Il Word Wide Web

Il ***World Wide Web*** (detto anche ***Web***, ***WWW*** o ***W***[3)] è nato al Cern nel 1989 per consentire una agevole cooperazione fra i gruppi di ricerca di fisica sparsi nel mondo.

E' un'architettura software volta a fornire l'accesso e la navigazione a un enorme insieme di documenti collegati fra loro e sparsi su milioni di elaboratori.

Ipertesti

Tale insieme di documenti forma un ***ipertesto*** (***hypertext***), cioè un testo che viene percorso in modo non lineare. Il concetto di ipertesto risale alla fine degli anni ' 40, e si deve a vari scienziati:

- Vannevar Bush (sistema Memex, basato su microfilm);
- Douglas Engelbart (sistema NLS/Augment, basato su elaboratori interconnessi);
- Ted Nelson (sistema Xanadu, con enfasi sulla tutela dei diritti d' autore: un documento poteva contenere un riferimento ad altri documenti, che venivano inclusi "al volo" in quello referente e mantenevano così la loro unicità e originalità).

Caratteristiche del WWW

- architettura di tipo client-server:
  - ampia scalabilità;
  - adatta ad ambienti di rete;
- architettura distribuita:
  - perfettamente in linea con le esigenze di gestione di un ipertesto;
- architettura basata su standard di pubblico dominio:
  - possibilità per chiunque di proporre una implementazione;
  - uguali possibilità di accesso per tutte le piattaforme di calcolo;
- capacità di gestire informazioni di diverso tipo (testo, immagini, suoni, filmati, realtà virtuale, ecc.):
  - grande interesse da parte di tutti gli utenti.

I documenti che costituiscono l' ipertesto gestito dal Web sono detti ***pagine web***, e possono contenere, oltre a normale testo formattato, anche:

- rimandi (detti ***link*** o ***hyperlink***) ad altre pagine web;
- immagini fisse o in movimento;
- suoni;
- scenari tridimensionali interattivi;
- codice eseguibile localmente.

L' utilizzo del Web è semplicissimo:

- un utente legge il testo della pagina, vede le immagini, ascolta la musica, ecc.;
- se seleziona col mouse un link (che di solito appare come una parola sottolineata e di diverso colore) la pagina di partenza viene sostituita sullo schermo da quella relativa al link selezionato.

La nuova pagina può risiedere su un qualsiasi server.

Architettura WWW

Il Web è una architettura software di tipo ***client-server***, nella quale sono previste due tipologie di componenti software: il ***client*** e il ***server***, ciascuno avente compiti ben definiti.

Client WWW

Il client (o ***user agent***) è la parte SW dell'utente per la navigazione nel WWW

Ha i compiti di:

- trasmettere all' opportuno server le richieste di reperimento dati che derivano dalle azioni dell' utente;
- ricevere dal server le informazioni richieste;
- visualizzare il contenuto della pagina Web richiesta dall' utente, gestendo in modo appropriato tutte le tipologie di informazioni in esse contenute;
- consentire operazioni locali sulle informazioni ricevute (ad esempio salvarle su disco, stamparle).

Questi client vengono comunemente chiamati ***browser*** (sfogliatori). Gli esempi più noti sono:

- NCSA Mosaic (il primo)
- Netscape Navigator
- Microsoft Internet Explorer
- Mozzilla

L'architettura dei Browser è modulare, non gestisce nativamente tutti i tipo di documenti ma solo:

- testo formattato;
- immagini fisse;
- codice eseguibile.

Gli altri tipi di informazioni vengono gestiti:

- consegnandoli a un programma esterno (***helper***) che provvederà alla corretta gestione (ad esempio, un file contenente un filmato verrà consegnato a un programma per il playback di filmati);
- se il browser ha un' architettura modulare le sue funzionalità possono essere estese per mezzo di ***plug-in***, ossia librerie di codice eseguibile specializzato che possono essere caricate in memoria secondo le necessità. In questa situazione, se il necessario plug-in è installato, il browser provvede a caricarlo e gli affida la gestione delle informazioni da trattare.

Multithreading

Una importante caratteristica di tutti i browser moderni è di essere ***multithreaded***, cioè di consentire che, quando la cpu è sotto il loro controllo, si alternino fra loro multipli ***thread di controllo***, cioè flussi di elaborazione concorrenti.

Ad esempio, nel caso di un sistema operativo (S.O.) che offre il multitasking, si può avere una situazione come quella seguente.

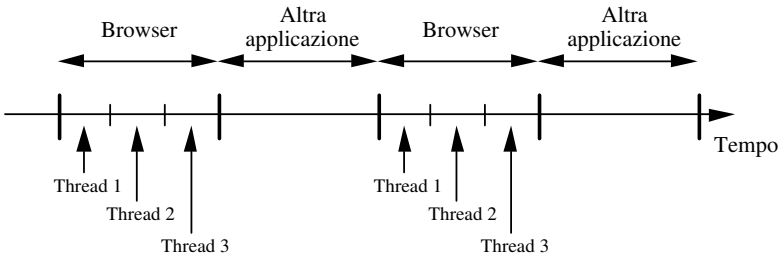

Figura 1-1: Uso della CPU in un browser multithreaded

Un thread, a differenza di un vero processo, è un contesto di esecuzione il cui spazio di indirizzamento viene ricavato all' interno di quello del processo che lo ha generato.

Il Server WWW

Il server è tipicamente un processo in esecuzione su un elaboratore. Esso, di norma, è sempre in esecuzione e deve:

- rimanere in ascolto di richieste da parte dei client;
- fare del suo meglio per soddisfare ogni richiesta che arriva:
  - se possibile, consegnare il documento richiesto;
  - altrimenti, spedire un messaggio di notifica di errore (documento non esistente, documento protetto, ecc.).

Problematiche dei Server

- efficienti e veloci
- gestire più richieste contemporanee, e continuare a rimanere in ascolto di nuove richieste.

Il secondo requisito in particolare implica una qualche forma di concorrenza nel lavoro del server.

La soluzione dipende dal S.O.

Principali tecniche:

1) Clonazione del server

- per ogni nuova richiesta che arriva, il server (che è sempre in ascolto):
  - crea una nuova copia di se stesso alla quale affida la gestione della richiesta;
  - si mette subito in attesa di nuove richieste;
- la copia clonata si occupa di soddisfare la richiesta e poi termina.

Le varie copie del server sono dei processi indipendenti gestiti dal S.O.

Questo è un metodo tipico di S.O. multitasking quali UNIX, e si ottiene con l' uso della **`fork()`**.

Vantaggi:

- il codice del server rimane semplice, poiché la clonazione è demandata in toto al S.O.

Svantaggi

- poiché in genere la gestione di una richiesta è piuttosto rapida, il tempo di generazione del clone può non essere trascurabile rispetto al tempo di gestione della richiesta, introducendo così un overhead che può penalizzare l' efficienza del sistema.

2) Server multithreaded

Esiste una sola copia del server, che però è progettato per essere in grado di generare thread multipli:

- il thread principale (quello iniziale) rimane sempre in ascolto delle richieste;
- quando ne arriva una, esso genera un nuovo thread che prima la gestisce e poi termina.

Questo metodo richiede che il S.O. offra librerie di supporto al multithreading, che ormai sono presenti in tutti i S.O. moderni (UNIX, Windows 95 e NT, MacOS, Linux) per cui di fatto è universalmente applicabile.

Vantaggi:

- la creazione di un thread è molto più veloce di una **`fork()`** (anche 30 volte sotto UNIX), quindi in generale è più efficiente per gestire operazioni veloci come l' esaudire la richiesta del client.

Svantaggi:

- il codice del server diviene un pò più complesso, perché al suo interno si dovranno gestire la creazione dei thread ed il loro avanzamento, anche in termini di sincronizzazione.

Il protocollo HTTP

Il protocollo HTTP gestisce il dialogo fra un client e un server web.

HTTP è un ***protocollo ASCII***, cioè i messaggi scambiati fra client e server sono costituiti da sequenze di caratteri ASCII (e questo, come vedremo, è un problema se è necessaria la riservatezza delle comunicazioni).

In questo contesto per ***messaggio*** si intende la richiesta del cliente oppure la risposta del server, intesa come informazione di controllo; viceversa, i dati della URL richiesta che vengono restituiti dal server non sono necessariamente ASCII (esempi di dati binari: immagini, filmati, suoni, codice eseguibile).

Il protocollo prevede che ogni singola interazione fra client e server si svolga secondo il seguente schema:

- apertura di una connessione a livello di trasporto fra client e server (TCP è lo standard di fatto, ma qualunque altro può essere usato);
- invio di una singola richiesta da parte del client, che specifica la URL desiderata;
- invio di una risposta da parte del server e dei dati di cui alla URL richiesta;
- chiusura della connessione a livello di trasporto.

Dunque, il protocollo è di tipo ***stateless***, cioè non è previsto il concetto di ***sessione*** all' interno della quale ci si ricorda dello stato dell' interazione fra client e server. Ogni singola interazione è storia a se ed è del tutto indipendente dalle altre.

Il client

Quando un client effettua una richiesta invia diverse informazioni:

- il metodo (cioè il comando) che si chiede al server di eseguire;
- il numero di versione del protocollo HTTP in uso;
- l' indicazione dell' oggetto al quale applicare il comando;
- varie altre informazioni, fra cui:
  - il tipo di client;
  - i tipi di dati che il client può accettare.

I metodi definiti in HTTP sono:

GET	Richiesta di ricevere un oggetto dal server
HEAD	Richiesta di ricevere la sola parte head di una pagina html
PUT	Richiesta di mandare un oggetto al server
POST	Richiesta di appendere sul server un oggetto a un altro (vedremo che si usa molto)
DELETE	Richiesta di cancellare sul server un oggetto
LINK e UNLINK	Richieste di stabilire o eliminare collegamenti fra oggetti del server

il metodo che si usa per ricevere le pagine è GET;

- POST ha il suo più significativo utilizzo in relazione all' invio di dati tramite form;
- HEAD si usa quando il client vuole avere delle informazioni per decidere se richiedere o no la pagina;
- PUT, DELETE, LINK, UNLINK non sono di norma disponibili per un client, tranne che in quei casi in cui l' utente sia abilitato alla configurazione remota (via Web) del server Web.

Ad esempio, supponiamo che nel file HTML visualizzato sul client vi sia un' ancora:

<A HREF="http://www.unica.it/index.html"> ..... </A>

e che l' utente attivi tale link. A tal punto il client:
chiede al DNS l' indirizzo IP di www.unica.it;
apre una connessione TCP con www.unica.it, porta 80;
invia la richiesta.

Essa è costituita da un insieme di comandi (uno per ogni linea di testo) terminati con una linea vuota:

GET /index.html HTTP/1.0	Metodo, URL e versione protocollo
User-agent: Mozilla/3.0	Tipo del client
Host: 160.10.5.43	Indirizzo IP del client
Accept: text/html	Client accetta pagine HTML
Accept: image/gif	Client accetta immagini
Accept: application/octet-stream	Client accetta file binari qualunque
If-modified-since: data e ora	Inviare il documento solo se è più recente della data specificata

Lato server

La risposta del server è articolata in più parti, perché c' è un problema di fondo: come farà il client a sapere in che modo dovrà gestire le informazioni che gli arriveranno?

Ovviamente, non si può mostrare sotto forma di testo un' immagine o un file sonoro! Dunque, si deve informare il client sulla natura dei dati che gli arriveranno prima di iniziare a spedirglieli.

Per questo motivo la risposta consiste di 3 parti:
una ***riga di stato***, che indica quale esito ha avuto la richiesta (tutto ok, errore, ecc.);
delle ***metainformazioni*** che descrivono la natura delle informazioni che seguono;
le ***informazioni*** vere e proprie (ossia, l' oggetto richiesto).

La riga di stato, a sua volta, consiste di tre parti:
Versione del protocollo http;
Codice numerico di stato;
Specifica testuale dello stato.

Tipici codici di stato sono:

Esito	Codice numerico	Specifica testuale
Tutto ok	200	OK
Documento spostato	301	Moved permanently
Richiesta di autenticazione	401	Unauthorized
Richiesta di pagamento	402	Payment required
Accesso vietato	403	Forbidden
Documento non esistente	404	Not found
Errore nel server	500	Server error

Dunque, ad esempio, si potrà avere

HTTP/1.0 200 OK

Le metainformazioni dicono al client ciò che deve sapere per poter gestire correttamente i dati che riceverà.

Sono elencate in linee di testo successive alla riga di stato e terminano con una ***linea vuota***.

Tipiche metainformazioni sono:

Server:	Identifica il tipo di server
Date:	Data e ora della risposta
Content-type:	Tipo dell' oggetto inviato
Content-length:	Numero di byte dell' oggetto inviato
Content-language:	Linguaggio delle informazioni
Last-modified: ...	Data e ora di ultima modifica
Content-encoding:	Tipo di decodifica per ottenere il content

Il Content-type si specifica usando lo standard ***MIME*** (***Multipurpose Internet Mail Exchange***), nato originariamente per estendere la funzionalità della posta elettronica.

Un tipo MIME è specificato da una coppia

MIME type/MIME subtype

Vari tipi MIME sono definiti, e molti altri continuano ad aggiungersi. I più comuni sono:

Type/Subtype	Estensione	Tipologia delle informazioni
text/plain	.txt, .java	testo
text/html	.html, .htm	pagine html
image/gif	.gif	immagini gif
image/jpeg	.jpeg, .jpg	immagini jpeg
audio/basic	.au	suoni
video/mpeg	.mpeg	filmati
application/octet-stream	.class, .cla, .exe	programmi eseguibili
application/postscript	.ps	documenti Postscript
x-world/x-vrml	.vrml, .wrl	scenari 3D

Il server viene configurato associando alle varie estensioni i corrispondenti tipi MIME. Quando gli viene chiesto un file, deduce dall' estensione e dalla propria configurazione il tipo MIME che deve comunicare al client.

Se la corrispondenza non è nota, si usa quella di default (tipicamente text/html), il che può causare errori in fase di visualizzazione.

Anche la configurazione del client (in merito alle applicazioni helper) si fa sulla base dei tipi MIME.

Tornando al nostro esempio, una richiesta del client quale:
GET /index.html HTTP/1.0 User-agent: Mozilla/3.0 ecc.
riceverà come risposta dal server (supponendo che non ci siano errori) le metainformazioni, poi una riga vuota e quindi il contenuto del documento (in questo caso una pagina HTML costituita di n byte):

HTTP/1.0 200 OK
Server: NCSA/1.4
Date: Tue, july 4, 1996 19:17:05 GMT
Content-type: text/html
Content-length: 6528
Content-language: en
Last-modified: Mon, july 3, 1996 15:05:35 GMT
<----- riga vuota
<HTML>
<HEAD>
...
<TITLE>...</TITLE>
...
</HEAD >
<BODY>
...
</BODY>
</HTML>

Conclusioni

il protocollo HTTP è molto semplice, essendo basato su interazioni che prevedono esclusivamente l' invio di una singola richiesta e la ricezione della relativa risposta;
questa semplicità è insieme un punto di forza e di debolezza:

- di forza perché è facilissimo, attraverso la definizione di nuovi tipi MIME e di corrispondenti funzioni sui client, estendere le tipologie di informazioni gestibili (il server non entra nel merito di ciò che contengono i file: si limita a consegnare i dati che gli vengono richiesti, senza preoccuparsi della loro semantica);
- di debolezza perché queste estensioni di funzionalità talvolta mal si adattano alla concezione originaria (stateless) del protocollo, come ad esempio è il caso delle transazioni commerciali.