

PARTE 11

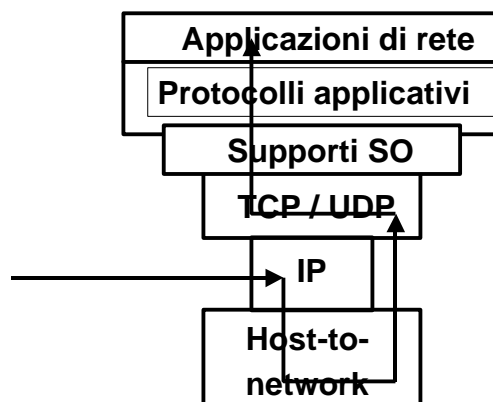
LIVELLO APPLICAZIONI

- Esempio: *World Wide Web*

Dove ci troviamo?

- Abbiamo descritto tutti i livelli TCP/IP fino al trasporto
- Non verranno trattati gli aspetti di *supporto del SO*

DA FARE:



Livello 5 (*application*)

- Il livello application utilizza il livello di trasporto dell'informazione tra processi in esecuzione su host terminali per realizzare applicazioni di rete
- Esempi protocolli applicativi
 - ftp
 - telnet
 - http
 - smtp
 - irc
 - ...

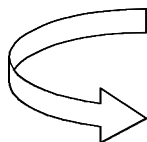
NOTA: Applicazioni di rete ≠ protocolli applicativi

Applicazioni di rete e protocolli applicativi

- Tipicamente, ciascuna applicazione di rete definisce un nuovo protocollo di livello applicativo ed un'interfaccia utente
- Esempi
 - *Trasferimento file*: usa protocollo applicativo **ftp**
 - *Collegamento a terminale remoto*: usa protocollo **telnet**
 - *World Wide Web*: usa protocollo **http**
 - *Posta elettronica*: usa protocollo **smtp**
 - *Chat*: usa protocollo **irc**
 - ...

Principali applicazioni Internet

- Domain Name System
- Posta elettronica (SMTP)
- Login remoto (Telnet)
- Trasferimento file (FTP)
- World Wide Web



Tutte usano il modello *client/server*

I chiari segnali del successo del Web

[Load misurato in hit]

Yahoo, Netscape, Microsoft, Pointcast, AltaVista, CNN, ... (>50 Milioni hits/day)

Evento	Periodo	Peak hits/day	Peak hits/minute
NCSA server (Oct. 1995)		2 Milioni	
Olympic Games 1996 (Atlanta, 1996)	180 Milioni	8 Milioni	
NASA Pathfinder (July 1997)	942 Milioni (14 days)	40 Milioni	
Olympic Winter Games (Japan, 1998)	634.7 Milioni (16 days)	55 Milioni	110.000
FIFA World Cup (France, 1998)	1.350 Milioni (90 days)	73 Milioni	209.000
Wimbledon (July, 1999)	942 Milioni (14 days)	125 Milioni	430.000
Wimbledon (July, 2000)		282 Milioni	964.000
Olympic Games 2000		875 Milioni	1.200.000

I veri motivi alla base del successo del Web

("Perché anche i semi migliori devono essere piantati nel periodo giusto ed in un terreno fertile")

- **Digitalizzazione dell'informazione**
(Qualsiasi informazione come sequenza di 0 e 1)
- **Diffusione di Internet (dagli anni '70)**
(Trasporto dell'informazione ovunque, in tempi rapidissimi e a costi bassissimi)
- **Diffusione dei PC (dagli anni '80)**
(Accesso, memorizzazione ed elaborazione dell'informazione da parte di chiunque a costi bassissimi)

Ingredienti del Web

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

Parte 6

Modulo 1: 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.

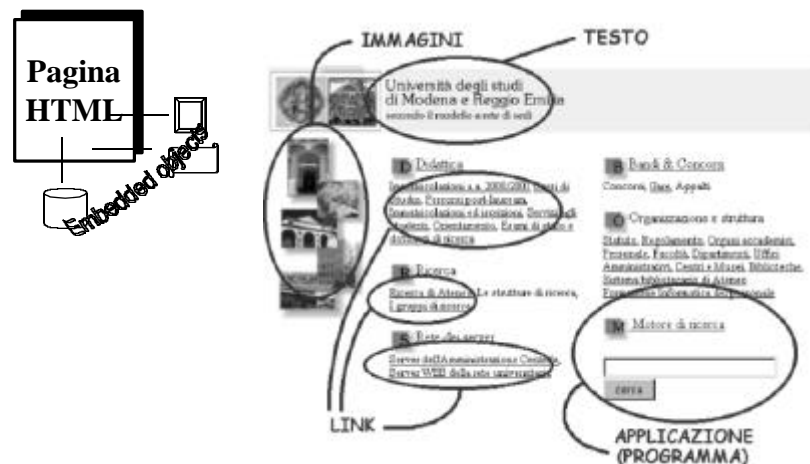
Uniform Resource Locator (URL)

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:

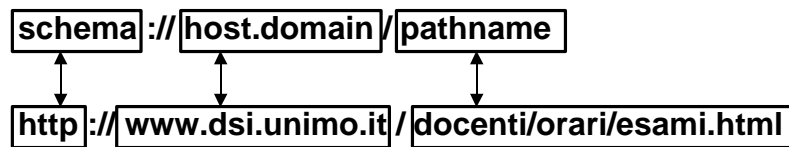
- **pagine** (testo, immagini, suoni, video, ...)
- **risultati di esecuzioni**
- **programmi eseguibili**

Tipica risorsa Web

- **Pagina HTML + *embedded objects*** (file di qualunque tipo)



Campi dell'URL

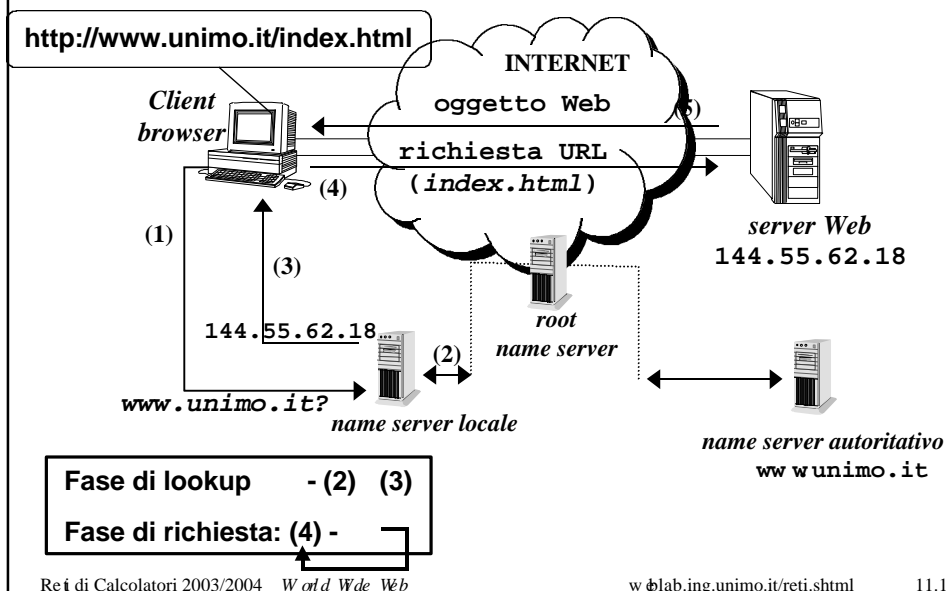


- **schema:** indica il modo con cui accedere alla risorsa, cioè quale protocollo bisogna usare per interagire con il server che controlla la risorsa. Il metodo di accesso più comune è **HTTP** (protocollo nativo del WWW per il recupero di risorse Web)
- **host.domain:** è l'hostname del nodo nel quale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

Altre definizioni

- **Sessione utente:** serie di *richieste di risorse* effettuate dallo stesso utente al medesimo sito Web
- **Richiesta di risorsa (o pagina):** una richiesta che ti ~~ti~~ ~~pc~~ ~~me~~ ~~ne~~ ~~co~~ ~~si~~ ~~se~~ di ~~multipli~~ ~~hit~~ inviati dal client dell'utente al sito Web
- **Hit:** una richiesta per un singolo oggetto effettuata client al server Web

Richiesta per una risorsa Web



Parte 11

Modulo 2: Linguaggio di markup

- **Mechanismi di comunicazione e naming di Internet**

- Protocollo TCP/IP e Sistema DNS

- **Sistema client-server**

- **Informazione digitalizzata**

- **“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

-

- 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

Connessa in rete è stato necessario definire un markup per la formattazione

- Fornisce delle linee guida generali per la rappresentazione
- Non specifica esattamente il formato e la posizione del



Due browser potrebbero visualizzare lo stesso documento in modo differente.

Linguaggio HTML

Sembrano stati proposti modifiche ed altri
*, a tutt'oggi
linguaggio del Web".

La pagina HTML è un file di solo testo ASCII.

Il testo è *formattato*.

Contenuto del testo e specifiche di formato sono
inseriti nello stesso file.

*** XML è lo standard più importante**

La descrizione dei contenuti dell'ipertesto viene effettuata
inserendo all'interno del testo stesso alcune istruzioni dette
marcatori o *markup tags* che producono le visualizzazioni

- Le istruzioni HTML sono racchiuse tra parentesi angolari,
nella forma **<tag>**, e vengono terminate da un tag di
</tag>. Es

```
<CENTER> <B> <FONT="Arial">  
<FONT COLOR="#336633"> <FONT SIZE=+1>  
Informazioni generali  
</FONT> </FONT> </FONT> </B> </CENTER>
```

Schema di un documento HTML

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

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>
```

Tag àncora

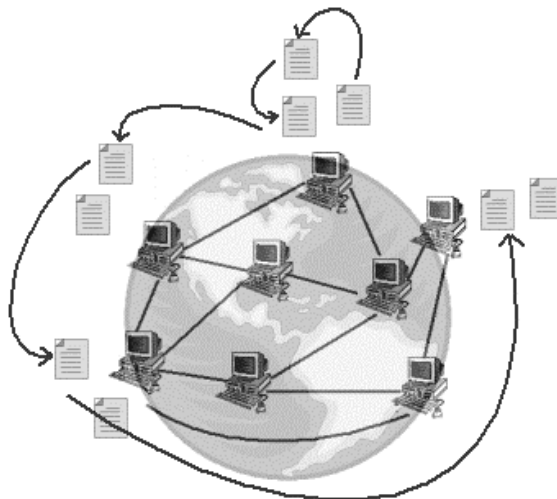
- L'istruzione più innovativa dell'HTML è l'**àncora** delimitata dai tag **<A>...**, in quanto tale elemento permette di trasformare un normale testo in ipertesto multimediale.
- Un àncora può far riferimento ad una sezione della stessa pagina oppure **ad una qualsiasi risorsa** (testuale, multimediale, 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



Modulo 3: Protocollo HTTP

Ingredienti del Web

- **Meccanismi di comunicazione e naming di Internet**
 - Protocollo TCP/IP e Sistema DNS
- **Sistema client-server**
- **Informazione digitalizzata**
- **“Solo” tre nuovi standard**
 - **URL**: Sistema di indirizzamento delle risorse
 - **HTML**: Linguaggio di markup ipertestuale
 - **HTTP**: Protocollo per le richieste di 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 (2)

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)

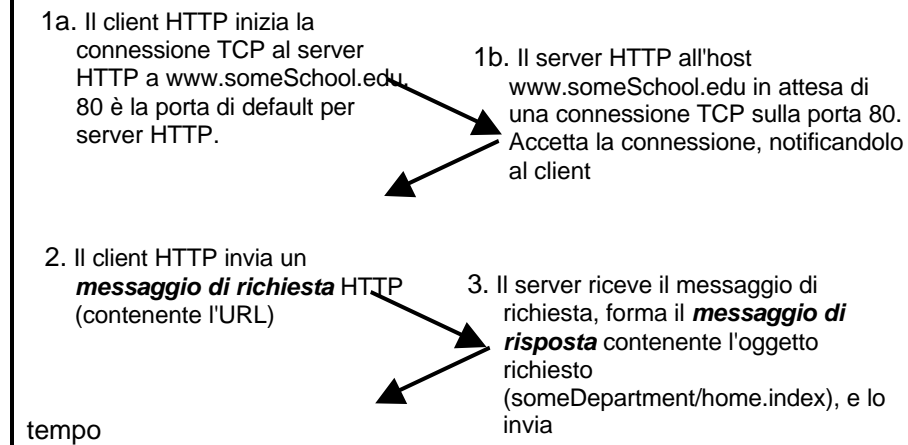
- **il server non conserva nessuna informazione riguardante le richieste dei client passati**
- **I protocolli che conservano lo stato sono complessi!**
 - . La storia passata (lo stato) deve essere memorizzata
 - . Se il server/client subiscono un crash, la vista dello stato può essere inconsistente e deve essere ristabilita

Esempio HTTP

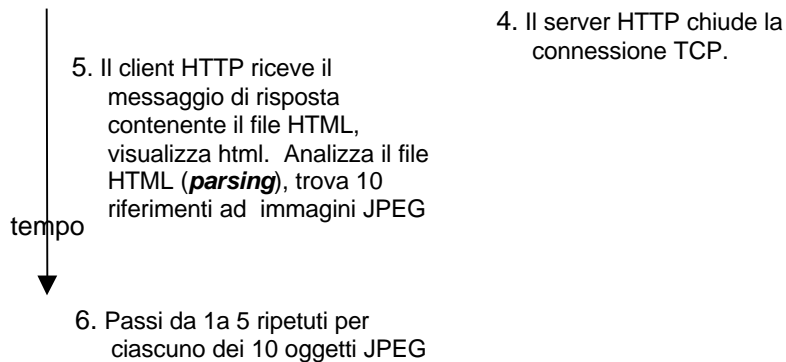
L'utente digita l'URL

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

(il documento contiene testo e i riferimenti a 10 immagini JPEG)



Esempio HTTP (cont.)



Connessioni TCP nell'HTTP

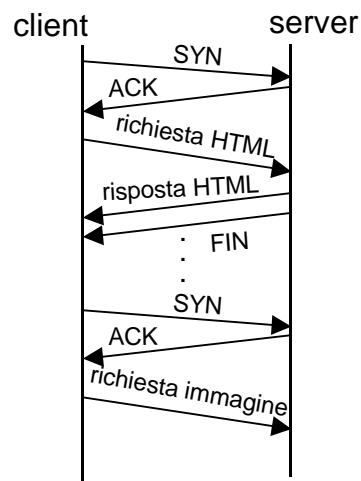
- **Connessione non-persistente:** ogni oggetto viene trasferito usando una nuova connessione TCP

- **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



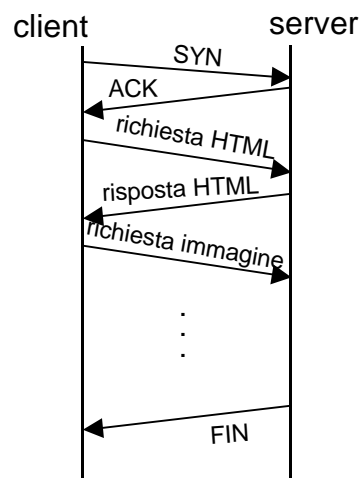
Connessioni TCP nell'HTTP (2)

- **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



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 aggiuntive, 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)

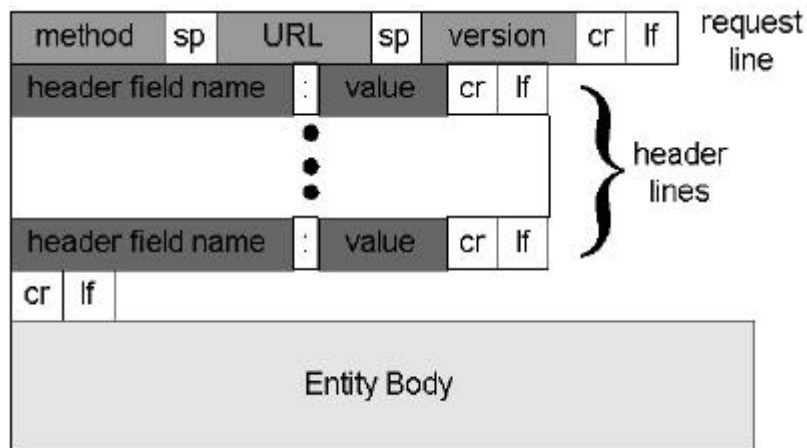
Linee
header

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)

Formato messaggio di richiesta HTTP



Messaggio di 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 delle richieste

<i>Metodo</i>	<i>Richiesta</i>	<i>Versione protocollo</i>
GET	Ricevere una risorsa dal server	Dalla HTTP/0.9
HEAD	Ricevere il solo header di una risorsa	Dalla HTTP/1.0
POST	Appendere un oggetto ad un altro sul server	Dalla HTTP/1.0
PUT	Inviare un oggetto al server	Dalla HTTP/1.1
DELETE	Cancellare un oggetto dal server	Dalla HTTP/1.1
LINK e UNLINK	Creare o eliminare collegamenti fra oggetti del server	Dalla HTTP/1.1
TRACE	Individuare la catena dei server proxy	Dalla HTTP/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

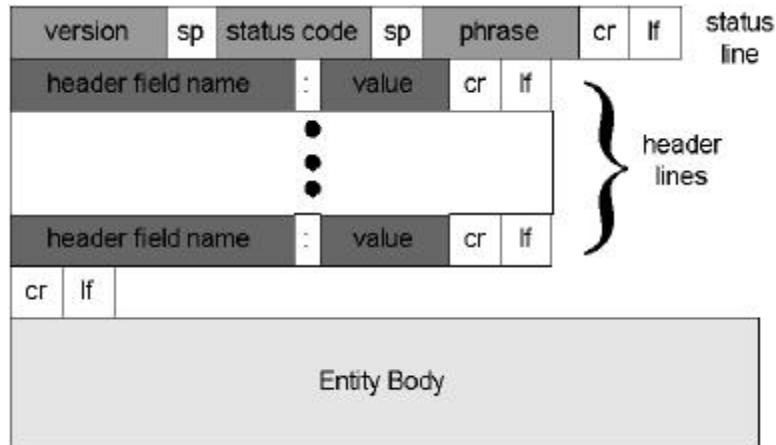
Linea di stato
(codice dello stato
del protocollo,
frase corrispondente)

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
data data data data data ...
```

Messaggio di risposta HTTP (cont.)



Messaggio di risposta HTTP (cont.)

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

Esito	Codice numerico (<i>status code</i>)	Specifica testuale (<i>reason phrase</i>)
OK, la risorsa è stata trovata e viene inviata	200	OK
Risorsa spostata	301	Moved permanently
Risorsa non modificata	304	Not Modified
Richiesta non valida	400	Bad request
Richiesta di autenticazione	401	Unauthorized
Richiesta di pagamento	402	Payment required
Accesso vietato	403	Forbidden
Risorsa non esistente	404	Not found
Errore nel server	500	Server error
Non implementato	501	Not Implemented
Gateway difettoso	502	Bad Gateway
Servizio non disponibile	503	Service Unavailable

Risposte HTTP

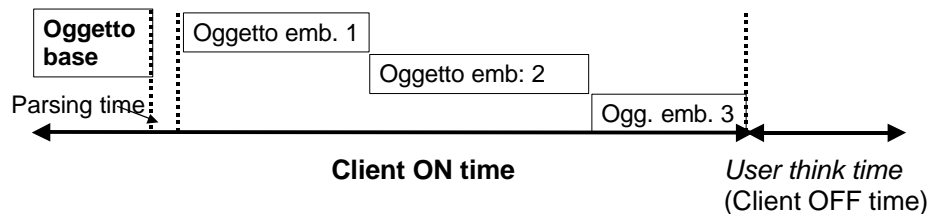
Header line

- . **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, ...)

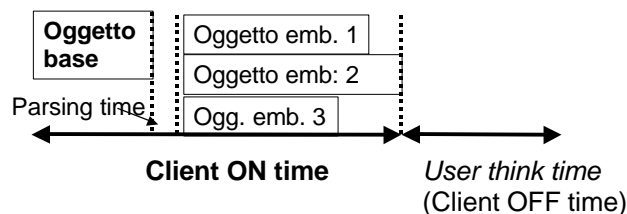
Entity body: oggetto

Protocolli HTTP/1.0 e HTTP/1.1

HTTP/1.0



Pipelining HTTP/1.1



Reti di Calcolatori 2003/2004 – World Wide Web

weblab.ing.unimo.it/reti.shtml

11.47

Interazione utente-server: autenticazione

Scopo dell'autenticazione:

controllare gli accessi ai documenti del server

HTTP stateless: il client deve presentare l'autorizzazione in ogni richiesta

autorizzazione: solitamente nome, password

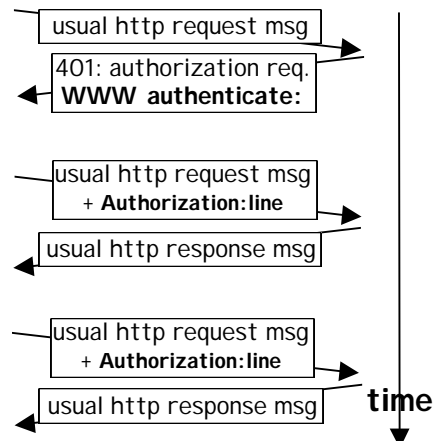
- **autorizzazione:** linea di header nella richiesta

- se non c'è autorizzazione, il server rifiuta l'accesso ed invia

WWW authenticate: come linea di header nella risposta

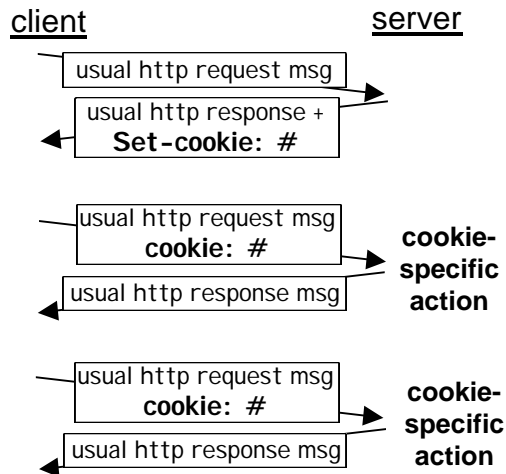
client

server



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 confronta il cookie presentato con i cookie da lui memorizzati
 - autenticazione
 - memoria delle preferenze degli utenti, delle scelte effettuate in precedenza



Esperimenti con HTTP

Provare ad usare un client HTTP (senza browser!):

1. Telnet ad un Web server sulla porta 80:

telnet www.eurecom.fr 80 Apre connessione TCP sulla porta 80 (porta HTTP server di default) di www.eurecom.fr. Qualsiasi cosa digitata viene inviata alla porta 80 di www.eurecom.fr

2. Digitare una richiesta HTTP GET:

GET /~ross/index.html HTTP/1.0 Digitando questa riga (seguito da un duplice invio), viene inviata una richiesta GET minima (ma completa) al server HTTP

3. Messaggio di risposta inviata dal server!

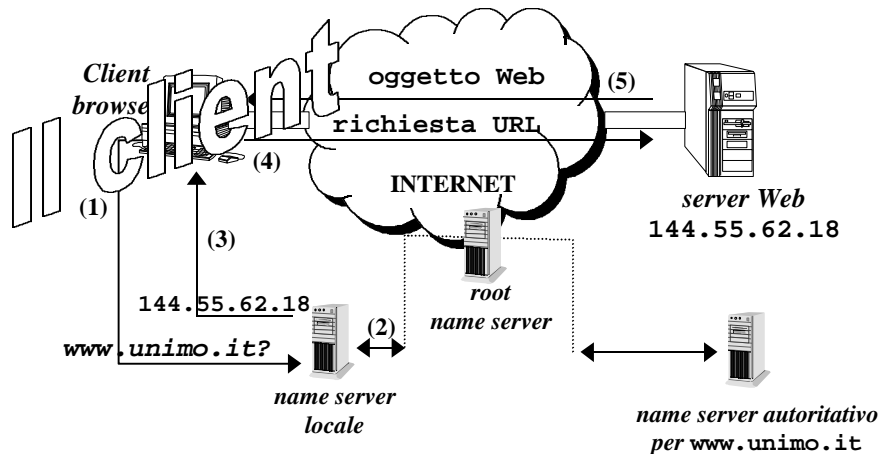
Parte 11

Modulo 4: Client HTTP

World Wide Web

(Lato Client)

Macro-componenti del Web



Browser

- Il browser è un'applicazione software che svolge il ruolo di interfaccia fra l'utente ed il WWW, mascherando la complessità di Internet.
- Diventa un client Web per recuperare informazioni dai server Web.
- Funzioni principali
 - instaura una connessione TCP con il server tramite cui invia opportuni messaggi al server Web per ottenere le risorse richieste
 - interpreta il codice ipertestuale HTML
 - elabora il codice allo scopo di visualizzare in modo appropriato il contenuto delle pagine sullo schermo

Breve storia

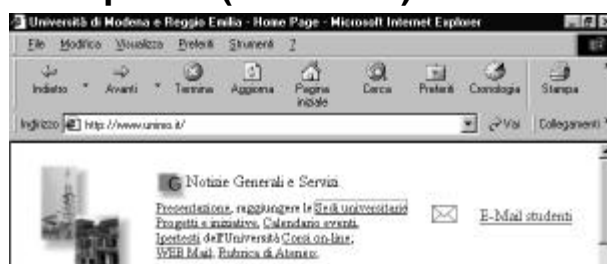
- **FrameMaker e Acrobat PDF**
 - consentono generazione e incorporazione di iperlink.
- **Gopher** (University of Minnesota)
 - primo browser con iperlink verso siti remoti
 - uso di standard, quali testo ASCII e socket Unix
 - difetti: link separati dal testo, immagini non gestite
- **HTML** **Mosaic**
- **Mosaic** (NCSA, 1993)
 - **Netscape**
 - **Microsoft Explorer** (tramite Spyglass Co.)

Browser più diffusi

- **Netscape Navigator (AoL)**















- **Internet Explorer (Microsoft)**



Finestra del browser

- **Barra del titolo della pagina scaricata**
- **Barra dei menu delle funzioni del browser suddivise in classi**
 - File, Edit, View, Go, Help
- **Barra degli strumenti veloci**
 - per funzioni frequenti: back, reload, home, ...
- **Finestra indirizzo URL**
- **Area di visualizzazione pagina Web**
- **Barra di stato**
 - per messaggi all'utente: URL dell'iperlink, informazioni su operazioni in corso, ...

Pulsanti azione dei browser

Netscape	Explorer	Azione
 Indietro	 Indietro	Ritorna all'ultima pagina Web visitata precedentemente
 Arresta	 Termina	Interrompe il trasferimento della pagina Web corrente
 Ricarica	 Aggiorna	Richiede nuovamente al server Web la risorsa visualizzata, per aggiornare eventuali modifiche della pagina
 Home	 Pagina iniziale	Richiama la home page per il browser, o pagina di partenza, scelta dall'utente
 Ricerca	 Cerca	Attiva un motore di ricerca per trovare risorse che contengono specifiche parole
 Segnalibri	 Preferiti	Consente di memorizzare una serie di indirizzi di pagine Web scelte dall'utente

Funzioni di un browser

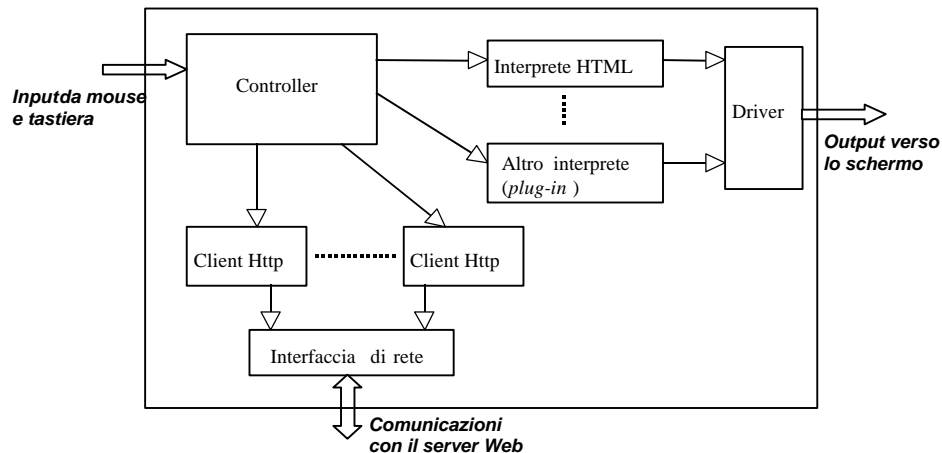
- Richiedere la risorsa al server Web
- Ricevere il file con la codifica della risorsa in linguaggio HTML
- Decodificare ed interpretare secondo le specifiche di HTML le caratteristiche grafiche, di formato e di comportamento dei vari oggetti contenuti nella risorsa (titoli, testi, immagini, pulsanti ...)
- Visualizzare la risorsa sullo schermo del computer dell'utente

Richiesta di una risorsa

- Una qualsiasi risorsa Web può essere richiesta o attraverso un link presente su un'altra risorsa, oppure digitando direttamente il suo indirizzo nella barra dell'indirizzo del browser.



Componenti di un browser



Azioni di un browser (1)

Fase iniziale

- Il browser analizza l'indirizzo inserito esplicitamente nella finestra indirizzo o l'iperlink selezionato nella pagina allo scopo di individuare la risorsa specificata nell'URL.
- Controlla se la risorsa richiesta è contenuta nella cache disco del browser. [mediante ricerca hash]
 - Se la copia è valida, il browser la visualizza.
 - Se l'utente ha esplicitamente effettuato un RELOAD, un buon browser cercherà di risparmiare tempo effettuando solo una richiesta HTTP con metodo GET al server con linea "If-modified-since" per controllare se la copia nella cache è ancora valida o meno.
- Si passa alla fase successiva nel caso in cui l'oggetto vada acquisito.

Azioni di un browser (2)

Fase di lookup

- Il browser acquisisce dall'utente l'URL da richiedere.
- Il browser invoca il *resolver* per conoscere, tramite il sistema di naming del DNS, l'indirizzo IP dell'URL cercato.
- Se esistente, il DNS restituisce l'indirizzo IP.

*Il lookup è quasi sempre implementato con una **chiamata di sistema bloccante**, per cui il browser non può effettuare altre operazioni fino a che l'operazione di lookup si conclude con successo o insuccesso.*

Azioni di un browser (3)

Fase di richiesta

- A questo punto, il browser attiva una connessione TCP con l'host che ha l'indirizzo IP individuato.
- Sfruttando questa connessione, il browser richiede mediante il protocollo HTTP la risorsa specificata nell'URL.
- Il server Web invia il file richiesto.
- *Operazione di parsing:*
 - il browser analizza se vi sono oggetti allegati alla pagina (*embedded URL*)
 - in caso affermativo, il browser effettua una richiesta per ciascuna risorsa collegata

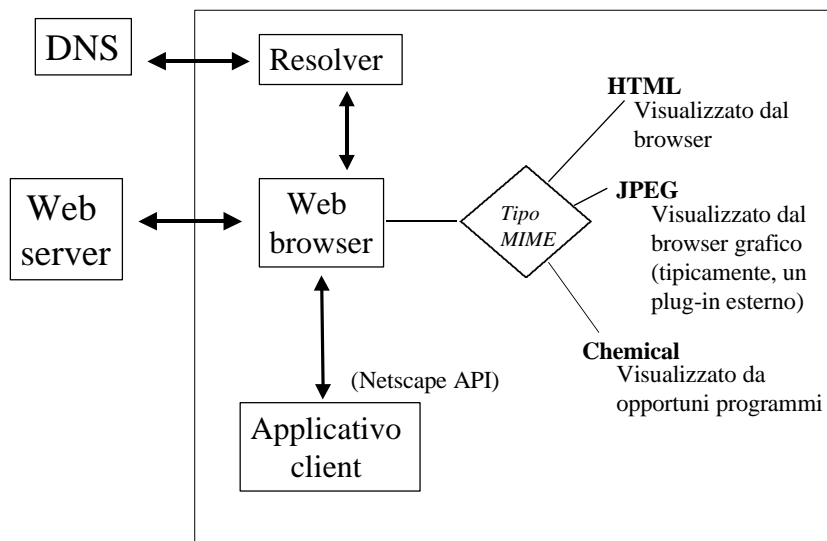
(Ricordare la differenza tra protocollo HTTP/1.0 e HTTP/1.1)

Azioni di un browser (4)

Fase di visualizzazione

- Una volta inviati tutti gli oggetti (nel caso di protocollo HTTP/1.1) o dopo aver inviato ciascun oggetto (protocollo HTTP/1.0), il server chiude la connessione TCP.
- Non appena riceve il primo file, il browser analizza come visualizzare sul monitor il testo contenuto nella pagina.
- Il browser carica e mostra gli eventuali oggetti allegati alla pagina. Nel caso in cui l'oggetto ricevuto è in qualche formato non direttamente interpretabile dal browser, questi può attivare un apposito **programma plug-in** che ne consente la visualizzazione.

Software in esecuzione sulla macchina client



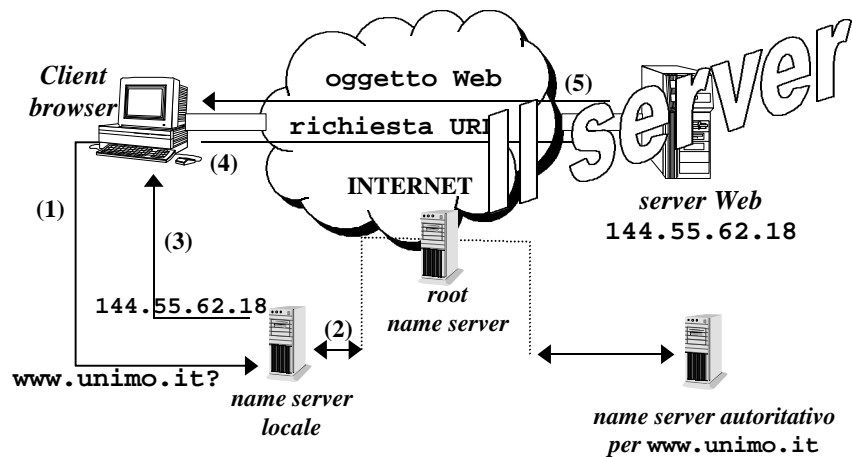
Caching nel browser

- Il browser gestisce uno spazio disco (di dimensione predefinita) in cui memorizza i file dei server Web recuperati corrispondenti a:
 - pagine HTML
 - immagini
- In realtà, prima di effettuare una richiesta al server Web (vedi “azioni di un browser – fase 3”), il browser controlla se nella cache vi è una copia dell’oggetto richiesto.
- L’utente può forzare il prelievo dell’oggetto dal server Web (bypassando la cache del browser) mediante un opportuno pulsante del browser:
 - **REFRESH/AGGIORNA** (*Microsoft Explorer*)
 - **RELOAD/RICARICA** (*Netscape Navigator*)
- Gli oggetti nella cache possono avere un **timestamp** che indica il periodo di validità dell’oggetto (stabilito dal server). Il client eventualmente, effettua una richiesta **if-modified-since** al server Web.

Parte 11

Modulo 5: Server HTTP

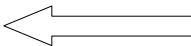
Macro-componenti del Web



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

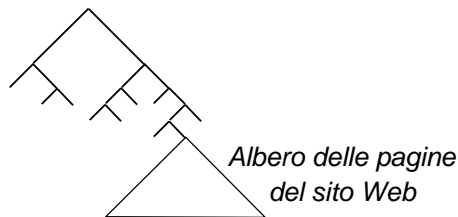
- Generalmente organizzata in risorse iper-mediali con link verso altre risorse (interne ed esterne al sito Web).
- L'organizzazione delle pagine è tipicamente gerarchica ad albero, in cui vi è un punto di partenza e tutte le altre pagine sono poste al di sotto della radice dell'albero.
- **L'albero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

File system del server



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*.

Organizzazione delle pagine (3)

- **Mirroring**
 - l'intero albero è replicato su più dischi o piattaforme
- **Soluzioni intermedie**
 - le parti superiori dell'albero, ovvero le pagine più frequentemente richieste, sono replicate
 - le altre pagine possono essere o meno partizionate

Ciascuna di queste organizzazioni deve essere trasparente per l'utente, che deve poter navigare e richiedere le pagine nello stesso identico modo, indipendentemente dall'organizzazione fisica dei file e dei servizi Web.

Tipi di pagine - 1 (classificazione 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 pagine - 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.

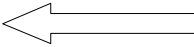
Risorse dinamiche

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.

Interazione protocollo di trasporto con applicazioni

- Il client ed il server utilizzano un protocollo di trasporto (**TCP**) per comunicare
- Il software di gestione del protocollo di trasporto si trova all'interno del Sistema Operativo
- Il software dell'applicativo (ovvero il processo HTTP) si trova all'esterno del Sistema Operativo
- **Come possono interagire?**

Si utilizza un meccanismo che svolge il ruolo di ponte tra S.O. e applicativo di rete: ***Application Program Interface (API)***

Application Program Interface

- E' parte del Sistema Operativo
- Consente ai processi applicativi di utilizzare i protocolli di rete, definendo:
 - Le operazioni consentite
 - Gli argomenti per ciascuna operazione
- **Socket API**
 - Definiti inizialmente per **BSD Unix** per utilizzare i protocolli TCP/IP
 - Ora divenuti uno standard industriale, disponibile su vari Sistemi Operativi

Socket

- I socket sono delle API che consentono ai programmatori di gestire le comunicazioni tra processi
- A differenza degli altri costrutti di comunicazione (*pipe*, *code di messaggi* e *memoria condivisa*) **i socket consentono la comunicazione tra processi che possono risiedere su macchine diverse**, e dunque costituiscono lo strumento di base per realizzare servizi di rete
- In pratica, consentono ad un programmatore di effettuare trasmissioni TCP e UDP senza curarsi dei dettagli “di più basso livello” che sono uguali per ogni comunicazione (*three-way handshaking*, *finestre*, *buffer*,...)

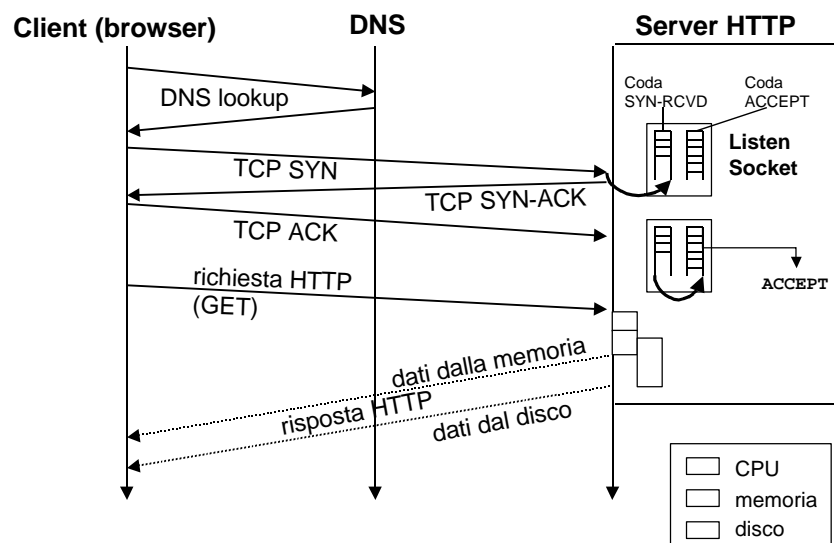
Fasi di una comunicazione TCP

- 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

Socket e network programming

- Per saperne un po' di più ...
D.E. Comer, "Computer Networks and Internet"
(*third edition*), Prentice Hall, 2001
- Per sapere tutto su ...
W.R. Stevens, "Unix Network Programming",
(*vol. 1, second edition*), Prentice Hall, 1998
NUOVA EDIZIONE SU LINUX: Comer, Stevens
(*third edition*)

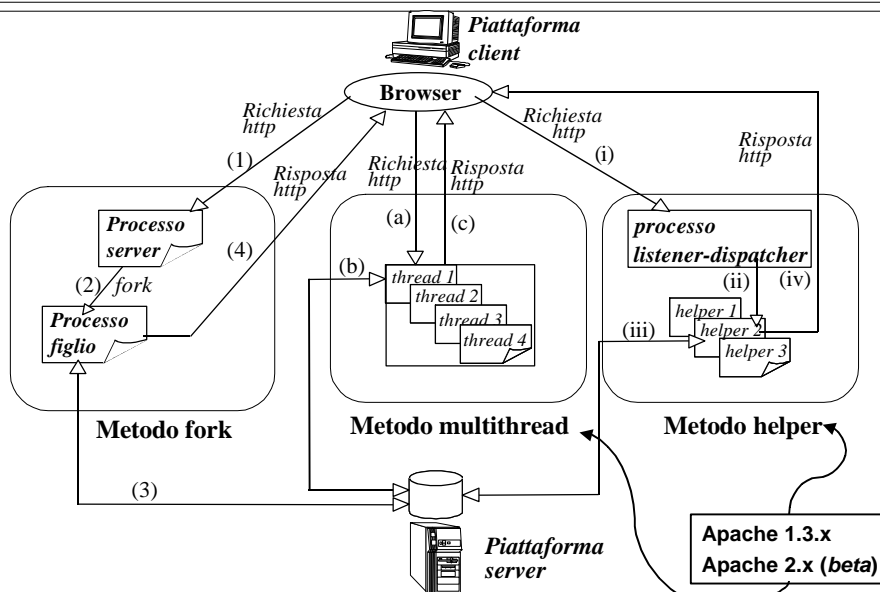
Server HTTP (es., richiesta oggetto statico)



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

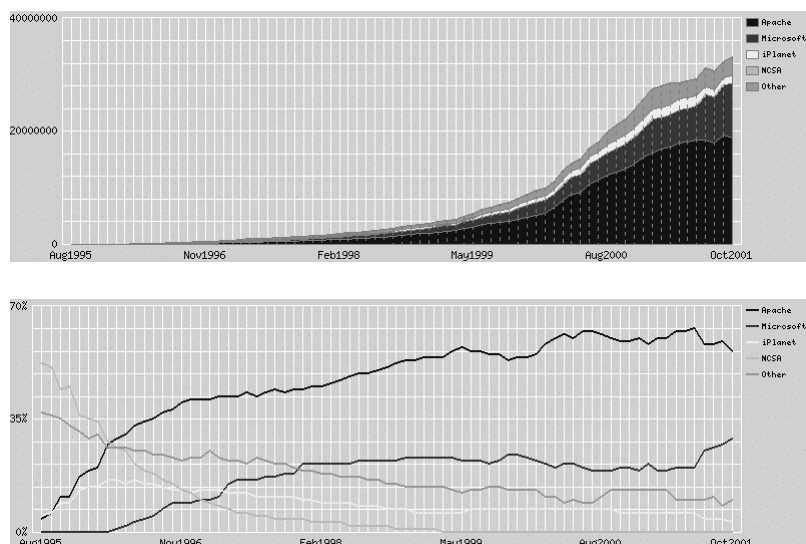
Tre modalità di gestione *richieste HTTP*



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

Software per server Web



Operazioni server Web

Pagine statiche



pagine il cui contenuto è relativamente stabile nel tempo.

Pagine volatili

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

Pagine dinamiche

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

Pagine attive

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

Sintesi azioni server Web

Il compito del server che fornisce pagine Web statiche è molto semplice:

- attendere da parte dei browser richieste di connessione TCP attraverso cui ricevere una richiesta HTTP
- decifrare il campo pathname dell'URL della richiesta per determinare il file cercato
- controllare se il file è disponibile
- in caso affermativo, spedire il file al browser attraverso la connessione TCP aperta
- chiudere la connessione subito dopo aver inviato l'ultimo pacchetto concernente il file (nel caso di protocollo HTTP/1.0)

1. Richiesta lato server (*processo server*)

- Il *processo server*, una volta attivato, si pone in attesa di una richiesta proveniente da un qualsiasi client Web di Internet. In particolare, il server rimane in ascolto sulla porta 80 finché qualche processo client attiva una connessione. In questa fase, il processo server si dice **dormiente in attesa**.
- Il processo diventa **attivo** nel momento in cui un utente mediante un browser richiede una risorsa sull'host su cui è eseguito il server.
- Esempio: Si supponga che venga inviata la richiesta per la risorsa
`http://www.unimo.it/Docenti/index.html`

specificata nella pagina HTML come

```
<A HREF="http://www.unimo.it/Docenti/index.html" >  
Elenco </A>
```

2. Richiesta lato server (*azione client*)

- Il browser, dopo aver localizzato l'indirizzo IP del computer corrispondente all'hostname **www.unimo.it**, attiva una connessione TCP sulla porta 80 del server.
Questa connessione richiede l'esecuzione di tutto il meccanismo di three-way handshaking del TCP.
- Una volta stabilita la connessione TCP, il client invia sulla rete la richiesta per il file "Docenti/index.html" seguendo le regole del protocollo HTTP.

Esempio:

```
GET /Docenti/index.html HTTP/1.0  
Connection: close  
User-agent: Mozilla/4.0  
Accept: text/plain  
Accept: text/html  
Accept: image/*
```

3. Richiesta lato server (*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 e trasmetterlo al client;
 - la connessione deve essere chiusa dal server, dopo aver inviato l'ultimo pacchetto;
 - il cammino della pagina (/Docenti/index.html) da individuare;
 - il 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).

4. Richiesta lato server (*risposta OK*)

- Assumendo che non si sia verificato alcun errore, il server deve eseguire il **metodo GET**:
 - Come prima azione, cerca di individuare il file "/Docenti/index.html" nel suo albero delle pagine Web. (*Operazione effettuata mediante una system call al file system della piattaforma Web.*)
 - Se il cammino è corretto ed è stato individuato il file corrispondente a "**index.html**" il server è pronto per la trasmissione della risposta (**non c'è bisogno di lookup perché TCP è bidirezionale**).
- Il server invia il codice del risultato del metodo eseguito e varie altre informazioni:

```
HTTP/1.0 200 OK
Server: Apache/1.3.0 (Unix)
Date: Fri, 14 Jan 2000 12:30:00 GMT
Content-type: text/html
Content-length: 10256
Last-modified: Sun, 14 Mar 1999 10:24:00 GMT
< ... dati dati dati ...>
```

5. Richiesta lato server (*risposta dati*)

- Dopo la descrizione di ciò che sta arrivando, il processo server preleva il contenuto del file dal disco (oppure dalla memoria RAM usata come *disk cache* nel caso in cui il server Web ne gestisca una) e lo scrive nella porta della connessione di rete.
- Inviare i dati sulla rete richiede la trasmissione di uno o più *segmenti* TCP (ovvero pacchetti IP con l'aggiunta di un header TCP). Ciò significa che il server deve suddividere il messaggio in parti, creando dei pacchetti IP, ciascuno contenente un header con l'indirizzo di destinazione.

6. Richiesta lato server (*risposta NO*)

- Nell'ipotesi in cui il file non sia trovato, la richiesta non può essere soddisfatta ed il codice di risposta è differente. Il problema più comune è che vi è stato un errore di digitazione nella scrittura del nome del file.
- 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
```


7. Richiesta lato server (*chiusura*)

- Se la pagina non contiene delle immagini o altri oggetti allegati, ovvero si sta utilizzando un protocollo antecedente all'HTTP/1.1, l'operazione del server è completata quando è stata completata la trasmissione di tutto il file o l'invio del messaggio di errore.
- Il processo server chiude il file (nel caso fosse stato aperto) e chiude la connessione TCP.
- Le operazioni successive sono completamente a carico del browser che deve ricevere i dati e visualizzarli sul monitor secondo il formato specificato. Il server Web non conosce nulla di questi passi successivi, in quanto si è già posto in attesa di un'altra richiesta.

8. Richiesta lato server (*embedded objects*)

Protocollo HTTP/1.0

- Ciascun oggetto allegato alla pagina principale viene richiesto secondo la stessa sequenza vista per la prima pagina, come se fosse un file del tutto indipendente.

Protocollo HTTP/1.1

- Il server non chiude la connessione TCP dopo aver trasmesso il primo file, ma rimane in attesa di ricevere le richieste per gli oggetti allegati alla pagina principale.
- La connessione TCP rimane aperta fino a quando non arrivano altre richieste o al massimo per un time-out che tipicamente è fissato a 15 secondi.