PARTE 8

LIVELLO APPLICAZIONI
- Esempio: World Wide Web

Livello 5 (application protocol)

- Il ***livello application*** utilizza il livello di trasporto dell'informazione tra processi in esecuzione su host terminali per realizzare ***servizi di rete***

- **<u>Esempi protocolli applicativi</u>**

  ftp

  telnet

  http

  smtp

  irc

  …

Servizi di rete

Applicazioni di rete diverso da Protocolli applicativi

Quindi, si preferirà parlare di

Servizi di rete (=Applicazioni di rete)

Tipicamente, ciascun nuovo servizio di rete definisce:

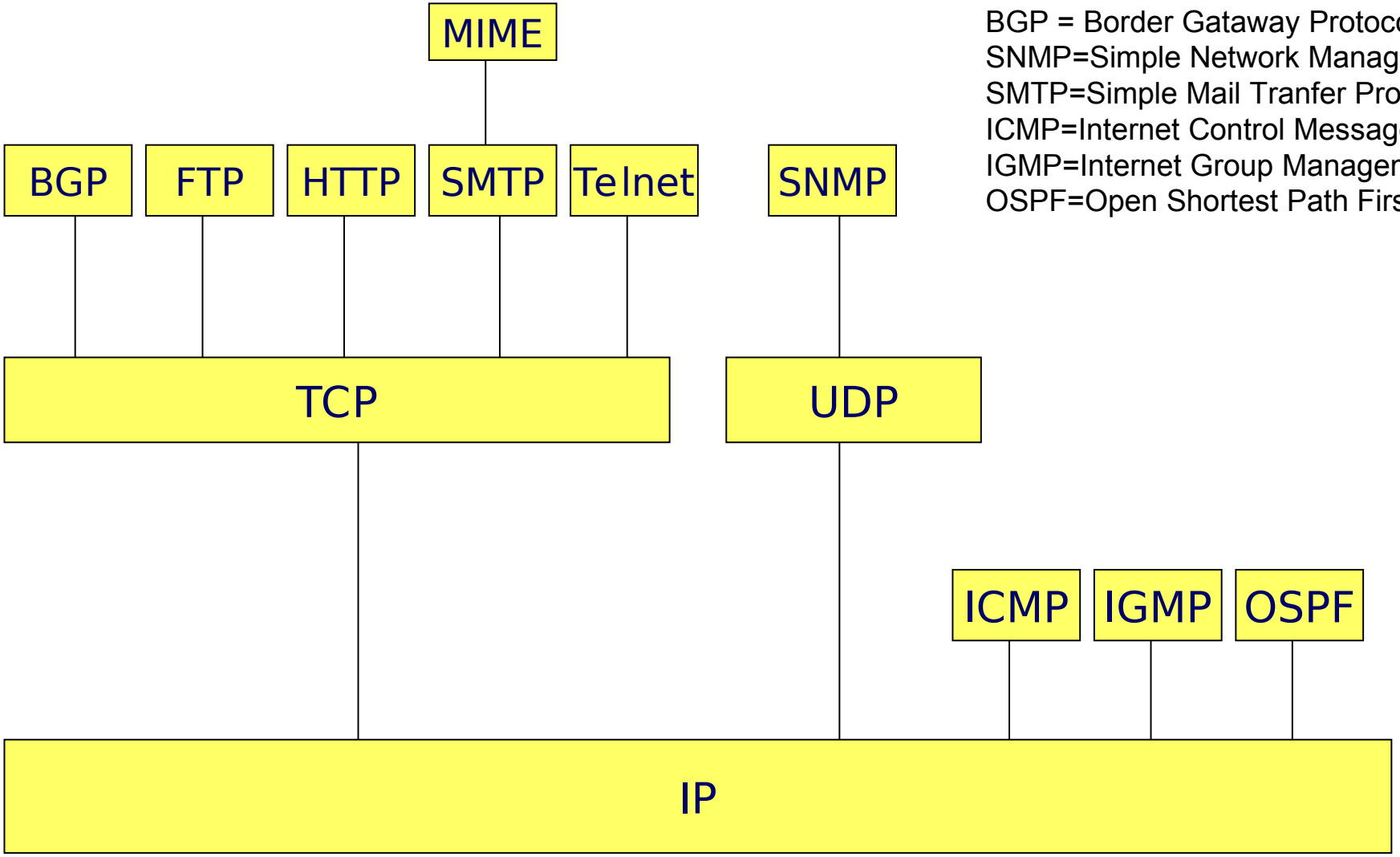
- Un software per il client (compresa un'interfaccia utente)
- Un software per il server
- Un nuovo protocollo di livello applicativo

Servizi di rete e protocolli applicativi

Esempi

- Trasferimento file: usa protocollo applicativo ftp
- Collegamento a terminale remoto: usa protocollo applicativo telnet
- World Wide Web: usa protocollo applicativo http
- Posta elettronica: usa protocollo applicativo smtp
- Chat: usa protocollo applicativo irc
- ...

Esempi di protocolli nello stack TCP/IP



BGP = Border Gataway Protocol
SNMP=Simple Network Management Protocol
SMTP=Simple Mail Tranfer Protocol
ICMP=Internet Control Message Protocol
IGMP=Internet Group Management Protocol
OSPF=Open Shortest Path First

Applicazioni e protocollo trasporto

1) Posta elettronica, accesso da terminale remoto, Web, trasferimento di file → TCP

Necessario il servizio di trasferimento dati affidabile fornito da TCP

2) Aggiornamento tabelle di routing in RIP → UDP

Aggiornamenti periodici delle tabelle → eventuali informazioni perse sostituite da informazioni più aggiornate

3) DNS query → **UDP**

Si evitano i ritardi dovuti all'instaurazione della connessione

4) Telefonia Internet → **UDP**

Tolleranza alla perdita di dati (no servizio affidabile)

Tasso di trasmissione costante (no controllo di congestione)

5) Applicazioni multimediali → **UDP**

Principali servizi Internet TCP-based

- **Posta elettronica (SMTP)**
- **Login remoto (Telnet)**
- **Trasferimento file (FTP)**
- **World Wide Web**

Tutti usano modello *client/server*

Modulo 1:
Introduzione al Web

Quando e perché nasce

Tim Berners-Lee (1989, CERN di Ginevra)

- “The current incompatibilities of the platforms and tools make it impossible to access existing information through a common interface, leading to waste of time, …”
- “A link is specified as an ASCII string from which the browser can deduce a suitable method of contacting an appropriate server. When a link is followed, the browser addresses the request for the node [document] to the server.”

Ingredienti del Web

- **Meccanismi di comunicazione e naming di Internet**
  - Protocollo TCP/IP
  - Sistema DNS
- **Sistema client-server**
- **“Solo” tre nuovi standard**
  - URL: Sistema di indirizzamento delle risorse
  - HTML: Linguaggio di markup ipertestuale
  - HTTP: Protocollo per le richieste risorse

Modulo 2:
Meccanismi di naming

Meccanismi di naming del Web

L'insieme di tutti i meccanismi standard di naming delle risorse Web è l'Uniform Resource Identifiers (URI) che si compone di:

- Uniform Resource Locator (URL): specifica la locazione fisica delle risorse e le modalità di accesso
- Uniform Resource Name (URN): proposta per la rappresentazione univoca dei nomi delle risorse in modo da garantire persistenza e disponibilità
- Uniform Resource Characteristics (URC): proposta per la descrizione delle risorse basata su coppie attributo-valore. Es.: autore, data, copyright

Uniform Resource Locator (URL)

Il sistema di indirizzamento delle risorse è basato su Uniform Resource Locator (URL), un meccanismo standard per fare riferimento a tutte le risorse presenti nel Web:

- pagine (testo, immagini, suoni, video, ...)
- risultati di esecuzioni
- programmi eseguibili

Campi dell'URL

schema :// **host.domain** / **pathname**

http :// **www.dsi.unimo.it** / **docenti/esami.html**

- **schema**: indica il modo con cui accedere alla risorsa, cioè quale protocollo bisogna usare per interagire con il server che controlla la risorsa. Il metodo di accesso più comune è ***HTTP*** (protocollo nativo del WWW per il recupero di risorse Web)
- **host.domain**: è l'hostname del nodo nel quale risiede la risorsa Web.
- **pathname**: identifica la risorsa presso il server Web.

Campi dell'URL

schema :// host.domain:porta / pathname

http :// www.dsi.unimo.it:80 / esami.html

- schema: indica il modo con cui accedere alla risorsa, cioè quale protocollo bisogna usare per interagire con il server che controlla la risorsa. Il metodo di accesso più comune è HTTP (protocollo nativo del WWW per il recupero di risorse Web)
- host.domain: è l'hostname del nodo nel quale risiede la risorsa Web. Può essere preceduto da username e password
- pathname: identifica la risorsa presso il server Web.

Altre definizioni

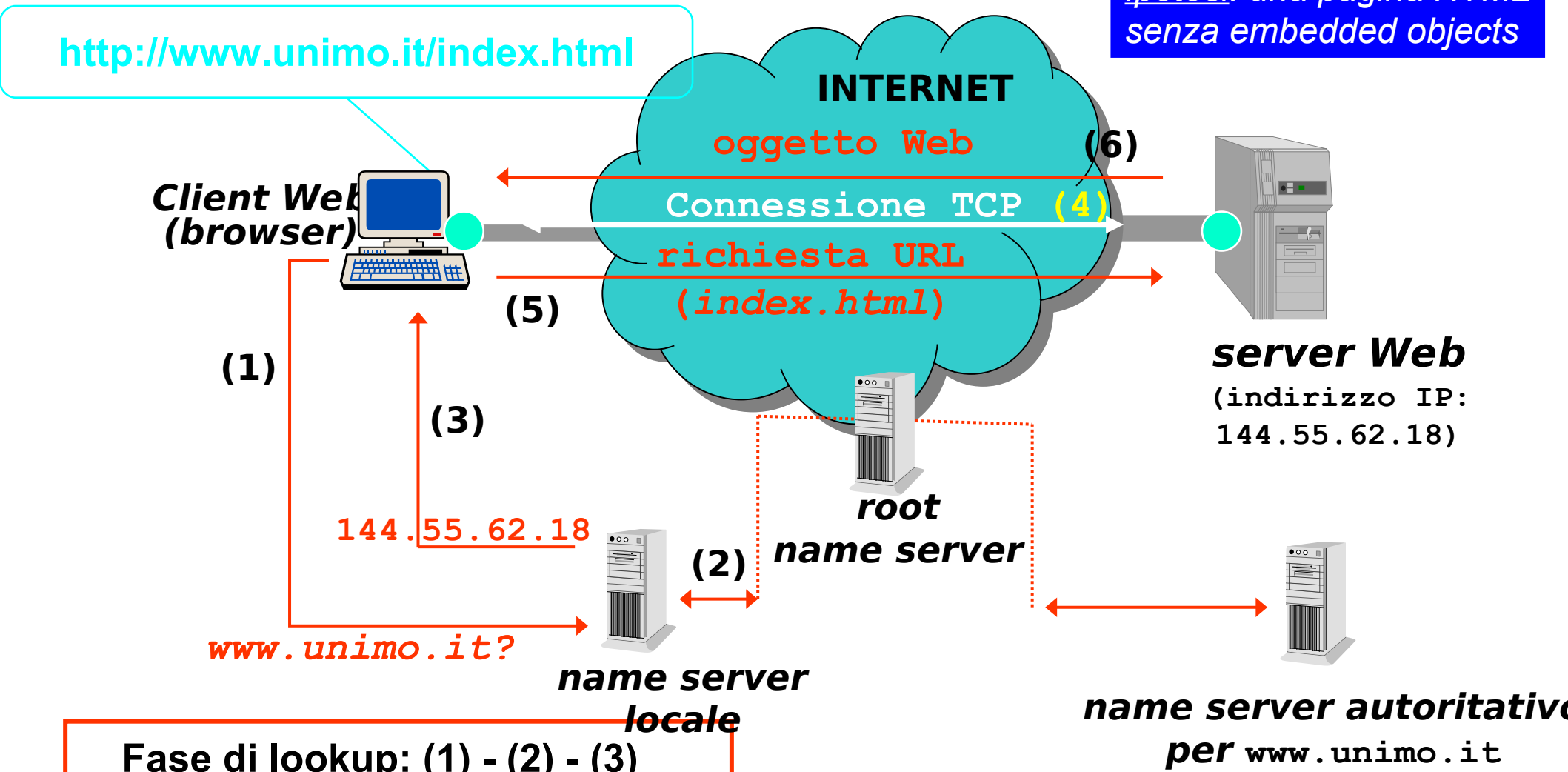
- **Sessione utente: serie di richieste di risorse effettuate dallo stesso utente al medesimo sito Web**

- **Richiesta di risorsa (o pagina): una singola richiesta effettuata dall'utente (→ il tipico click), che tipicamente consiste di multiple richieste di hit inviate dal client dell'utente al sito Web**

- **Hit: una richiesta per un singolo oggetto effettuata dal client al server Web**

Richiesta di una risorsa Web
(cosa succede ad ogni click del mouse?)

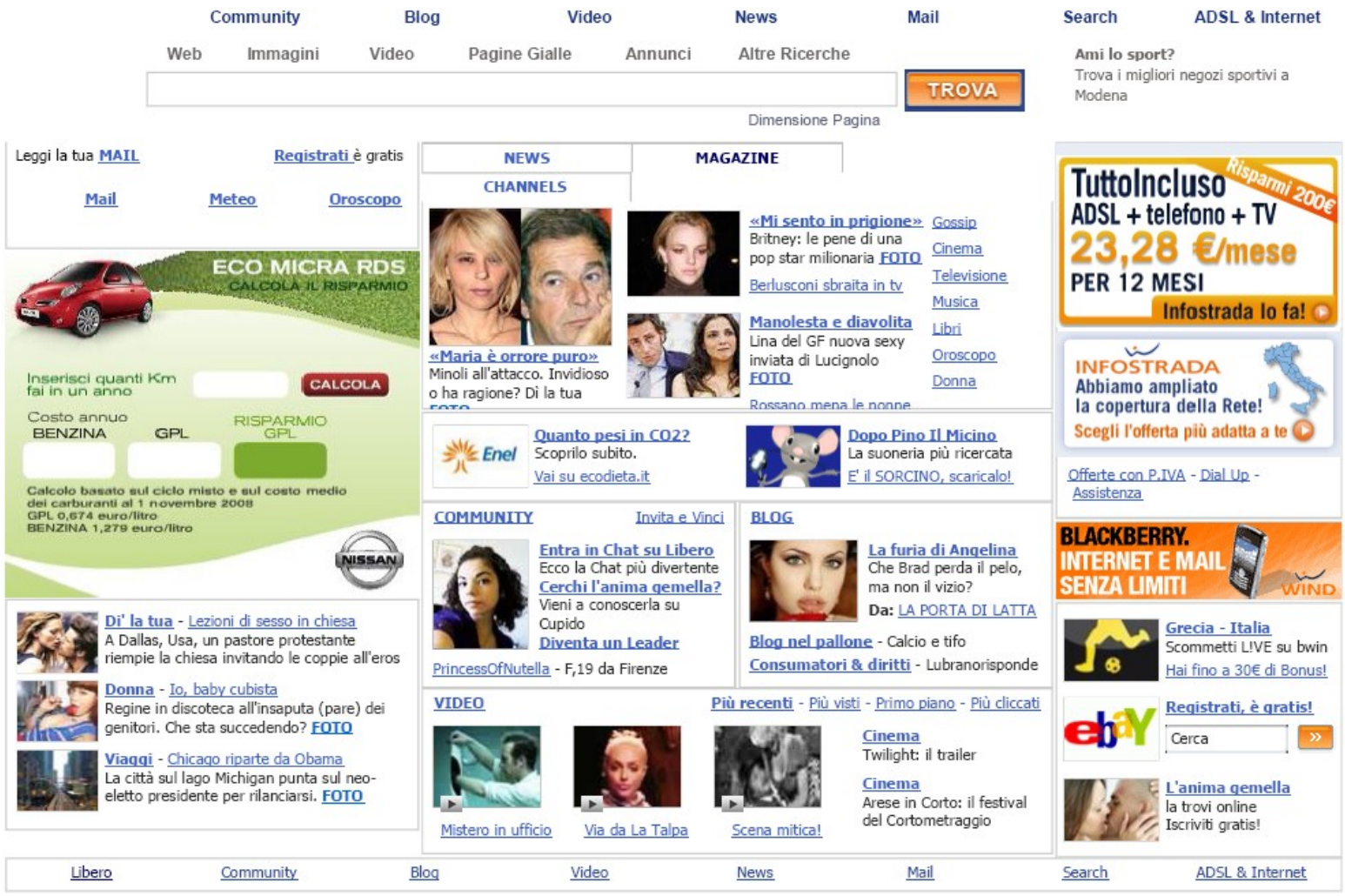
Ipotesi: una pagina HTML senza embedded objects



Fase di lookup: (1) - (2) - (3)
Fase di connessione TCP: (4)
Fase di richiesta: (5) - (6)

Tipica risorsa Web

Pagina HTML + *embedded objects* (risorse di ogni tipo)



Modulo 3:
Linguaggio di markup

Ingredienti del Web

- **Meccanismi di comunicazione e naming di Internet**
  - Protocollo TCP/IP
  - Sistema DNS
- **Sistema client-server**
- **“Solo” tre nuovi standard**
  - URL: Sistema di indirizzamento delle risorse
  - HTML: Linguaggio di markup ipertestuale
  - HTTP: Protocollo per le richieste risorse

Documento Web

- **Una pagina è costituita da vari oggetti (testo, immagini binarie, …), detti *embedded objects*.**
- **Ad ogni oggetto corrisponde un file.**
- **Caratteristiche del testo tipico:**

  Rappresentazione in standard ASCII

  Si specifica sia il contenuto sia la rappresentazione (layout)

  Scritto nel **linguaggio di markup HTML**

Concetto di Ipermedia

- **I documenti Web tipicamente contengono un insieme di**

  testo

  immagini

  puntatori selezionabili ad altre risorse

  (audio)

  (video) **Ipermedia**

- **Uso mediante “point-and-click”**

Linguaggio di markup

Per permettere la corretta visualizzazione dell'informazione su qualsiasi piattaforma hardware connessa in rete è stato necessario definire un nuovo linguaggio di markup per la formattazione delle pagine:

- Fornisce delle linee guida generali per la rappresentazione del contenuto.
- Non specifica esattamente il formato e la posizione del testo, lasciando ai browser la definizione dei dettagli.

→ **Due browser potrebbero visualizzare lo stesso documento in modo differente**

Linguaggio HTML

- Sebbene siano stati proposti modifiche ed altri standard*, a tutt'oggi *HyperText Markup Language* (HTML) nelle sue varie incarnazioni rimane il "**linguaggio del Web**".
- La pagina HTML è un file di solo testo ASCII.
- Il testo è *free-format*.
- Contenuto del testo e specifiche di formato possono essere inseriti nello stesso file.

* **XML è lo standard più importante**

Statement HTML

- La descrizione dei contenuti dell’ipertesto viene effettuata inserendo all’interno del testo stesso alcune istruzioni dette *marcatori* o *markup* o *tag* che producono le visualizzazioni e le azioni specificate.
- Gli statement HTML sono racchiuse tra parentesi angolari, nella forma **<tag>**, e vengono terminate da un tag di chiusura nella forma **</tag>**.

Immagini in un documento HTML

- Ciascuna immagine è contenuta in un file differente dal testo.

- Uso di espliciti tag per le immagini:

```
<IMG SRC "Pathname_del_file" >
```

- Possibilità di allineare le immagini al testo:

```
<IMG SRC "Pathname_del_file" align=middle>
```

Schema di un documento HTML

```
<HTML>
    <HEAD>
        <TITLE>
            Titolo del documento
        </TITLE>
    </HEAD>
    <BODY>
        Corpo del documento
    </BODY>
</HTML>
```

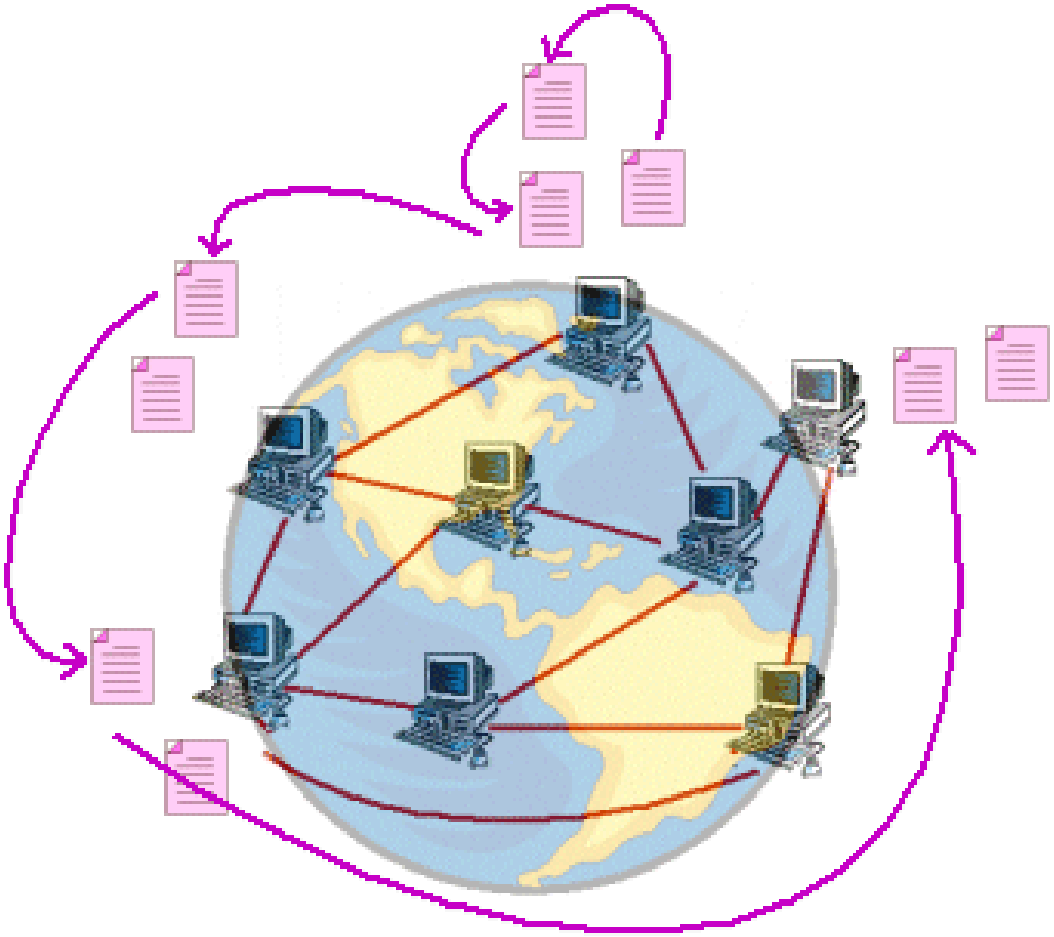
Tag àncora

- L’istruzione più innovativa dell’HTML è l’àncora delimitata dai tag <A>...</A>, in quanto tale elemento permette di trasformare un normale testo in ipertesto multimediale.
- Un àncora può far riferimento ad una sezione della stessa pagina oppure ad una qualsiasi risorsa (testuale, multimediale, eseguibile) presente sul Web, denotata mediante un URL che va inserito all’interno del tag àncora.

<A HREF="http://www.unimo.it/studenti/erasmus.html”>
Programma Erasmus </A>

- Il testo Programma Erasmus viene visualizzato in modo differente e risulta un link simbolico selezionabile via mouse.

Il tag àncora consente l'ipermedialità su scala geografica



Conseguenza dirompente dal punto di vista storico e sociale: l'informazione non è più organizzata in rigidi schemi (es., biblioteca), ma è accessibile (anche) mediante "link" decisi da singole persone, aziende, organizzazioni, …

Ogni link inserito tra una pagina Web e un'altra determina una nuova modalità di accesso all'informazione

Modulo 4:
Protocollo HTTP

Ingredienti del Web

- **Meccanismi di comunicazione e naming di Internet**
  - Protocollo TCP/IP
  - Sistema DNS
- **Sistema client-server**
- **“Solo” tre nuovi standard**
  - URL: Sistema di indirizzamento delle risorse
  - HTML: Linguaggio di markup ipertestuale
  - HTTP: Protocollo per le richieste risorse

Protocollo HTTP

- *HyperText Trasmission Protocol* (HTTP) è il protocollo che permette il reperimento delle risorse Web

- E’ un protocollo applicativo di tipo **request/reply** basato sulla suite di protocolli TCP/IP

- Tutti i client e server Web devono supportare il protocollo HTTP per poter scambiare richieste e risposte. Per questa ragione i **client** e i **server Web** sono chiamati anche **client HTTP** e **server HTTP**

Protocollo HTTP

- **HTTP usa TCP come protocollo di trasporto**
  - il client inizia la connessione TCP verso il server sulla porta 80
  - il server accetta la connessione TCP dal client
  - messaggi HTTP di tipo testuale scambiati tra browser e Web server
  - chiusura della connessione TCP
  - TCP offre un servizio di trasferimento affidabile: i messaggi di richiesta/risposta sono consegnati integri al destinatario
- **HTTP è un protocollo stateless (senza stato)**
  - il server non conserva nessuna informazione riguardante le richieste dei client passati
  - I protocolli che conservano lo stato sono complessi!
    - La storia passata (lo stato) deve essere memorizzata
    - Se il server/client subiscono un crash, la vista dello stato può essere inconsistente e deve essere ristabilita

Richiesta HTTP

- **Una richiesta HTTP comprende**
  - metodo
  - URL
  - identificativo della versione del protocollo HTTP
  - insieme di extension header
- **Il metodo specifica il tipo di operazione che il client richiede al server. Il metodo più comune è GET che serve per acquisire pagine Web.**
- **Gli header contengono informazioni addizionali, quali la data e l'ora della comunicazione, il tipo di software utilizzato dal client, i tipi di dato che il browser è in grado di visualizzare, per un totale di circa 50 tipi di header differenti.**

Messaggi HTTP

Due tipi di messaggi:

- messaggi di **richiesta** HTTP
- messaggi di **risposta** HTTP

- **Messaggio di richiesta HTTP: ASCII**

Linea di richiesta (comandi GET, POST, HEAD)

Linee header

Carriage return, line feed indicanti la fine del messaggio

```
GET /somedir/page.html HTTP/1.1
Connection: close
User-agent: Mozilla/4.0
Accept: text/html, image/gif, image/jpeg
Accept-language:fr
```

(extra carriage return, line feed)

[optional] request body

Richiesta HTTP

- **Method: tipo di operazione richiesta dal client**
  - GET: richiesta di un oggetto
  - POST: il client richiede una pagina Web il cui contenuto è specificato dall'utente (es. richiesta ad un motore di ricerca) nel campo entity body
  - HEAD: il client richiede che il server invii soltanto l'header della risposta senza l'oggetto (usato per debugging dei Web server)
  - PUT, DELETE
  - LINK, UNLINK (HTTP 1.0)
  - TRACE, CONNECT, OPTIONS (HTTP 1.1)
- **URL: identificatore dell’oggetto richiesto**
- **version: versione del protocollo HTTP**

Metodi HTTP

Metodo	Richiesta	Versione
GET	Ricevere una risorsa dal server	>=0.9
HEAD	Ricevere il solo header di una risorsa	>=1.0
POST	Inviare dati al server	>=1.0
PUT	Inviare un oggetto al server	>=1.1
DELETE	Cancellare un oggetto dal server	>=1.1
LINK / UNLINK	Creare o eliminare collegamenti fra oggetti del server	<=1.0
TRACE	Individuare la catena dei server proxy	>=1.1
CONNECT	Creare connessione	>=1.1

Messaggio di richiesta HTTP (cont.)

Header lines

Connection: tipo di connessione richiesta dal client (persistente, non-persistente)

- User-agent: browser utilizzato dall'utente
- Accept: tipo di oggetti che il client accetta
- Accept-language: preferenza della lingua
- Accept-encoding, Accept-charset
- Host: specifica host che ha la risorsa (HTTP 1.1)
- Altri header per garantire la consistenza delle informazioni (es., If-Match o If-Modified-Since)

Risposta HTTP

- **Una risposta HTTP comprende, oltre al contenuto della risorsa richiesta, un header contenente l'identificativo della versione del protocollo HTTP, il codice di stato, l'informazione di stato in forma testuale, ed un insieme di possibili altre informazioni di risposta.**
- **Se la pagina richiesta, oltre al testo HTML, contiene altri oggetti, ciascuno di essi sarà identificato da un URL differente, per cui è necessario che il browser invii un esplicito messaggio di richiesta per ognuno degli elementi collegati alla pagina.**

Messaggio di risposta HTTP

Linea di stato
(codice dello stato del
protocollo,
frase corrispondente)

Linee
header

dati, ad es.
il file HTML
richiesto

```
HTTP/1.1 200 OK
Connection: close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 …...
Content-Length: 6821
Content-Type: text/html
(extra carriage return, line feed)
response body
```

Messaggio di risposta HTTP

Version: versione del protocollo HTTP

Status code, phrase: esito della richiesta (codice e frase)

- **1xx: Informazioni**
- **2xx: Successo**
- **3xx: Redirezione**
- **4xx: Errore del client**
- **5xx: Errore del server**

Alcuni status code di risposta

HTTP Status Codes

For great REST services the correct usage of the correct HTTP status code in a response is vital.

1xx – Informational	2xx – Successful	3xx – Redirection	4xx – Client Error	5xx – Server Error
This class of status code indicates a provisional response, consisting only of the Status-Line and optional headers, and is terminated by an empty line	This class of status code indicates that the client's request was successfully received, understood, and accepted.	This class of status code indicates that further action needs to be taken by the user agent in order to fulfill the request.	The 4xx class of status code is intended for cases in which the client seems to have erred.	Response status codes beginning with the digit "5" indicate cases in which the server is aware that it has erred or is incapable of performing the request.
100 – Continue 101 – Switching Protocols 102 – Processing	200 – OK 201 – Created 202 – Accepted 203 – Non-Authoritative Information 204 – No Content 205 – Reset Content 206 – Partial Content 207 – Multi-Status	300 – Multiple Choices 301 – Moved Permanently 302 – Found 303 – See Other 304 – Not Modified 305 – Use Proxy 307 – Temporary Redirect	400 – Bad Request 401 – Unauthorised 402 – Payment Required 403 – Forbidden 404 – Not Found 405 – Method Not Allowed 406 – Not Acceptable 407 – Proxy Authentication Required 408 – Request Timeout 409 – Conflict 410 – Gone 411 – Length Required 412 – Precondition Failed 413 – Request Entity Too Large 414 – Request URI Too Long 415 – Unsupported Media Type 416 – Requested Range Not Satisfiable 417 – Expectation Failed 422 – Unprocessable Entity 423 – Locked 424 – Failed Dependency 425 – Unordered Collection 426 – Upgrade Required	500 – Internal Server Error 501 – Not Implemented 502 – Bad Gateway 503 – Service Unavailable 504 – Gateway Timeout 505 – HTTP Version Not Supported 506 – Variant Also Negotiates 507 – Insufficient Storage 510 – Not Extended

Examples of using HTTP Status Codes in REST

201 – When doing a POST to create a new resource it is best to return 201 and not 200.
204 – When deleting a resources it is best to return 204, which indicates it succeeded but there is no body to return.
301 – If a 301 is returned the client should update any cached URI's to point to the new URI.
302 – This is often used for temporary redirect's, however 303 and 307 are better choices.
409 – This provides a great way to deal with conflicts caused by multiple updates.
501 – This implies that the feature will be implemented in the future.

Special Cases

306 – This status code is no longer used. It used to be for switch proxy.
418 – This status code from RFC 2324. However RFC 2324 was submitted as an April Fools' Joke. The message is *I am a teapot.*

Key	Description
Black	HTTP version 1.0
Blue	HTTP version 1.1
Aqua	Extension RFC 2295
Green	Extension RFC 2518
Yellow	Extension RFC 2774
Orange	Extension RFC 2817
Purple	Extension RFC 3648
Red	Extension RFC 4918

BARONE, BUDGE & DOMINICK
Business•Technology•Solutions

REST is a style of development for exposing data in such a way that it is easy to consume, easy to produce and makes use of HTTP.
REST is not a replacement for SOAP or WS* but an alternative to solve certain problems better.

Architecture

From SADev.co.za

Risposte HTTP

Header line

- Connection: tipo di connessione usata dal server
- Date: data e ora della richiesta
- Server: tipo di Web server e di sistema operativo
- Last-Modified: data e ora creazione o modifica dell'oggetto (caching)
- Content-Length: dimensione in byte dell'oggetto
- Content-Type: tipo di oggetto (es. HTML, GIF, ...)

- **Entity body: oggetto**

Esempio HTTP

L'utente digita l'URL
www.someSchool.edu/someDepartment/home.index
(il documento contiene testo e i riferimenti a 10 immagini JPEG)

1a. Il client HTTP inizia la connessione TCP al server HTTP a www.someSchool.edu. 80 è la porta di default per server HTTP.

1b. Il server HTTP all'host www.someSchool.edu in attesa di una connessione TCP sulla porta 80. Accetta la connessione, notificandolo al client

2. Il client HTTP invia un ***messaggio di richiesta*** HTTP (contenente l'URL)

3. Il server riceve il messaggio di richiesta, forma il ***messaggio di risposta*** contenente l'oggetto richiesto (someDepartment/home.index), e lo invia

tempo

Esempio HTTP

4. Il server HTTP chiude la connessione TCP.

5. Il client HTTP riceve il messaggio di risposta contenente il file HTML, visualizza html. Analizza il file HTML (***parsing***), trova 10 riferimenti ad immagini JPEG

tempo

6. Passi da 1a 5 ripetuti per ciascuno dei 10 oggetti JPEG

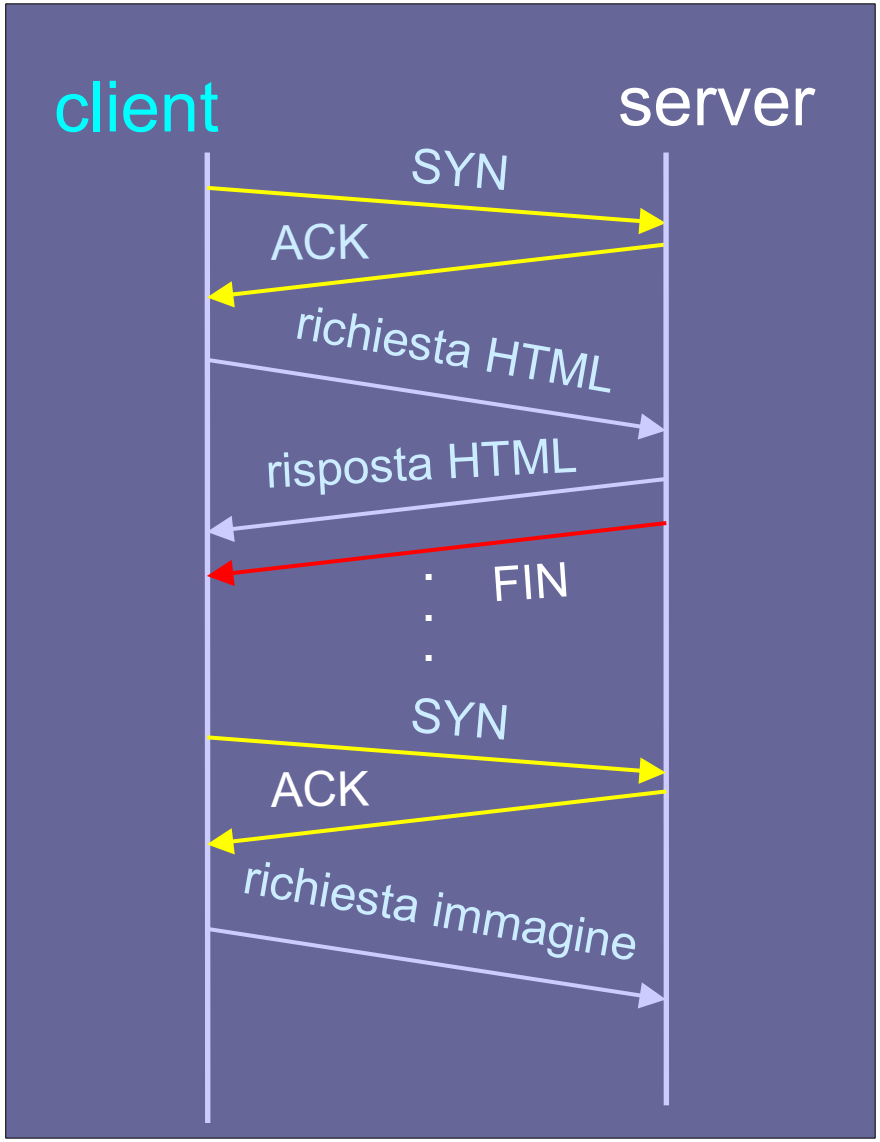
Connessioni TCP nell'HTTP 1.0

- **Connessione non-persistente**: ogni oggetto viene trasferito usando una nuova connessione TCP

Versione HTTP/1.0 (RFC 1945)

- problema: three-way handshake del TCP: per ogni trasferimento di oggetto, in più anche un round-trip time per instaurare la connessione
- problema: slow-start del TCP (la finestra ha dimensione pari a 1 all'inizio di ogni nuova connessione)

Soluzione parziale: alcuni browser creano *simultaneamente* connessioni TCP multiple (una per oggetto), dopo aver analizzato i riferimenti presenti nel file HTML



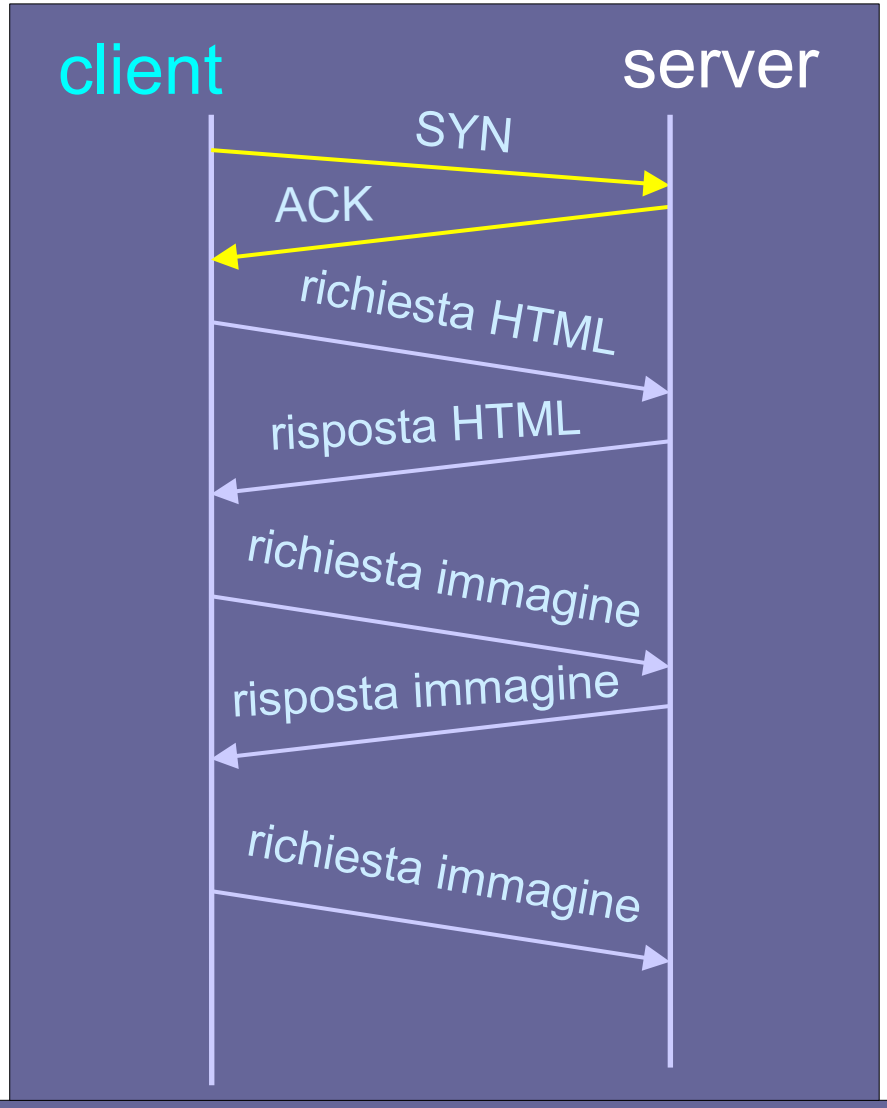
Connessioni TCP nell'HTTP 1.1

- **Connessione persistente**: un numero multiplo di oggetti (ad es. tutti gli oggetti che compongono una pagina) vengono trasferiti entro una stessa connessione TCP

Versione HTTP/1.1 (RFC 2616)

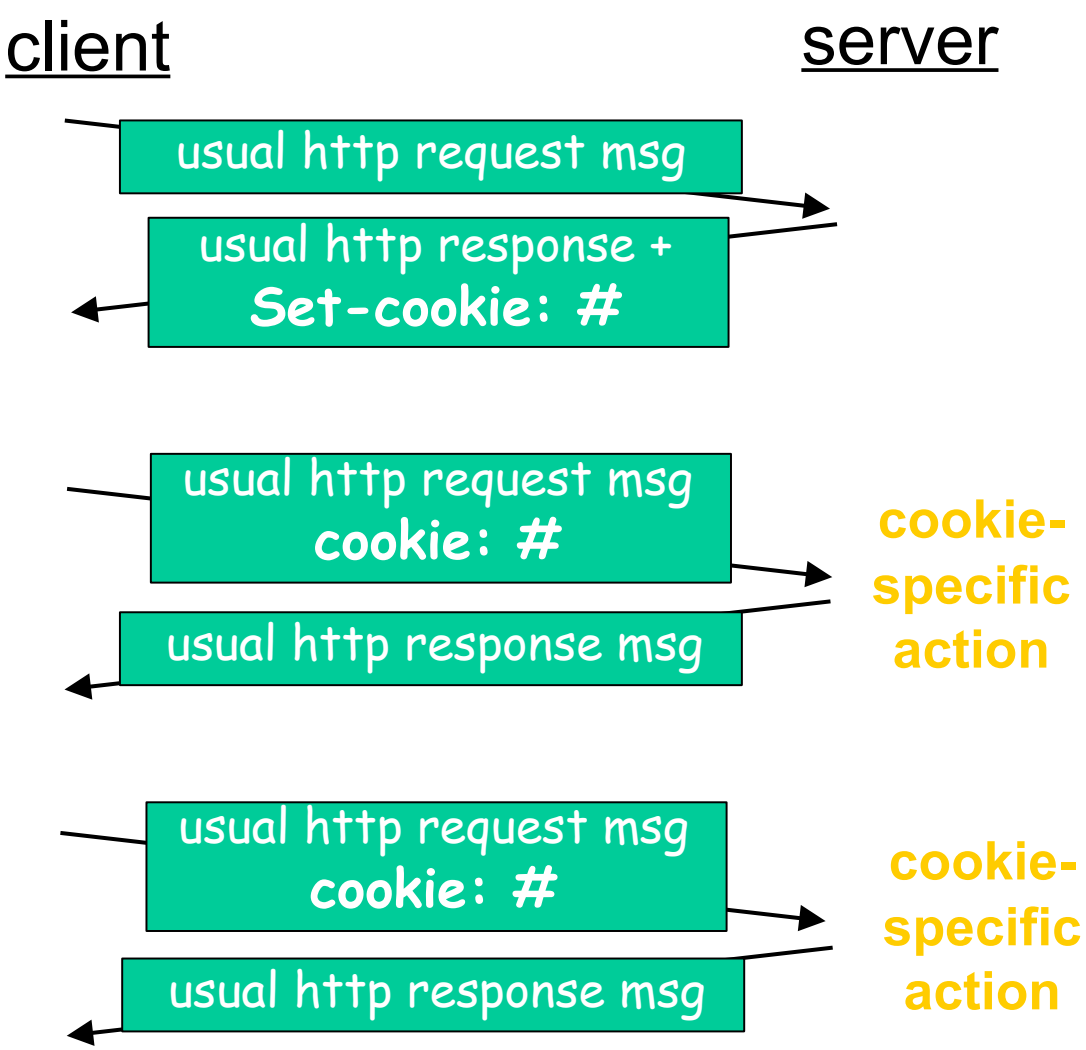
- three-way handshake del TCP: solo per instaurare la connessione iniziale
- controllo di congestione a regime

- **Pipelining** in HTTP/1.1: il browser, dopo aver ricevuto ed analizzato il file HTML, invia più richieste consecutive sulla stessa connessione TCP senza aspettare di ricevere la risposta



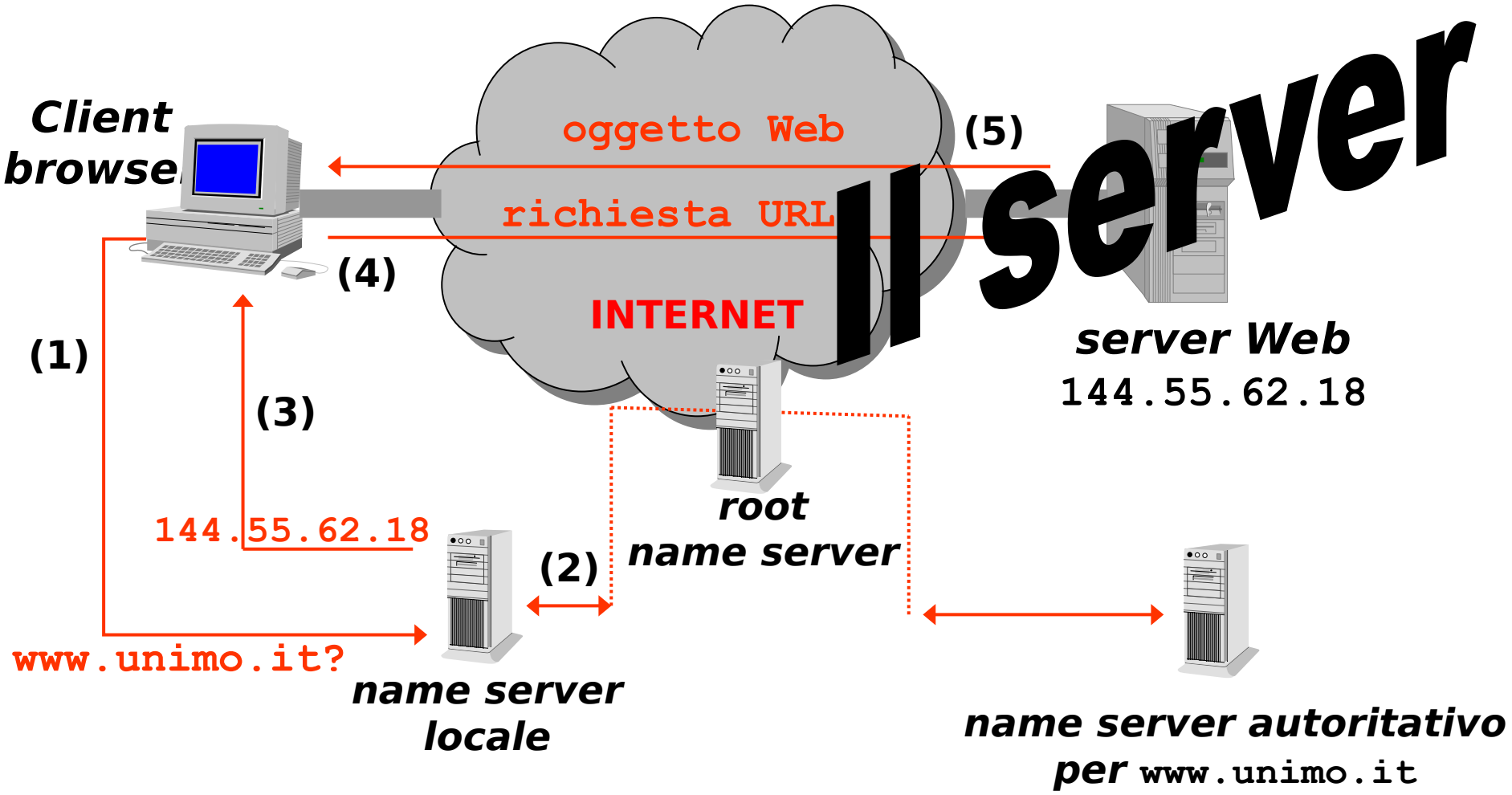
Interazione utente-server: cookie

- Il server invia al client un "cookie" in risposta a
  - **Set-cookie: #**
- Il client presenta il cookie nelle richieste successive
  - **cookie: #**
- Il server può confrontare il cookie presentato con i cookie da lui memorizzati per
  - identificare il client
  - cercare di ricostruire la “sessione” per seguire le preferenze e la navigazione degli utenti
- L’utente può cancellare i cookie (funzione dei browser)



Modulo 5: Server HTTP

Macro-componenti del Web



Componenti di un sito Web

- **Piattaforma hardware**
- **Software di base**
- **Server Web**
  - Il processo server HTTP ed il relativo software che viene eseguito sulla piattaforma stabilisce il collegamento tra la piattaforma (hardware - software di base) e la parte informativa del sito Web.
- **Parte informativa e servizi del sito Web**
  - Il sito deve mettere a disposizione un insieme di risorse Web che possono essere richieste dai client con cui vengono instaurate delle connessioni HTTP.

Componenti di un sito Web

- **Piattaforma hardware**
- **Software di base**
- **Server Web**
  - Il processo server HTTP ed il relativo software che viene eseguito sulla piattaforma stabilisce il collegamento tra la piattaforma (hardware - software di base) e la parte informativa del sito Web.
- **Parte informativa e servizi del sito Web**
  - Il sito deve mettere a disposizione un insieme di risorse Web che possono essere richieste dai client con cui vengono instaurate delle connessioni HTTP.

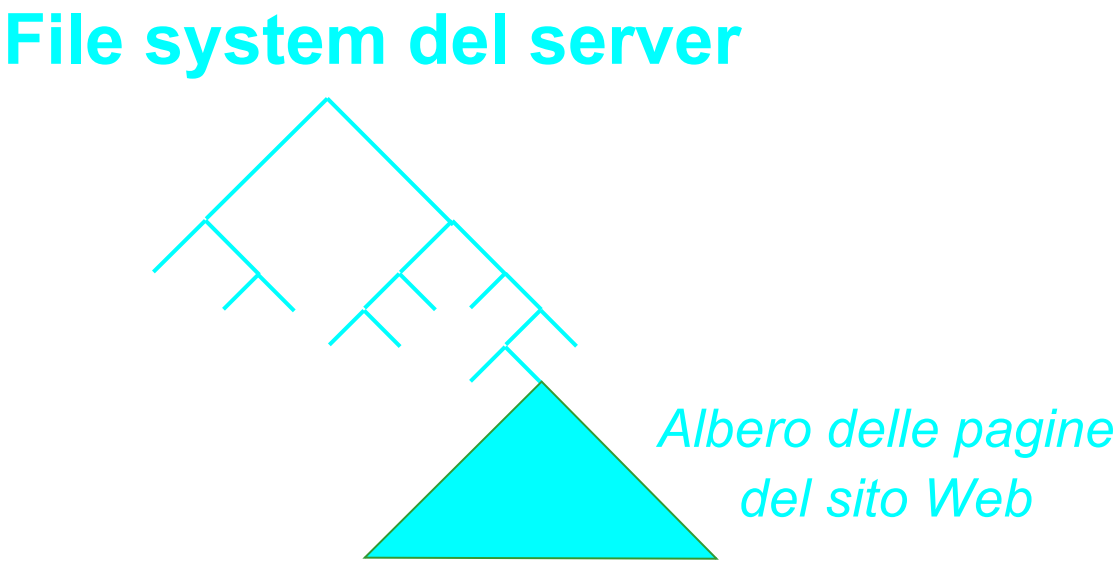
Parte informativa

- **Generalmente organizzata in risorse iper-mediali con link verso altre risorse (interne ed esterne al sito Web).**
- **L'organizzazione delle pagine è tipicamente gerarchica ad albero, in cui vi è un punto di partenza e tutte le altre pagine sono poste al di sotto della radice dell'albero.**
- **L'albero delle pagine Web è organizzato in modo simile ad un file system gerarchico con una radice e directory che contengono altre directory o file.**

Organizzazione delle pagine (1)

- Ogni pagina Web ha un nome unico, che corrisponde al cammino assoluto dalla radice “/” dell’albero delle pagine.

(Questo cammino è proprio quello specificato nella parte *pathname* dell’URL richiesto dal client.)



Organizzazione delle pagine (2)

- L’albero delle pagine Web non riflette necessariamente la vera organizzazione dei file all’interno del file system.
- L’organizzazione fisica che rispecchia fedelmente quella logica è solo una delle possibili alternative.
- Per motivi di efficienza organizzativa (gruppi differenti possono creare o fornire le informazioni per le pagine) o di efficienza nella risposta, l’albero delle pagine Web può essere partizionato tra due o più dischi della stessa piattaforma o addirittura tra piattaforme differenti, utilizzando o meno meccanismi di Network File System.

Tipi di risorse - 1 (sulla base del contenuto)

- pagina HTML
- testo in formato ASCII
- pagina preformattata (come PostScript, PDF)
- immagine in diversi formati (tipo GIF, JPEG)
- suono codificati in diversi formati (quali AU, AIFF, MP3)
- video in diverse rappresentazioni (quali Quicktime, MPEG)
- rappresentazione VRML di scene tridimensionali
- codice eseguibile in linguaggi interpretati (tipo Perl e shell)
- codice eseguibile in linguaggi compilati, tipo C
- codice Java

Tipi di risorse - 2 (classificazione funzionale)

Risorse statiche

pagine il cui contenuto è relativamente stabile nel tempo.

Risorse volatili

pagine il cui contenuto viene modificato da eventi in corso. Es., ultime notizie, avvenimenti sportivi, titoli in borsa.

Risorse dinamiche

pagine il cui contenuto è creato dinamicamente sulla base della richiesta del client. Es., CGI.

Risorse attive

pagine in cui il server Web invia un applet Java al browser, che esegue il contenuto sulla piattaforma client.

Componenti di un sito Web

- **Piattaforma hardware**
- **Software di base**
- **Server Web**
  - Il processo server HTTP ed il relativo software che viene eseguito sulla piattaforma stabilisce il collegamento tra la piattaforma (hardware - software di base) e la parte informativa del sito Web.
- **Parte informativa e servizi del sito Web**
  - Il sito deve mettere a disposizione un insieme di risorse Web che possono essere richieste dai client con cui vengono instaurate delle connessioni HTTP.

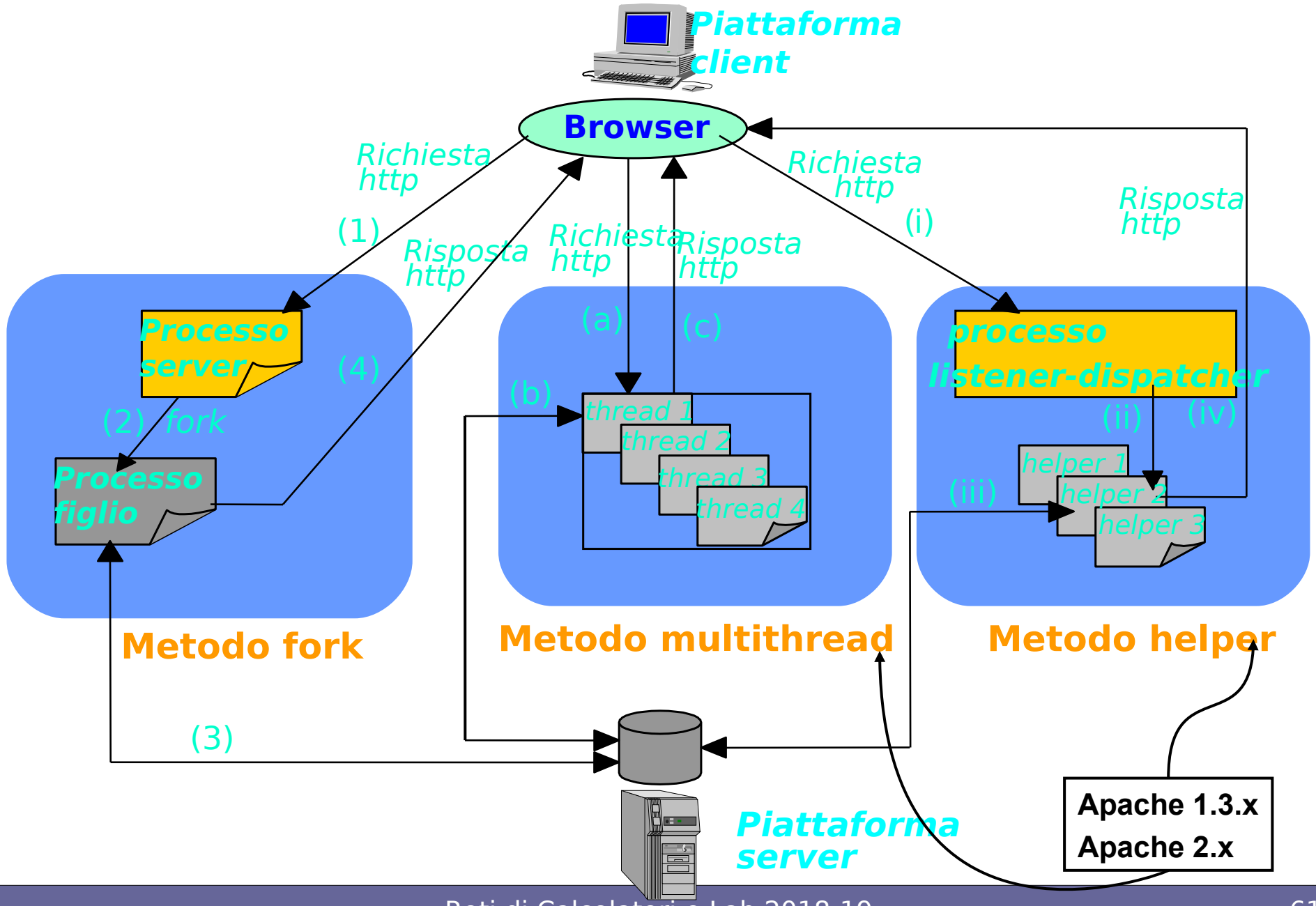
Fasi di una comunicazione TCP

- **Dichiarazione al sistema operativo che si intende instaurare una connessione con specifica delle caratteristiche**
- **Apertura della connessione (differente dal lato server rispetto al lato client):**
  - il server assume di definire la connessione prima del client, e rimane in attesa che il client si connetta alla porta specificata
  - il client assume che il server sia già attivo e prova a connettersi specificando indirizzo e porta del server
- **Scambio di dati bidirezionale (trasmissione e ricezione)**
- **Chiusura della connessione**

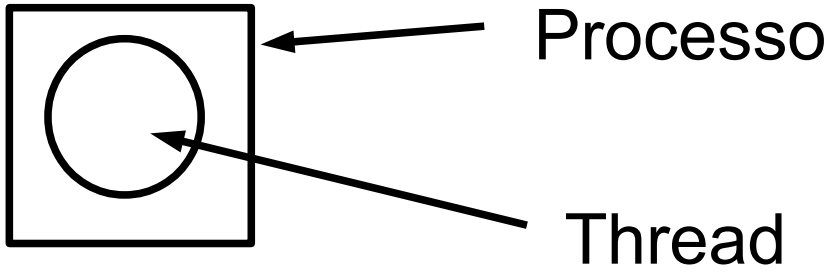
Gestione richieste multiple

- **Il server deve specificare una porta che identifica il servizio sull'host**
- **Tuttavia, più client possono richiedere il servizio in rapida successione**
- **Due soluzioni:**
  - Accodamento della richiesta client arrivata dopo Gestito automaticamente dal sistema operativo; il processo server deve specificare solo la lunghezza (backlog) della coda
  - Gestione contemporanea di più richieste client Possibile mediante la gestione del multitasking da parte del sistema operativo

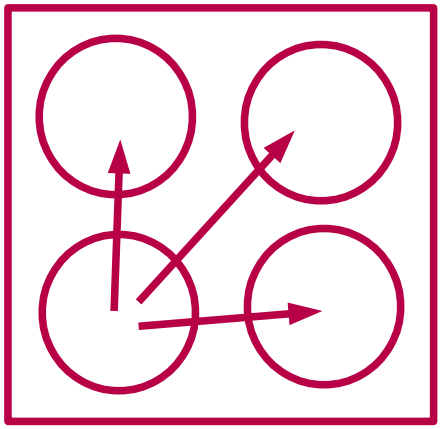
Tre modalità di gestione richieste HTTP



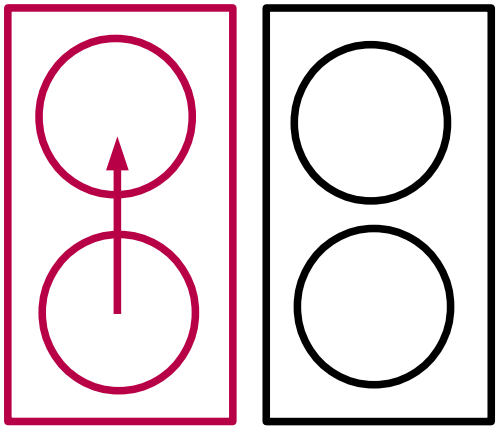
Due parole su sicurezza e prestazioni



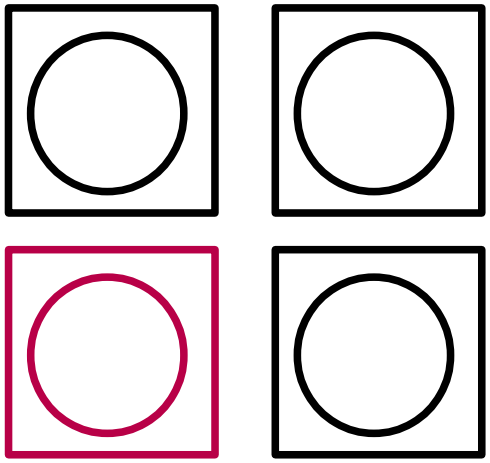
un processo molti thread



molti processi più thread per processo



molti processi un thread per processo



Web server esoterici

- **Sistemi con criticità prestazionali**
- **Colli di bottiglia:**
  - Invocazione fork()
  - Context switch
- **Approccio alternativo:**
  → programmazione event-driven
  - Single process → no fork()
  - No context switch
  - Uso massiccio di callback
  - Programmazione basata su puntatori a funzione
- **Ulteriore step: in-kernel HTTPd (es, tux)**

Web server esoterici

- **Modello molto complesso per debugging**
- **Recentemente adottato in altri contesti**
  - Diffusione di linguaggi di alto livello (es. Java Script, Python)
  - Supporto semplificato per callback (garbage collection, duck typing, no puntatori esplicity)
- **Prestazioni non sono più un problema**
- **Approccio event driven per semplificare lo sviluppo**
- **Adatto a modelli con logica client-side**

Un problema apparente ...

- Una porta viene assegnata ad un servizio, ma nel caso del multitasking vi potrebbero essere più processi server attivi
- D'altro canto, le richieste di un client devono essere inviate al processo server corretto

- **Soluzione:**

  Usare sia le informazioni del server sia le informazioni del client per indirizzare i pacchetti

  Il TCP usa 4 informazioni per identificare una connessione:

  - indirizzo IP del server
  - numero di porta del protocollo lato server
  - indirizzo IP del client
  - numero di porta del protocollo lato client

Motivazione

- **Il Web trascende la sua funzione di tecnologia per**
  - semplificare il request/reply di file statici (*get*)
  - facilitare il reperimento di file (*navigazione point&click*)

- **Per diversi motivi, la tecnologia Web è ormai l'interfaccia eletta per l'interazione della maggior parte (tutti?) dei servizi che si possono usufruire via rete:**
  - interrogazioni: a database, a motori di ricerca, ...
  - comunicazioni: posta elettronica, blog, ...
  - download/upload file
  - fruizione di contenuti multimediali
  - svariati applicativi: dai gestionali in su
  - ...

Evoluzione

- **Terminale “stupido” di mainframe**
- **Client/server, dove il client va inteso come software specifico per interagire con quel server. Forte interazione tra i due software.**
- **Il client diventa un software browser-*like* e il server è in grado di rispondere**
- **Web services**

Dipendenza totale

Forte interazione

Interazione lasca

Completo svincolo

Modulo 6:
Lo “storico” CGI
(Common Gateway Interface)

Script e Gateway

Meccanismi che consentono

- **al client Web la capacità di eseguire un'applicazione sulla piattaforma server**
- **al server Web di connettersi ad altri servizi e poter recuperare informazioni dall'utente, eventualmente mediante form interattive**
- **SCRIPT: definizione generica di un qualsiasi programma eseguito dal server (tipicamente, ma non necessariamente implementato in un lingua interpretato)**
- **GATEWAY: uno script che fornisce accesso ad un servizio online (es., ad un database) svolgendo un ruolo da interfaccia tra il server Web ed un altro server**

Azioni di uno script/gateway

- Tradurre l’input del client (il file HTML proveniente dalla connessione HTTP) in forma comprensibile ai servizi a cui si collega (es., query nel linguaggio del database)
- Invocare l’attivazione di altri programmi eseguibili
- Tradurre l’output del programma in una forma comprensibile al client (es., l’informazione restituita dal database in un formato compatibile con il protocollo HTTP)

Implementazione di script

- Uno script Web, nella sua forma più generale, è un qualsiasi programma che può essere eseguito dal server in risposta ad una richiesta Web
- Non importa il linguaggio con cui lo script è implementato
  - C/C++
  - Perl
  - TCL
  - Unix shell
  - Visual Basic
  - Altri strumenti proprietari (AppleScript, …)
- L'importante è che sia in grado di leggere da STDIN, scrivere su STDOUT e leggere le variabili ambiente

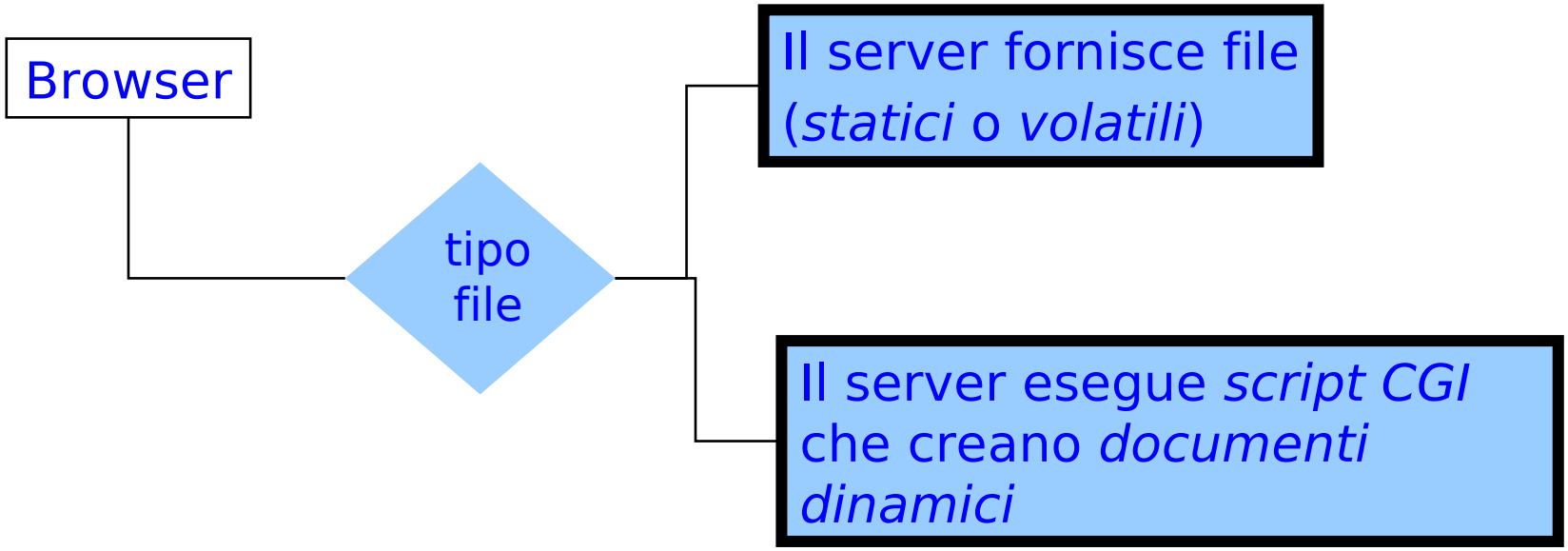
Tecnologia CGI (primo standard de facto)

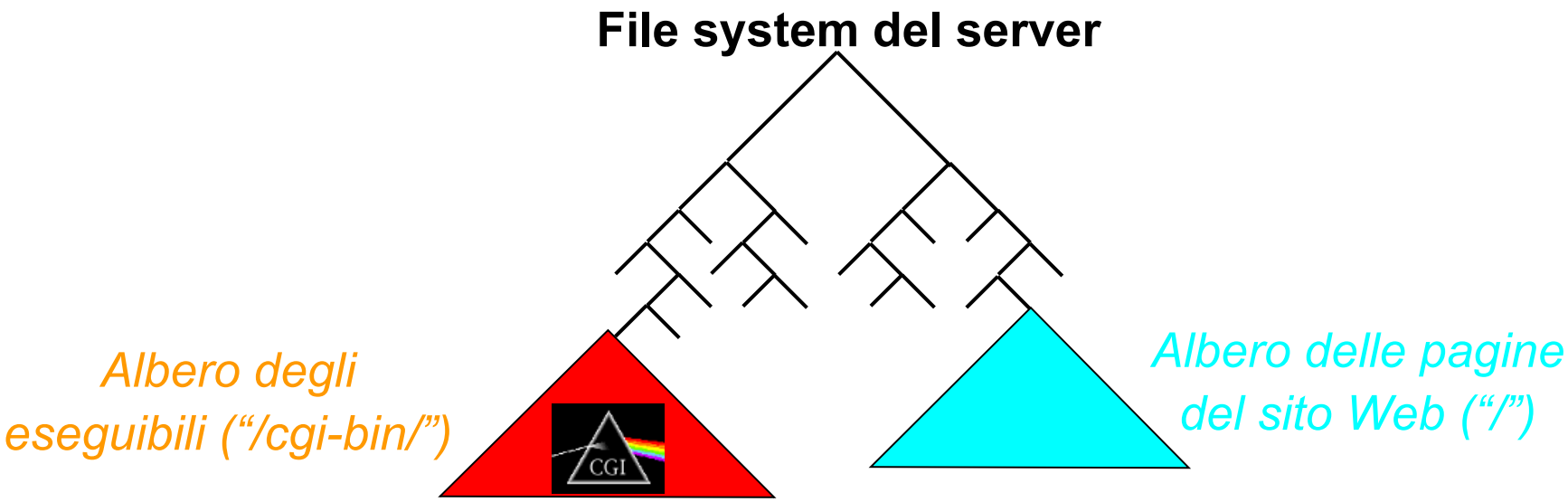
Come garantire che il server e lo script possano interagire correttamente tenendo conto che i due software sono tipicamente implementati da persone differenti in linguaggi potenzialmente diversi?

- Una soluzione è lo standard NCSA
  Common Gateway Interface (**CGI**)
  che specifica
  - come gli script devono essere invocati
  - come devono essere passati i dati dal processo server allo script e viceversa

- **Lo standard CGI svolge tra processo server e script lo stesso ruolo che il protocollo HTTP svolge tra browser e processo server.**
- Lo standard è così importante che spesso gli script e i gateway vengono chiamati rispettivamente ***script CGI*** e ***gateway CGI***

Suddivisione dello spazio server





Azioni del server HTTP

- **Il server HTTP deve determinare che la risorsa richiesta nell'URL non è una pagina Web, ma un programma.**
- **Il processo deve localizzare il programma e controllare se può essere eseguito. Possibili scelte:**
  - directory /cgibin
  - qualunque file abbia estensione .cgi in qualunque directory
  - qualunque file abbia estensione .EXE (server su Windows NT)

Azioni del server HTTP

- **Nel caso in cui le precedenti operazioni siano andate a buon fine, il server HTTP deve attivare il programma script e assicurare che l'eventuale input proveniente dal browser sia passato allo script**
- **Il server HTTP deve leggere l'output dello script e passarlo al client, ovvero deve inviare un messaggio di errore**
- **Il server HTTP deve chiudere la connessione nel momento in cui l'esecuzione dello script è completata**

Configurazione CGI

In Apache si usa la direttiva ScriptAlias

ScriptAlias URL Path

es., ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/

Alternativa:

AddHandler cgi-script cgi pl

Option +ExecCGI

Interazione tra browser e server HTTP

- **Invocazione diretta di un URL (METHOD=GET)**
  - l'utente specifica l'URL di un CGI
  - l'utente seleziona un link all'URL di un CGI

- **Invio di una "fill-in form"**
  - mediante METHOD=GET
  - mediante METHOD=POST

Fill-in form

- **Gli script/gateway acquisiscono l'informazione dall'utente mediante una speciale pagina HTML detta fill-in form inviata dal server alla macchina client**
- **Il browser visualizza il form all'utente come una normale pagina HTML, ma in più raccoglie i dati di input inseriti dall'utente**
- **Il browser rispedisce al server il fill-in form compilato**

Fill-in form

- **I form possono essere utilizzati per raccogliere vari tipi di informazioni, come nome e password dell'utente per poter accedere ai siti Web ad accesso limitato, o una o più parole chiave per consentire la ricerca in database.**
- **I form potrebbero anche essere embedded (es., imagemap o mappe sensibili)**

Tag <FORM> di HTML

<FORM> </FORM> *Attributi*:

METHOD=GET
METHOD=POST
ACTION="*http://www.unimo.it/cgi-bin/script.cgi*"

<INPUT> *Attributi*:

TYPE=TEXT	(input testuale)
TYPE=CHECKBOX	(selezione)
TYPE=RADIO	(selezione esclusiva)
TYPE=SUBMIT	(tasto per inviare i dati)
TYPE=RESET	(tasto per cancellare i dati)
NAME="*etichetta*"	
VALUE="*valore_associato*"	(se selezionato)
CHECKED	(selezionato per default)

<SELECT> <OPTION>...</SELECT> (menù a tendina)

Attributi:

NAME="*etichetta*"

Un esempio di Fill-in Form

un esempio di form

Dove studi?

Quale browser stai usando?

1) Netscape
2) Explorer

Submit Reset

```
<FORM METHOD=POST
            ACTION= “http://www.dsi.unimo.it/cgi-bin/miofile”>
<P>Dove studi?
<INPUT NAME=“city” TYPE=text SIZE=“20”>
<P>Quale browser stai usando?
<P> Netscape <INPUT NAME=“brow” TYPE=radio
VALUE=“netscape”>
<P> Explorer <INPUT NAME=“brow” TYPE=radio
VALUE=“explorer”>
<P><INPUT TYPE=submit> <INPUT TYPE=reset>
</FORM>
```

Metodo GET

- **L'informazione della fill-in form è inviata al server come parte dell'URL, aggiungendo alla stringa URL una QUERY_STRING, separata dal carattere "?"**
- **La query_string è composta da coppie "nome=valore":**
  - nome è il nome della variabile
  - valore è il valore assegnatole

Metodo GET

- **Ogni coppia “nome=valore” è separata dal carattere “&”. Es.,**

  /cgi-bin/script.cgi?city=modena&brow=netscape
- **La query_string è “URL encoded” e si deve decodificare**
- **Lo svantaggio maggiore del metodo GET è che la dimensione massima della QUERY_STRING è limitata dalla dimensione dell’input buffer del server HTTP (tipicamente fissata a 1024 byte)**

Metodo POST

- **L'informazione della fill-in form non è aggiunta all'URL, ma viene inviata dopo tutte le informazioni dell'header della richiesta HTTP**
- **In particolare, tra le informazioni dell'header vi è il campo content-length, che contiene la dimensione (in byte) dell'informazione inviata al server. In questo modo si supera il problema maggiore del metodo GET**
- **Un altro vantaggio del metodo POST è la possibilità di inviare informazioni anche non testuali (utilizzando opportunamente il campo content-type)**
- **L'informazione viene passata dal server HTTP allo script CGI mediante STDIN (si vedrà dopo)**

Esempio: Metodo POST

```
POST /cgi-bin/script.cgi HTTP/1.0
Referer: http://www.spc.com/form.html
User-Agent: Mozilla/4.05 [en] (Win95; I)
Host: www.spc.com
Accept: image/gif, image/x-xbitmap, image/jpeg, */*
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
Content-type: application/x-www-form-urlencoded
Content-length: 23

city=modena&brow=netscape
```

GET	POST
Indicato per pochi parametri (<1024 byte)	Indicato per grandi quantità di dati
Solo parametri testuali	Dati in qualsiasi formato (testi, immagini, video)
Possibilità di inserire un URL con parametri opportuni all'interno di una pagina HTML. Esempio: <A HREF= "http://www.unimo.it /cgi-bin/search?query=ingegneria> Ricerca Facoltà Ingegneria </A>	Solo in risposta ad una fill-in form

URL encoding

- **Caratteri quali spazi, slash, tilde, sono illegali in un URL. Pertanto devono essere "codificati" per poter essere trasmessi dal browser**
- **Spazio → "+"**

Esempio:

Campo "nome" in una form: John Smith

www.unimo.it/cgi-bin/script.cgi?myname=John+Smith

URL encoding

- **Caratteri illegali → “%codice ASCII esadecimale”**

  Esempio:

  Campo “file” in una form: ~filename

www.unimo.it/cgi-bin/script.cgi?myfile=%7Efilename

- **Sarà compito del programmatore dello script CGI “decodificare” l’URL prima di utilizzarne i valori**

Interazione tra server HTTP e CGI

- **Nel momento in cui un client richiede ad un server di lanciare un CGI, può anche passargli delle informazioni da fornire al CGI. Esistono 3 modalità per passare parametri dal server HTTP allo script CGI:**
  - Variabili ambiente
  - Argomenti linea di comando
  - Standard input
- **Esiste un modo per passare risultati dallo script CGI al server HTTP:**
  - Il CGI invia il risultato dell'elaborazione sempre sullo standard output verso il server HTTP, che a sua volta prepara i dati e li rispedisce al client.

Variabili ambiente

- **Una variabile ambiente è un parametro con un nome per trasferire informazioni dal server allo script CGI. Non è necessariamente una variabile dell'ambiente del Sistema Operativo, anche se questa è l'implementazione più comune.**
- **Esempi (rappresentazione canonica: "maiuscole_maiuscole"):**
  - SERVER_SOFTWARE, nome e versione del server
  - SERVER_NAME, hostname o indirizzo del nodo server
  - GATEWAY_INTERFACE, versione CGI (uso: CGI/1.1)
  - QUERY_STRING, informazione contenuta dopo il ? nell'URL
  - REQUEST_METHOD, metodo della richiesta HTTP (GET, PUT, ...)
  - REMOTE_ADDR, indirizzo IP del nodo client che effettua la richiesta
  - REMOTE_USER, se lo script è protetto da autenticazione utente
  - CONTENT_TYPE, per query HTTP POST e PUT
  - CONTENT_LENGTH

Input 1: Variabile ambiente QUERY_STRING

- Il client aggiunge nell'URL un campo query "*nome1=valore1&nome2=valore2&*..." separato dal resto dell'URL mediante il carattere "?". Ad esempio,

GET http://www.unimo.it/cgi-bin/corsi.cgi?attr1=val1&attr2=val2 HTTP/1.0

- Il server HTTP esegue **corsi.cgi** dopo aver inserito la stringa

  "attr1=val1&attr2=val2"

  nella variabile ambiente QUERY_STRING

- Nello script CGI **corsi.cgi** ci dovranno essere:
  - un'istruzione del tipo: **char * str = getenv("QUERY_STRING");**
  - le istruzioni per la decodifica della stringa, in formato *URL encoded*
  - le istruzioni per il parsing della stringa

Input 2: Argomenti linea di comando

- **Metodo semplice, ma non molto utilizzato per script CGI**
- **Utile, tuttavia, quando si vogliono passare pochi parametri (uno, due) senza dover effettuare il parsing di una stringa.**
- **Utilizzate, ad esempio, per richieste di tipo ISINDEX, per ricerche di testo nei file.**
- **Le parole da ricercare sono inserite direttamente nell'URL, senza utilizzare il formato "nome=valore".**
- **Esempio 1 (linea di comando)**
  - http://www.dsi.unimo.it/cgi_bin/corsi.cgi?reti
- **Esempio 2 (variabile di ambiente QUERY_STRING)**
  - http://www.dsi.unimo.it/cgi_bin/corsi.cgi?corso=reti

Passaggio mediante linea di comando

- **Il client aggiunge nell'URL un campo query separato dal resto dell'URL mediante il carattere "?". Ad esempio,**

GET http://www.unimo.it/cgi-bin/corsi.cgi?valore1+valore2 HTTP/1.0

- **Il server HTTP esegue corsi.cgi valore1 valore2**
- **Nello script CGI corsi.cgi i parametri passati sulla linea di comando potranno essere recuperati mediante tipiche istruzioni C:**

```
main(int argc, char** argv)
  {  ...
argv[1];     /* contiene valore1 */
    argv[2];     /* contiene valore2 */
    ...
  }
```

Input 3: Standard input

- **Metodo possibile solo quando l'applicativo CGI viene invocato mediante i metodi POST o PUT dell'HTTP**

  (utilizzati per completare la fill-in form)
- **Il corpo della richiesta del client inviata al server HTTP viene ridiretto dal server sullo standard input dello script CGI**
- **Il numero di byte è nella variabile ambiente CONTENT_LENGHT, mentre il tipo dei dati MIME è nella variabile di ambiente CONTENT_TYPE**
- **Anche in questo caso, l'informazione trasferita dal metodo POST è URL encoded e deve essere decodificata**

Passaggio mediante standard input

- **Il client invia una richiesta del tipo:**

POST http://www.unimo.it/cgi-bin/corsi.cgi HTTP/1.0

...

(linea vuota)

attr1=valore1&attr2=valore2

- **Il server HTTP passa i parametri a corsi.cgi tramite lo standard input**

Passaggio mediante standard input

- **Nello script CGI corsi.cgi ci dovranno essere:**
  - un’istruzione del tipo: scanf(“%s”, stringa);
  - O un’istruzione del tipo: char c=getchar( );
  - le istruzioni per la decodifica della stringa, in formato URL encoded
  - le istruzioni per il parsing della stringa

Input a CGI (riepilogo)

BROWSER	***METHOD HTTP***	***SERVER → CGI***
Invocazione URL	GET (senza =)	**Linea comando**
Invocazione URL	GET (con =)	**Variabili ambiente**
Invio form	GET (con =)	**Variabili ambiente**
Invio form	POST	**Standard input**

Output da CGI

- **Quando il server HTTP restituisce un oggetto (statico o dinamico) al client, include nell'header di risposta alcune informazioni sull'oggetto.**
- **Queste informazioni sono inviate indipendentemente dal risultato dell'applicazione CGI.**
- **Lo script CGI può aggiungere tre tipi di informazione:**
  - Tipo del contenuto: formati standard MIME
  - Posizione: alternativa per localizzare l'oggetto

    Location: URL
  - Status: risposta HTTP

    Status: XXX message

Esempio

http://www.dsi.unimo.it/cgi-bin/phone

Un semplice esempio di script shell che accede al file Unix

`/apps/data/Phone_Directory`

mediante il comando `grep`. Per trovare il numero telefonico della persona specificata nella prima parte (*fill-in form*).

Si noti l'uso di tag HTML per la creazione e la visualizzazione del testo con layout.

```
#!/bin/sh
echo Content-type: text/html
echo
if [ $# = 0 ]
then
   echo "<HEAD>"
   echo "<!-- Written by B Kelly --!>"
   echo "<TITLE>Search Phone
     Directory</TITLE>"
   echo "<ISINDEX>"
   echo "</HEAD>"
   echo "<BODY>"
   echo "<H1>University Phone Directory</H1>"
   echo "Enter name of the person in the
     box.<P>"
   echo "</BODY>"
else
   echo "<HEAD>"
   echo "<TITLE>Results of Search</TITLE>"
   echo "</HEAD>"
   echo "<BODY>"
   echo "<H1>Results of Search for $* </H1>"
   echo "<PRE><TT>"
   grep -i "$*" /apps/data/Phone_Directory
   echo "</PRE></TT>"
   echo "</BODY>"
fi
```

1. Esecuzione script (azione client)

- Il browser, dopo aver localizzato l'indirizzo IP del computer corrispondente all'hostname **www.dii.unimo.it**, attiva una connessione TCP sulla porta 80 del server.

Una volta stabilita la connessione TCP, il client invia la richiesta per conoscere il numero di telefono di un docente del DII.

Esempio:

```
GET  /cgi_bin/phone  HTTP/1.0
Connection: close
User-agent: Mozilla/4.0
Accept: text/plain
Accept: text/html
Accept: image/*
```

2. Esecuzione script (informazioni)

Il server riceve la stringa dal client e decodifica la richiesta secondo le regole del protocollo HTTP/1.0 per determinare le azioni da intraprendere

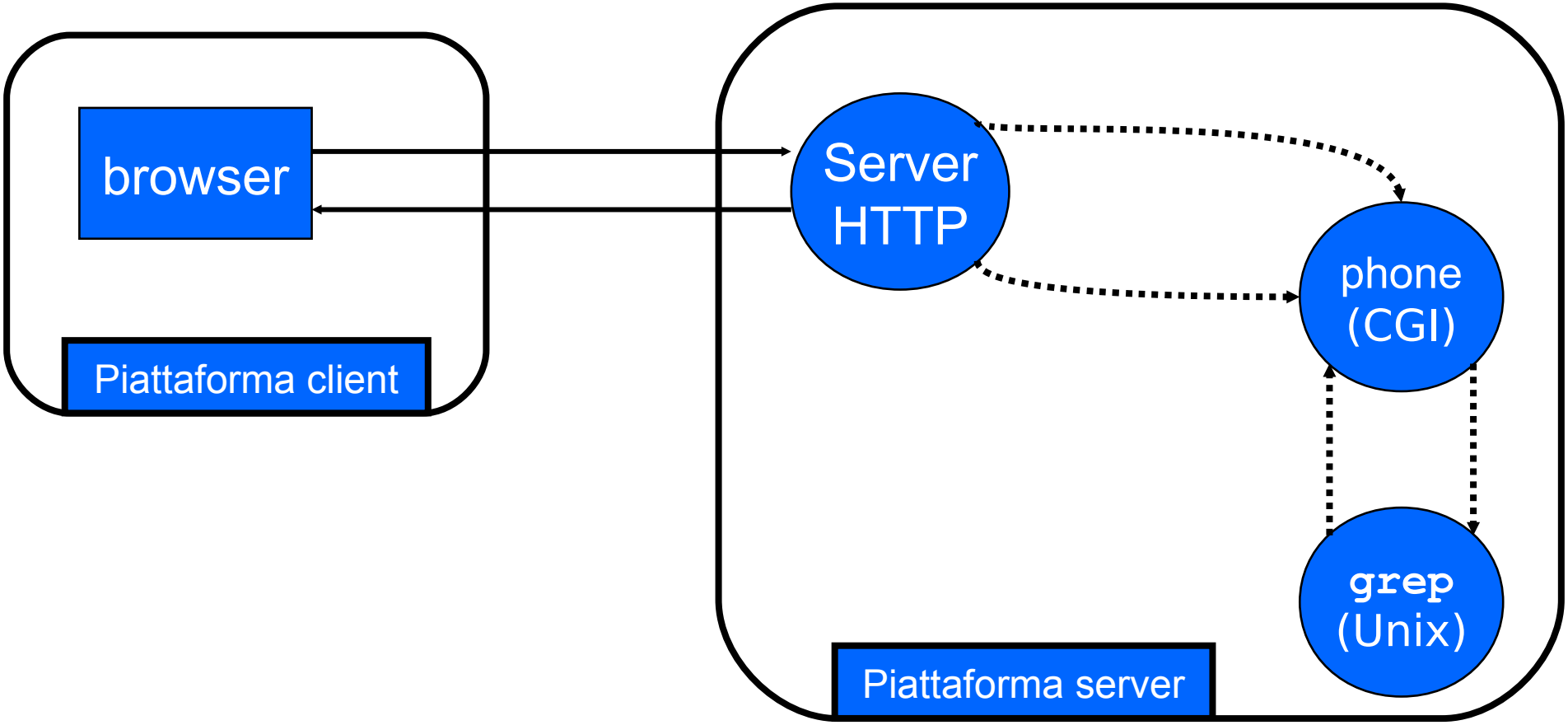
La richiesta contiene quattro importanti **informazioni**:

- il metodo (GET) che specifica le azioni, ovvero localizzare il file, leggerlo da disco, **eseguirlo**, e trasmettere il risultato al client
- la connessione deve essere chiusa dal server, dopo aver inviato l'ultimo pacchetto
- il pathname dello script CGI (**/cgi_bin/phone**) da eseguire
- la versione del protocollo utilizzato dal browser

La richiesta contiene anche delle **meta-informazioni** sul browser:

- browser utilizzato (es., versione 4.2 di Netscape)
- configurazione (es., configurato per visualizzare testo ASCII, file HTML ed immagini codificate in qualsiasi modo)

3. Esecuzione script (esecuzione)



4. Esecuzione script (risposta OK)

- Assumendo che non si sia verificato alcun errore, il server invia il codice del risultato del metodo eseguito.
- A differenza del caso in cui la richiesta era per una pagina statica, in questo caso il server Web non valuta il “Content-type” né il “Content-length”, in quanto è lo script stesso che determina il tipo di informazione restituita:

```
HTTP/1.0  200 OK
Server: Apache/1.3.0 (Unix)
Date: Thu, 29 Apr 2000  11:21:00 GMT
```

- Lo script si occupa di determinare la seconda parte dell’header, seguita dalla pagina di risposta costituita dall’output del processo “grep”:

```
Content-type: text/html
< … dati del processo grep ... >
```

5. Esecuzione script (risposta NO)

- Nell'ipotesi in cui il file non sia trovato o non sia eseguibile, la richiesta non può essere soddisfatta ed il codice di risposta è "**404 – Not found**" e la risposta del server è del tipo:

```
HTTP/1.0  404 Not found
Server: Apache/1.3.0 (Unix)
Date: Fri, 14 Jan 2000  12:30:00 GMT
Content-type: text/html
Content-length: 0
```

- Nell'ipotesi in cui lo script "phone" sia individuato, ma la sua esecuzione generi un errore (es., non viene trovato il processo Unix "grep"), la risposta è:
  - "200 – OK" perché il processo sta restituendo qualche informazione.
  - Tuttavia, l'informazione restituita sarà un messaggio di errore del tipo "Cannot find *grep* command on this system".

Esecuzione

1. **Il server capisce che l'URL richiesto è un CGI**

2. **Istanzia un processo che esegue il CGI**
   - Il processo a sua volta crea un nuovo processo che esegue il comando grep

3. **Invia l'output dello script al client**

CGI: Pro e Contro

- **Vantaggi**
  - Semplice da programmare
  - Indipendente dal linguaggio e dalla piattaforma
  - Primo standard *de facto* per il Web dinamico

- **Svantaggi**
  - Poco efficiente nelle prestazioni: è necessario eseguire una FORK-EXEC per ogni richiesta
  - E' difficile verificare e garantire la sicurezza

    (es., l'applicazione CGI viene lanciata dal processo server, con i privilegi di quest'ultimo)

Modulo 7
Cenni di linguaggio PHP

Overview di PHP

- **PHP = PHP Hypertext processor**
- **Tecnologia per la generazione di pagine dinamiche**

  Tecnologia server side (cioè la generazione dei documenti dinamici avviene direttamente sul server)

  Le pagine PHP si presentano come un mix di statement HTML standard e di istruzioni PHP
- **Linguaggio molto diffuso**

  Secondo Netcraft i siti che usano PHP sono più di 14M
- **Ottima documentazione**

  Manuale estremamente ricco e chiaro, con reference per la consultazione puntuale e tutorial per cominciare

Overview di PHP

- **Versioni PHP**

  3 e 4 versioni ormai obsolete e non più molto diffuse

  5 la nuova versione che sta guadagnando popolarità (è la versione che usiamo nel nostro corso)

- **Aspetto di un documento PHP**

```
<html>
<head><title>prova</title></head>
<body>
   <?php
      echo “hello world”;
   ?>
</body>
</html>
```

Codice HTML

Codice PHP

Codice PHP

- **Il codice PHP è embedded nelle pagine HTML**
- **Viene racchiuso tra i tag <?php e ?>**
- **Il codice PHP ha una sintassi che ricorda il C, C++**
- **Alcuni elementi però lo fanno classificare come linguaggio di scripting**
  - PHP è interpretato
  - PHP non è un linguaggio strettamente tipato
  - PHP non richiede che le variabili siano dichiarate
- **Caratteristica particolare di PHP è la ricchezza delle funzioni per interfacciarsi a Database**
  - Punto di forza importante per lo sviluppo di siti Web dinamici

Sintassi PHP

- **Commenti**

  // stile C++

  /* stile C */

  # stile shell
- **Costrutti di controllo di flusso:**

  if/else/elseif (salto condizionale)

  for, while, do/while (iteratori classici)

  foreach (iteratore sugli elementi di un insieme)

  switch (selezione multipla)
- **Blocchi di istruzioni delimitati da parentesi graffe {,}**

Tipi di dato

- **Tipi primitivi semplici:**
  - interi
  - float
  - boolean
  - stringhe
- **Tipi complessi**
  - array
  - oggetti
- **Tipi *molto complessi***
  - callback

Gestione dei tipi di dato

- **Linguaggio non tipato**
  - non specifico il tipo delle variabili
  - casting automatico in caso di overflow nelle operazioni
- **Per i tipi di dati semplice è possibile il casting esplicito**
  - $j=5.4; $i=(int) $j

Variabili

- **Non strettamente tipate**
- **Non c'è bisogno di dichiararle**
- **Il nome di una variabile non deve iniziare con un numero**
- **Notazioni usate per rappresentare delle variabili:**

  $variabile1

  ${variabile1}

  {$variabile1}

Stringhe

- **3 sintassi di dichiarazione:**

```
$s='stringa'
$s=”stringa”
$s=<<<EOD
stringa
molto
lunga
EOD;
```

Sintassi Heredoc

Delimitatore

- **Attenzione: le stringhe Heredoc richiedono che il delimitatore sia da solo sulla riga finale della stringa (senza neanche degli spazi)**

Note su stringhe e variabili

- **Stringhe tra apici semplici non hanno espansione di variabili**

  $i=1; echo '$i'; → stampa '$i'

- **Stringhe tra apici doppi o in forma heredoc prevedono espansione di variabili**

  $i=1; echo “$i”; → stampa '1'

- **Escape di apici:**

  'aaa”bbb' //OK

  “aaa'bbb” //OK

  “aaa\”bbb” //OK

Operazione di assegnamento

- **Il default è l'assegnamento per valore**

```
$b=1
$a=$b;
$b++;
if ($a==$b)
    {echo “a==b”;}
else
    {echo “a!=b”;}
```

Stampa “a!=b”

- **Esiste tuttavia anche l'assegnamento per indirizzo**

```
$b=1
$a=&$b;
$b++;
if ($a==$b)
    {echo “a==b”;}
else
    {echo “a!=b”;}
```

Stampa “a==b”

Esempio di programma PHP

Programma banale con struttura di controllo if

```
<html><head><title>if</title></head>
<body>
<?php
$a=5;
if ($a>5) {
  echo “a &egrave; maggiore di 5”;
}
else {
  echo “a &egrave; minore o uguale a 5”;
}
?>
</body></html>
```

Esempio di programma PHP

Programma banale con struttura di controllo if

```
<html><head><title>if</title></head>
<body>
<?php
$a=5;
if ($a>5) { ?>
    a &egrave; maggiore di 5
<?php
} else { ?>
    a &egrave; minore o uguale a 5
<?php
} ?>
</body></html>
```

Modulo 8
Features avanzate del linguaggio PHP

Array in PHP

- **In PHP un array è una struttura dati associativa**
- **Comunemente queste strutture dati sono chiamate anche:**

  Dictionary

  Hashtables
- **L'array contiene dei valori ciascuno identificato da una chiave**

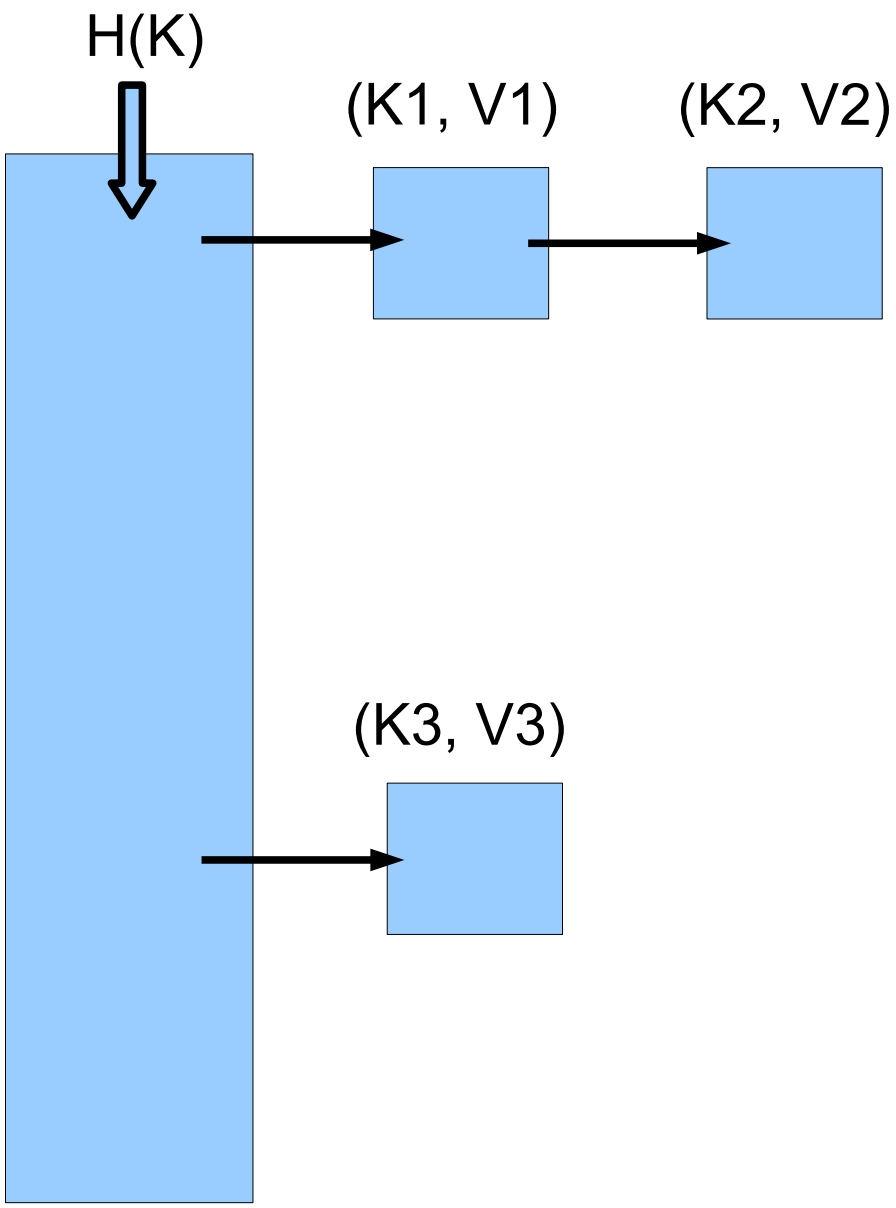
Struttura di una Hashtable

- **Hastable: array di puntatori a liste concatenate**
- **Gestisce coppia di tipo chiave, valore (K, V)**
- **Operazioni supportate:**

  add (K, V)

  V=get(K)

  Del (K)
- **Data la chiave K si applica una funzine Hash per identificare quale lista di bucket usare**
- **L'inserimento avviene nella lista selezionata**



Creazione di un array

- **Di norma un array viene creato usando il costrutto array**

  $arr=array(chiave1=>valore1, chiave2=>valore2)

- **Le chiavi possono essere dati di tipo intero o stringa**
- **Accesso al contenuto di un array:**

  Inserire/modificare un valore in un array:
  $arr[chiave]=valore

  Leggere un valore in un array:
  echo $arr[chiave];

  Cancellare un elemento
  unset($arr[chiave]);

  Stampare il contenuto di un array
  print_r($arr)

Iterare su un array

- **Uso del costrutto foreach**
- **foreach ($arr as $val)**
  - ad ogni iterazione $val contiene il valore di un elemento dell'array
- **foreach ($arr as $key=>$val)**
  - ad ogni iterazione $val contiene il valore di un elemento dell'array e $key contiene il valore corrisponsente della chiave

Cenni di programmazione strutturata

- **E' posibile definire nuove funzioni utilizzando la dirttiva “function”**
- **Sintassi della direttiva:**

  function <nome>(argomenti) {}
- **Esempio:**

```
function my_print_r ($arr){
        foreach ($arr as $k=>$v) {
                echo "$k => $v <br />";
        }
}
```

Supporto database in PHP

- **Approccio modulare**
  - Il supporto di ogni database deve essere specificato a tempo di compilazione
  - Ogni database ha proprie funzioni di supporto
- **Molti database supportati**
  - MySQL
  - PostgreSQL
  - Oracle
  - Tanti altri...

Interazione con database

- **Schema base dell'interazione con un database**
  - Creazione delle connessione
    - XX_connect(parametri di connessione)
  - Sottomissione della query
    - XX_query(connessione,''query-string'')
  - Parsing dei risultati
    - XX_fetchObject() o altro
- **Possibilita' di usare connessioni persistenti**

Esempio di interazione con DBMS

```
<?php
$conn=mysql_connect("hostname", "utente", "password");
mysql_select_db($conn, $db);
$risultato = mysql_query($conn, "select * from tabella");
while ($riga = mysql_fetch_object($risultato)) {
    echo $riga->id_utente;
    echo $riga->nome_intero;
}
mysql_free_result($risultato);
?>
```

Modulo 9
Interazione PHP/Web server

Interazione con HTTP

- **Array “superglobals” (introdotti in PHP 4)**
  - $_SERVER: variabili relative al server e alla richiesta
  - $_GET: variabili fornite dal metodo GET HTTP
  - $_POST: variabili fornite dal metodo POST HTTP
  - $_COOKIE: variabili fornite dai cookie HTTP
  - $_FILES: variabili riguardanti file trasmessi al server tramite upload
  - $_ENV: variabili d'ambiente
  - $_REQUEST: qualsiasi input fornito dall'utente (confluenza delle variabili GET, POST e COOKIE)

Confronto: Richiesta HTTP/ Array _SERVER

RICHIESTA HTTP

GET /script.php HTTP/1.1

Host: localhost:8080

User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.7.7) Gecko/20050420 Epiphany/1.6.3 (Debian)

Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5

Accept-Language: it-it,it;q=0.7,en;q=0.3

Accept-Encoding: gzip,deflate

Accept-Charset: ISO-8859-15,utf-8;q=0.7,*;q=0.7

Keep-Alive: 300

Connection: keep-alive

ARRAY SERVER

HTTP_HOST => localhost:8080

HTTP_USER_AGENT => Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.7.7) Gecko/20050420 Epiphany/1.6.3 (Debian)

HTTP_ACCEPT => text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5

HTTP_ACCEPT_LANGUAGE => it-it,it;q=0.7,en;q=0.3

HTTP_ACCEPT_ENCODING => gzip,deflate

HTTP_ACCEPT_CHARSET => ISO-8859-15,utf-8;q=0.7,*;q=0.7

HTTP_KEEP_ALIVE => 300

HTTP_CONNECTION => keep-alive

Confronto: Richiesta HTTP/ Array _REQUEST

RICHIESTA HTTP

GET
/script.php?
giorno=13&
mese=10&
anno=2009
HTTP/1.1

ARRAY REQUEST

giorno => 13

mese => 10

anno => 2009