



**UNIVERSITÀ
DEGLI STUDI
DI UDINE**
hic sunt futura

TECNOLOGIE WEB E LABORATORIO (a.a. 2022-2023)

WORLD WIDE WEB

Daniele Salvati

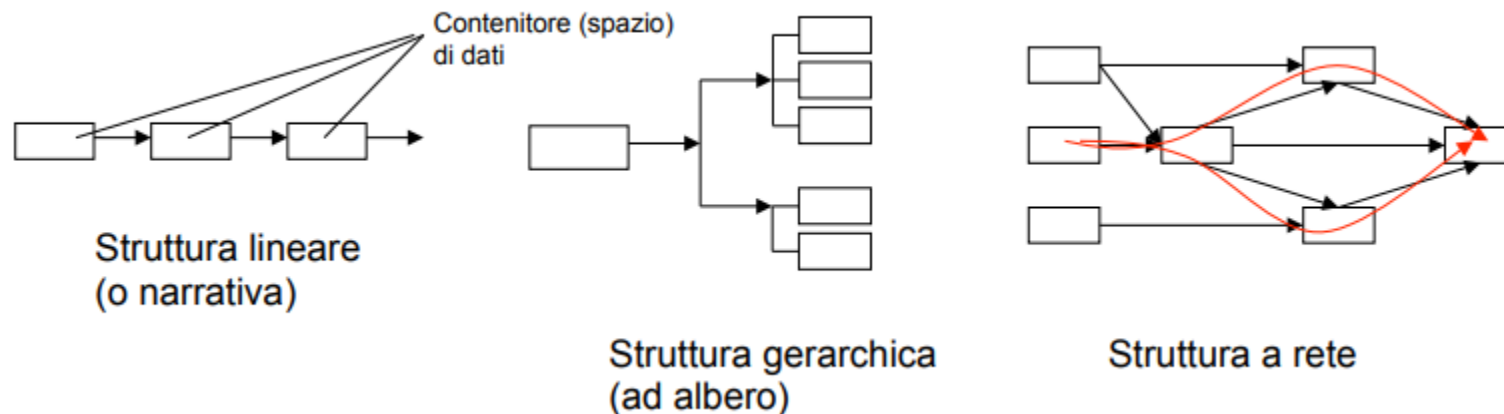
Dipartimento di Scienze Matematiche, Informatiche e Fisiche
Università degli Studi di Udine

Informazioni slide

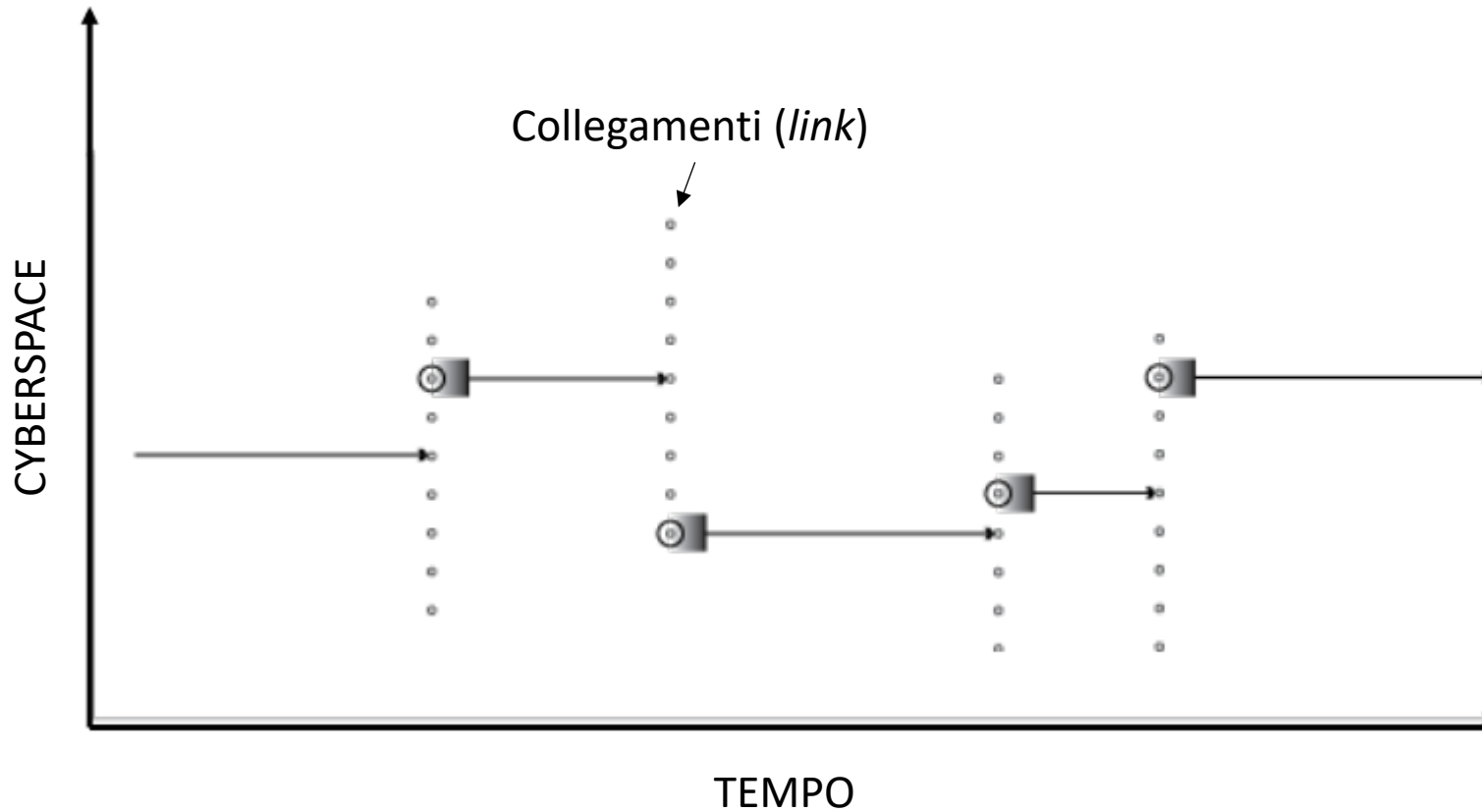
- Il materiale contenuto in queste slide è riservato esclusivamente agli studenti del corso di **Tecnologie Web e Laboratorio** del Corso di Studio in **Internet of Things, Big Data, Machine Learning** dell'Università degli Studi di Udine.
- Non è consentita la diffusione del materiale contenuto in queste slide, ma solo l'utilizzo inerente la preparazione dell'esame del suddetto corso.

World Wide Web

- Il **Web** è un **servizio del livello applicativo di Internet** con architettura client-server.
- Il **Web** è uno **spazio informativo e operativo** con una **struttura ipertestuale**.
- **Ipertesto**: struttura multilineare (esistono più percorsi possibili) in cui l'interazione è vista come navigazione.



Navigazione



Cyberspace: l'insieme delle pagine web disponibili

Ipermedia

- Il **Web** è un **Ipermedia**: documento con struttura multilineare che utilizza diverse modalità di rappresentazione dell'informazione:
 - Testi scritti
 - Immagini
 - Audio
 - Video
 - Animazioni
 - Ecc.

Le risorse multimediali sono connesse tra loro in maniera non sequenziale.

Pagine Web

- Una **pagina web** (*web page*), detta anche documento, è costituita da oggetti.
- Un **oggetto** è un file (quale un file HTML, un'immagine JPEG, un file Javascript, un foglio di stile CSS, una clip video, un file audio, ecc.) caratterizzato da un indirizzo.
- La maggioranza delle pagine web consiste di un **file HTML principale** (*base HTML file*) e **diversi oggetti referenziati da esso**.
- Esempio:
 - Se una pagina web contiene testo in HTML e cinque immagini JPEG, allora la pagina nel complesso presenta sei oggetti. Il file HTML referencia gli altri oggetti nella pagina tramite il loro Indirizzo di collegamento.

Le origini del Web

- Nasce tra la fine del 1989 e gli inizi del 1990, al CERN (Centro Europeo per le Ricerche Nucleari) di Ginevra.
- L'obiettivo era di trasformare "Internet" in un **servizio di pubblicazione e distribuzione di dati e informazioni scientifiche**.
- Si ispira ad una **idea di ipertesto** (il Memex) illustrata nel **1945** da Vannevar Bush nell'articolo "As we may think" per la rivista The Atlantic Monthly.
- Il **termine ipertesto** fu coniato più tardi (nel 1963) da Ted Nelson.

Le origini del Web (2)

- Nel 1991 vengono messi a punto da **Tim Berners Lee** i componenti principali del web:
 - Un programma fornitore del servizio (web server)
 - Un programma per la consultazione delle informazioni (web client o browser)
 - Le prime versioni del protocollo **HTTP** (*HyperText Transfer Protocol*) e del linguaggio di marcatura **HTML** (*HyperText Markup Language*).

Le origini del Web (3)

- Nel 1994, viene fondato dal MIT e CERN il **World Wide Web Consortium (W3C)**.
- La missione del **W3C** è portare il World Wide Web al suo pieno potenziale, sviluppando protocolli e linee guida che assicurino la crescita a lungo termine del Web.



<https://www.w3.org/>

Standard

- La principale attività svolta dal W3C consiste nello stabilire **standard** tecnici per il World Wide Web sui **linguaggi di markup** e sui **protocolli di comunicazione**.
- **Standard**: documentazione su linee guida, che riflette accordi su prodotti, pratiche operazioni da parte di associazioni o enti governativi, professionali o commerciali riconosciuti a livello nazionale o internazionale.

Principi fondamentali del Web

- **Accesso Universale:** il Web deve essere accessibile e utilizzabile da parte di chiunque, dovunque, in qualsiasi momento e con qualsiasi modalità.
- **Universo Aperto:** il Web deve potersi sviluppare con continuità tramite l'introduzione di nuove risorse, di innovazioni tecnologiche, di adeguamenti e personalizzazioni richiesti localmente.

Accesso Universale

- La realizzazione di questo principio ha dato luogo a tre iniziative importanti del WC3:
 - **Web Accessibility Initiative:** si propone di rendere il web accessibile a chiunque, inclusi i portatori di disabilità (es., Governo Italiano, legge "Stanca" del 2004 e successive).
 - **Internationalization Activity:** ha lo scopo di rendere il Web accessibile dovunque (es., sviluppa supporti per le lingue locali, vedi Google Translator).
 - **Device Independence Activity** (**Ubiquitous Web Domain**): ha lo scopo di rendere accessibile il Web in qualsiasi momento e con qualsiasi modalità.

Usabilità e Accessibilità

- **Usabilità:** è la capacità di un sistema (es. sito web) di permettere a determinati utenti di raggiungere determinati scopi con efficacia, efficienza, sicurezza e soddisfazione nell'ambito di determinati contesti d'uso.
- **Accessibilità:** un sistema è accessibile se persone disabili (o diversamente abili!) possono accedervi ed usarlo con la stessa efficacia e sicurezza di persone non disabili. Nota: il sistema può permettere alle persone disabili di realizzare gli stessi scopi delle persone non disabili, ma attraverso percorsi (sequenze di azioni) diversi ed esperienze d'uso diverse.

Universo Aperto

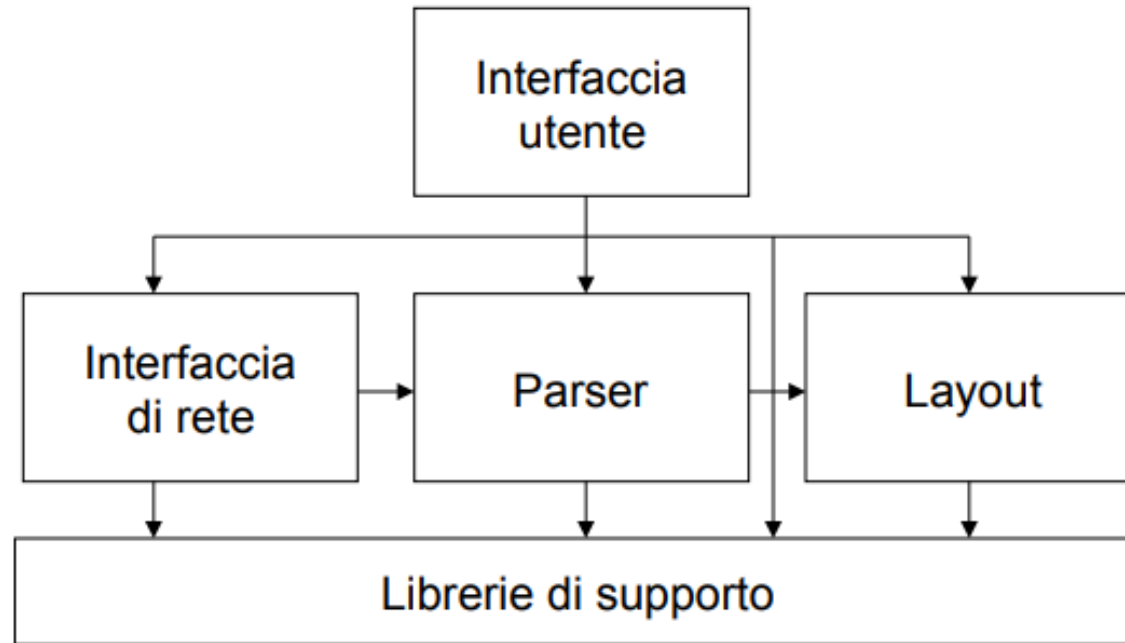
- **Interoperabilità:** capacità di due o più sistemi o componenti di scambiarsi informazione e usare l'informazione che è stata scambiata.
- **Evolvibilità:** nel Web deve essere sempre possibile aggiornare un sistema o un componente e aggiungere nuove realizzazioni, purché aderenti alle linee guida emesse dagli organismi ufficiali (es. evoluzione dei protocolli (http), dei linguaggi (html), dei software (browser), ecc.).
- **Uso di linguaggi misti:** tipico è l'utilizzo di linguaggi diversi per realizzare pagine o applicazioni Web (es. pagine che contengono codice HTML, CSS, e Javascript).

Applicazione Web

- L'applicazione web è caratterizzata da:
 - Struttura **ipertestuale** dei documenti
 - Standard per i **formati dei documenti** (es. HTML5, XHTML)
 - **Browser Web**
 - **Server Web**
 - Protocolli a livello di applicazione: **HTTP** e **HTTPS**

Architettura di un Browser Web

Elementi principali di un Browser Web



Architettura di un Browser Web (2)

- **Interfaccia utente:** programma che interagisce direttamente con l'utente e permette la connessione e il recupero dei file da remoto attraverso l'Interfaccia di rete, trasferisce il codice HTML a Parser e fa riferimento a Layout sul lato grafico.
- **Interfaccia di rete:** gestisce il flusso dati attraverso Internet e il mantenimento della memoria cache e dei cookies.
- **Parser:** è responsabile dell'analisi sintattica (*parsing*) del file richiesto.
- **Layout:** gestisce la presentazione dei componenti del sistema. Ha il compito di organizzare e visualizzare i contenuti delle pagine web.
- **Librerie di supporto:** forniscono le componenti software per l'interfacciamento del browser con la specifica piattaforma su cui il browser è installato.

Uniform Resource Locator (URL)

- Richiesta di una risorsa:
 - Nello spazio web è importante **identificare univocamente** le **risorse** che lo compongono.
 - Esiste un modo standard per farlo mediante l'**URL** (*Uniform Resource Locator*).
- L'URL è lo "strumento" che permette di identificare le risorse mediante un indirizzo.
- Esempio:
<https://www.uniud.it>

Uniform Resource Locator (URL) (2)

- Ogni URL è identificato da uno schema tipico:

protocol://hostname:port/path-and-file-name

- **Protocol:** protocollo a livello applicazione utilizzato da client e server (HTTP, HTTPS, FTP, ecc.).
- **Hostname:** nome dominio DNS (*Domain Name System*) (es. www.uniud.it) o indirizzo IP (es. 158.110.3.46) del server.
- **Port:** numero della porta del servizio applicativo del server.
- **Path-and-the-file-name:** nome ed ubicazione della risorsa richiesta all'interno della directory del server.

Uniform Resource Locator (URL) (3)

- Un URL può comprendere anche altri campi:

protocol://username:password@hostname:port/path-and-file-name?querystring#fragment

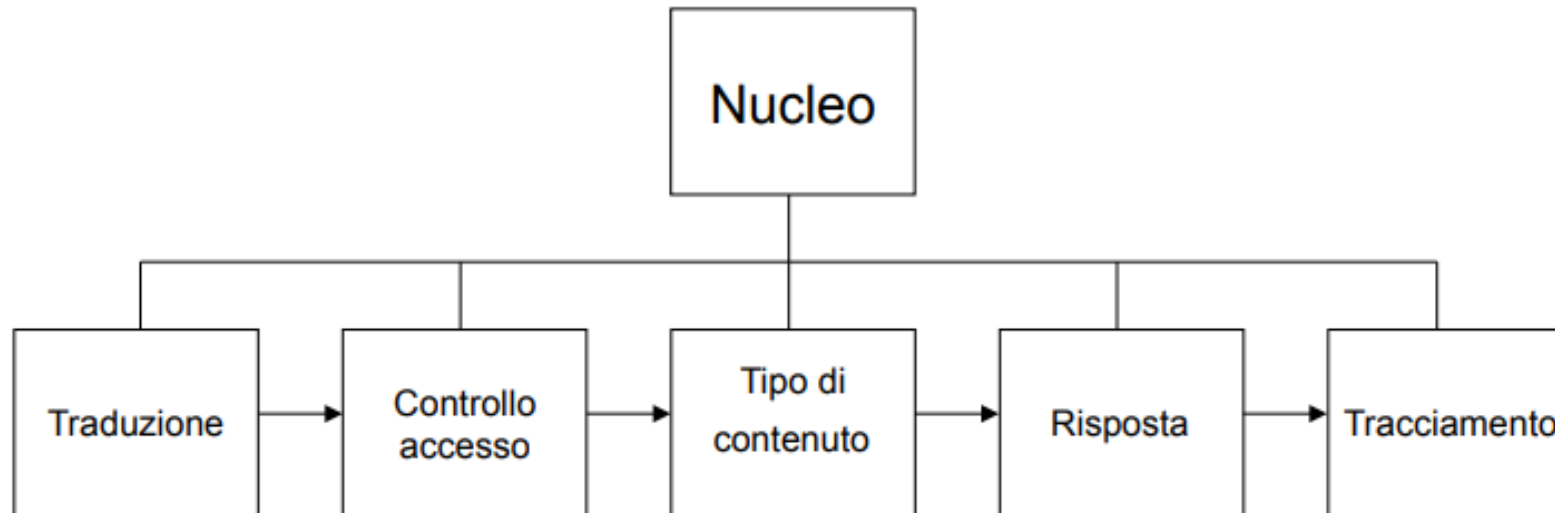
- Componenti opzionali:
 - **username:password** (credenziali autenticazione)
 - **querystring**: è aggiunta all'URL con l'utilizzo del simbolo «?» (la *query string* è una stringa di caratteri che consente di passare al server uno o più parametri, ad es. utilizzando i form).
 - **fragment**: si utilizza il simbolo «#» (indica una parte o una posizione all'interno della risorsa)
- Il numero di porta può essere omesso quando corrisponde alla porta standard (es. **HTTP: 80**, **HTTPS: 443**).

Funzionamento del Browser Web

- L'utente **inserisce l'URL** di una pagina web nel campo location del browser.
- L'URL è spedito al modulo **Interfaccia di rete**. Il modulo, prima analizza la stringa di URL per estrarre l'indirizzo IP a cui connettersi e il nome del file da recuperare; quindi determina il protocollo da usare. Infine **stabilisce la connessione** con la macchina specificata dall'indirizzo estratto.
- Una volta che la connessione è stata stabilita avviene la **richiesta del file** (es. tramite HTTP o HTTPS).
- Ricevuto il file, questo viene inviato al **Parser** (es. un HTML parser). Il parser inizia l'analisi. L'output di questa fase è la decomposizione gerarchica del file HTML in base alla sua struttura, chiamata albero di decomposizione. L'albero è passato al modulo **Layout** che determina la posizione di ogni oggetto presente nel codice HTML e lo visualizza.

Architettura di un Server Web

Elementi principali di un Server Web



Architettura di un Server Web (2)

- **Nucleo:** coordina e gestisce il flusso di dati tra i moduli.
- **Traduzione:** determinazione e localizzazione delle informazioni richieste dal client (browser). Quale file è richiesto? Dove si trova?
- **Controllo accesso:** verifica dell'identità dell'utente e controlla l'accesso in base alle autorizzazioni.
- **Tipo di contenuto:** individuazione degli attributi MIME (*Multipurpose Internet Mail Extensions*) dell'oggetto richiesto: tipo di contenuto, codifica e linguaggio.
- **Risposta:** invio della risposta al client (browser).
- **Tracciamento:** registrazione di dati relativi alla richiesta (*logging*).

Esempio Server Web: Apache

- **Apache HTTP Server** è uno dei più usati Server Web di Internet.
- Apache è un software libero sviluppato dalla **Apache Software Foundation**.
- Altri Web Server molto diffusi sono:
 - **nginx** (<https://nginx.org>): licenza libera
 - **Microsoft Internet Information Services**

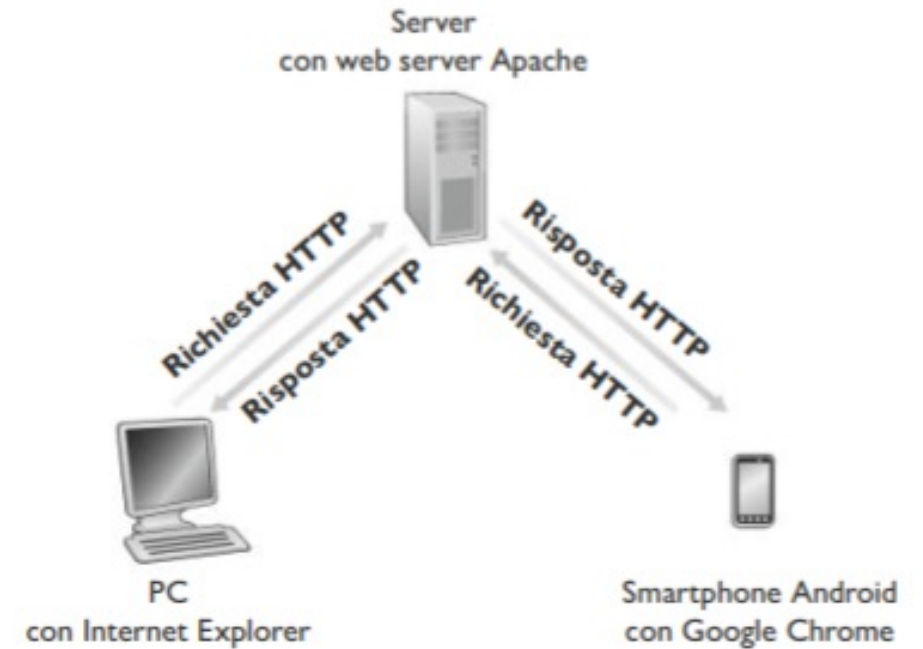


<https://httpd.apache.org/>

Il protocollo HTTP

HTTP

- **HTTP** (*HyperText Transfer Protocol*) è il protocollo a livello di applicazione del Web.
- Il protocollo definisce:
 - la **struttura dei messaggi**
 - la **modalità** con cui client e server si scambiano i messaggi



HTTP (2)

- HTTP utilizza TCP come protocollo di trasporto.
- Il client HTTP per prima cosa inizia una connessione TCP con il server (**handshake a tre vie**).
- Una volta stabilita la connessione, i processi client e server accedono a TCP attraverso le proprie **socket**.
- Ogni messaggio di richiesta HTTP emesso da un processo client arriverà intatto al server e viceversa (**servizio trasferimento dati affidabile**).

HTTP (3)

- HTTP è classificato come **protocollo senza memoria di stato** (*stateless protocol*).
- Il server invia i file richiesti ai client **senza memorizzare alcuna informazione di stato** a proposito del client.
- In caso di ulteriore richiesta dello stesso oggetto da parte dello stesso client, anche nel giro di pochi secondi, il server procederà nuovamente all'invio, non avendo mantenuto alcuna traccia di quello precedentemente effettuato.

Versioni HTTP

- Versione originale di HTTP si chiama **HTTP/1.0** e risale agli inizi degli anni 1990. La versione HTTP/1.0 è obsoleta.
- La maggior parte delle transazioni HTTP avviene oggi sulla versione **HTTP/1.1** (introdotta nel 1997).
- Tuttavia, sempre più browser e web server supportano una nuova versione di HTTP chiamata **HTTP/2** (introdotta nel 2015).

Connessioni persistenti e non persistenti

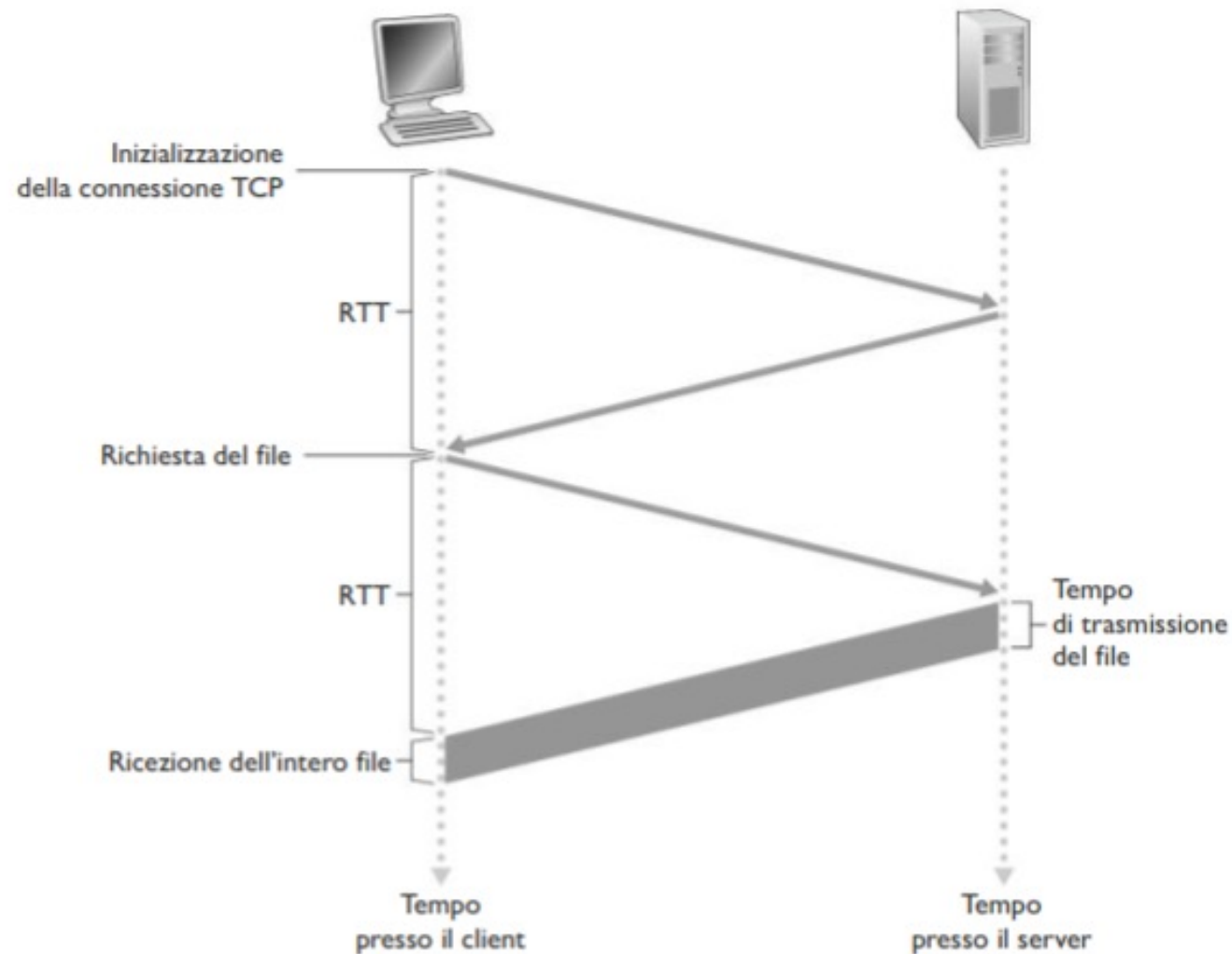
- **Connessioni non persistenti:** ciascuno oggetto (coppia richiesta/risposta) deve essere inviato su una connessione TCP separata.
- **Connessioni persistenti:** ciascuno oggetto (coppia richiesta/risposta) deve essere inviato sulla stessa connessione TCP.

HTTP con connessioni non persistenti

- **Ipotesi:** supponiamo che il client abbia richiesto lo scaricamento di un file base HTML che contiene 10 immagini e che tutti questi 11 oggetti (il file base e le 10 immagini) risiedano sullo stesso server web.
- **Azioni:**
 - Il processo client HTTP inizializza una connessione TCP con il server.
 - Il client HTTP, tramite la propria socket, invia al server un messaggio di richiesta HTTP.
 - Il processo server HTTP riceve il messaggio di richiesta attraverso la propria socket, recupera l'oggetto e lo incapsula in un messaggio di risposta HTTP che viene inviato al client attraverso la socket.
 - Il processo server HTTP comunica a TCP di chiudere la connessione. Questo, però, non termina la connessione finché non sia certo che il client abbia ricevuto integro il messaggio di risposta.
 - Il client HTTP riceve il messaggio di risposta. La connessione TCP termina.
 - Il messaggio indica che l'oggetto incapsulato è un file HTML. Il client estrae il file dal messaggio di risposta, esamina il file HTML e trova i riferimenti ai 10 oggetti JPEG. Vengono quindi ripetuti i primi quattro passi per ciascuno degli oggetti JPEG referenziati.

HTTP/1.0 utilizza connessioni non persistenti

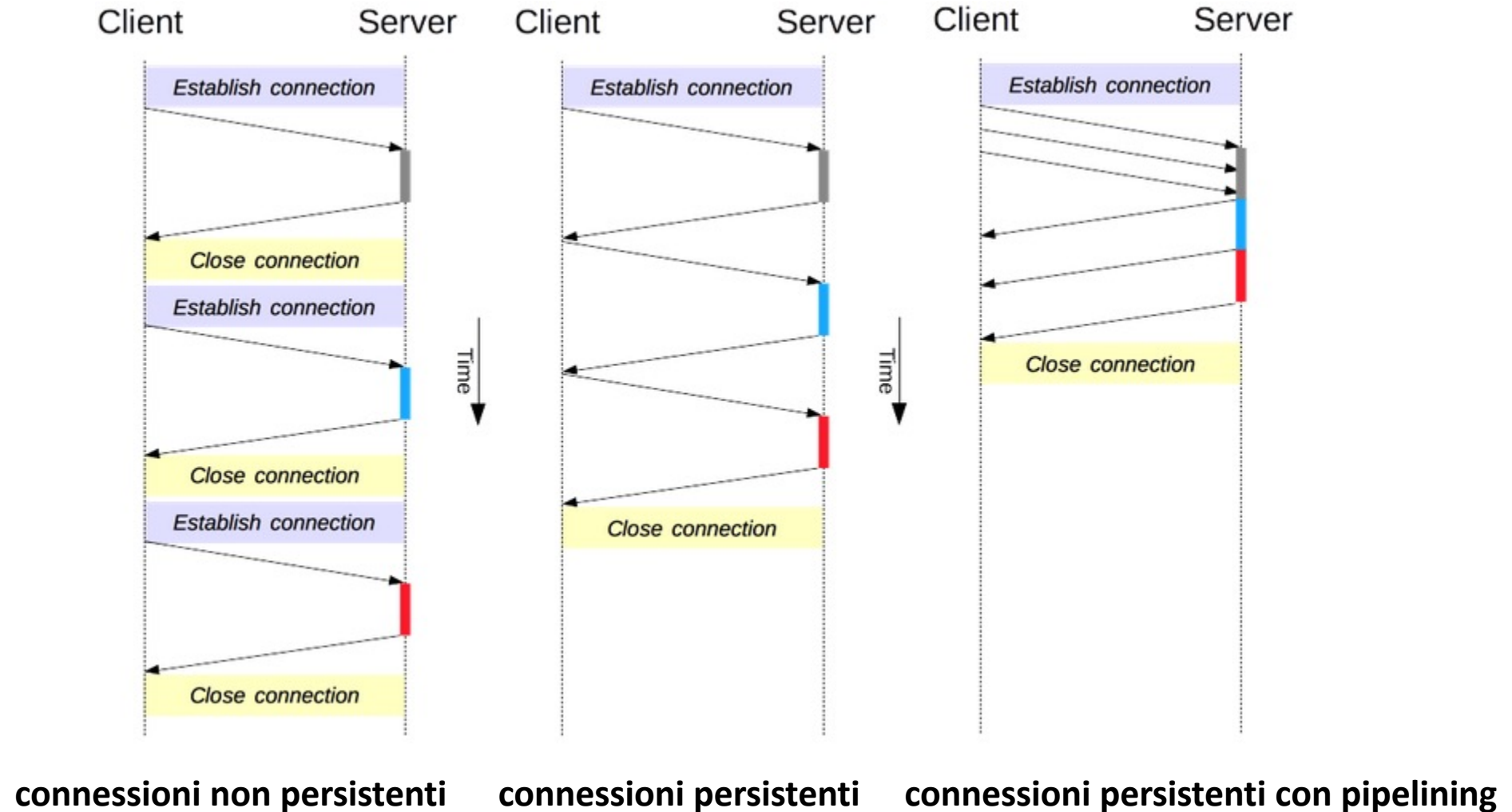
Stima dei tempi di richiesta di una risorsa



HTTP con connessioni persistenti

- Con **HTTP 1.1** nelle **connessioni persistenti** il server lascia la connessione TCP aperta dopo l'invio di una risposta, per cui le richieste e le risposte successive tra gli stessi client e server possono essere trasmesse sulla stessa connessione.
- La modalità di default di HTTP impiega **connessioni persistenti con pipelining** (il client può richiedere più oggetti e il server invia una di seguito all'altra gli oggetti senza aspettare le risposte delle richieste pendenti).

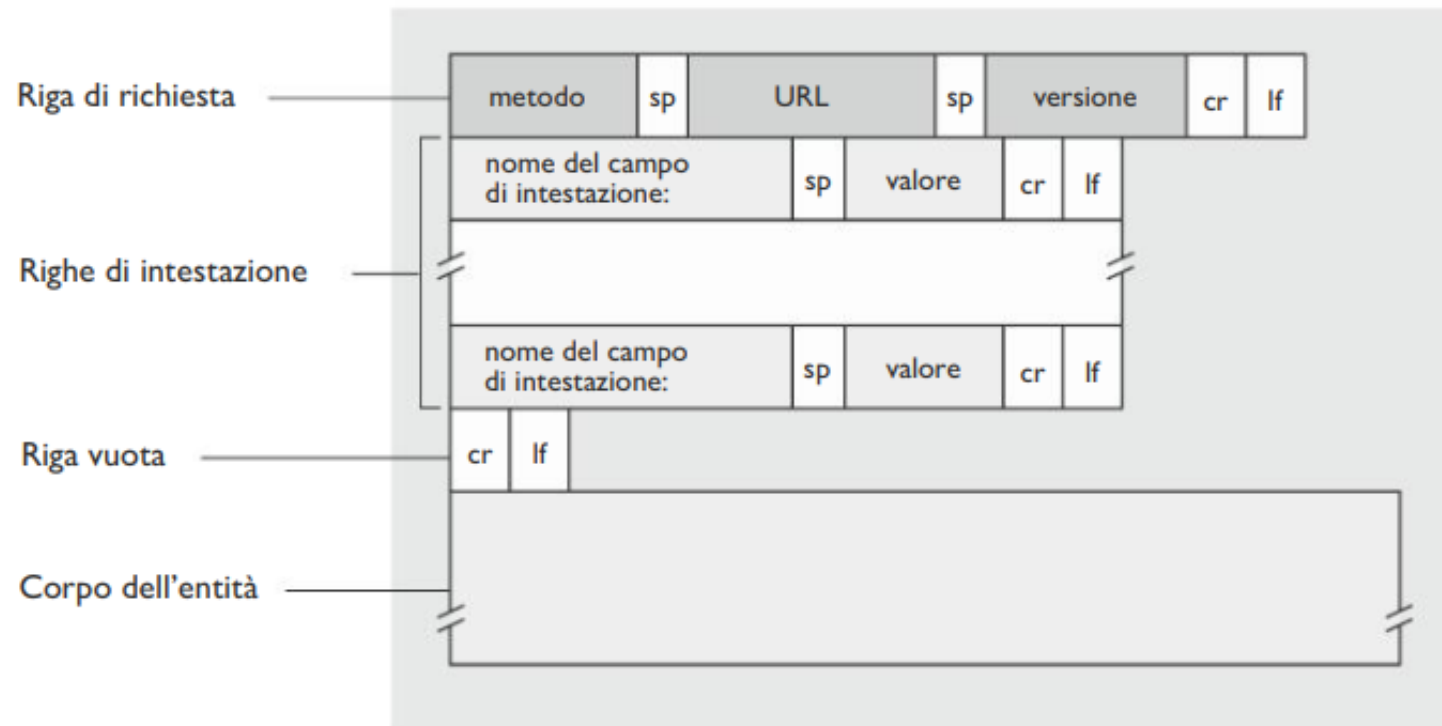
Connessioni persistenti con pipelining



Formato dei messaggi HTTP

- Le specifiche HTTP includono la definizione dei due formati dei messaggi HTTP:
 - **Messaggi HTTP di richiesta**
 - **Messaggi HTTP di risposta**
- Il messaggio è scritto in **testo ASCII** (*American Standard Code for Information Interchange*).
- L'utente quindi è in grado di leggere i messaggi HTTP.

Formato dei messaggi HTTP di richiesta



Formato dei messaggi HTTP di richiesta (2)

- La prima riga è detta **riga di richiesta** (*request line*), quelle successive **righe di intestazione** (*header lines*).
- La riga di richiesta presenta tre campi:
 - il campo metodo
 - il campo URL
 - il campo versione di HTTP
- Il campo metodo può assumere diversi valori, tra cui GET, POST, HEAD, PUT e DELETE. La maggioranza dei messaggi di richiesta HTTP usa **il metodo GET**, adottato quando il browser richiede un oggetto identificato dal campo URL.

Formato dei messaggi HTTP di richiesta (3)

- Esempio:

Richiesta risorsa

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
```

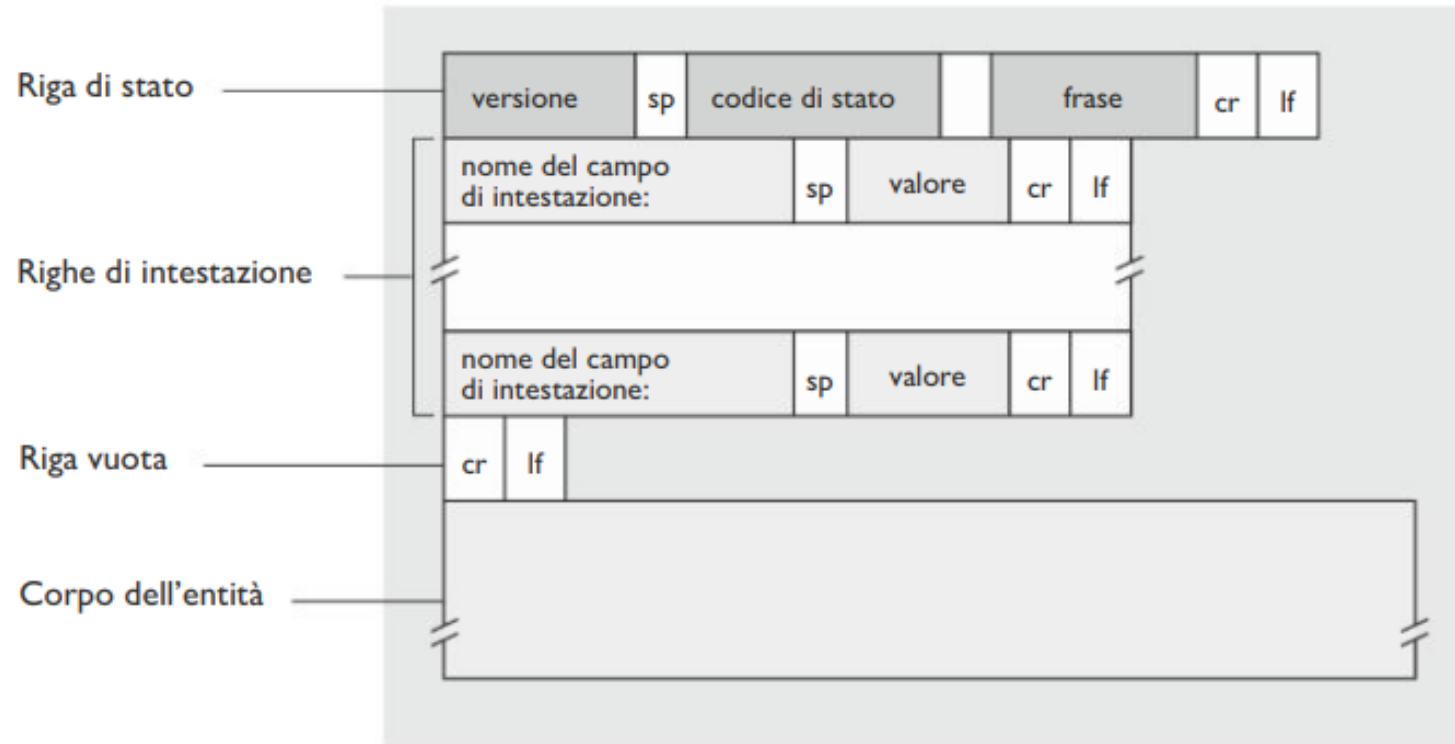
Serve per la cache dei proxy

Il browser sta comunicando al server che non si deve occupare di connessioni persistenti, ma vuole che questi chiuda la connessione dopo aver inviato l'oggetto richiesto.

Formato dei messaggi HTTP di richiesta (4)

- **Corpo dell'entità:** quest'ultimo è vuoto nel caso del **metodo GET**, ma viene utilizzato dal **metodo POST**. Un client HTTP usa in genere il metodo POST quando l'utente **riempie un form**.
- Esempio: quando un utente fornisce le voci da trovare a un motore di ricerca. Nel caso di messaggio POST, l'utente sta ancora richiedendo una pagina web al server, ma i contenuti specifici della pagina dipendono da ciò che l'utente ha immesso nei campi del form. Se il valore del campo metodo è POST, allora il corpo contiene ciò che l'utente ha immesso nei campi del form.

Formato dei messaggi HTTP di risposta

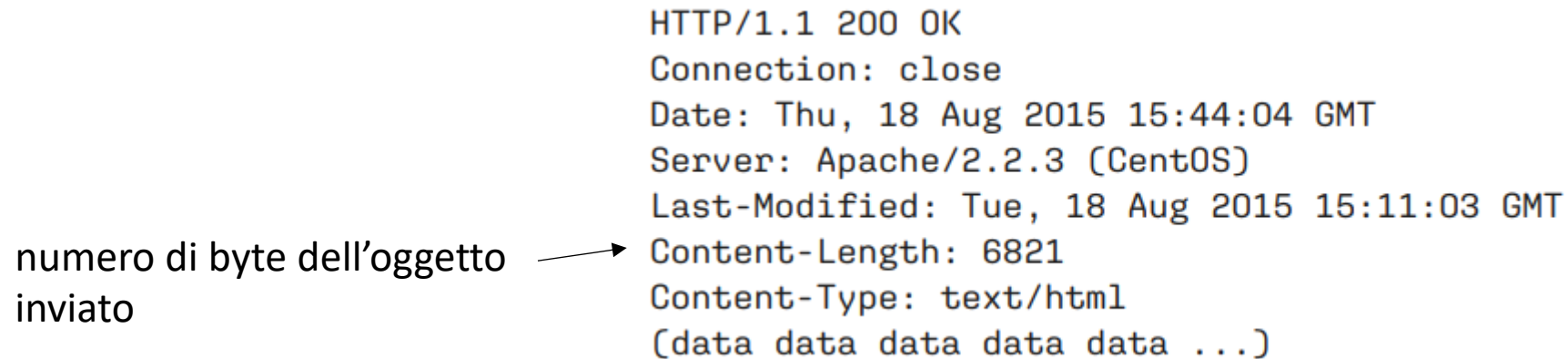


Formato dei messaggi HTTP di risposta (2)

- Esempio:

```
HTTP/1.1 200 OK
Connection: close
Date: Thu, 18 Aug 2015 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 18 Aug 2015 15:11:03 GMT
Content-Length: 6821
Content-Type: text/html
(data data data data data ...)
```

numero di byte dell'oggetto
inviato



contiene l'oggetto richiesto

Formato dei messaggi HTTP di risposta (3)

- Tra i più comuni codici di stato e relative espressioni troviamo:
 - **200 OK**: la richiesta ha avuto successo e in risposta si invia l'informazione.
 - **301 Moved Permanently**: l'oggetto richiesto è stato trasferito in modo permanente; il nuovo URL è specificato nell'intestazione Location del messaggio di risposta. Il client recupererà automaticamente il nuovo URL.
 - **400 Bad Request**: si tratta di un codice di errore generico che indica che la richiesta non è stata compresa dal server.
 - **404 Not Found**: il documento richiesto non esiste sul server.
 - **505 HTTP Version Not Supported**: il server non dispone della versione di protocollo HTTP richiesta.

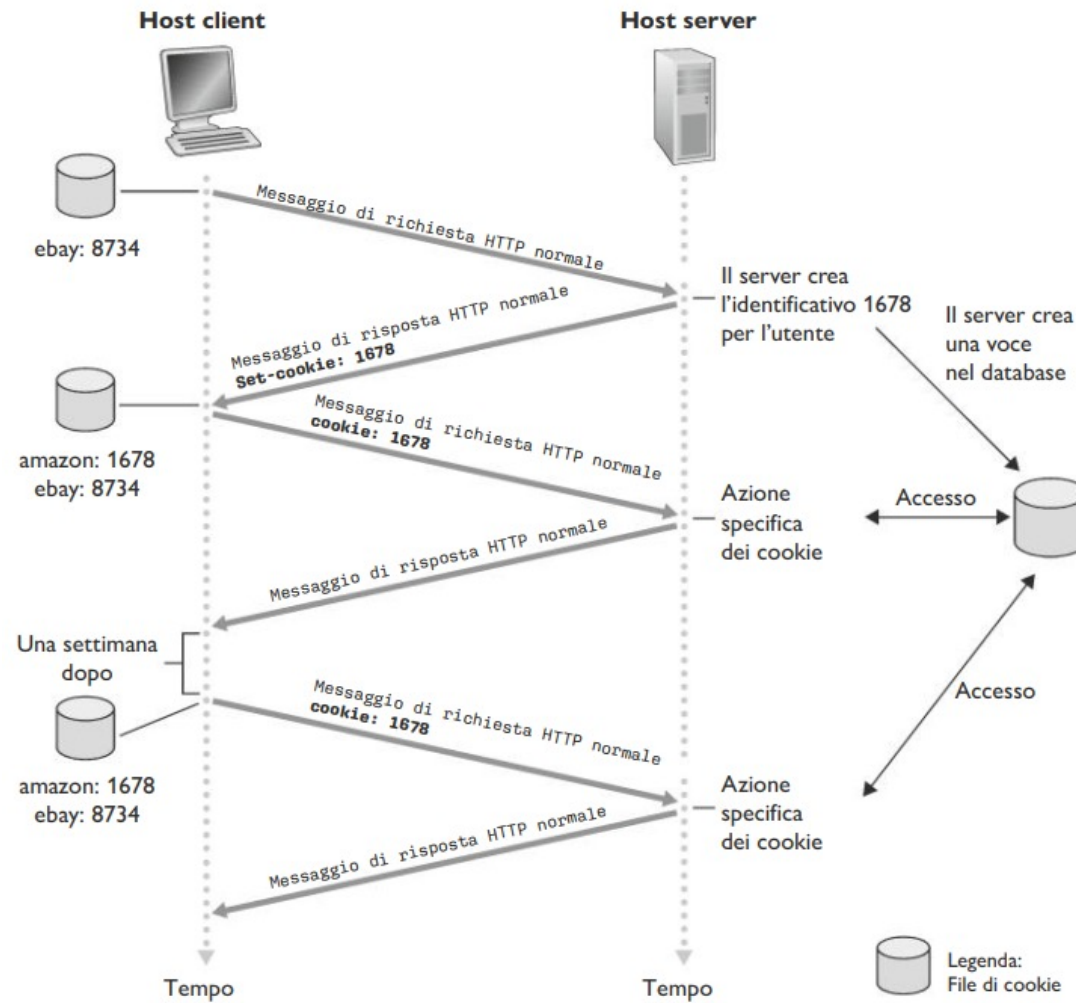
Interazione utente-server: i cookie

- HTTP adotta i **cookie** per **autenticare gli utenti**, sia per limitare l'accesso da parte di questi ultimi sia per fornire contenuti in funzione della loro identità, visto che i server HTTP sono privi di stato.
- La tecnologia dei cookie presenta quattro componenti:
 - una riga di intestazione nel messaggio di risposta HTTP
 - una riga di intestazione nel messaggio di richiesta HTTP
 - un file mantenuto sul sistema dell'utente e gestito dal browser
 - un database sul sito

Funzionamento dei cookie

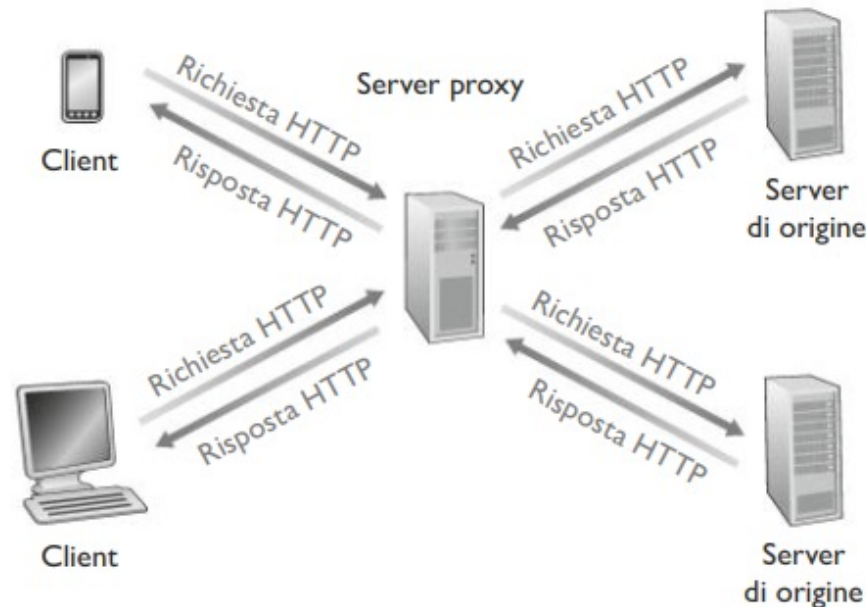
- L'utente A accede al sito S (es. Amazon) per la prima volta (S usa i cookie).
- Quando la richiesta di A arriva al server, il sito S crea un numero di identificazione **ID-x** univoco e crea nel suo database locale una nuova voce indicizzata da ID-x.
- Il server risponde ad A introducendo nella risposta HTTP una intestazione del tipo: **Set cookie: ID-x**.
- Quando il browser di A riceve la risposta crea una nuova voce nel file cookie da esso gestito. Questa nuova voce contiene il nome (*hostname*) del server e il numero di identificazione ricevuto (ID-x).
- Ogni volta che A navigando nel sito esegue nuove richieste il browser consulta il file cookie estrae il numero di identificazione per il sito S e lo mette nella richiesta HTTP in una linea di intestazione del tipo: **Cookie: ID-x**.
- In questo modo il server è in grado di **tracciare l'attività di A nel sito S**.

Funzionamento dei cookie (2)

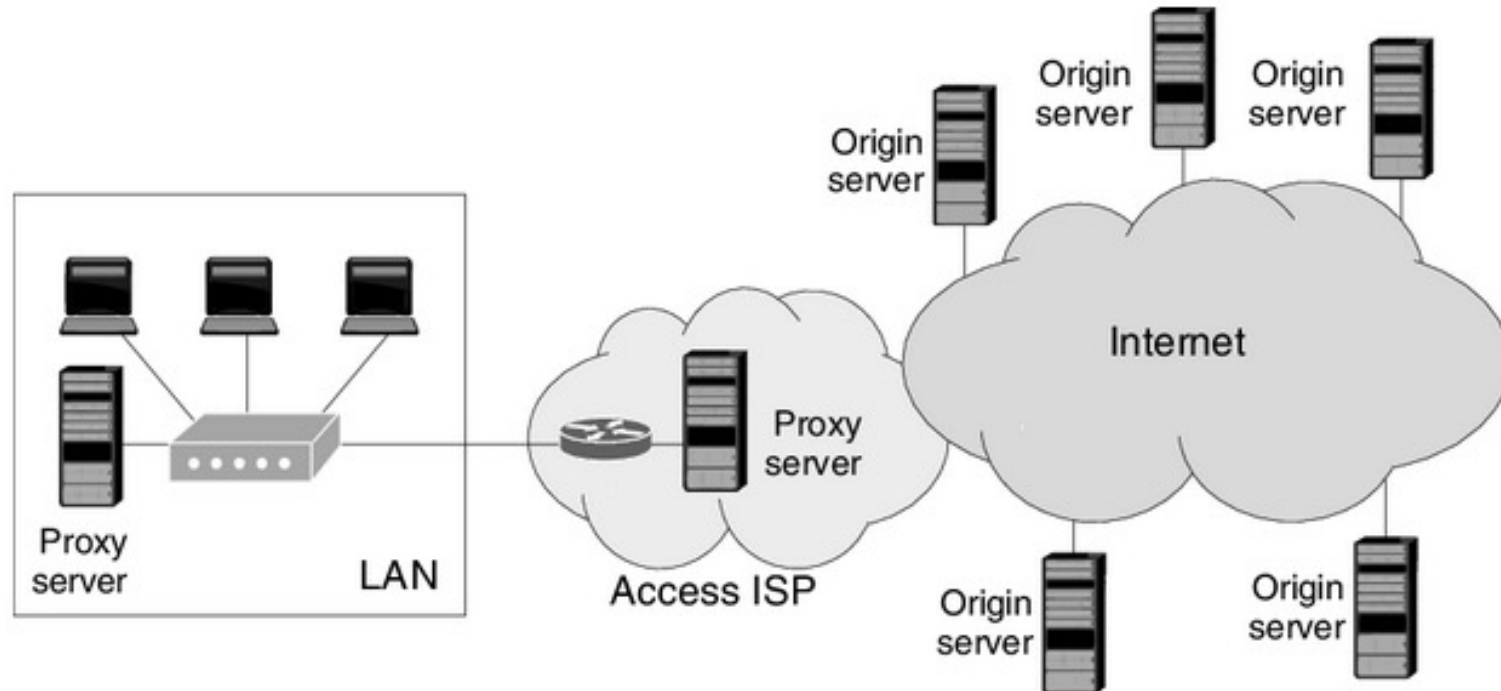


Web caching

- Una **web cache**, nota anche come **server proxy**, è un'entità di rete che soddisfa richieste HTTP al posto del web server effettivo.
- Il proxy ha una propria memoria su disco (una cache) in cui conserva copie di oggetti recentemente richiesti.



Web caching (2)



Web caching (3)

- **Vantaggi:**

- Un proxy può ridurre in modo sostanziale i tempi di risposta alle richieste dei client, in particolare se l'ampiezza di banda che costituisce il collo di bottiglia tra il client e il server di origine è molto inferiore rispetto all'ampiezza di banda minima tra client e proxy.
- I proxy possono ridurre sostanzialmente il traffico sul collegamento di accesso a Internet, con il vantaggio di non dover aumentare l'ampiezza di banda frequentemente e ottenere quindi una riduzione dei costi.
- Inoltre i proxy possono ridurre in modo sostanziale il traffico globale del Web in Internet, migliorando di conseguenza le prestazioni di tutte le applicazioni.

GET condizionale

- Sebbene il web caching riduca i tempi di risposta percepiti dall'utente, introduce un nuovo **problema**: la copia di un oggetto che risiede in cache potrebbe essere scaduta.
- Il **GET condizionale** è un meccanismo dell'HTTP che permette alla cache di verificare se i suoi oggetti sono aggiornati.
- Un messaggio di richiesta HTTP viene detto messaggio di GET condizionale se:
 - usa il metodo GET
 - include una riga di intestazione **If-modified-since**:

GET condizionale (2)

- Esempio: Il proxy invia al server web la richiesta del client

```
GET /fruit/kiwi.gif HTTP/1.1  
Host: www.exotiquecuisine.com
```

Il server web invia l'oggetto al proxy

```
HTTP/1.1 200 OK  
Date: Sat, 3 Oct 2015 15:39:29  
Server: Apache/1.3.0 (Unix)  
Last-Modified: Wed, 9 Sep 2015 09:23:24  
Content-Type: image/gif  
(data data data data data ...)
```

Il proxy inoltra l'oggetto al browser richiedente e pone anche l'oggetto nella cache locale

GET condizionale (3)

- Esempio: Un nuovo client richiede l'oggetto dopo un certo tempo. Il proxy effettua un controllo di aggiornamento inviando un GET condizionale.

```
GET /fruit/kiwi.gif HTTP/1.1
Host: www.exotiquecuisine.com
If-modified-since: Wed, 9 Sep 2015 09:23:24
```

Il server web risponde

```
HTTP/1.1 304 Not Modified
Date: Sat, 10 Oct 2015 15:39:29
Server: Apache/1.3.0 (Unix)
[corpo vuoto]
```

Non ci sono modifiche
l'oggetto nella cache del
proxy è quindi aggiornato.

HTTP/2

- Gli **obiettivi principali di HTTP/2** sono quello di:
 - Ridurre la latenza percepita attivando il **multiplexing di richiesta** e risposta su una singola connessione TCP
 - Fornire supporto per la **priorità delle richieste** e il **server push**
 - Fornire una compressione efficiente dei campi di intestazione HTTP
- Inoltre elimina il problema dell'**Head of Line (HOL) blocking** (“blocco in testa alla coda”) dell'HTTP/1.1.

HTTP/2 (2)

- **HOL blocking:** oggetti di grandi dimensioni caricate in testa ad una pagine web, creano un collo di bottiglia per oggetti più piccoli caricati più in basso.
- HTTP/1.1 attraverso TCP può arginare il problema creando più connessioni in parallelo. Questo però fa accrescere notevolmente il numero di connessioni al server web.
- Il **framing HTTP/2** è una tecnica implementata nel protocollo per limitare l'HOL blocking.

Framing HTTP/2

- La soluzione HTTP/2 del blocco HOL consiste nel suddividere ogni messaggio in **piccoli frame** e alternare i messaggi di richiesta e risposta sulla stessa connessione TCP.
- I frame, man mano che arrivano al client, vengono prima ricostruiti nei messaggi di risposta originali nel sottolivello di framing e quindi elaborati dal browser.
- In questo modo gli oggetti più piccoli presenti in coda ad una pagina web arrivano comunque a destinazione, senza alterare il ritardo percepito dall'utente.

Priorità dei messaggi e server pushing

- Il meccanismo di **priorità dei messaggi** consente agli sviluppatori di personalizzare la priorità relativa delle richieste per ottimizzare le prestazioni dell'applicazione. Il server può inviare prima i frame per le risposte con la priorità più alta.
- Il **server push** è la capacità di un server di inviare più risposte per una singola richiesta del cliente. Oltre alla risposta alla richiesta originale, il server può effettuare un invio push al client di oggetti aggiuntivi, senza che il client debba richiedere ognuno di essi.

Il protocollo HTTPS

HTTPS

- **HTTPS** (*HyperText Transfer Protocol over Secure Socket Layer*): la comunicazione tra browser web e server web avviene su **Transport Layer Security (TLS)** (il suo predecessore era Secure Sockets Layer (SSL)).
- **TLS**: **protocolli crittografici** che permettono una comunicazione sicura dalla sorgente al destinatario su reti TCP/IP.
- Servizi importanti di sicurezza:
 - La riservatezza dei dati
 - L'integrità dei dati
 - L'autenticazione end-to-end

HTTPS (2)

- **Riservatezza:** solo mittente e destinatario devono comprendere il contenuto del messaggio (si usa crittografia).
- **Integrità del messaggio:** mittente e destinatario devono essere sicuri che il contenuto non subisca alterazioni durante la trasmissione (per cause fortuite o per manipolazioni).
- **Autenticazione:** mittente e destinatario devono essere sicuri della loro identità.

Crittografia

- **A chiave simmetrica:** i due host utilizzano la stessa chiave
- Esempio:
 - La chiave è un pattern di sostituzione monoalfabetico (sostituzione di una lettera con un'altra)

Lettere in chiaro:	abcdefghijklmnopqrstuvwxyz
	↓ ↓
Lettere cifrate:	mnbvcxzasdfghjklpoiuytrewq

Testo in chiaro: bob. i love you. alice

Testo cifrato: nkn. s gktc wky. mgsbc

Crittografia (2)

- **A chiave pubblica:** si hanno due chiavi (la chiave pubblica, che deve essere distribuita e la chiave privata, personale e segreta)

