Corso di Laurea in Ingegneria Informatica

Reti di Calcolatori

Livello di applicazione

Uso dei socket, il World Wide Web

A.A. 2013/2014

Livello di applicazione

Obiettivi:

- Fornire i concetti base e gli aspetti implementativi dei protocolli delle applicazioni di rete
- Esaminare i protocolli delle applicazioni di rete più diffuse
  - HTTP
  - FTP
  - SMTP / POP3 / IMAP
  - DNS
- Programmare le applicazioni di rete
  - socket e datagrammi

Creare un'applicazione di rete

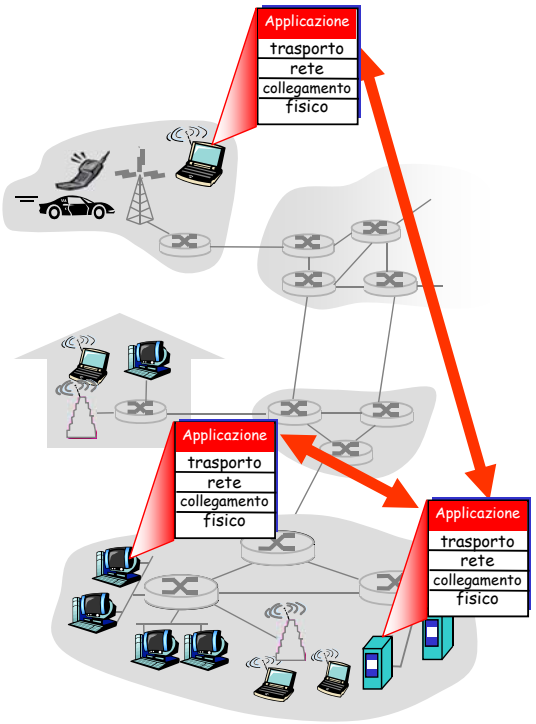
Scrivere programmi che

- girano su computer remoti
- comunicano attraverso la rete inviandosi messaggi
- *es. il software di un browser comunica con il software di un server Web*

Processo client: processo che dà inizio alla comunicazione

Processo server: processo che attende di essere contattato

Nel caso delle applicazioni P2P, ogni processo può svolgere funzione di client e di server



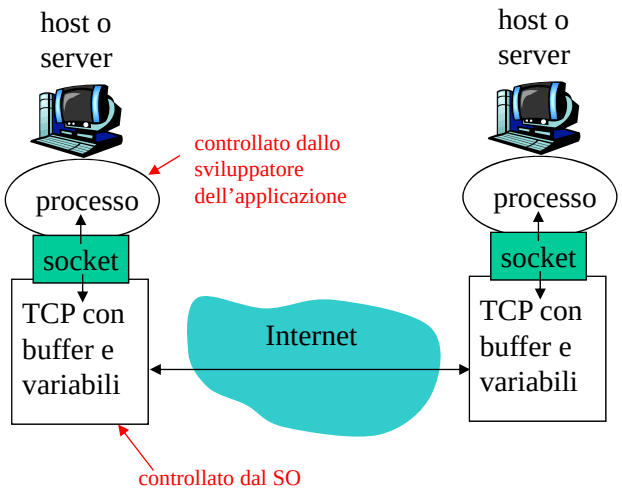
Indirizzamento

- Affinché un processo su un host invii un messaggio a un processo su un altro host, il mittente deve identificare il processo destinatario.
- Un host ha un indirizzo IP univoco a 32 bit
- Domanda: È sufficiente conoscere l'indirizzo IP dell'host su cui è in esecuzione il processo per identificare il processo stesso?
- Risposta: No, sullo stesso host possono essere in esecuzione molti processi.

- L'identificatore comprende sia l'indirizzo IP che il numero di porta associato al processo in esecuzione su un host.
- Esempi di numeri di porta:
  - HTTP server: 80
  - Mail server: 25
- Per inviare un messaggio HTTP al server gaia.cs.umass.edu:
  - Indirizzo IP: 128.119.245.12
  - Numero di porta: 80

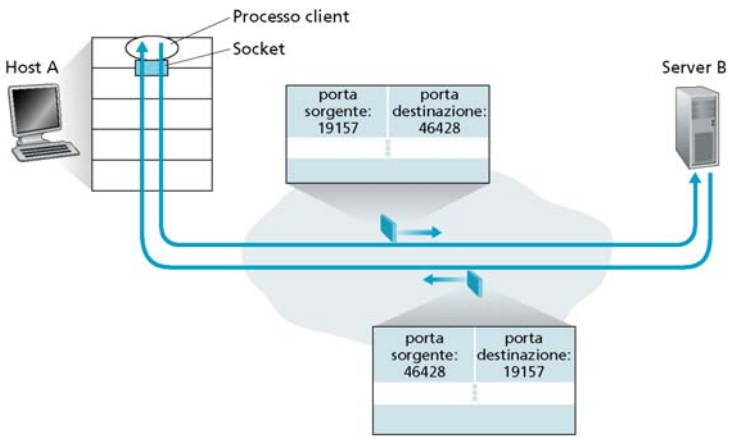
Socket

- un processo invia/riceve messaggi a/da il suo socket
- un socket è analogo a una porta
  - un processo che vuole inviare un messaggio, lo fa uscire dalla propria "porta" (socket)
  - il processo presuppone l'esistenza di un'infrastruttura esterna che trasporterà il messaggio attraverso la rete fino alla "porta" del processo di destinazione



Connessione full duplex: inversione delle porte

- Il processo server utilizza la porta del client per inviare a sua volta dei dati al client (es. risposta del server Web)



Programmazione dei socket

Obiettivo: imparare a costruire un'applicazione client/server che comunica utilizzando i socket

Socket API

- introdotta in BSD4.1 UNIX (Berkeley, USA) nel 1981
- paradigma client/server
- due tipi di servizio di trasporto:
  - A datagrammi, inaffidabile
  - Flusso di byte, affidabile,

socket

Interfaccia *creata dalle applicazioni*, e *controllata dal SO*, tramite cui il processo di un'applicazione può inviare e ricevere messaggi al/dal processo di un'altra applicazione

Primitive Socket di Berkeley Unix

Primitiva	Significato	Uso sul
SOCKET	Crea un socket e restituisce il suo descrittore	C / S
BIND	Assegna una porta locale al socket	Server
LISTEN	Il socket si mette in attesa di richieste di connessione; a tal scopo gestisce una coda	Server
ACCEPT	Se la coda è vuota, attende una richiesta di connessione; altrimenti gestisce la prima richiesta in coda	Server
CONNECT	Richiede una connessione alla controparte	Client
SEND	Invia dati alla controparte	C / S
RECEIVE	Riceve dati dalla controparte	C / S
CLOSE	Rilascia la connessione	C / S

Sequenze di primitive socket

Client
- Socket
- Connect
- (Send/ Receive)*
- Close

Server semplice
- Socket
- Bind
- Listen
- Accept
- (Send/ Receive)*
- Close

Server "multiplo"
- Socket
- Bind
- Listen
- Accept
- Fork
  - (Send / Receive)*
  - Close
  - Exit

Attenzione: i metodi java delle classi Socket e ServerSocket sono definiti ad un livello di astrazione leggermente superiore

Programmazione dei socket con TCP

Punto di vista dell'applicazione

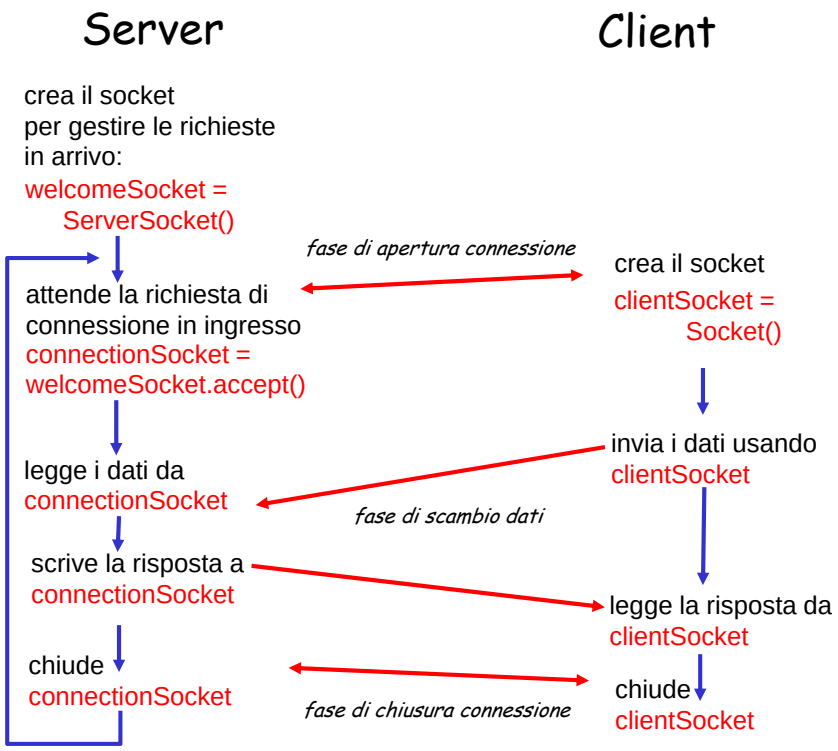
TCP fornisce un trasferimento di byte affidabile e ordinato ("pipe") tra client e server

Operazioni preliminari del server
- Il processo server deve essere in esecuzione
- Il server deve avere creato un socket (porta) che dà il benvenuto quando contattato dal client

Il client contatta il server:
- Creando un socket TCP
- Specificando l'indirizzo IP e il numero di porta del processo server

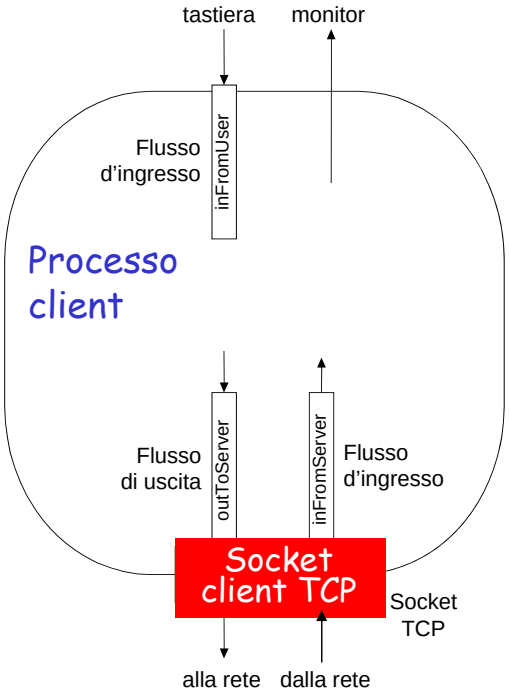
- Quando viene contattato dal client, il server TCP crea un nuovo socket per comunicare con il client
  - questo consente al server di comunicare contemporaneamente con più client

Interazione dei socket client/server: TCP



Stream

- Un flusso (*stream*) è una sequenza di caratteri che fluisce verso/da un processo.
- Un flusso d'ingresso (*input stream*) è collegato a una sorgente di dati del processo, ad esempio la tastiera o un socket.
- Un flusso di uscita (*output stream*) è collegato a un'uscita, ad esempio il monitor o un socket.



Programmazione dei socket con TCP

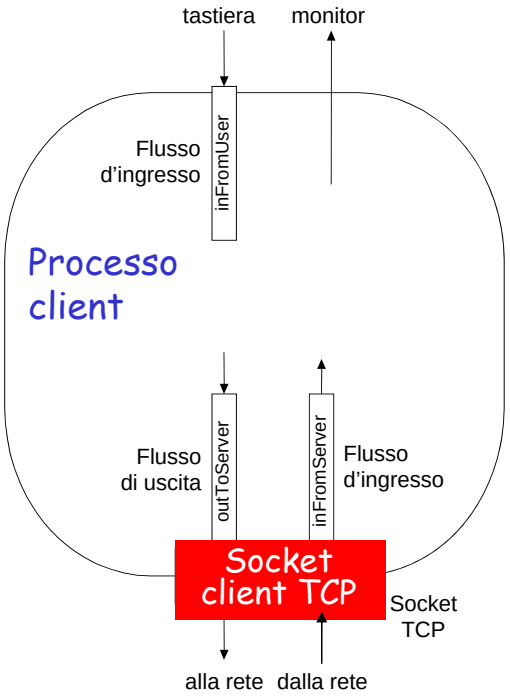
Esempio di applicazione client-server:

1) Il client legge una riga dall'input standard (flusso inFromUser) e la invia al server tramite il socket (flusso outToServer)

2) Il server legge la riga dal socket

3) Il server converte la riga in lettere maiuscole e la invia al client

4) Il client legge nel suo socket la riga modificata e la visualizza (flusso inFromServer)



Programmazione dei socket con UDP

UDP: non c'è "connessione" tra client e server

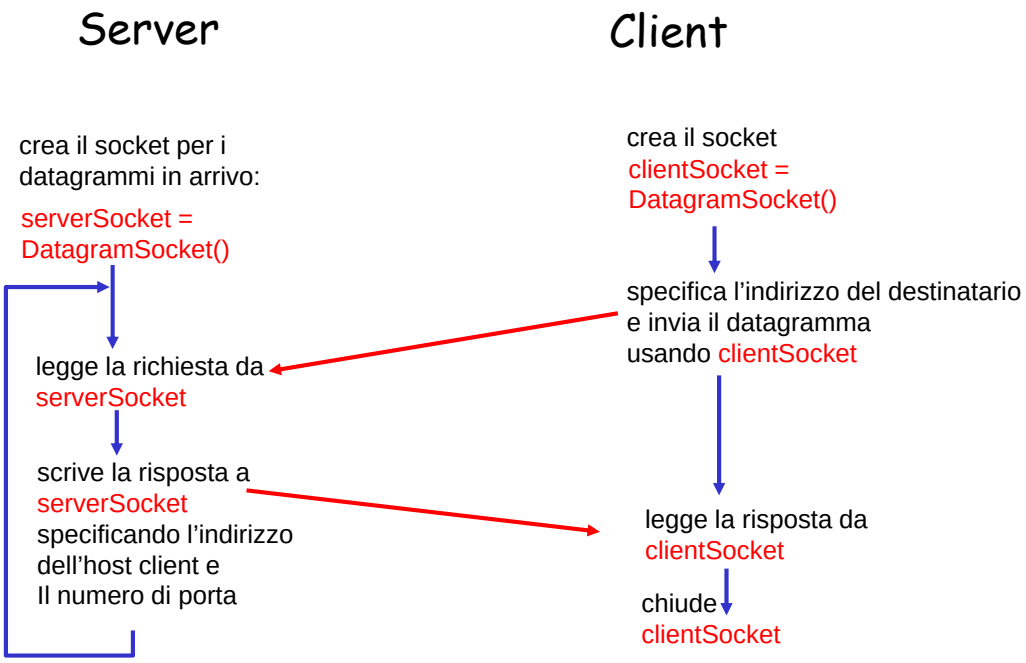
- Non c'è handshake
- Il mittente allega esplicitamente a ogni pacchetto l'indirizzo IP e la porta di destinazione
- Il server deve estrarre l'indirizzo IP e la porta del mittente dal pacchetto ricevuto

UDP: i dati trasmessi possono perdersi o arrivare a destinazione in un ordine diverso da quello d'invio

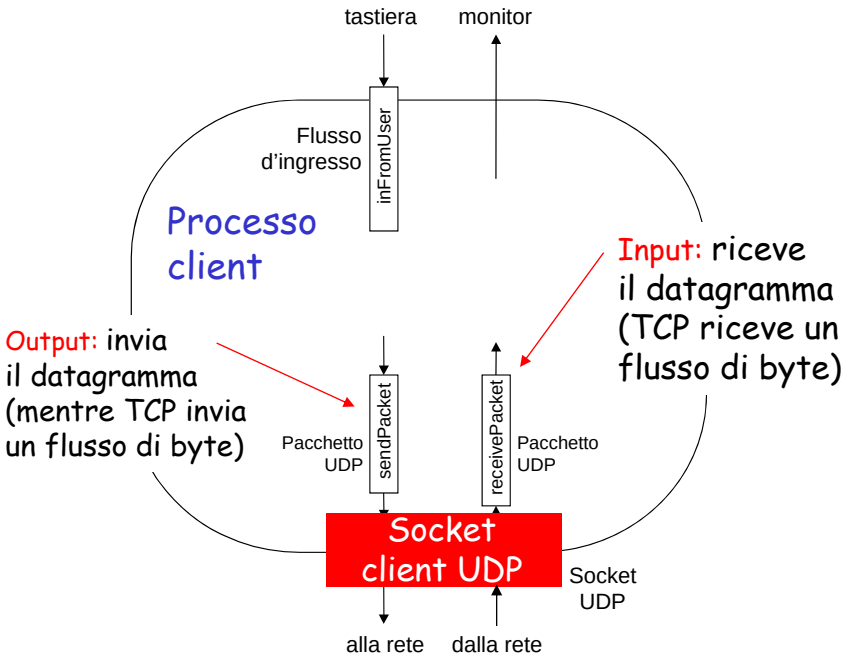
Punto di vista dell'applicazione

UDP fornisce un trasferimento inaffidabile di gruppi di byte ("datagrammi") tra client e server

Interazione tra client e server con UDP



Esempio: client Java (UDP)



Protocolli a livello di applicazione

Un protocollo definisce:

- il tipo dei messaggi scambiati, ad esempio i messaggi di richiesta e di risposta
- La sintassi dei messaggi: quali sono i campi nel messaggio e come sono descritti
- La semantica dei campi, ovvero il significato delle informazioni nei campi
- Le regole per determinare quando e come un processo invia e risponde ai messaggi

Protocolli di pubblico dominio:

- Definiti nelle RFC
- Consentono l'interoperabilità
- Ad esempio, HTTP, SMTP

Protocolli proprietari:

- Ad esempio, Skype
- Voi stessi potete ideare dei protocolli applicativi!

Quale servizio di trasporto richiede un'applicazione?

Perdita di dati

- alcune applicazioni (ad esempio, audio) possono tollerare qualche perdita
- altre applicazioni (ad esempio, trasferimento di file, telnet) richiedono un trasferimento dati affidabile al 100%

Temporizzazione

- alcune applicazioni (ad esempio, telefonia Internet, giochi interattivi) per essere "realistiche" richiedono ritardi limitati

Throughput

- alcune applicazioni (ad esempio, quelle multimediali) per essere "efficaci" richiedono una certa ampiezza di banda
- altre applicazioni ("le applicazioni elastiche") utilizzano l'ampiezza di banda che si rende disponibile

Sicurezza

- Cifratura, integrità dei dati, ...

Requisiti del servizio di trasporto di alcune applicazioni comuni

Applicazione	Tolleranza alla perdita di dati	Throughput	Sensibilità al tempo
Trasferimento file	No	Variabile	No
Posta elettronica	No	Variabile	No
Documenti Web	No	Variabile	No
Audio/video in tempo reale	Sì	Audio: da 5 Kbps a 1 Mbps Video: da 10 Kbps a 5 Mbps	Sì, centinaia di ms
Audio/video memorizzati	Sì	Come sopra	Sì, pochi secondi
Giochi interattivi	Sì	Fino a pochi Kbps	Sì, centinaia di ms
Messaggistica istantanea	No	Variabile	Sì e no

Servizi dei protocolli di trasporto Internet

Servizio di TCP:

- *orientato alla connessione:* è richiesto un setup fra i processi client e server
- *trasporto affidabile* fra i processi d'invio e di ricezione
- *controllo di flusso:* il mittente non sovraccarica il destinatario
- *controllo della congestione:* "rallenta" il mittente quando le rete è sovraccaricata
- *non offre:* temporizzazione, garanzie su un'ampiezza di banda minima, sicurezza

Servizio di UDP:

- trasferimento dati inaffidabile fra i processi d'invio e di ricezione
- non offre: setup della connessione, affidabilità, controllo di flusso, controllo della congestione, temporizzazione né ampiezza di banda minima e sicurezza

D: Allora perché esiste UDP?

Applicazioni Internet: protocollo a livello applicazione e protocollo di trasporto

Applicazione	Protocollo a livello applicazione	Protocollo di trasporto sottostante
Posta elettronica	SMTP [RFC 2821]	TCP
Accesso a terminali remoti	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
Trasferimento file	FTP [RFC 959]	TCP
Multimedia in streaming	HTTP (es. YouTube) RTP [RFC 1889]	TCP o UDP
Telefonia Internet	SIP, RTP, proprietario (es. Skype)	Tipicamente UDP

Livello di applicazione

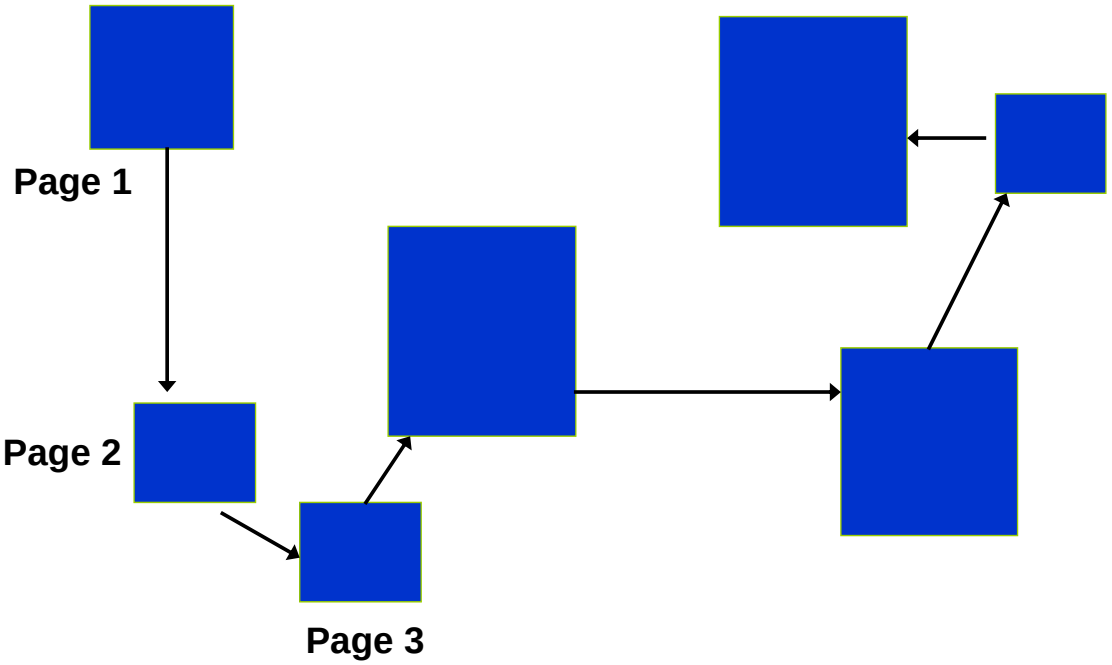
- **Web e HTTP**
- FTP
- Posta elettronica
  - SMTP, POP3, IMAP
- DNS
- Applicazioni P2P

World Wide Web (WWW o Web)

- Il World Wide Web è una architettura software per l'accesso ai documenti pubblicati sui vari siti di Internet.
- Nacque nel 1989 al CERN di Ginevra come mezzo per scambiare informazioni tra scienziati di diverse nazioni.
- Il primo browser grafico, Mosaic, fu sviluppato nel 1993. Poi nacquero Netscape, Explorer, Firefox ecc.
- Nel 1994 il CERN ed il MIT di Boston fondarono il W3C - World Wide Web Consortium (www.w3c.org), cui poi si associarono centinaia di università e società di tutto il mondo.
- Scopo del W3C: emanare standard (denominati RFC – Request For Comment) e favorire lo sviluppo del Web.

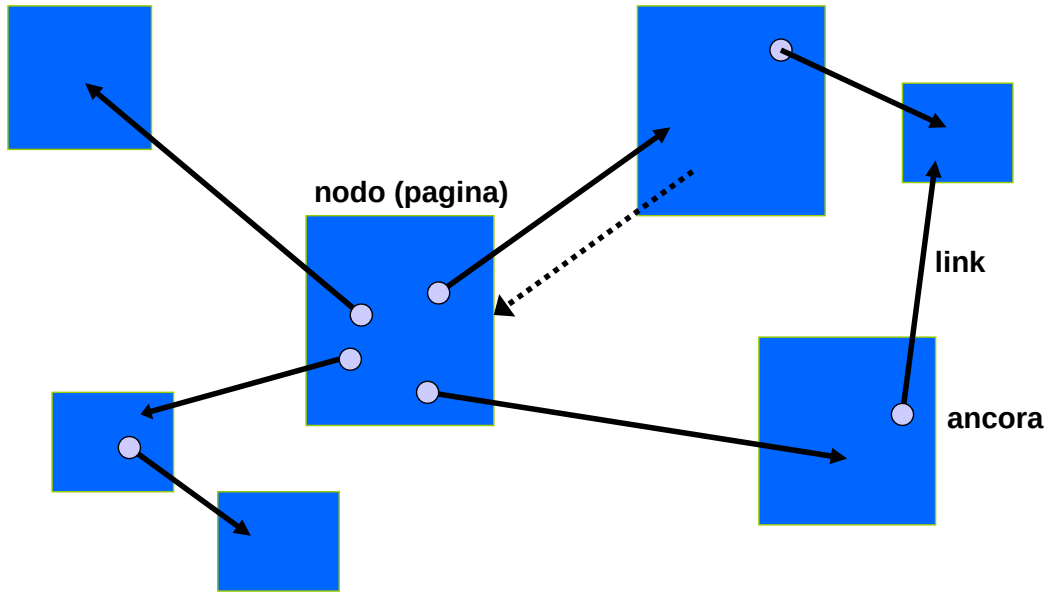
Organizzazione di un testo

lettura sequenziale



Organizzazione di un ipertesto

navigazione (browsing)



Terminologia

- **Browser**: programma applicativo client per navigare in rete.
- **Web server**: programma applicativo server eseguito sull'elaboratore del fornitore di informazioni (**provider**).
- **Pagina Web**: singolo "nodo" di un ipertesto. La pagina è scritta nel linguaggio HTML.
- **Home-page**: pagina di ingresso di un sito web.
- **Link, iperlink**: porzione di una pagina che, se selezionata, permette di raggiungere un altro punto della stessa pagina o una nuova pagina.

Document Model: HTML

- Gli ipertesti del Web sono scritti usando il linguaggio **HyperText Markup Language** (**HTML**).
- HTML descrive la disposizione degli elementi presenti all'interno di un documento ipertestuale. Può includere documenti in altri formati (immagini, file audio, video).
- Il **browser** interpreta il file HTML e lo visualizza.
- I documenti non HTML sono visualizzati tramite **plug-in** (es. per le immagini JPEG) o tramite **helper applications** (es. per documenti PDF o Word). Questi ultimi sono applicativi esterni al browser.

HTML

```
<HTML>                                          <!- Start of HTML document    -->
<HEAD>
<TITLE>page title</TITLE>
</HEAD>
<BODY>                                          <!- Start of the main body      -->
<H1>Hello World/H1>                             <!- Basic text to be displayed  -->
<P>                                             <!- Start of a new paragraph   -->
<SCRIPT type = "text/javascript">               <!- identify scripting language -->
  document.writeln ("<H1>Hello World</H1>;      // Write a line of text
</SCRIPT>                                       <!- End of scripting section    -->
</P>                                            <!- End of paragraph section -->
</BODY>                                         <!- End of main body           -->
</HTML>                                         <!- End of HTML section        -->
```

- Questa pagina HTML contiene uno script, nel linguaggio Javascript, che produce una parte del testo HTML che sarà visualizzato dal browser.

Uniform Resource Locator

Gli indirizzi dei documenti messi a disposizione sui siti Web sono espressi tramite gli **Uniform Resource Locator (URL)**

L'URL determina:

- il nome della pagina
- su quale sito si trova la pagina (IP o nome host)
- come è possibile accedere alla pagina (protocollo e porta)

Formato: **protocollo://indirizzodelserver[:portaTCP]/pathname**

URL: formati

Scheme		Host name	Pathname
http	://	www.cs.vu.nl	/home/steen/mbox

(a)

Scheme		Host name		Port	Pathname
http	://	www.cs.vu.nl	:	80	/home/steen/mbox

(b)

Scheme		Host name		Port	Pathname
http	://	130.37.24.11	:	80	/home/steen/mbox

(c)

a) Con nome DNS.
b) Con nome DNS e numero di porta.
c) Con indirizzo IP e numero di porta.

URL: protocolli utilizzati

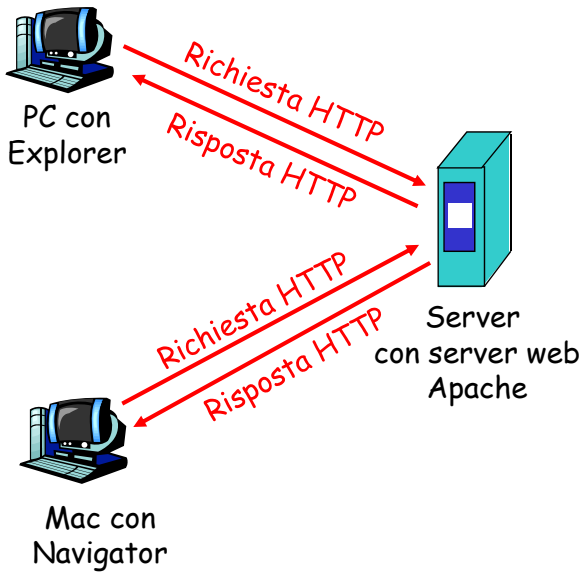
- I browser web sono in grado di gestire non solo HTTP ma molti altri protocolli che è possibile specificare nell'URL.

http	HTTP	http://www.icar.cnr.it/dir/index.html
https	HTTPS	HTTP "sicuro" (sulla porta 443)
ftp	FTP	ftp://ftp.lcs.mit.edu/pub/README
file	file locale	C:\doc\lezione-reti.ppt
mailto	invio e-mail	mailto:louiqa@disney.com
telnet	login remoto	telnet://si.deis.unical.it

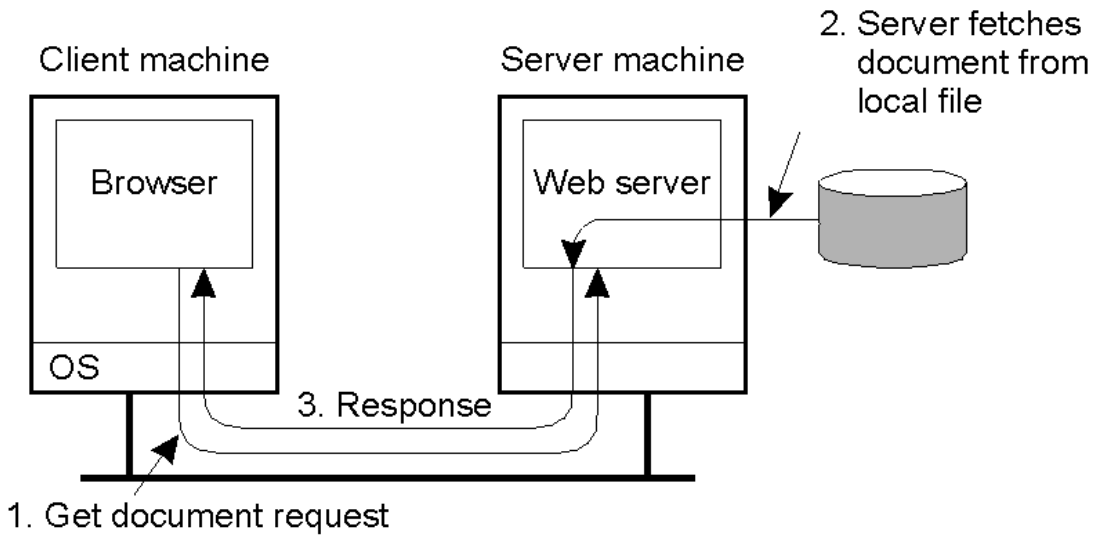
Il protocollo del Web: HTTP

HTTP: hypertext transfer protocol

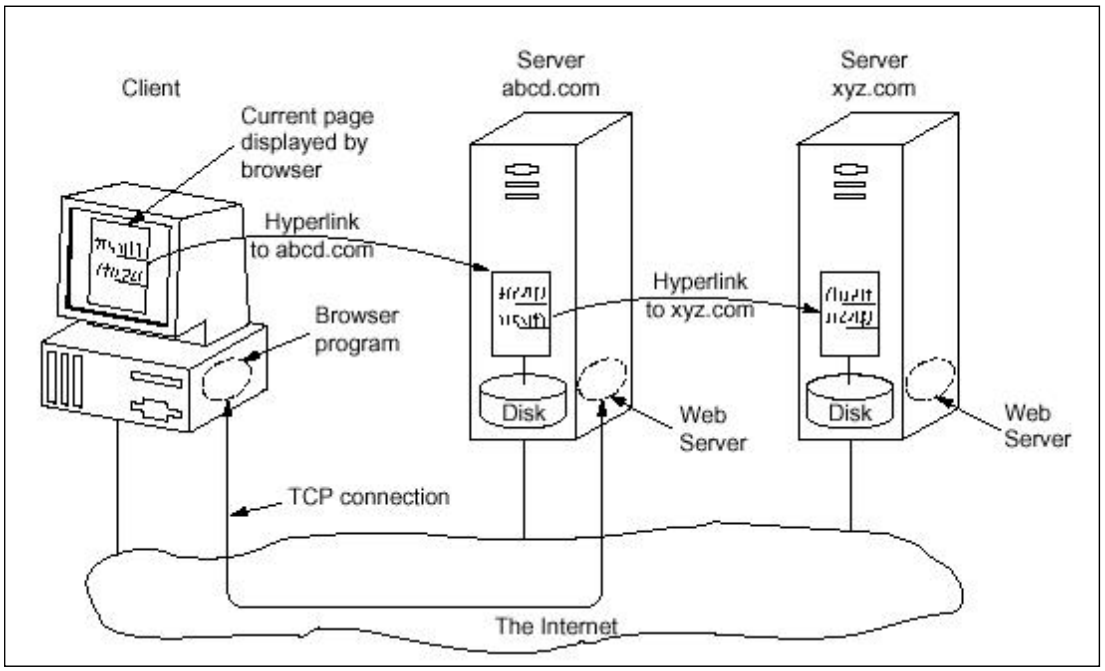
- Protocollo a livello di applicazione del Web
- Modello client/server
  - *client:* il browser che richiede, riceve, "visualizza" gli oggetti del Web
  - *server:* il server web invia oggetti (in genere pagine HTML) in risposta ad una richiesta



Architettura client-server del Web



Link tra due siti Web



Panoramica su HTTP

HTTP usa TCP:

- Il client inizializza la connessione TCP (crea una socket) con la porta 80 del server
- Il server accetta la connessione TCP dal client
- Messaggi HTTP scambiati fra browser (client HTTP) e server web (server HTTP)
- Chiusura connessione TCP

HTTP è un protocollo "senza stato" (stateless)

- Il server non mantiene informazioni sulle richieste fatte dal client

nota

I protocolli che mantengono lo "stato" sono complessi!

- La storia passata (*stato*) deve essere memorizzata
- Se il server e/o il client si bloccano, le loro viste dello "stato" potrebbero essere contrastanti e dovrebbero essere riconciliate

Connessioni HTTP

Connessioni non persistenti

- Su una connessione TCP viene trasmesso un solo oggetto

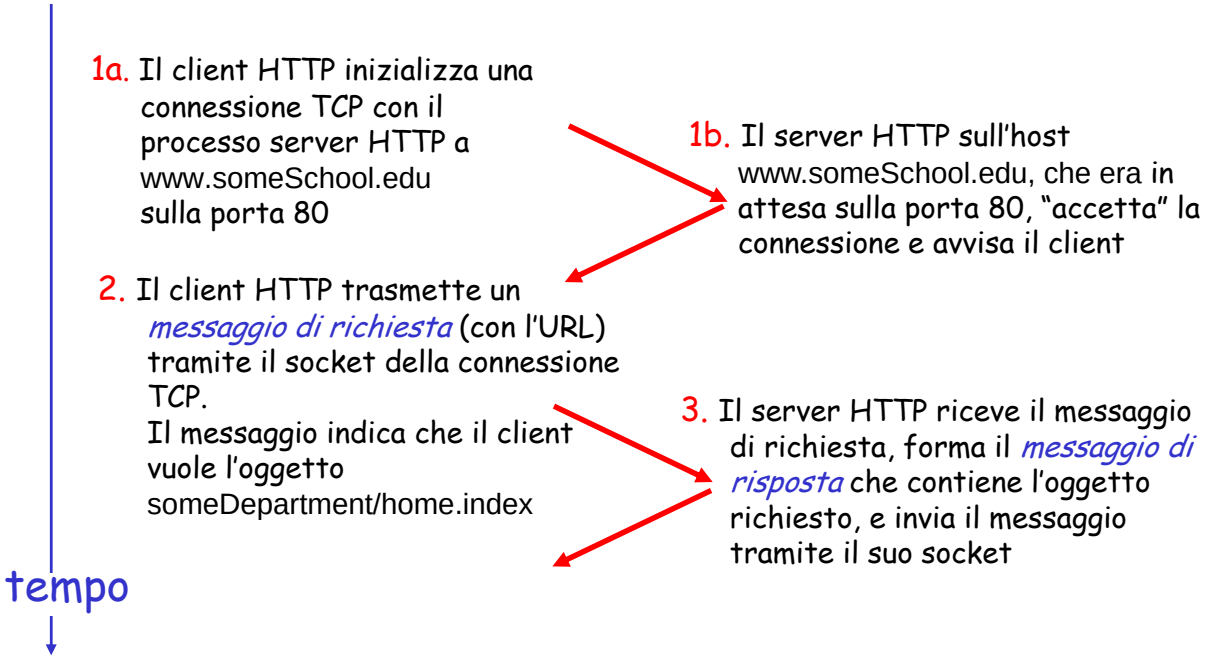
Connessioni persistenti

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

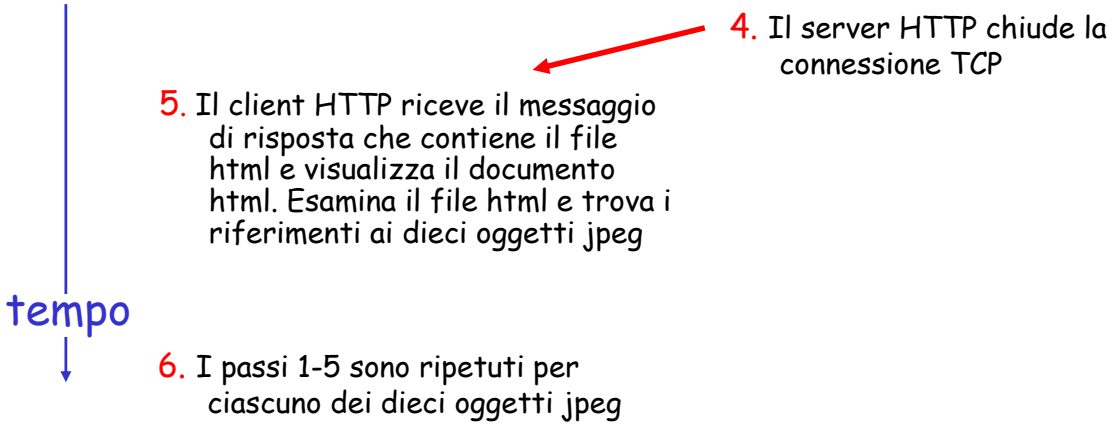
Connessioni non persistenti

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

La pagina web contiene testo e riferimenti a dieci immagini jpeg



Connessioni non persistenti (continua)

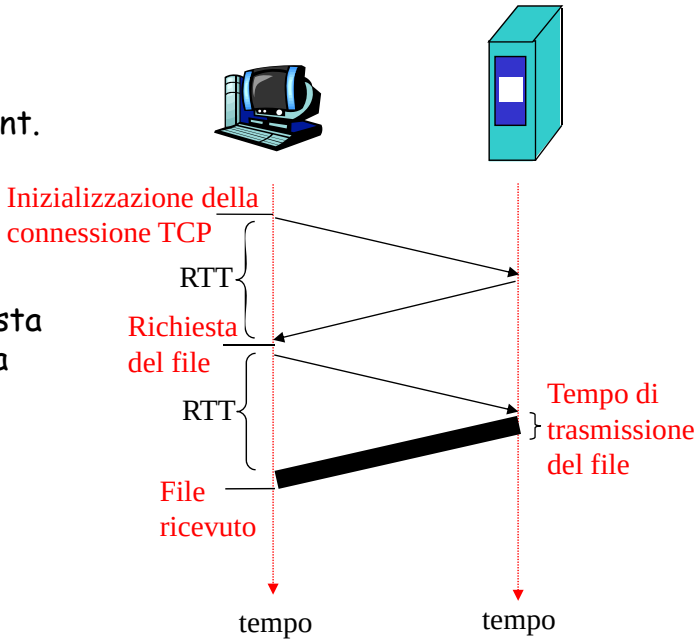


Schema del tempo di risposta

Definizione di RTT: tempo impiegato da un pacchetto per andare dal client al server e ritornare al client.

Tempo di risposta:

- un RTT per inizializzare la connessione TCP
- un RTT perché ritornino la richiesta HTTP e i primi byte della risposta HTTP
- tempo di trasmissione del file



totale = 2RTT + tempo di trasmissione

Connessioni persistenti

Svantaggi delle connessioni non persistenti:

- richiedono 2 RTT per oggetto
- overhead del sistema operativo per *ogni* connessione TCP

Connessioni persistenti

- il server lascia la connessione TCP aperta dopo l'invio di una risposta
- i successivi messaggi tra gli stessi client/server vengono trasmessi sulla connessione aperta
- il client invia le richieste non appena incontra un oggetto referenziato
- un solo RTT per tutti gli oggetti referenziati

Messaggi HTTP

- due tipi di messaggi HTTP: *richiesta, risposta*
- Messaggio di richiesta HTTP:
  - ASCII (formato leggibile dall'utente)

Riga di richiesta (comandi GET, POST, HEAD)

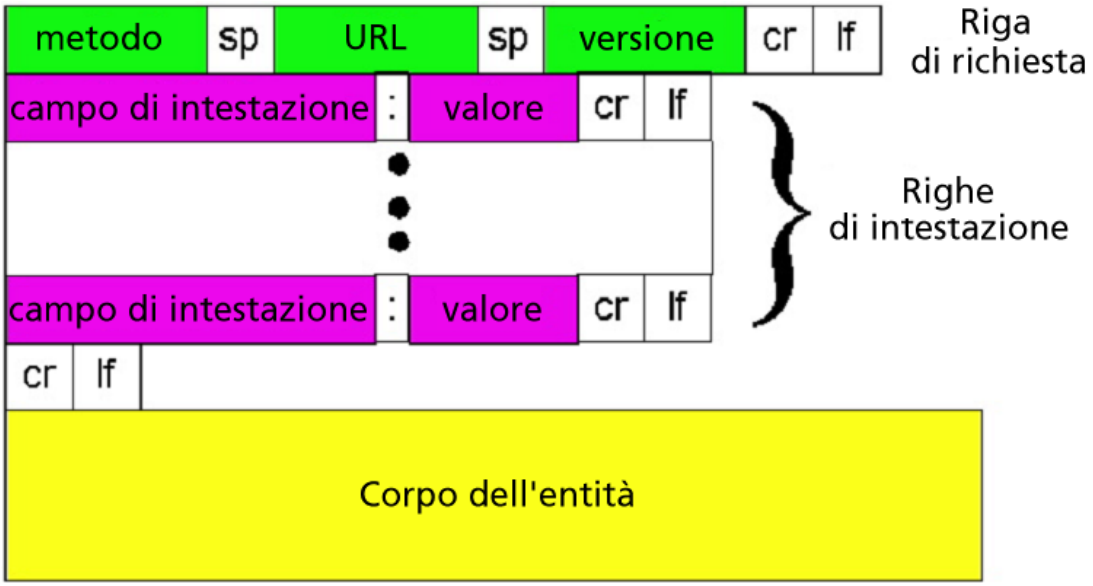
Righe di intestazione

Un carriage return e un line feed indicano la fine del messaggio

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language:fr
```

(carriage return e line feed extra)

Messaggio di richiesta HTTP: formato generale



Web interattivo: le form

- Il linguaggio **HTML 1.0** permetteva di visualizzare i documenti, ma non di fornire dati al server.
- **HTML 2.0** ha introdotto le **form**. Una form contiene campi (caselle, pulsanti) con i quali l'utente può inserire informazioni ed inviarle al server.
- Ad ogni campo è associato un valore. Ad esempio il valore di una area di testo è il testo inserito dall'utente, il valore di un gruppo di pulsanti di scelta (*radiobutton*) è l'etichetta del pulsante scelto dall'utente.
- È compito del server interpretare i dati ricevuti tramite la form.

Invio dei dati di una form

Metodo POST:

- La pagina web spesso include un form per l'input dell'utente
- I dati della form sono messi nel corpo della richiesta HTTP

Metodo GET:

- I dati della form sono messi in coda al campo URL della riga di richiesta:

```
www.somesite.com/animalsearch?weight=heavy&food=banana
```

Tipi di metodi

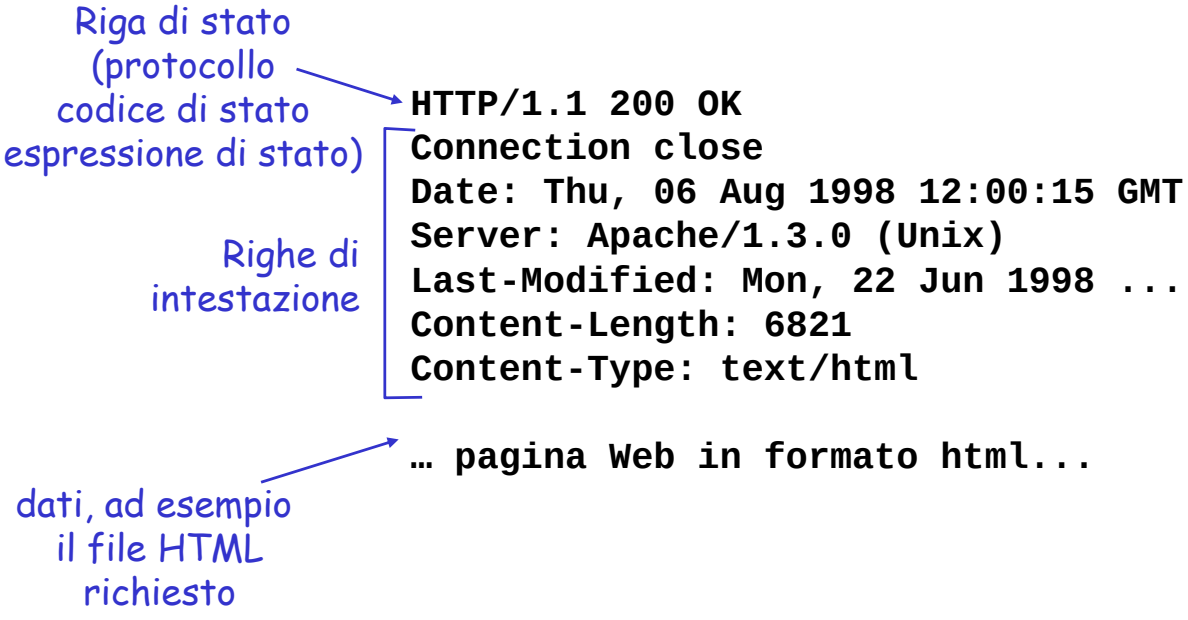
HTTP/1.0

- GET
- POST
- HEAD
  - chiede al server di inviare solo i campi di intestazione

HTTP/1.1

- GET, POST, HEAD
- PUT
  - include il file nel corpo dell'entità e lo invia al percorso specificato nel campo URL
- DELETE
  - cancella il file specificato nel campo URL

Messaggio di risposta HTTP



Header HTTP: un elenco

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

Codici di stato della risposta HTTP

Nella prima riga nel messaggio di risposta server->client.
Alcuni codici di stato e relative espressioni:

200 OK

- La richiesta ha avuto successo; l'oggetto richiesto viene inviato nella risposta

301 Moved Permanently

- L'oggetto richiesto è stato trasferito; la nuova posizione è specificata nell'intestazione **`Location:`** della risposta

400 Bad Request

- Il messaggio di richiesta non è stato compreso dal server

404 Not Found

- Il documento richiesto non si trova su questo server

505 HTTP Version Not Supported

- Il server non ha la versione di protocollo HTTP specificata

Provare HTTP (lato client)

1. Collegarsi via Telnet al vostro server web preferito:

```
telnet cis.poly.edu 80
```

Apre una connessione TCP alla porta 80 (porta di default per un server HTTP) dell'host cis.poly.edu.
Tutto ciò che digitate viene trasmesso alla porta 80 di cis.poly.edu

2. Digitare una richiesta GET:

```
GET /~ross/ HTTP/1.1
Host: cis.poly.edu
```

Digitando questo (premete due volte il tasto Invio), trasmettete una richiesta GET minima (ma completa) al server HTTP

3. Guardare il messaggio di risposta trasmesso dal server HTTP!

Per attivare telnet su Windows 7, andare su Start -> Pannello di controllo -> Programmi -> Attivazione o disattivazione delle funzionalità di Windows

Interazione utente-server: i cookie

Molti dei più importanti siti web usano i cookie

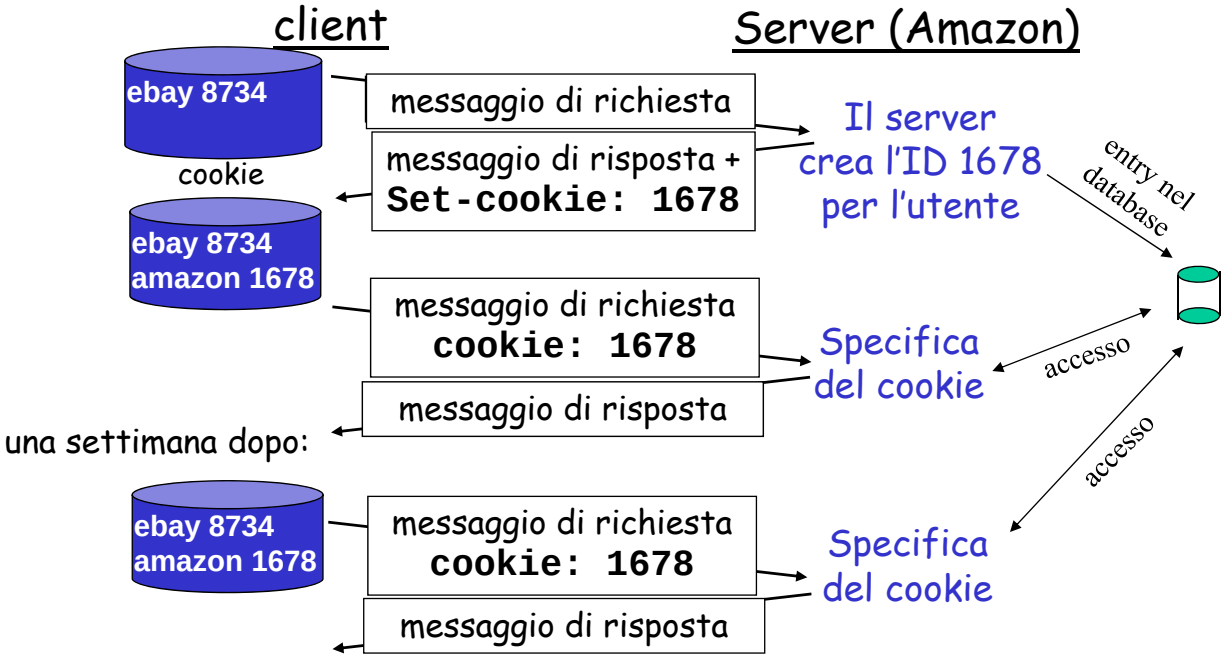
Quattro componenti:

1) Una riga di intestazione nel messaggio di *risposta* HTTP
2) Una riga di intestazione nel messaggio di *richiesta* HTTP
3) Un file cookie mantenuto sul sistema terminale dell'utente e gestito dal browser dell'utente
4) Un database sul sito del server Web

Esempio:

- Susan accede sempre a Internet dallo stesso PC
- Visita per la prima volta un particolare sito di commercio elettronico
- Quando la richiesta HTTP iniziale giunge al sito, il sito crea un identificativo unico (ID) e una *entry* nel database per Susan

Cookie (continua)



Cookie (continua)

Cosa possono contenere i cookie:

- autorizzazione
- carta per acquisti
- raccomandazioni
- stato della sessione dell'utente (e-mail)

Lo "stato"

- I cookie mantengono lo stato del mittente e del ricevente per più transazioni
- I messaggi http trasportano informazioni di stato

Cookie e privacy: nota

- i cookie permettono ai siti di imparare molte cose sugli utenti
- l'utente può fornire al sito il nome e l'indirizzo e-mail
- quindi abilitando i cookie si rinuncia ad un certo grado di privacy

Web dinamico

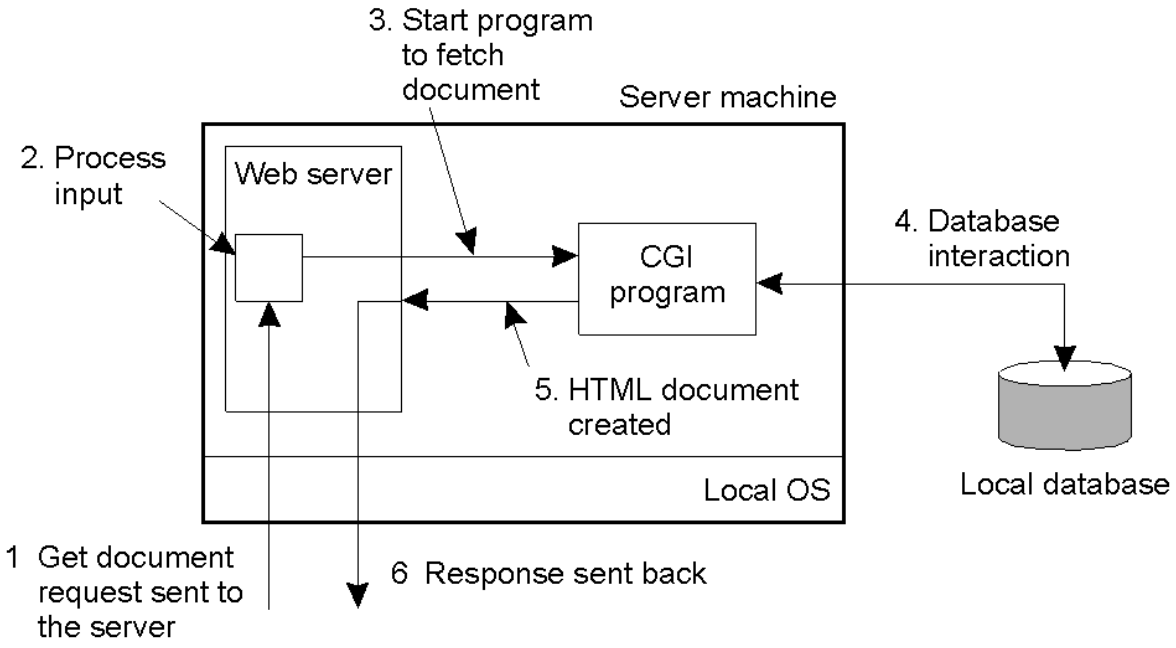
- Le pagine Web sono **statiche** se il loro contenuto non è modificabile.
- Il contenuto delle pagine **Web dinamiche** è invece generato al momento della richiesta, sul server e/o sul client.

Web dinamico: lato server

- **Interfaccia CGI**
- Esempio: il server elabora i dati della form, in base ad essi effettua una query su un database, inserisce il risultato della query nella pagina HTML, ed infine invia la pagina al client.
- La pagina HTML contiene quindi sia testo statico, sia testo dinamico, cioè il risultato della query che ovviamente può variare da una esecuzione all'altra.

- **Linguaggi di Scripting sul lato server**
- Alcuni linguaggi molto diffusi sono:
  - PHP
  - JSP e Servlet (Java)
  - ASP
  - Javascript
  - PERL

Uso di programmi CGI



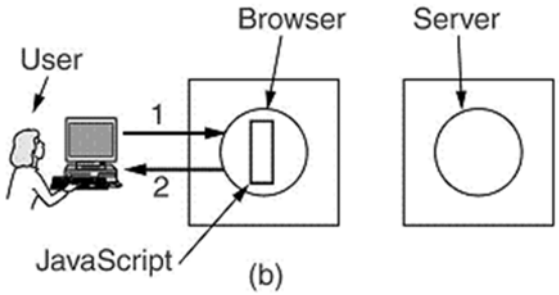
Esempio di script eseguito sul server

```
(1)     <HTML>
(2)     <BODY>
(3)     <P>The current content of <EM>/datafile.txt</EM>is:</P>
(4)     <P>
(5)     <SERVER type = "text/javascript");
(6)         clientFile = new File("/datafile.txt");
(7)         if(clientFile.open("r")){
(8)                 while (!clientFile.eof())
(9)                     document.writeln(clientFile.readln());
(10)                clientFile.close();
(11)        }
(12)    </SERVER>
(13)    </P>
(14)    <P>Thank you for visiting this site.</P>
(15)    </BODY>
(16)    </HTML>
```

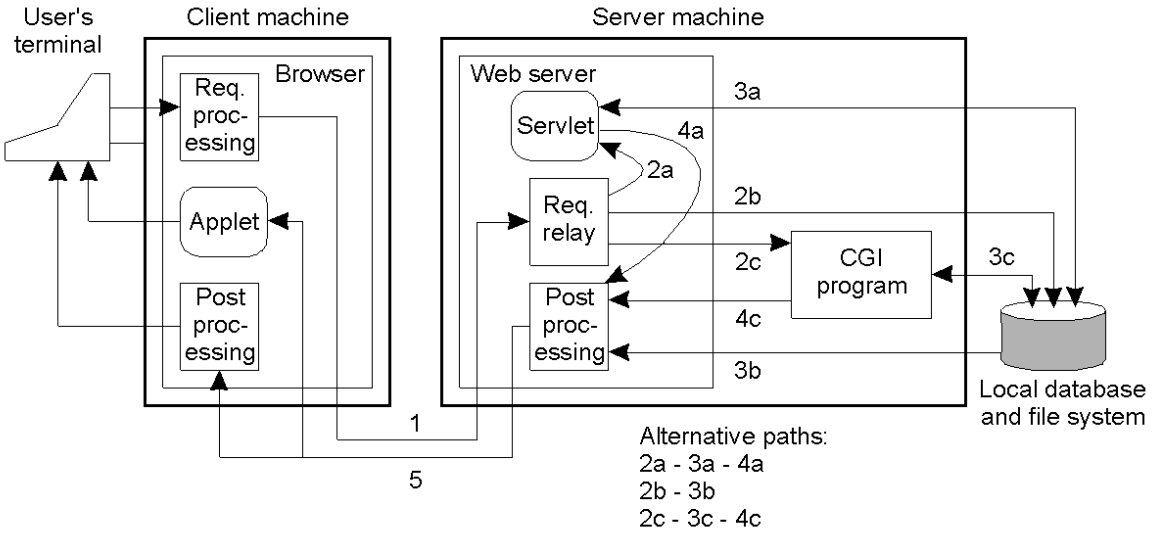
- Il codice Javascript inserisce nella pagina HTML il contenuto del file datafile.txt

Web dinamico: lato client

- E' possibile elaborare i dati di una form **direttamente sul browser**, senza inviarli al server.
- Questo può servire per verificare la correttezza dei dati inseriti dall'utente (es. la correttezza di un codice fiscale o l'effettivo inserimento di dati in un campo obbligatorio).
- Il linguaggio di **scripting** più usato sul lato client è **Javascript**.
- Altri metodi: **applet Java** e controlli **ActiveX** di Microsoft.

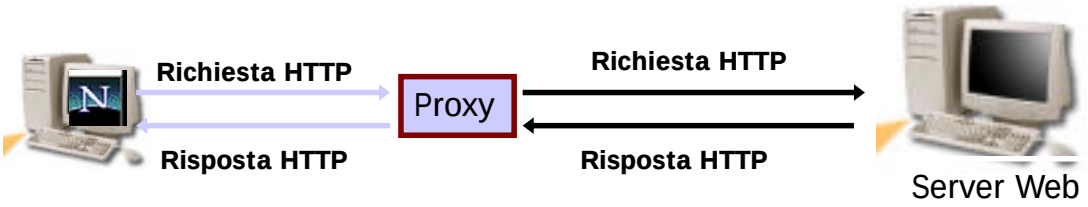


Generazione di pagine dinamiche



Proxy HTTP

- Un **proxy HTTP** (o Web Proxy) agisce da intermediario fra il client e il server: riceve le richieste dal client ed eventualmente le inoltra al server.
- I proxy HTTP sono utilizzati per:
  - gestire il caching (ad esempio si evita di aprire connessioni multiple per ottenere una stessa pagina Web)
  - controllare e filtrare gli accessi alla rete da parte degli host di una Intranet



Caching

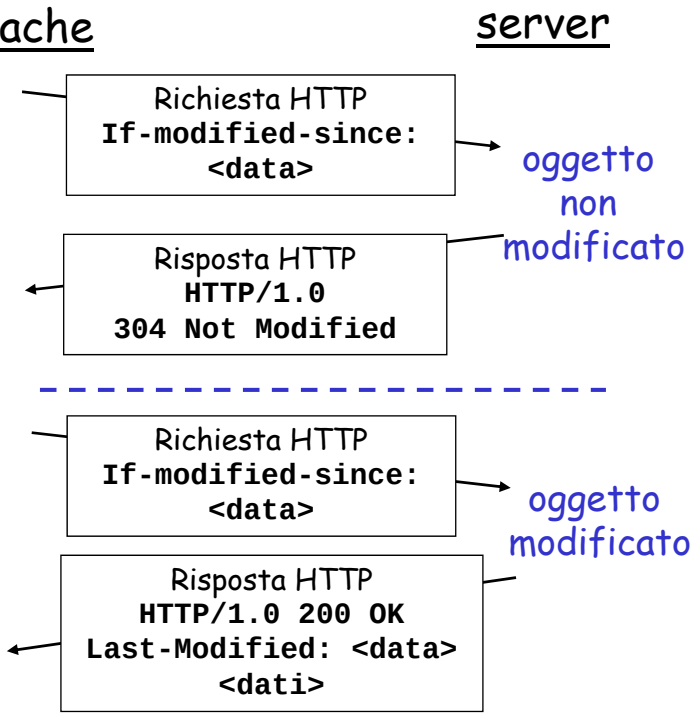
- Il caching può essere effettuato:
  - sulla macchina del client
  - sul proxy della rete locale del client
  - sul proxy del provider
  - sul server Web del provider
- Le operazioni di caching utilizzano gli header:
  - **Last-Modified:** (nella risposta HTTP)

    serve a stabilire quanto tempo tenere la pagina nella cache; se è stata modificata di recente è probabile che sarà di nuovo modificata a breve.
  - **If-Modified-Since:** (nella richiesta HTTP)

    la pagina viene scaricata solo se è stata modificata sul server *dopo* la data specificata.
- Una pagina dinamica non dovrebbe essere mai essere messa nella cache.

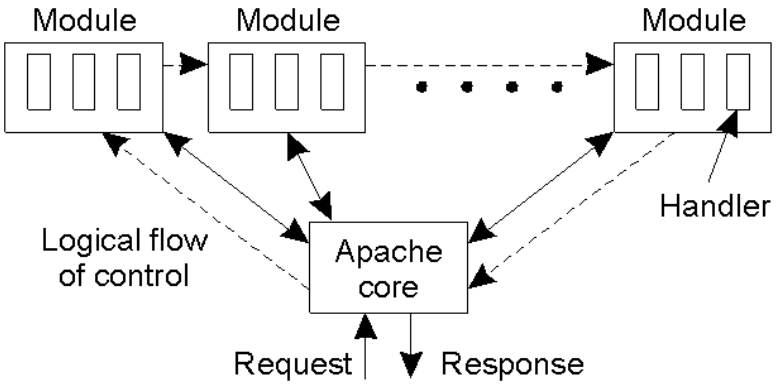
GET condizionale

- Obiettivo: non inviare un oggetto se la cache ha una copia aggiornata dell'oggetto
- cache: specifica la data della copia dell'oggetto nella richiesta HTTP
  - **`If-modified-since: <data>`**
- server: la risposta non contiene l'oggetto se la copia nella cache è aggiornata:
  - **`HTTP/1.0 304 Not Modified`**



Server Web

- Organizzazione di un server Web Apache.

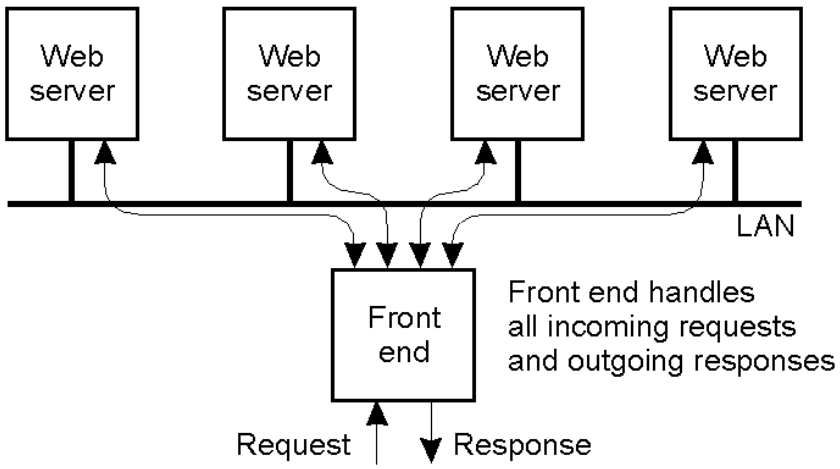


Idea di base: poiché un server deve eseguire diverse operazioni in cascata, è possibile assegnare tali operazioni a diversi moduli, con una tecnica di pipeline.

Cluster di Server Web

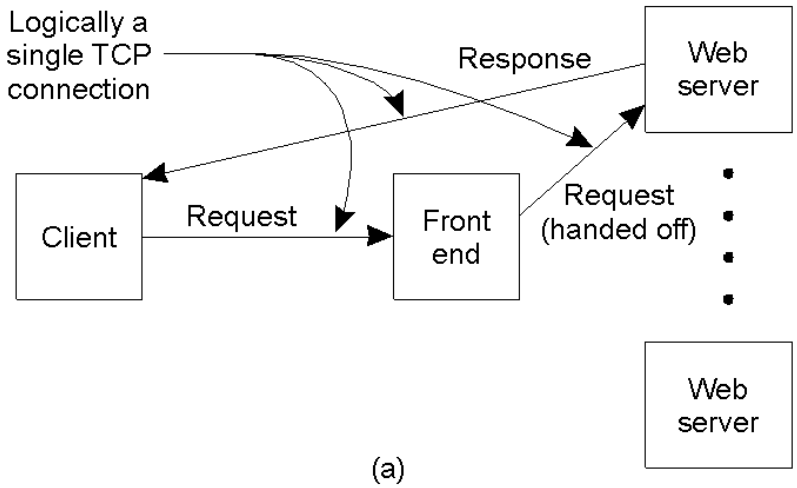
Per migliorare l'efficienza, è possibile smistare le richieste a più server Web che operano in parallelo.

Le richieste sono catturate da un server di "front end", che le smista ai vari server a seconda del loro carico.



Cluster di Server Web: TCP handoff

Usando l'architettura della pagina precedente, le risposte devono sempre passare per il front end. La tecnica del "TCP handoff" consente al server di "back end" di inviare la risposta direttamente al client. Il client non si accorge di nulla, e vede la risposta come se gli provenisse dal front end.



(a)