server
Content-Encoding	Response	How the content is encoded (e.g., gzip)
Content-Language	Response	The natural language used in the page
Content-Length	Response	The page's length in bytes
Content-Type	Response	The page's MIME type
Last-Modified	Response	Time and date the page was last changed
Location	Response	A command to the client to send its request elsewhere
Accept-Ranges	Response	The server will accept byte range requests
Set-Cookie	Response	The server wants the client to save a cookie

Passaggio dei dati di un form: display testuale

```
This is a practice form.

   Please help us to improve the World Wide Web by filling in the
   following questionaire:

   Your organization? ____________________________________

   Commercial? ( ) How many users? ____________________

   Which browsers do you use?
    1. Cello ( )
    2. Lynx ( )
    3. X Mosaic ( )
    4. Others ____________________________________


   A contact point for your site:

   ____________________________________________

   Many thanks on behalf of the WWW central support team.

   Submit Reset
```

Passaggio dei dati di un form: HTML source

```
<html>
<head>
<title>This is a practice form.</title>
</head>
<body>
<FORM METHOD=POST
   ACTION="http://www.cc.ukans.edu/cgi-bin/post-query">
Please help us to improve the World Wide Web by filling in
the following questionaire:
   <P>Your organization? <INPUT NAME="org" TYPE=text
   SIZE="48">
   <P>Commercial? <INPUT NAME="commerce" TYPE=checkbox>
           How many users? <INPUT NAME="users" TYPE=int>
```

Passaggio dei dati di un form: HTML source (cont.)

```
<P>Which browsers do you use?
   <OL>
    <LI>Cello <INPUT NAME="browsers" TYPE=checkbox
   VALUE="cello">
    <LI>Lynx <INPUT NAME="browsers" TYPE=checkbox
   VALUE="lynx">
    <LI>X Mosaic <INPUT NAME="browsers" TYPE=checkbox
   VALUE="mosaic">
    <LI>Others <INPUT NAME="others" SIZE=40>
   </OL>
A contact point for your site: <INPUT NAME="contact"
   SIZE="42">
<P>Many thanks on behalf of the WWW central support team.
    <P><INPUT TYPE=submit> <INPUT TYPE=reset>
</FORM>
</body>
</html>
```

Passaggio dei dati di un form: campi avvalorati

This is a practice form.

Please help us to improve the World Wide Web by filling in the following questionaire:

Your organization? Academic Computing Services____

Commercial? ( ) How many users? 10000_____

Which browsers do you use?

1. Cello (*)
2. Lynx (*)
3. X Mosaic (*)
4. Others
   Mac Mosaic, Win Mosaic_____

A contact point for your site:
Michael Grobe grobe@kuhub.cc.ukans.edu___

Many thanks on behalf of the WWW central support team.

Submit Reset

HTTP request mediante metodo POST

```
POST /cgi-bin/post-query
   HTTP/1.0
Accept: www/source
Accept: text/html
Accept: video/mpeg
Accept: image/jpeg
Accept: image/x-tiff
Accept: image/x-rgb
Accept: image/x-xbm
Accept: image/gif
Accept: application/postscript
User-Agent:  Lynx/2.2
   libwww/2.14
From:  grobe@www.cc.ukans.edu
Content-type: application/x-www-
   form-urlencoded
Content-length: 150
     * a blank line *
org=Academic%20Computing%20Services
&users=10000
&browsers=lynx
&browsers=cello
&browsers=mosaic
&others=MacMosaic%2C%20WinMosaic
&contact=Michael%20Grobe%20grobe@kuh
   ub.cc.ukans.edu
```

HTTP request mediante metodo GET

- In HTML form: <FORM METHOD=GET ACTION="http://www.cc.ukans.edu/cgi-bin/get-query">

```
GET /cgi-bin/get-query?org=Academic%20Computing%20Services
&users=10000&browsers=lynx&browsers=cello&browsers=mosaic
&others=MacMosaic%2C%20WinMosaic
&contact=Michael%20Grobe%20grobe@kuhub.cc.ukans.edu HTTP/1.0
Accept: www/source
Accept: text/html
Accept: video/mpeg
Accept: image/jpeg
Accept: image/x-tiff
Accept: image/x-rgb
Accept: image/x-xbm
Accept: image/gif
Accept: application/postscript
User-Agent:  Lynx/2.2  libwww/2.14
From:  grobe@www.cc.ukans.edu
     * a blank line *
```

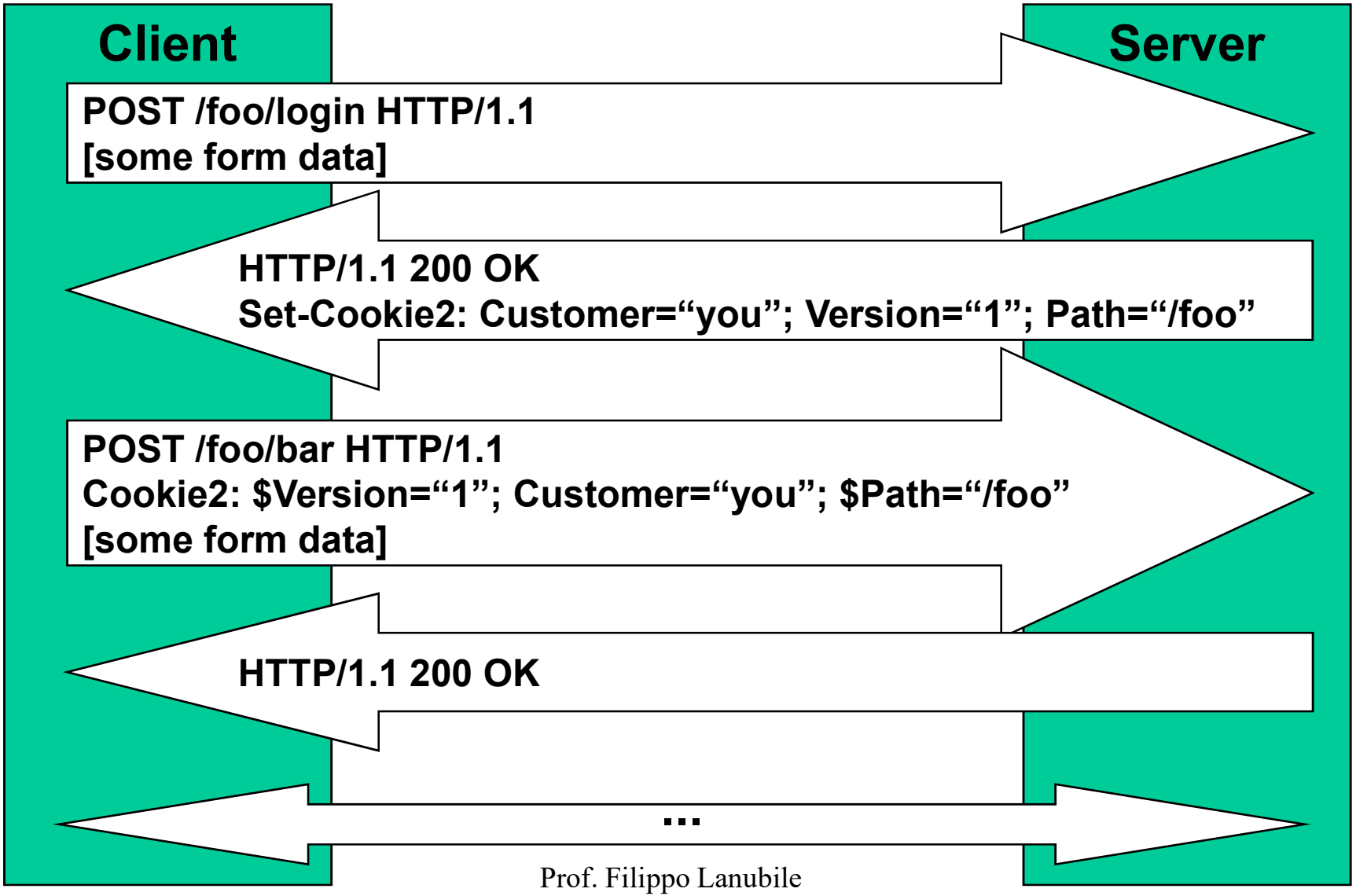
Gestione della sessione

- HTTP e' un protocollo stateless
  - Il server non conosce la storia delle richieste/risposte precedenti
- Sessione
  - sequenza di richieste/risposta come parte di un dialogo
  - La gestione della sessione è un requisito indispensabile per l'uso del web come infrastruttura di applicazioni
    - Autenticazione dell'utente
    - Acquisti
    - Segnalibro
  - La gestione della sessione richiede che le informazioni di stato siano condivise tra client (browser) e server
- Come mantenere lo stato di una sessione
  - HTML link con URL parametrizzata
  - HTML form contenente campi "hidden"
  - Cookies

Cookies

- Estensione di HTTP
  - Proposta da Netscape e poi standardizzata (RFC 2965, 2000)
  - Headers Cookie, Cookie2, and Set-Cookie2
- Un cookie è formato da una serie di coppie attributo-valore
  - Creato e memorizzato client-side in un file testuale
    - Da non usare per password o altre informazioni critiche
  - Un cookie per host o gruppi di host
- Scambio delle informazioni di stato iniziato dal server
  - Il server spedisce un HTTP reply con Set-Cookie2
    - Alcuni attributi sono predefiniti: Max-Age, Version, Discard, …
  - Il client può rifiutare il cookie o spedire un HTTP request con Cookie, Cookie2
    - Coppie di attributi-valori ricevuti in precedenza da quel server

Cookies


Client
Server
POST /foo/login HTTP/1.1
[some form data]
HTTP/1.1 200 OK
Set-Cookie2: Customer="you"; Version="1"; Path="/foo"
POST /foo/bar HTTP/1.1
Cookie2: $Version="1"; Customer="you"; $Path="/foo"
[some form data]
HTTP/1.1 200 OK
...

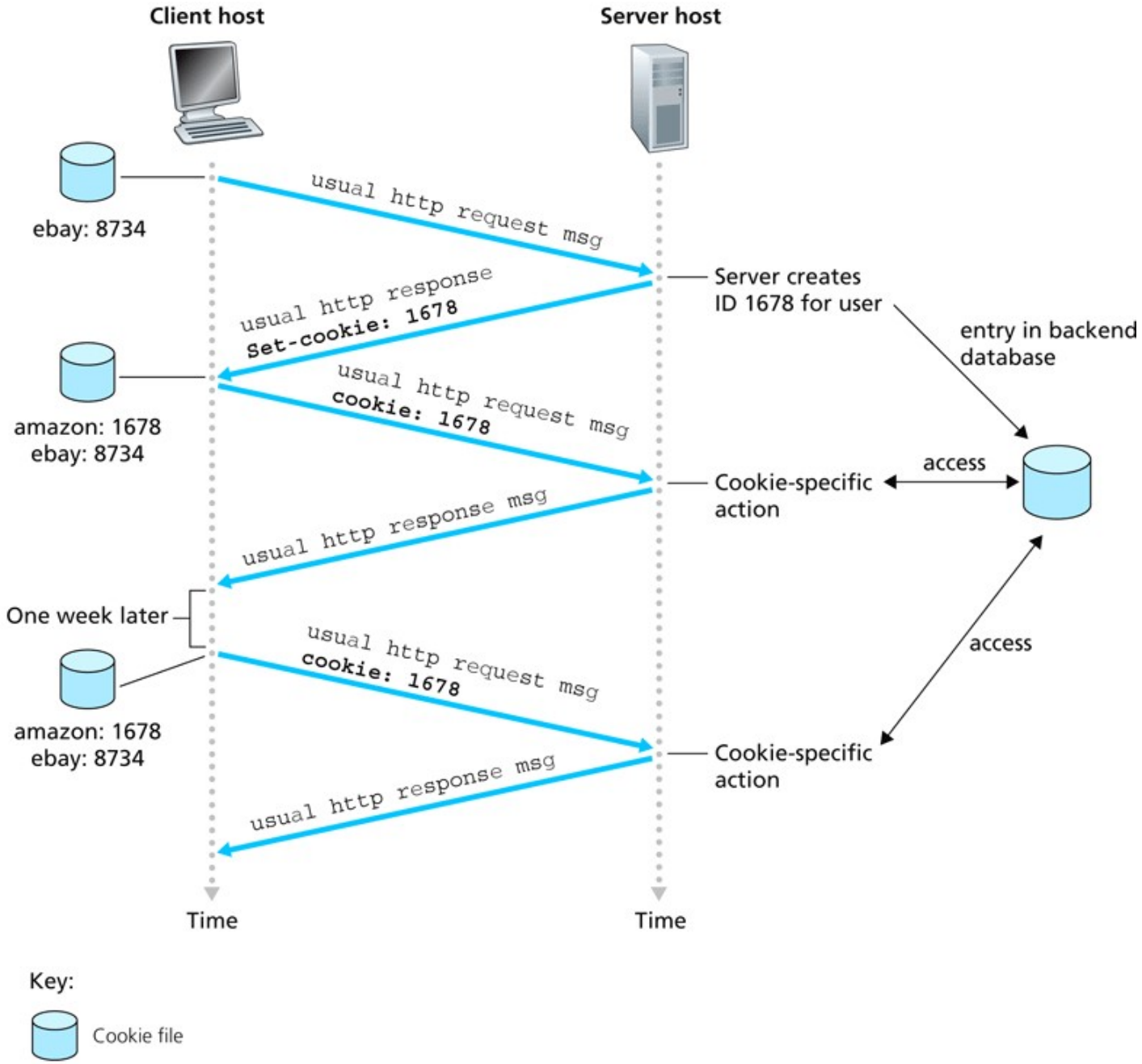


Figure 2.10 ♦ Keeping user state with cookies