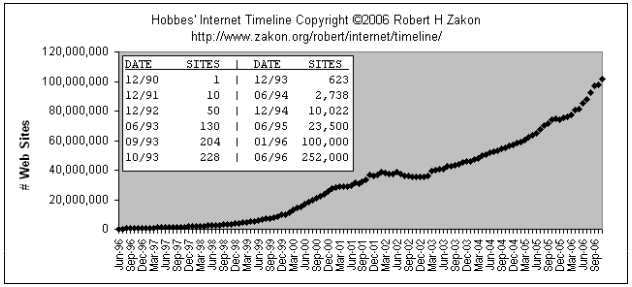
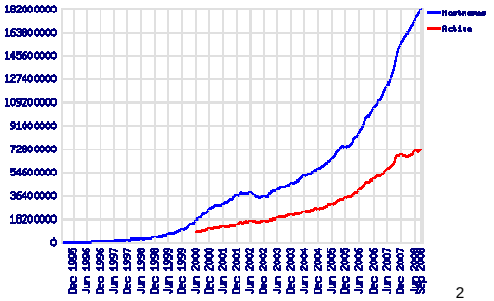
World Wide Web: introduzione e componenti

I segnali del successo del Web



Dal 2007 aumento esponenziale del numero di siti presso fornitori di servizi di blogging e social networking (MySpace, Live Spaces, Blogger, ..)

Fonte: Netcraft Web Server Survey (http://news.netcraft.com/archives/web_server_survey.html)



I segnali del successo del Web (2)

- Fino all'introduzione dei sistemi P2P, il Web è stata l'applicazione killer di Internet (75% del traffico di Internet nel 1998)

Event	*Period*	*Peak day*	*Peak minute*
NCSA server (Oct. 1995)	-	2 Million	-
Olympic Summer Games (Aug. 1996)	192 Million (17 days)	8 Million	-
Nasa Pathfinder (July 1997)	942 Million (14 days)	40 Million	-
Olympic Winter Games (Feb. 1998)	634.7 Million (16 days)	55 Million	110,000
Wimbledon (July 1998)	-	-	145,000
FIFA World Cup (July 1998)	1,350 Million (84 days)	73 Million	209,000
Wimbledon (July 1999)	-	125 Million	430,000
Wimbledon (July 2000)	-	282 Million	964,000
Olympic Summer Games (Sept. 2000)	-	875 Million	1,200,000

[Carico misurato in contatti]

- Inoltre: google.com, msn.com, yahoo.com (> 200 milioni di contatti al giorno)

I motivi alla base del successo del Web

- Digitalizzazione dell'informazione
  - Qualsiasi informazione rappresentabile in binario come sequenza di 0 e 1
- Diffusione di Internet (dagli anni 1970)
  - Trasporto dell'informazione ovunque, in tempi rapidissimi e a costi bassissimi
- Diffusione dei PC (dagli anni 1980)
  - Accesso, memorizzazione ed elaborazione dell'informazione da parte di chiunque a costi bassissimi
- Semplicità di utilizzo e trasparenza dell'allocazione delle risorse
  - Uso di interfacce grafiche

Una definizione di World Wide Web

- "L'universo dell'informazione globale accessibile tramite rete" (T. Bernars-Lee, l'inventore del WWW o Web)
- Il Web è un sistema *ipermediale*, *distribuito globalmente* che supporta accessi interattivi a risorse e servizi
  - ipermediale: diverse forme di rappresentazione delle risorse (testo, immagini, audio, video, …) tra loro collegate
    - *ipertesto*: sistema non lineare per la strutturazione di informazioni
  - distribuito globalmente: risorse distribuite e scalate su l'intera Internet
    - La ragnatela (web) di collegamenti è di ampiezza mondiale (world-wide)
    - risorse Web amministrate indipendentemente → possibile perdita di consistenza nel tempo

Una definizione di World Wide Web (2)

- Il Web è un sistema *aperto*
  - Può essere esteso e implementato con nuove modalità senza alterare le sue funzionalità esistenti
  - Il suo funzionamento è basato su standard di comunicazione e di documenti che sono pubblicamente disponibili ed ampiamente implementati
- Il Web è un sistema aperto rispetto al tipo di risorse che possono essere pubblicate e condivise
- Aspetto più interessante per gli utenti: a differenza di altri mezzi informativi, il Web è *on demand*
- Il Web non è sinonimo di Internet, ma rappresenta un'applicazione dell'infrastruttura Internet!

Un po' di storia: ipertesti

- L'esigenza di evidenziare visivamente riferimenti e connessioni implicite od esplicite tra testi esiste da sempre
- 1945: sistema Memex proposto da V. Bush
  - Sistema elettromeccanico basato su microfilm per la memorizzazione e interconnessione di tutte le carte, libri ed informazioni utili per la vita d'ufficio
- 1965: termine ipertesto coniato da T. Nelson
  - Progettazione del sistema integrato Xanadu per la gestione della letteratura, ovvero dei contenuti, dei riferimenti espliciti ed impliciti, e del processo che sostiene la produzione letteraria (creazione, pubblicazione, modello economico)
  - Di ispirazione per T. Berners-Lee

Un po' di storia: ipertesti (2)

- 1968: D. Engelbart iniziò a lavorare sul concetto di personal computing
  - Realizzazione di un sistema di video-conferenza, editing di testi gerarchici ed ipertestuali e di supporto per il lavoro cooperativo dotato di interfaccia a finestre, mouse e altri meccanismi di input ed output rivoluzionari per l'epoca
- Anni 1980: con il progredire della tecnologia, i sistemi ipertestuali escono dai laboratori di pochi studiosi e diventano un argomento ufficiale di ricerca e produzione software
  - 1987: Hypercard, prodotto dalla Apple, un software di produttività personale

Un po' di storia: WWW

- 1989: un gruppo di ricercatori informatici del CERN di Ginevra (tra cui T. Berners-Lee e R. Cailliau) furono incaricati di realizzare un meccanismo di collaborazione scientifica in un contesto internazionale
  - Progetto di integrazione in forma ipertestuale delle risorse esistenti in Internet
  - Internet, ipertesti e SGML come elementi chiave
- 1991: il gruppo mostrò (con scarso successo) il primo prototipo dell'applicazione realizzata secondo il paradigma client-server: World-Wide Web
- Prototipo composto di:
  - un server che spediva risorse memorizzate localmente a chiunque lo richiedesse secondo il protocollo stabilito, e che memorizzava risorse spedite da remoto, senza autorizzazione o verifica
  - un editor di testi parzialmente WYSIWYG che permetteva di visualizzare documenti ipertestuali e di modificarli, creando link e blocchi di testo

Un po' di storia: WWW (2)

- I principali competitor del Web su Internet erano:
  - FTP: protocollo di scambio di file (no visualizzazione)
  - WAIS: server con notevoli potenzialità di ricerca su documenti di solo testo
  - Gopher: meccanismo di organizzazione di documenti di testo in gerarchie distribuite su più server
- 1992: l'NCSA (National Centre for Supercomputing Applications) esaminò il prototipo di WWW e decise di realizzarne una versione propria
- Con la realizzazione del server NCSA e del primo browser Web chiamato Mosaic, l'NCSA decretò l'inizio del successo esplosivo del Web
- 1993: fondata la Mosaic Corporation (poi rinominata Netscape Corporation)
- 1994: successo immediato di Netscape Navigator

Un po' di storia: WWW (3)

- Berners-Lee e Cailliau fondarono il W3C per mantenere il controllo sull'evoluzione del Web
- Microsoft, dopo una falsa partenza, realizzò un browser Web (Internet Explorer) ed un server Web (Microsoft Internet Information Server)
- 1997: nascita di XML; caso unico nella storia dell'informatica, sembra accontentare tutti (fornitori di contenuti, sviluppatori di applicazioni Web, progettisti di DBMS, …)
- 1998: Netscape rilasciò il codice sorgente di Navigator
  - Progetto mozilla.org per sfruttare gli stessi principi di organizzazione della comunità Linux per un browser freeware
- 2000: esplosione della bolla della new-economy
  - Bloccati molti sviluppi interessanti ed innovativi
  - Sopravvivono le grandi aziende, o quelli che fanno applicazioni più tradizionali, anche su Web: gestionali, integrazione di sistemi informativi, applicazioni di database

Evoluzione del Web

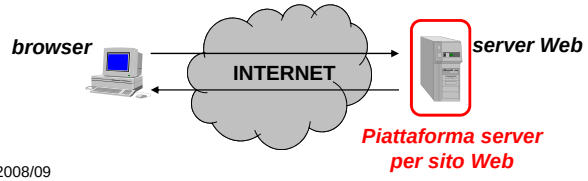
- Una classificazione "ufficiale" dell'evoluzione del Web
  - Web 1.0
  - Web 1.5
  - Web 2.0
- Una classificazione "alternativa" dell'evoluzione del Web
  - Prima generazione
  - Seconda generazione
  - Terza generazione
  - Servizi Web e Web semantico
  - Non solo evoluzione delle applicazioni ma anche prestazioni

Da Web 1.0 a Web 2.0

- Non è un'evoluzione di tipo tecnologico ma sociale
  - Approccio con il quale gli utenti si rivolgono al Web
- Web 1.0: modello di interazione statico
  - Semplice consultazione e fruizione di contenuti statici
- Web 1.5: modello di interazione dinamico
  - Consultazione e fruizione di contenuti dinamici
  - Tecnologie per la generazione di contenuti dinamici client-side e server-side
- Web 2.0: applicazioni online che permettono uno spiccato livello di interazione sito-utente
  - Termine coniato da T. O'Reilly e D. Dougherty nel 2004
  - Blog, forum, chat, sistemi quali Wikipedia, YouTube, Flickr, Facebook, Myspace, Gmail, …
  - Fruizione e creazione/modifica di contenuti multimediali

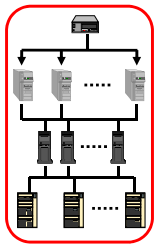
La prima generazione del Web

- Prima generazione (*ieri*) = Web publishing
  - Un ulteriore canale di comunicazione
  - 95% dell'informazione costituita da testo ed immagini
  - Siti Web prevalentemente **statici**, con alcune tecnologie (ad es. CGI) per la generazione di contenuti dinamici
  - Manutenzione ed aggiornamenti occasionali
  - Prestazioni molto variabili
  - Affidabilità non garantita
  - Sicurezza non indispensabile



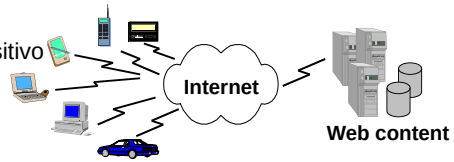
La seconda generazione del Web

- Seconda generazione (*ieri-oggi*) = Web-based information systems
  - Canale di informazione critica, divenuto un mezzo di informazione privilegiato per molti utenti
  - "Vetrina" importante per industrie e organizzazioni
  - Contenuti **dinamici** ed attivi in continuo aumento
  - Servizi di streaming audio e video
  - Servizi personalizzati, servizi a pagamento (diretto o indiretto)
  - Interfaccia di accesso per molti altri servizi informatici usufruiti via rete (anche se non propriamente servizi di rete)
    - E-mail, trasferimento di file
    - Accesso ad archivi, basi di dati, banche dati, …
  - Necessaria la **qualità di servizio** (QoS)
    - Prestazioni garantite (*regola degli X secondi*)
      - X=8, poi X=4, …
    - Affidabilità come capacità di tollerare guasti
    - Sicurezza



La terza generazione del Web

- Terza generazione (*oggi-domani*) = Ubiquitous Web
  - Web for Everyone and Web on Everything
  - Possibilità di usufruire di tutti i servizi Web della seconda generazione in modalità AAA
    - *Anytime:* sempre (24/7)
    - *Anywhere*: ovunque
    - *Anyway*: da qualunque dispositivo



  - In più: servizi time-aware, location-aware, device-aware e personalizzazione dei servizi
  - Prima fase:
    - Si accettano prestazioni ed affidabilità *variabili*
  - Seconda fase:
    - Prestazioni ed affidabilità *garantiti*
  - La sicurezza è comunque un requisito indispensabile

Ulteriori evoluzioni del Web

- Servizi Web (Web services):
  - Il Web come infrastruttura per la comunicazione e l'interazione tra applicazioni distribuite
  - Sistemi eterogenei possono lavorare insieme per realizzare il service oriented computing (SOC)
    - Programmazione con componenti distribuite sul Web
- Web semantico (Semantic Web):
  - Non più solo documenti, ma informazioni e dati relativi ai documenti stessi (metadati) in un formato adatto all'interrogazione, interpretazione e, più in generale, all'elaborazione automatica
  - Tale possibilità trasformerà il Web da *machine-readable* a *machine-understandable*, permettendo la nascita di applicazioni sofisticate in grado di interpretare ed elaborare dati in maniera semi-intelligente

Organismi di standardizzazione per il Web

- Internet Engineering Task Force (IETF, http://www.ietf.org/)
  - Realizzazione e discussione degli standard in ambito Internet
  - Pubblicazione delle specifiche approvate in forma di Request for Comments (RFC, http://www.rfc-editor.org/)
- World Wide Web Consortium (W3C, http://www.w3.org/)
  - Organizzazione fondata e diretta da T. Berners-Lee al fine di "sviluppare protocolli comuni per migliorare l'interoperabilità e guidare l'evoluzione del World Wide Web"
  - Produzione di specificazioni di interoperabilità e codice d'esempio
  - Pubblicazione di technical report (http://www.w3.org/TR/): W3C Note e Recommendation Track

Stack tecnologico del Web (secondo il W3C)

Fonte: http://www.w3.org/Consortium/technology

Ingredienti basilari del Web

- Informazione digitalizzata
- Architettura client-server
  - Componenti, funzionamento ed architettura del Web client (browser)
  - Componenti, funzionamento ed architettura del Web server
- Meccanismi di comunicazione e naming delle risorse (a livello di nodi) di Internet
  - Protocollo TCP/IP e Domain Name System (DNS)
- Tre principali componenti tecnologici standard alla base del Web
  - **URI**: meccanismo di naming universale per identificare le risorse
  - **HTTP**: protocollo per il trasferimento delle risorse
  - **HTML**: linguaggio di markup ipertestuale

Architettura client-server

- Browser (client Web)
  - Agente dell'utente per il Web
  - Guida l'interazione client/server nei confronti di un server per volta
  - Visualizza la pagina Web richiesta dall'utente; fornisce molte caratteristiche per la navigazione e configurazione
  - Implementa il client nel protocollo HTTP
- Server Web
  - Fornisce su richiesta le risorse Web
    - Memorizzate su disco o generate dinamicamente
  - Implementa il server nel protocollo HTTP

Documenti, risorse, pagine

- Documento: un file di dati
  - Testo o immagine o video o suono o PDF, …
  - Ogni documento ha un tipo
  - Alcuni tipi hanno funzionalità di ipertesto: HTML, PDF, …
- Risorsa: sinonimo di documento
- Pagina: cosa l'utente visualizza sullo schermo
  - Testo e immagini, video e suono, …
  - Quindi la pagina è un documento ipermediale composto da molteplici risorse
  - Ogni risorsa è identificabile univocamente
- Richiesta di pagina Web
  - Una singola richiesta da parte dell'utente, che tipicamente consiste di molteplici richieste (*hit*) inviate dal browser dell'utente al server Web per reperire le diverse risorse che compongono la pagina Web
  - *Hit*: una richiesta per una risorsa effettuata dal client al server Web
- Attenzione: questi termini non sono standardizzati

Meccanismi di naming delle risorse

- L'insieme di tutti i meccanismi standard di naming delle risorse (fisiche o astratte) è detto Uniform Resource Identifier (URI)
  - Riferimento: RFC 3986 (Gen. 2005)
- Le risorse sono considerate accessibili tramite l'utilizzo di protocolli esistenti, inventati appositamente, o ancora da inventare
- Gli URI si orientano a risolvere il problema di creare un meccanismo ed una ***sintassi di accesso unificata*** alle risorse disponibili
- Tutte le istruzioni d'accesso alle varie specifiche risorse disponibili secondo un dato schema di accesso sono codificate come una stringa di indirizzo

Meccanismi di naming delle risorse (2)

- Gli URI sono, per definizione:
  - *Universal Resource Locator* (URL): una sintassi che, oltre ad identificare la risorsa, fornisce informazioni immediatamente utilizzabili per accedere alla risorsa (es. la locazione di rete della risorsa)
    - Sottoinsieme di URI
    - Esempio: foo://example.com:8042/over/there?name=ferret#nose
  - *Universal Resource Names* (URN): una sintassi che permette un'etichettatura permanente e non ripudiabile della risorsa, indipendentemente dal riportare informazioni sull'accesso
    - Sottoinsieme di URI
    - Esempio: urn:example:animal:ferret:nose

URL

schema://host[:port][/path][?query][#fragment]

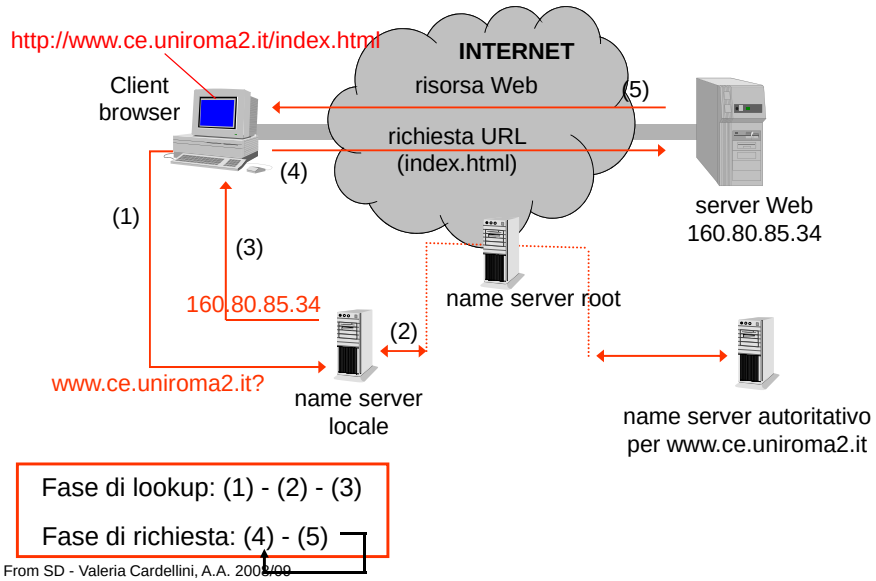
- Esempi:
  - http://web.uniroma2.it/
  - http://www.ce.uniroma2.it/courses/sd0809/#materiale
  - http://www.google.it/search?hl=it&q=url&meta=
- schema di naming (in pratica, il protocollo):
  - Indica il servizio con cui accedere alla risorsa, ossia quale protocollo usare per interagire con il server che controlla la risorsa; il protocollo di accesso più comune è http
- host:
  - Identifica la locazione della risorsa sulla rete, ovvero il nome o l'indirizzo IP del server sul quale risiede la risorsa e, opzionale, la porta del servizio (http: per default 80)
- path:
  - Identifica la risorsa presso il server

URL (2)

schema://host[:port][/path][?query][#fragment]

- query:
  - Per inviare informazioni al server, ad esempio una query ad un motore di ricerca
- fragment
  - Identifica un componente della risorsa
- URL assoluto e URL relativo

Richiesta per una risorsa Web



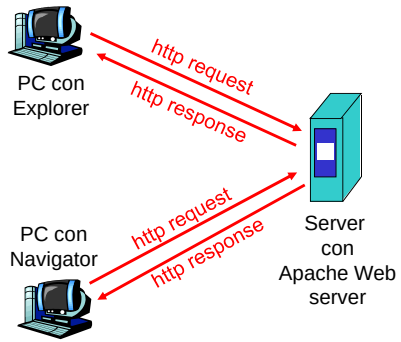
Protocollo HTTP

Tra i componenti standard basilari del Web vi è un protocollo per il trasferimento di informazioni: ***HyperText Transfer Protocol*** (***HTTP***)

- Riferimenti: RFC2616 (HTTP/1.1), RFC1945 (HTTP/1.0)

HTTP:

- protocollo di livello applicativo nativo del Web
- basato sul paradigma client/server
  - *Client HTTP:* browser che richiede, riceve, "visualizza" risorse Web
  - *Server HTTP:* Web server che invia risorse in risposta a richieste



Caratteristiche del protocollo HTTP

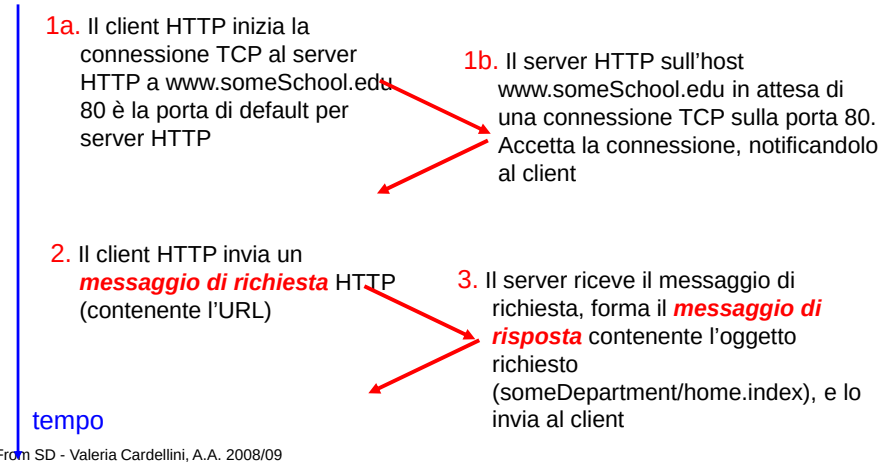
- HTTP è un protocollo di richiesta/risposta
  - Il client invia un messaggio di richiesta, in conseguenza del quale il server risponde con un messaggio di risposta
- HTTP definisce la sintassi dei messaggi ed il modo in cui dovrebbero essere interpretati i campi in ciascuna linea del messaggio
- HTTP 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 di richiesta e risposta HTTP scambiati tra browser e server Web
  - Chiusura della connessione TCP
- TCP offre un servizio di trasferimento affidabile: i messaggi di richiesta/risposta sono consegnati integri al destinatario
- HTTP potrebbe usare anche altri protocolli diversi da TCP, ma esistono solo alcuni tentativi sperimentali

Caratteristiche del protocollo HTTP (2)

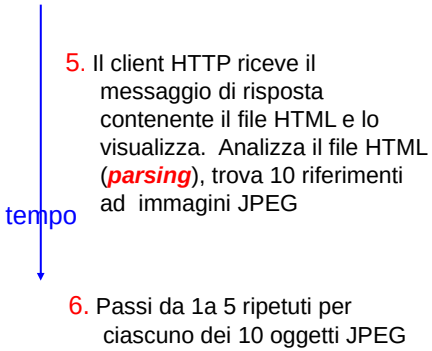
- HTTP è un protocollo stateless (senza stato)
  - Il server non mantiene nessuna informazione tra una richiesta di un client e la successiva
- I protocolli che conservano lo stato sono complessi!
  - La storia passata (lo stato) deve essere memorizzata
  - Se il server/client subiscono un crash, la vista dello stato può essere inconsistente e deve essere ristabilita

Esempio di base HTTP/1.0

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



Esempio di base HTTP/1.0



Linguaggio di markup

- Linguaggio di markup: descrive i meccanismi di rappresentazione (strutturali, semantici o presentazionali) del testo che, utilizzando convenzioni standardizzate, sono utilizzabili su più supporti
  - Markup (marcatura o annotazione): un modo per associare informazioni strutturali al testo
- Documento = contenuto + informazioni strutturali
  - Contenuto: l'informazione pura del testo
  - Informazioni strutturali: ciò che serve per rendere efficaciemente intellegibile l'informazione (anche visualizzazione)
- SGML (Standard Generalized Markup Language):
  - *Metalinguaggio* di markup descrittivo standardizzato che definisce dei metodi di rappresentazione del testo in forma elettronica in modo indipendente dall'hardware e dal sistema utilizzato

Linguaggio di markup (2)

- Web aperto: con la nascita del Web è stato necessario definire un nuovo linguaggio di markup per descrivere la struttura ipermediale delle pagine Web (*in forma aperta*):
  - Fornire delle linee guida generali per la rappresentazione del contenuto
  - In forma aperta: leggibile da tutti i calcolatori e scrivibile senza bisogno di fare ricorso a programmi particolari
  - Non specifica esattamente il formato e la posizione del testo, lasciando ai browser la definizione dei dettagli
    - Due browser possono visualizzare la stessa pagina in modo differente

Linguaggio HTML

- Sebbene siano stati proposti modifiche ed altri standard, nell'accezione comune HTML (HyperText Markup Language) rimane ancora il "linguaggio del Web"
- HTML è un linguaggio di markup non proprietario basato su SGML ed ideato nel 1989
- La risorsa HTML è un file di solo testo ASCII
- Il testo è *free-format*
- Contenuto del testo e specifiche di formato sono inseriti nello stesso file

- Riferimenti:
  - HTML 4.01, 1999
    (W3C Recommendation, http://www.w3.org/TR/html401/)
  - HTML 5, 2008
    (W3C Working Draft, http://www.w3.org/TR/html5/)

Linguaggio HTML (2)

- Quando un browser riceve una risorsa da un Web server:
  - la interpreta e la visualizza in base a determinate istruzioni in esso contenute se la risorsa è un documento HTML
  - la visualizza nella stessa forma in cui l'ha ricevuta se la risorsa è un'immagine oppure invoca un opportuno plug-in

- HTML non è un linguaggio di programmazione
  - un linguaggio di programmazione permette di computare qualcosa, di usare salti condizionali, cicli e di operare su dati contenuti in strutture di dati astratte
  - HTML è semplicemente un linguaggio di markup usato per definire la struttura di un documento

Struttura di un documento HTML

- Un documento HTML è composto da un'intestazione (*header*) ed un corpo (*body*):
  - intestazione contenente informazioni riguardanti il documento
  - corpo contenente il documento vero e proprio
- Un esempio minimale di documento HTML:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<HTML>
    <HEAD>                                   <- Usato dal browser, non visibile all'utente
        <TITLE>Titolo del documento</TITLE>
    </HEAD>
    <BODY>                                   <- Visto dall'utente come testo della pagina
        <P>Testo del paragrafo</P>
    </BODY>
</HTML>
```


XML

- XML (eXtensible Markup Language): metalinguaggio di markup, progettato per lo scambio e la interusabilità di documenti strutturati sul Web
- Sintassi semplificata rispetto a SGML
- Definizione di una serie (piuttosto lunga) di linguaggi associati e basati su XML, tra cui:
  - XML-Schema: per definire la struttura di un documento XML, inclusa la tipizzazione dei dati
  - XLink: per i collegamenti ipertestuali
  - XSL: per i fogli di stile di un documento XML
  - XPath: per identificare i nodi in un documento XML
- XML ha contribuito all'evoluzione di HTML
- Riferimento: XML 1.0 (W3C Recommendation, http://www.w3.org/TR/xml/

Esempio di documento in XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<utenti>
    <utente>
        <nome>Luca</nome>
        <cognome>Ruggero</cognome>
        <indirizzo>Milano</indirizzo>
    </utente>
    <utente>
        <nome>Max</nome>
        <cognome>Rossi</cognome>
        <indirizzo>Roma</indirizzo>
    </utente>
</utenti>
```


Vantaggi di XML

- Indipendenza dalla piattaforma
  - Standard aperto: chiunque può realizzare strumenti che lo usino come formato di dati
- Meta-linguaggio
  - Non è una grammatica, ma una *grammatica di grammatiche*, un modo per generare grammatiche personalizzate
- Documenti auto-descrittivi
  - Scelta del vocabolario in modo da facilitare la comprensione del ruolo strutturale di ogni elemento
- Sintassi universale, minimale e rigorosa
- Struttura navigabile dei documenti
  - La rigida struttura ad albero e la facilità di accesso alle varie sottoparti facilitano la visualizzazione e la programmazione di applicazioni
- Facile convertibilità a formati Web

XHTML

- XHTML (eXtensible HyperText Markup Language): linguaggio di markup basato su XML, che associa alcune proprietà di XML con le caratteristiche di HTML
  - file XHTML: file XML scritto con i termini dell'HTML
  - Riformulazione di HTML 4.01 come applicazione di XML 1.0
- Uso più restrittivo dei tag HTML
  - Solo la struttura della pagina è scritta in XHTML, mentre il layout è imposto dai fogli di stile a cascata (CSS, Cascading Style Sheets)
- Riferimento: XHTML 1.0 (W3C Recommendation, http://www.w3.org/TR/xhtml1/

Esempio di documento in XHTML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html public "-//W3C//DTD XHTML 1.1//EN"
   http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd>
    <html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>Questo &egrave; un bel titolo</title>
    </head>
    <body>
        <!-- qui compare il contenuto del corpo -->
        Hello world!
    </body>
</html>
```


Sviluppi di HTML

- Obiettivi principali del W3C nello sviluppo di HTML 5
  - Incorporare nello standard concetti e funzionalità che rispecchiano le modalità di utilizzo reali del linguaggio
  - Aggiungere nuove funzionalità dedicate alla creazione di applicazioni Web, in particolare nuovo insieme di API per:
    - grafica 2D
    - integrazione e controllo di contenuti audio e video
    - archiviazione persistente dei dati sul client
    - esecuzione di applicazioni web in modalità offline
    - modifica interattiva dei documenti da parte degli utenti
    - oltre ad API per messaggistica, networking, drag and drop, gestione della cache

Macro-componenti del Web: client

- Diversi tipi di Web client (user agent nel protocollo HTTP)
- Browser Web
  - Forma più popolare di Web client
- Spider o robot (bot)
  - Applicazione automatica usata dai motori di ricerca per ottenere informazioni sulle risorse fornite dai server Web a diversi scopi (indicizzazione, catalogazione)
- Agente software
  - Client in grado di svolgere compiti specifici (ad esempio, motori di meta-ricerca, agenti per auction, verifica di correttezza sintattica)

Browser

- Applicazione software che svolge il ruolo di interfaccia fra l'utente ed il Web, mascherando la complessità di Internet e del Web
- Principali servizi del browser
  - Consente di specificare le richieste (URL)
  - Implementa il client del protocollo HTTP (query al DNS per la risoluzione dell'hostname nell'URL, gestione delle connessioni TCP, invio dei messaggi di richiesta HTTP)
  - Visualizza in modo appropriato il contenuto delle risposte sullo schermo (ad es., attivando programmi esterni per elaborare specifiche parti di una pagina Web) e consente la navigazione
  - Invia informazioni al server per la generazione di risorse dinamiche (codificate nell'URL o inviate con il metodo POST dell'HTTP), gestisce informazioni di stato (cookie)
  - Caching locale delle risorse
  - Altri servizi (preferiti, stampa, salva, ricerca nel testo, …)
- Browser diversi, diverse compatibilità

Breve storia

- *FrameMaker* **e** *Acrobat PDF*
  - consentono generazione e incorporazione di hyperlink
- *Gopher* (University of Minnesota)
  - primo browser con hyperlink verso siti remoti
  - uso di standard, quali testo ASCII e socket Unix
  - difetti: link separati dal testo, immagini non gestite

HTML Mosaic

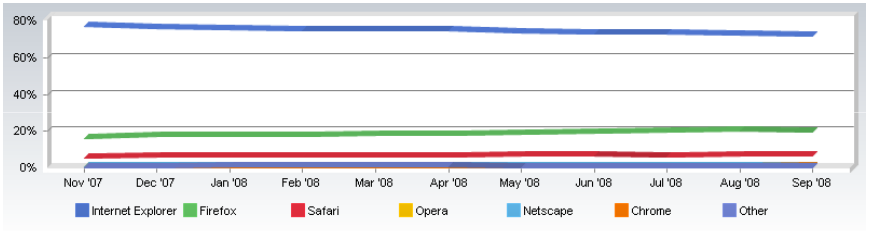
- *Mosaic* (NCSA, 1993)
  - *Netscape Navigator*
  - *Microsoft Explorer*

Principali browser

- Microsoft Internet Explorer
  - Leader di mercato
  - Trident come motore di rendering (o layout engine)
- Mozilla Firefox
  - Il maggiore competitor di IE
  - Gecko come layout engine
- Opera
  - Presto come layout engine
- Google Chrome
- Browser per dispositivi mobili
  - Es.: Nokia browser, Opera Mobile Browser, Internet Explorer Mobile per pocket PC e handheld PC
- Altri browser
  - Safari, Konqueror (browser di KDE), lynx (solo testuale)

Principali browser (2)

- Una stima approssimativa sulla diffusione dei principali browser



Fonte: http://marketshare.hitslink.com/

Componenti di un browser

- Il browser, di per sé, è in grado solo di visualizzare correttamente file HTML e immagini GIF e JPEG
- Può ricorrere ad applicazioni esterne (*plug-in* ed *helper*) per visualizzare o comunque per elaborare specifiche parti di una pagina Web; ad esempio:
  - Acrobat Reader per visualizzare file in formato pdf
  - RealAudio per video e video streams
  - Flash per animazioni
  - Java Virtual Machine per applet Java

Caching nel browser

- Il browser gestisce uno spazio su disco (di dimensione predefinita e configurabile) in cui memorizza le risorse recuperate corrispondenti a:
  - file HTML, file CSS, immagini (GIF, JPEG, PNG, …), file PDF, …
  - Ad es., con IE di default nella cartella "Temporary Internet Files"
- Prima di effettuare una richiesta al server Web, il browser controlla se nella cache vi è una copia della risorsa
  - Obiettivo: ridurre il tempo di risposta percepito dall'utente
- L'utente può forzare il prelievo della risorsa dal server o comunque la sua rivalidazione mediante un opportuno pulsante del browser (ad es., Aggiorna in IE)
- Le risorse nella cache possono avere un timestamp che indica il periodo di validità della risorsa (stabilito dal server)

Azioni di un browser

Fase iniziale

- Il browser analizza l'indirizzo inserito esplicitamente o il link selezionato in una 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 in un file indice)
  - Se la copia è valida, il browser la visualizza
  - Se l'utente ha esplicitamente effettuato un reload, un buon browser effettuerà una rivalidazione della risorsa in cache, inviando al server una richiesta HTTP con metodo GET ed header "If-modified-since"
- Controlla se è necessario gestire autorizzazioni o informazioni di stato (cookie)
- Si passa alla fase successiva solo se la risorsa deve essere acquisita

Azioni di un browser (2)

Fase di lookup

- Il browser invoca il *resolver* per conoscere, tramite il sistema di naming del DNS, la risoluzione dell'indirizzo IP corrispondente all'hostaname specificato nell'URL
  - Eventuale caching della risoluzione dell'indirizzo
- Se la risoluzione esiste, 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

- Ottenuta la risoluzione, il browser apre una connessione TCP con l'host avente l'indirizzo IP individuato
- Sfruttando la connessione, il browser richiede mediante il protocollo HTTP la risorsa specificata nell'URL

Fase di risposta (ricezione e processamento)

- Il server Web invia la risorsa richiesta
- Il browser effettua un'operazione di parsing (del file HTML):
  - Il browser analizza se vi sono risorse embedded nella pagina
  - In caso affermativo, il browser effettua una richiesta per ciascuna risorsa collegata (differente comportamento a seconda della versione del protocollo HTTP)

Azioni di un browser (4)

Fase di visualizzazione

- Una volta inviate tutte le risorse (HTTP/1.1) o dopo aver inviato ciascuna risorsa (HTTP/1.0), il server chiude la connessione TCP
- Non appena riceve la prima risorsa, il browser analizza come visualizzare sullo schermo il testo contenuto nella pagina
- Il browser carica e visualizza le risorse incorporate nella pagina
- Se la risorsa ricevuta è in qualche formato non direttamente interpretabile dal browser, questi può attivare un apposito programma plug-in che ne consente la visualizzazione

Cookie

- Il protocollo HTTP è stato progettato senza stato per ridurre il carico dei server
- Per gestire lo stato sul server (mantenere traccia sul delle richieste del client), il server memorizza sul client un cookie
  - Una stringa di caratteri di dimensione massima pari a 4 KB
  - Può essere temporaneo (in memoria fino alla chiusura del browser) o permanente (su disco con scadenza)
- Il meccanismo dei cookie è un'estensione del protocollo HTTP
  - Riferimento: RFC 2965
- Cookie utilizzati per
  - Gestione delle sessioni, tracking nei siti, personalizzazione dei contenuti in base alle preferenze dell'utente, pubblicità su misura, …
- La manipolazione dei cookie è uno degli attacchi di sicurezza più diffusi

Esempi di cookie

- Esempi di cookie memorizzati sul disco del client:

ebNewBandWidth_.www.repubblica.it1465%3A1223556946716www.repubblica.it/160035216126722303414315016014722996071 8*

CPnull*www.oreillynet.com/3200176193536030731590278090392029961355*

session-id-time23479600lamazon.com/258412994867202884268030280920802996139 5*session-id213-4388883-8575331amazon.com/10321299486720277656803028192080299613955*ubid-main214-8125996-7216155amazon.com/10242916341376319612693217062080299613955*session-tokenRbQ6MO0mWxC62LkZbGOm5aeGc8YNnkS7VMDIV6+PBl9uRYVpkjsiV1btTwbOmLt9N5Z6Ag0DDqF1hjIMBwj2etxofHx09DSM8IGu1+WUVv36grACksElh7f64gCVJD4QJTZpDLYf8e/FlYxDhN2XqZbUpNi5YL7hJaD3h9Mc/hPOdu0HGFPeyitOuBdG+CXi4FCtVu5n9zA=amazon.com/214743267242925928962996139926893301923286219 8*

Interazione con i cookie

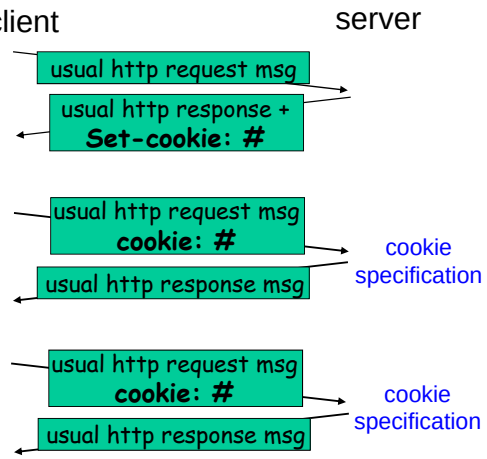
- Il server invia al client un "cookie" in risposta usando l'header Set-cookie

  Set-Cookie:
  <name>=<value>
  [; <name>=<value>]...
  [; expires=<date>]
  [; domain=<domain_name>]
  [; path=<some_path>]
  [; secure][; httponly]

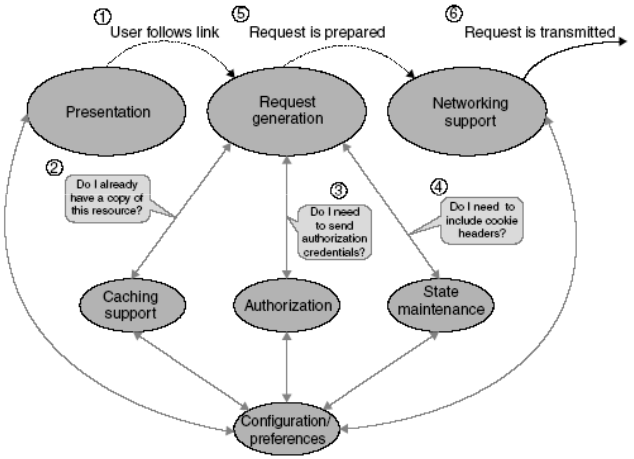
- Il client include il cookie nelle richieste successive usando l'header Cookie

  Cookie: <name>=<value>
  [;<name>=<value>]...

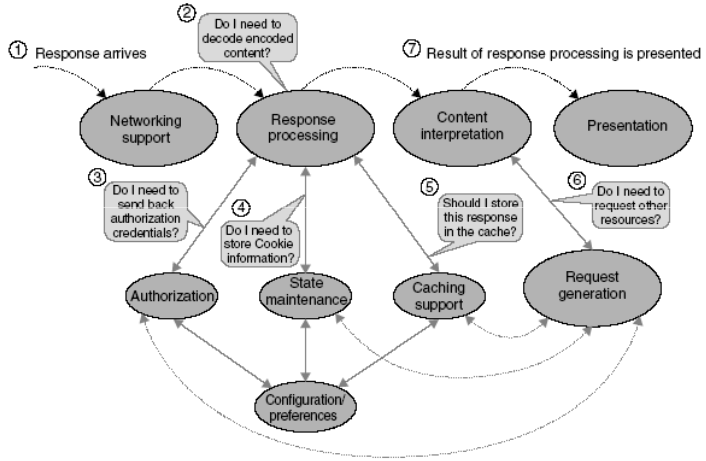
- Il server confronta il cookie del client con i propri cookie memorizzati



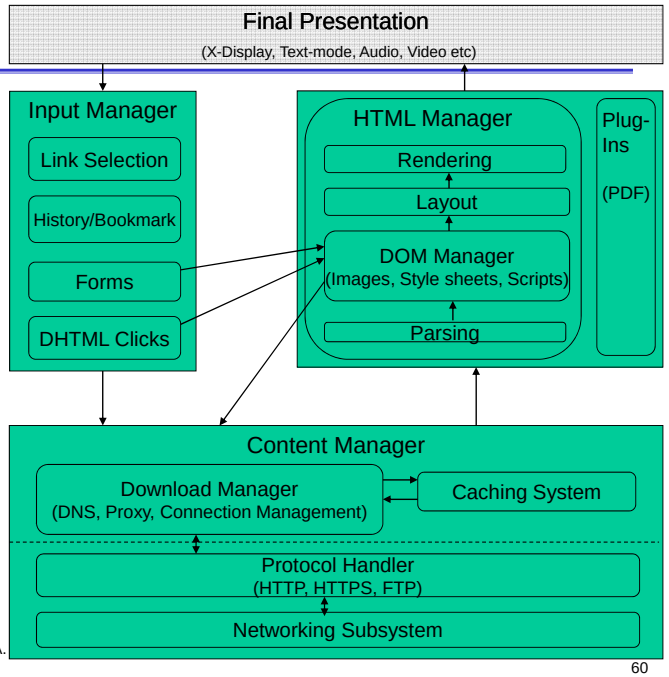
Browser: gestione della richiesta



Browser: gestione della risposta



Componenti dell'architettura di un browser



Componenti di un sito Web

- Piattaforma hardware
- Software di base
- Parte informativa ed applicativa del sito Web
  - Insieme di risorse Web che possono essere richieste dai client tramite messaggi di richiesta HTTP
  - Applicazioni per la generazione delle risorse
- Server Web
  - Il processo server ed il relativo software che viene eseguito sulla piattaforma (hardware - software di base)

Parte informativa

- Organizzata generalmente in pagine ipermediali con collegamenti verso altre pagine (interne o esterne al sito)
- 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
- Ogni pagina Web ha un nome unico, che è il cammino assoluto dalla radice "/" dell'albero delle pagine
  - Questo cammino è quello specificato nella parte path dell'URL richiesto dal client
- La parte informativa del sito è creata e mantenuta dal *content provider*

Memorizzazione delle pagine

- 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

Memorizzazione delle pagine (2)

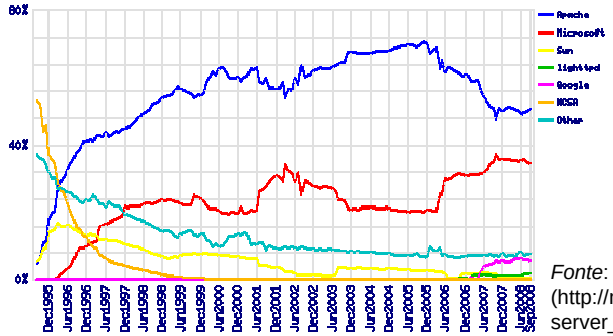
- *Mirroring*
  - l'intero albero è replicato su più dischi o piattaforme
- *Soluzioni intermedie*
  - le parti superiori dell'albero, ovvero le pagine più frequentemente richieste, sono replicate
  - le altre pagine possono essere o meno partizionate
- Ciascuna di queste organizzazioni deve essere trasparente per l'utente, che deve poter navigare e richiedere le pagine nello stesso identico modo, *indipendentemente* dall'organizzazione fisica dei file e dei servizi

Tipi di risorse (*classificazione funzionale*)

- Risorse statiche
  - risorse il cui contenuto è relativamente stabile nel tempo
- Risorse volatili
  - risorse il cui contenuto viene modificato da eventi in corso
    - Es.: ultime notizie, avvenimenti sportivi, titoli in borsa
- Risorse dinamiche
  - risorse il cui contenuto è creato dinamicamente sulla base della richiesta del client
- Risorse attive
  - risorse contenenti codice che viene eseguito dal client

Software per server Web

- I server Web più diffusi sono:
  - Apache (http://httpd.apache.org/)
  - Microsoft Internet Information Server (http://www.microsoft.com/)
  - Google Web server



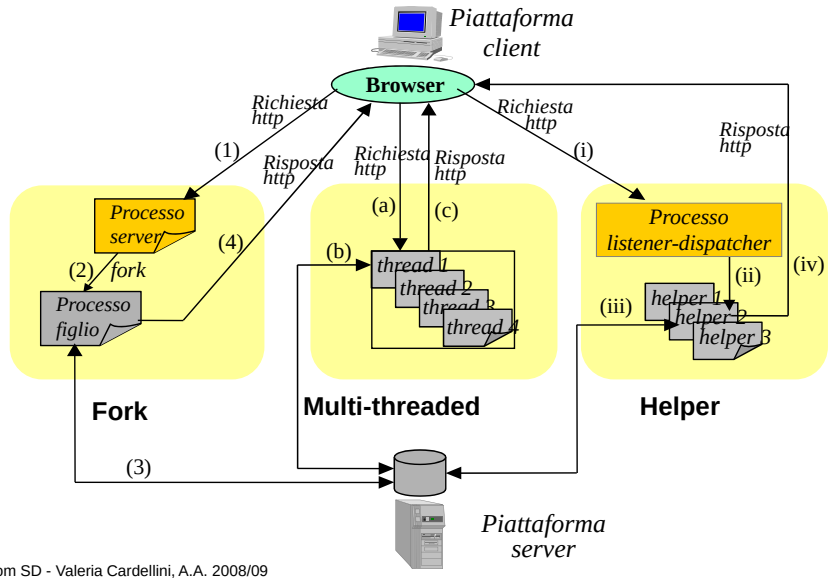
Fonte: Netcraft Web Server Survey (http://news.netcraft.com/archives/web_server_survey.html)

Quanti software per server Web?
http://www.serverwatch.com/stypes/compare/

Modelli architetturali di server Web

- Diversi approcci per l'architettura del server
  - Basato su processi
    - Fork e preforking
    - Esempio: Apache 1.X e Apache 2.X con mpm_prefork
  - Basato su thread
    - Esempio: Microsoft IIS
  - Ibrido (processi e thread)
    - Apache 2.X con mpm_worker
  - Basato su eventi
    - Esempio: Flash, Zeus
  - Interno al kernel
    - Esempio: Tux
- Nella scelta del modello tradeoff tra:
  - Prestazioni, robustezza, protezione, estensibilità, …

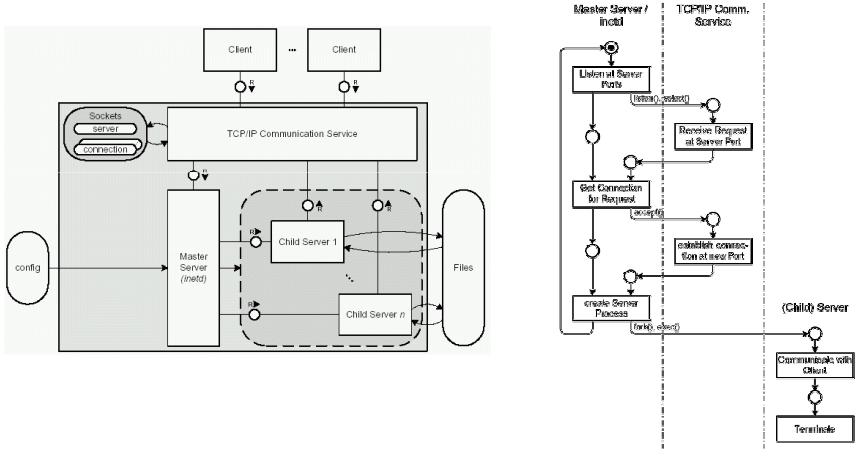
Principali approcci



Server multi-process: fork

- Server basato su processi
- Per ogni nuova richiesta che arriva il processo server (padre):
  - Crea una copia di se stesso (un processo child), alla quale affida la gestione della richiesta tramite fork()
  - Si pone in attesa di nuove richieste
  - Il processo child si occupa di soddisfare la richiesta e poi termina
    - Un processo child per ogni client
  - Con fork():
    - Copia di dati, heap e stack; condivisione del segmento testo
    - No copia completa ma *copy-on-write*: è una ottimizzazione

Server multi-process: fork (2)



Server multi-process: fork (3)

- Vantaggi:
  - Il codice del server rimane semplice, poiché la copia è demandata in toto al sistema operativo
- Svantaggi:
  - Overhead di fork() può penalizzare l'efficienza del sistema
    - Il tempo di generazione del processo child può non essere trascurabile rispetto al tempo di gestione della richiesta
  - In mancanza di un limite superiore al numero di richieste che possono essere gestite concorrentemente, nel caso di un elevato numero di richieste i processi child possono esaurire le risorse del sistema
  - Meccanismo di IPC per la condivisione di informazioni tra padre e figli successivamente a fork()
    - Ad esempio memoria condivisa

Server multi-process: helper

- Server basato su processi
- Un processo *dispatcher* (o listener) ed alcuni processi per il servizio delle richieste, detti processi *helper* (o worker)
  - All'avvio del servizio, il dispatcher effettua il *preforking* dei processi helper (*pool* di processi helper)
  - Il dispatcher rimane in ascolto delle richieste di connessione
  - Quando arriva una richiesta di connessione, il dispatcher la trasferisce ad un helper per la gestione
    - Occorre usare una forma di passaggio di descrittori tra processi distinti
  - Quando l'helper termina la gestione della richiesta, si rende disponibile per gestire una nuova richiesta
  - A regime, il dispatcher svolge compiti di supervisione e controllo

Server multi-process: helper (2)

- Vantaggi
  - Processi helper creati una sola volta e poi riusati
    - Si evita l'overhead dovuto alla fork() all'arrivo di ogni nuova richiesta
  - Maggiore robustezza (separazione dello spazio di indirizzi) e portabilità rispetto al server multi-threaded
  - Maggiore semplicità rispetto al server basato su eventi (linearità nel modo di pensare del programmatore)
- Svantaggi
  - Processo dispatcher potenziale collo di bottiglia
  - Gestione del numero di processi helper nel pool
  - Maggior uso di memoria rispetto a server multi-threaded
  - Gestione della condivisione di informazioni tra i processi helper (uso di lock)

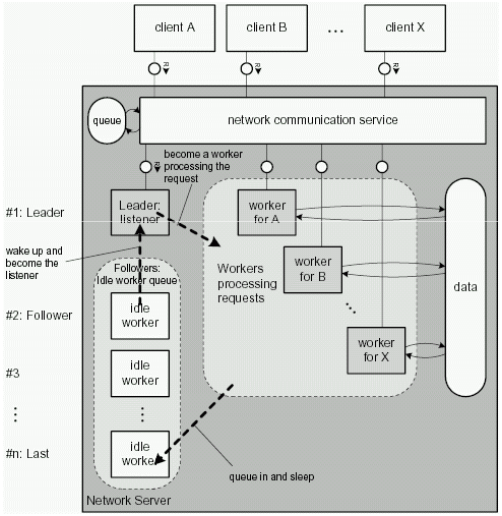
Schemi per preforking

- Schema con preforking considerato finora:
  - Il dispatcher effettua accept() e passa il descrittore del socket di connessione ad un helper
  - Anche noto come schema *job-queue*
    - Il dispatcher è il produttore; gli helper sono i consumatori
    - Il dispatcher accetta le richieste e le pone in una coda
    - Gli helper leggono le richieste dalla coda e le servono
- In alternativa, si può usare lo schema *leader-follower*

Schema leader-follower

- Dopo il preforking, ogni helper chiama accept() sul socket di ascolto
- Soluzione 1: nessuna forma di locking per accept
  - Problema a: *thundering herd*
  - Problema b: funziona correttamente su kernel Unix derivati da Berkeley (accept implementata nel kernel), ma non su kernel Unix derivati da System V (accept come funzione di libreria)
    - Il socket d'ascolto è una risorsa a cui accedere in mutua esclusione
- Soluzione 2: file locking per accept
  - Dopo il preforking, ogni helper chiama accept(), effettuando un *file locking* prima dell'invocazione di accept()
  - Vedi esempio: file locking Posix con funzione fcntl()
- Gli helper idle competono per accedere al socket d'ascolto
  - Al più uno (detto *leader*) si può trovare in ascolto, mentre gli altri (detti *follower*) sono accodati in attesa di poter accedere alla sezione critica per il socket d'ascolto

Schema leader-follower (2)



Preforking con file locking

```
static int          nchildren;
static pid_t        *pids;
int main(int argc, char **argv)
{
    int             listenfd, i;
    socklen_t       addrlen;
    void            sig_int(int);
    pid_t           child_make(int, int, int);

    …               /* creazione del socket di ascolto, bind() e listen() */
    pids = calloc(nchildren, sizeof(pid_t));
    my_lock_init("/tmp/lock.XXXXXX"); /* un file di lock per tutti i processi child */
    for (i = 0; i < nchildren; i++)
        pids[i] = child_make(i, listenfd, addrlen);
    …
}

pid_t child_make(int i, int listenfd, int addrlen)
{
    pid_t pid;
```


Preforking con file locking (2)

```
    if ( (pid = fork()) > 0)
        return(pid);                /* processo padre */
    child_main(i, listenfd, addrlen);   /* non ritorna mai */
}

void child_main(int i, int listenfd, int addrlen)
{
    int     connfd;
    socklen_t clilen;
    struct sockaddr *cliaddr;

    cliaddr = malloc(addrlen);
    printf("child %ld starting\n", (long) getpid());
    for ( ; ; ) {
        clilen = addrlen;
        my_lock_wait();             /* my_lock_wait() usa fcntl() */
        connfd = accept(listenfd, cliaddr, &clilen);
        my_lock_release();
        web_child(connfd);          /* processa la richiesta */
        close(connfd);
    }
}
```


Preforking con file locking (3)

```
#include <fcntl.h>
#include "basic.h“
static struct flock  lock_it, unlock_it;
static int  lock_fd = -1;
     /* fcntl() will fail if my_lock_init() not called */

void my_lock_init(char *pathname)
{
  char  lock_file[1024];  /* must copy caller's string, in case it's a constant */

  strncpy(lock_file, pathname, sizeof(lock_file));
  if ( (lock_fd = mkstemp(lock_file)) < 0) {
    fprintf(stderr, "errore in mkstemp");
    exit(1);
  }
  if (unlink(lock_file) == -1) {            /* but lock_fd remains open */
    fprintf(stderr, "errore in unlink per %s", lock_file);
    exit(1);
  }
```


Preforking con file locking (4)

```
  lock_it.l_type = F_WRLCK;
  lock_it.l_whence = SEEK_SET;
  lock_it.l_start = 0;
  lock_it.l_len = 0;

  unlock_it.l_type = F_UNLCK;
  unlock_it.l_whence = SEEK_SET;
  unlock_it.l_start = 0;
  unlock_it.l_len = 0;
}
```

Preforking con file locking (5)

```
void my_lock_wait()
{
  int rc;
  while ( (rc = fcntl(lock_fd, F_SETLKW, &lock_it)) < 0) {
    if (errno == EINTR) continue;
    else {
      fprintf(stderr, "errore fcntl in my_lock_wait");
      exit(1);
    }
  }
}

void my_lock_release()
{
  if (fcntl(lock_fd, F_SETLKW, &unlock_it) < 0) {
    fprintf(stderr, "errore fcntl in my_lock_release");
    exit(1);
  }
}
```


Server multi-threaded

- Server basato su thread
- Una sola copia del server che genera thread multipli di esecuzione
  - Il thread principale rimane sempre in ascolto delle richieste
  - Quando arriva una richiesta, esso genera un nuovo thread (un *request handler*), che la gestisce e poi (eventualmente) termina
  - Ogni thread possiede una copia privata della connessione gestita, ma condivide con gli altri thread uno spazio di memoria (codice del programma e variabili globali)
- In alternativa alla creazione del thread all'arrivo della richiesta, il pool di thread può essere pre-creato (*prethreading*)

Server multi-threaded (2)

- Vantaggi
  - Creazione di un thread più veloce della creazione di un processo
    - Da 10 a 100 volte
  - Minore overhead per il context switching
  - Condivisione delle informazioni per default
  - Mantiene una maggiore semplicità rispetto a server basato su eventi
- Svantaggi
  - Maggiore complessità del codice del server (gestione della sincronizzazione tra thread)
  - Minore robustezza rispetto al server multi-process: i thread non sono protetti uno dall'altro
  - Supporto da parte del sistema operativo al multithreading (ad es., Linux/Unix, Windows)
  - Maggiori limitazioni sul numero di risorse rispetto al preforking (ad es. numero di descrittori aperti)

Preforking e prethreading

- Riassumiamo le possibili alternative per realizzare un server con preforking o prethreading
- Preforked server
  - Dispatcher/listener
    - Il processo padre invoca accept; occorre passare da padre a figlio il descrittore del socket di connessione
  - Leader/follower
    - Nessuna forma di locking per accept
    - File locking per accept
    - Mutex per proteggere accept
- Prethreaded server
  - Dispatcher/listener
    - Il thread principale invoca accept; non occorre passare il descrittore da un thread all'altro, perché i thread condividono tutti i descrittori
  - Leader/follower
    - Mutex per proteggere accept

Dimensione del pool

- Comportamento della dimensione del pool di processi o thread
  - Scelta significativa nell'architettura software di un server basato su processi o thread
- Alternative: dimensione statica o dinamica
- Pool di dimensione *statica* (ad es. *p*)
  - Carico alto: se i *p* processi o thread del pool sono occupati, una nuova richiesta deve attendere
  - Carico basso: la maggior parte dei processi o thread sono idle (spreco di risorse)
- Pool di dimensione *dinamica*
  - Il numero di processi o thread varia con il carico: cresce se il carico è alto, diminuisce se il carico è basso
  - Tipicamente, c'è un minimo numero di processi o thread idle
  - Esempio: Apache

Server ibrido

- Server basato su processi e thread
- Molteplici processi, ciascuno dei quali è multi-threaded
  - Un singolo processo di controllo (processo padre) lancia i processi figli
  - Ciascun processo figlio crea un certo numero di thread di servizio ed un thread listener
  - Quando arriva una richiesta, il thread listener la passa ad un thread di servizio che la gestisce
- Combina i vantaggi delle architetture basata su processi e basata su thread, riducendo i loro svantaggi
  - In grado di servire un maggior numero di richieste usando una minore quantità di risorse rispetto all'architettura basata su processi
  - Conserva in gran parte la robustezza e stabilità dell'architettura basata su processi

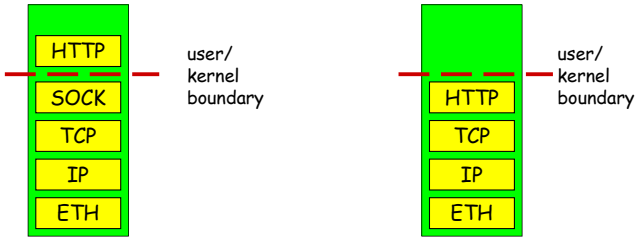
Server basato su eventi

- Un solo processo che gestisce le richieste in modo event-driven
  - Anziché servire una singola richiesta nella sua interezza, il server esegue una piccola parte di servizio per conto di ciascuna richiesta
- Uso di select(), opzioni non bloccanti sui socket, gestione asincrona dell'I/O
  - Il server continua l'esecuzione mentre aspetta di ricevere una risposta alla chiamata di sistema da parte del sistema operativo

Server basato su eventi (2)

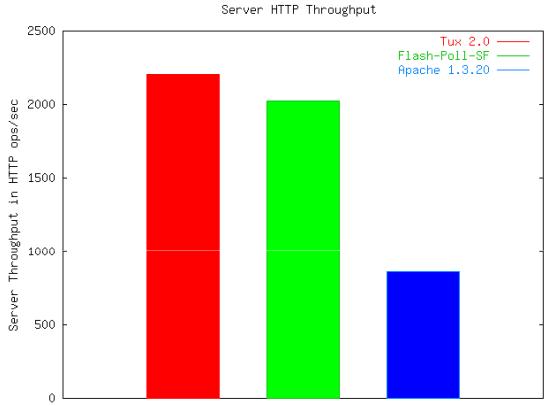
- Vantaggi
  - Molto veloce
  - La condivisione è intrinseca: un solo processo
  - Non c'è bisogno di sincronizzazione come nel server multi-threaded
  - Non ci sono overhead dovuti al context switching o consumi extra di memoria
- Svantaggi
  - Maggiore complessità nella progettazione ed implementazione
  - Meno robusto: una failure può fermare l'intero server
  - Limiti delle risorse per processo (es. descrittori di file)
  - Supporto in tutti i sistemi operativi di I/O asincrono

Server interno al kernel



- Thread del kernel dedicato per richieste HTTP:
  - Prima alternativa: tutto il server nel kernel
  - Seconda alternativa: gestione delle richieste GET statiche nel kernel, mentre richieste dinamiche gestite da un server (es. Apache) nello spazio utente
- Tux rimosso dal Linux kernel 2.6
  - Tuttavia, alcune distribuzioni (es. Fedora) lo hanno rimesso nel kernel 2.6

Confronto delle prestazioni



Tux: interno al kernel
Flash: basato su eventi
Apache 1.3: preforking

Fonte: E. Nahum, "Web servers: Implementation and performance"

- Il grafo mostra il throughput per Tux, Flash e Apache
- Esperimenti su P/II 400 MHz, gigabit Ethernet, Linux 2.4.9-ac10, 8 macchine client, WaspClient come generatore di carico

Macro-componenti del Web: server proxy

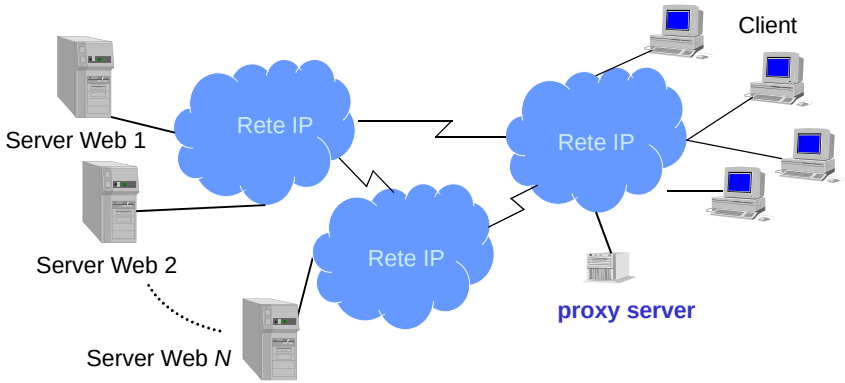
- Nel caso più semplice la comunicazione tra client e server Web avviene direttamente
- Più in generale, ove non sia possibile o non conveniente che il client contatti direttamente il server, vengono utilizzati degli intermediari, tra cui il più diffuso è il *server proxy*
- Il proxy agisce sia da server nei confronti del *client Web* sia da client nei confronti del *server Web*

Caching delle risorse

- I proxy sono nati come *gateway*
  - Intermediari per consentire la comunicazione tra client all'interno di Intranet con server Web esterni
- Successivamente, usati principalmente come locazioni per il *caching* delle risorse
- Principi di località spaziale e temporale:
  - Probabilità elevata che utenti che *condividono* la cache (ad es. appartenenti alla stessa organizzazione) possano essere interessati alle medesime risorse Web in un intervallo di tempo
- Quando una risorsa viene reperita per la prima volta da un server Web, può essere memorizzata sul disco del proxy, in modo che richieste successive per la stessa risorsa indirizzate al proxy possono essere soddisfatte localmente

Caching nel Web

- Una cache Web memorizza una copia locale delle risorse Web richieste (più recentemente) e reagisce come un *proxy* alle richieste degli utenti



Caching nel Web (2)

- Vantaggi in caso di *cache hit*
  - Riduzione della latenza percepita dall'utente
  - Riduzione del traffico Internet
  - Riduzione del carico sui server Web

Dove avviene il caching nel Web?

- Il caching ha luogo in diverse locazioni del Web: una richiesta Web può attraversare diversi sistemi di caching lungo il suo percorso verso il server Web
- Caching a livello del browser
- Caching a livello del server Web
- Caching a livello dei proxy
  - **forward proxy**: intermediario tra client e server Web, localizzato in prossimità del client (*consumer-oriented*) oppure in punti strategici della rete (*backbone core proxy*)
  - **reverse proxy** (o acceleratore HTTP): localizzato in prossimità del server Web (*provider-oriented*)

Come indirizzare le richieste al proxy

- Alternativa tra proxy *trasparente* e *non trasparente*
  - Proxy *non trasparente*: il client è a conoscenza dell'esistenza del proxy
  - Proxy *trasparente*: presenza di un elemento di rete (uno switch o un router) che intercetta tutto il traffico Web da client a server e lo redirige verso un proxy server
- Esistono varie modalità per configurare l'uso di un proxy *non trasparente* da parte di un client
  - Configurazione esplicita del browser
    - L'utente deve indicare l'indirizzo e la porta del proxy
    - Per Internet Explorer: Opzioni Internet -> Connessioni -> Impostazioni -> Server proxy
  - Configurazione automatica del browser tramite un file di configurazione, fornito ad es. dall'amministratore del sistema
  - Discovery automatico del file di configurazione

Problemi del proxy caching
(lato *consumer*)

- Trade-off tra contattare server Web e altri proxy
  - Non c'è garanzia di prestazioni migliori
  - Non c'è garanzia di migliore prossimità Internet
- Consistenza delle informazioni in cache
- Aumento di contenuti dinamici, personalizzati e sicuri sui siti Web

- Altri problemi economico-legali
  - Privacy
  - Copyright
  - Acquisizione di informazioni personalizzate
  - Siti con banner pubblicitari