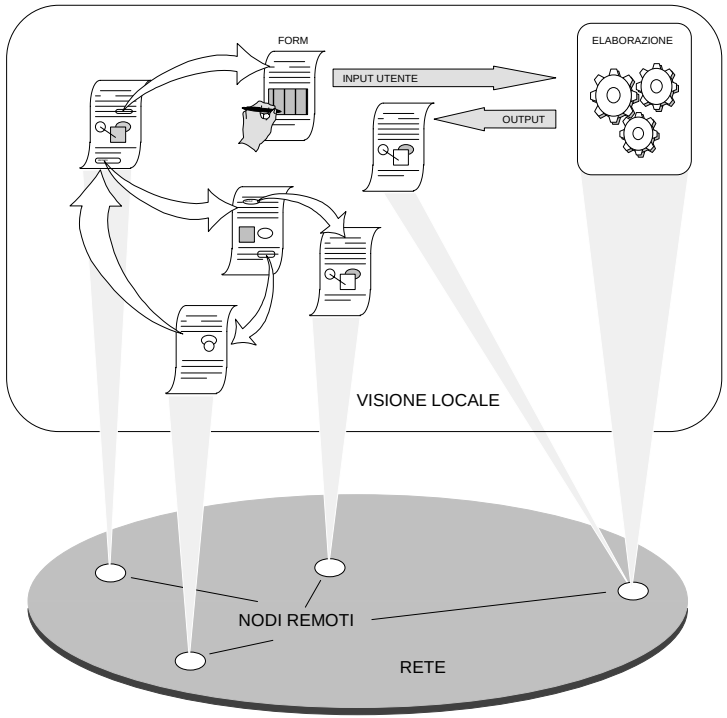
Strumenti trasparenti

unico insieme di informazioni accessibile a tutti i possibili utenti

Gopher

WWW (Mosaic, Netscape)

strutturazione ipertestuale delle informazioni (trasparenza della allocazione delle informazioni) e uso di interfacce grafiche (semplicità di utilizzo)



Primo passo (gopher)

strumenti di visualizzazione a **caratteri** di informazioni

Creazione di un unico direttorio che nasconde le informazioni di allocazione

UNICO indice di informazioni per argomento

Allargamento della fascia di utenza

Secondo passo (WWW)

trasparenza allocazione (strutturazione ipertestuale) gestione informazioni di tipo diverso (multimediali)
strumenti e **protocolli** con ***informazioni multimediali***

- evoluzione della mail e altri tool tradizionali
- informazioni trasparenti e multimediali
- protocolli eterogenei

Il secondo approccio ha *ampliato ulteriormente* la fascia di utenza (in modo esponenziale)

ma introdotto problemi
*di **banda di occupazione** di risorse*
*di **sicurezza** delle informazioni*
*di **standardizzazione** delle informazioni*
*di **visualizzazione rapida** delle informazioni*

World Wide Web (WWW)

CERN (1989)
Progetto di integrazione in forma ipertestuale delle risorse esistenti in INTERNET

Scopi

- Trasparenza accesso e allocazione
- Presentazione multimediale
- Interfaccia unica per protocolli diversi
  (integrazione con gli altri protocolli)
- Modificabilità e condivisione delle informazioni

Ampia scelta di interfacce testuali e grafiche
Possibilità di estensioni sperimentali del sistema

Componenti

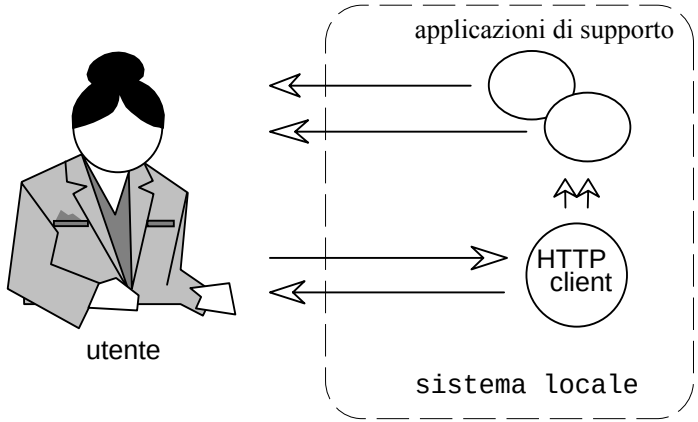
- Browser (presentazione e gestione richieste)
- Server (accesso e invio informazioni)
- Helper applications (particolari presentazioni)
- Applicazioni CGI (esecuzione remota)

Specifiche standard

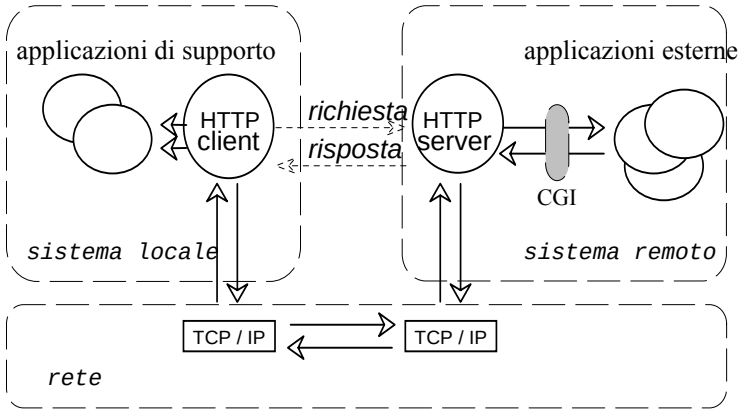
- Sistema di nomi universale URI e URL
  (Uniform Resource Identifier/Location)
- Protocollo HTTP (HyperText Transfer Protocol)
- Linguaggio HTML (HyperText Markup Language)
- Interfaccia CGI (Common Gateway Interface)

SISTEMA **WWW**

Cliente e sua interazione

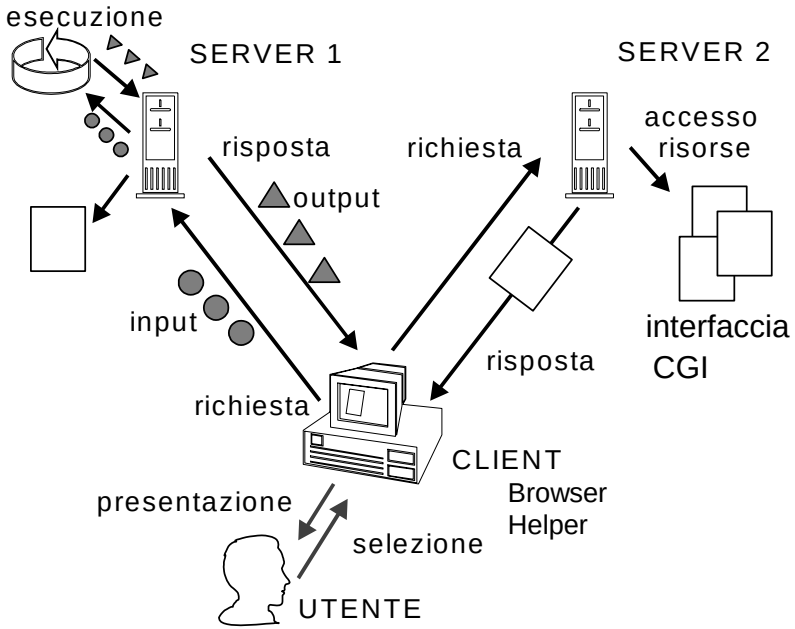


Il Cliente HTTP usa un modo cliente/servitore nei confronti di un server per volta e può anche interagire con risorse locali



Le interazioni cliente/servitore usano il protocollo TCP creando una connessione per ogni informazione da ritrovare (tipica porta fissa **80**)

Implementazione WWW



Modello di comunicazione
client / server
- server parallelo
- server stateless
  (ma uso di **Common Gateway Interface**)

Funzionalità offerte
- Accesso ipertestuale a risorse informative
- Esecuzione applicazioni remote
  - ◊ invio di input da utente
  - ◊ presentazione output con helper

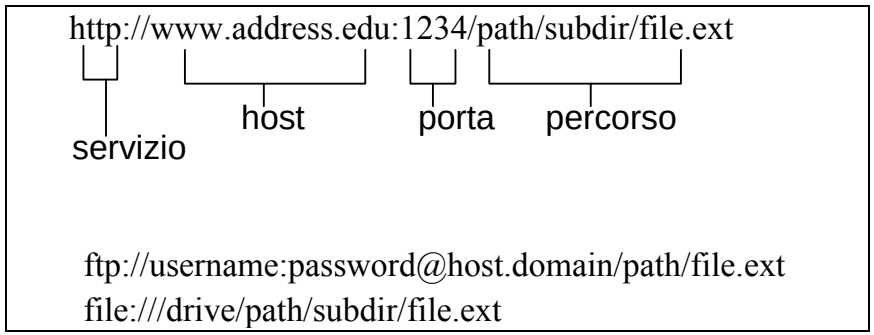
URL Uniform Resource Locators

nomi unici per le risorse del sistema
specificati dal cliente per determinare il servitore

- ***URN*** *(Uniform Resource Names) e servizio di name*
- **Uniform Resource Locators (URL):**
  - nodo contenente la risorsa (*documento o dati*)
  - protocollo di accesso alla risorsa (*e.g. http, gopher*)
  - numero di porta TCP (*porta di default del servizio*)
  - localizzazione della risorsa nel server.

```
<protocollo>[://<host>][:<porta>][<percorso>]
```

Sono riconosciuti i servizi internet e relativi protocolli =>
http, gopher, ftp, wais, telnet, news, nntp, e mail



Uso di default per localizzare risorse

Un **URL** può anche determinare un insieme di risorse:
ad esempio versioni multilingue tra cui scegliere

HTTP **H**yper**T**ext **T**ransfer **P**rotocol

protocollo di interfaccia tra cliente e servitore

Uso di TCP e di connessione (porta **80** default)

Caratteristiche HTTP:

- **request/response**
- **one-shot connection**
- **stateless**

Request/response: richiesta e ricezione di dati.
One-shot connection: la connessione TCP è mantenuta per il tempo necessario a trasmettere i dati
Stateless: non mantiene nessuna informazione tra una richiesta e la successiva

in genere:

- **richiesta** del cliente con **informazioni** per il **server**
- **risposta** con informazioni dal server

il cliente può determinare una forma di scelta (**negoziazione**) sulle informazioni ed i servizi

```
HTTP-message =
   / Simple-Request          ; HTTP/0.9
   / Simple-Response
   / Full-Request            ; HTTP/1.0
   / Full-Response
```

NON c'è **stato del server**

HTTP request/response

Formato del messaggio di **richiesta**:

indirizzo del server	metodo di richiesta	campi opzionali	path+ nome file (in generale, i dati)

Metodo di richiesta

- GET richiesta di leggere una pagina web
- HEAD richiesta di leggere l'header di una pagina web (es. contiene data ultima revisione del documento)
  (usato nelle tecniche di caching nei client)
- PUT richiesta di pubblicare una pagina web
- POST append di dati (di solito HTML form)
- DELETE rimuove una pagina web

Campi opzionali:

- from: identità dell'utente richiedente (debole forma di autenticazione degli accessi)
- referer: il documento contenente il link che ha portato al documento corrente

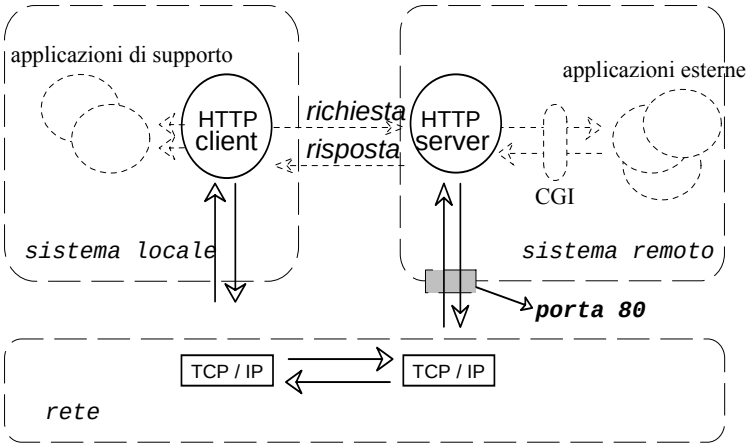
GET URL protocol vers.	headers	delimitatori	(dati opzionali)
	Browser tipi accettati	**CRLF**	**Dati Utente**

Formato del messaggio di **risposta** (CRLF)

status code	informazioni sull'oggetto	dati

- **status code**: successo o fallimento (es. file not found)
- **informazioni sull'oggetto**: data di modifica, tipo di oggetto. Ogni tipo di oggetto (immagine, HTML, audio) attiva una gestione specifica da parte del client

HTTP One-shot connection



1. Il browser (http client) riceve un URL dall'utente
2. Il browser richiede al DNS di risolvere il nome della macchina specificata nell'URL
3. Il DNS risponde con l'indirizzo IP
4. Il browser stabilisce una connessione TCP sulla porta 80 dell'indirizzo IP
5. Il browser manda una richiesta con il metodo:
   GET nome_pagina _web
6. Il server risponde mandando la pagina web richiesta

Attenzione: la presenza in una pagina web di altri oggetti (immagini, applets, ecc.) costringe il client ad aprire una **differente connessione** per ognuno degli oggetti necessari.

Il cliente http può eseguire richieste seguendo protocolli differenti (es. ftp) ⇒ porte differenti

Codice di stato di risposta

solite 3 cifre in chiaro

- **1xx: Informational**

risposta di successo parziale temporanea

- **2xx: Success**

Il server accetta la richiesta

200 OK (Get)
201 Created
206 Partial Content

- **3xx: Redirection**

altre informazioni richieste

301 la informazione mossa definitivamente
302 la informazione mossa temporaneamente
304 la informazione non modificata

- **4xx: Client error**

richiesta non soddisfatta per errore del cliente (errore sintattico o richiesta non autorizzata)

400 Bad Request
401 Unauthorized
402 Payment Required
403 Forbidden
404 Not Found

- **5xx: Server error**

richiesta corretta, ma il server non può soddisfarla (problema del server o di applicazioni CGI)

500 Internal Error
501 Extension Header

Interazione HTTP

Request

```
GET /beta.html HTTP/1.0
Referer: http://www.alpha.com/alpha.html
Connection: Keep-Alive
User-Agent: Mozilla/4.61
Host: www.alpha.com:80
Cookie: name=value
Accept: image/gif, image/jpeg, */*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
Expires: ...
If-modified-since: ...
```

Reply

```
HTTP/1.1 200 OK
Date: Fri, 12 Nov 2001 11:46:53 GMT
Server: Apache/1.3.3 (Unix)
Last-Modified: Mon, 12 Jul 2000 22:55:23 GMT
Accept-Ranges: bytes
Content-Length: 34
Content-Type: text/html
```

Si noti che le cose cominciano ad evolvere per facilitare le operazioni

Keep-alive *la connessione viene mantenuta*
condizioni *richieste condizionali*

proxy *agenti intermedi che fanno cache*

cookie *la possibilità di avere stato della interazione è delegata al cliente che mantiene associazioni nome=valore (cookie)*

HTTP1.1 RFC2616 (1999), state management RFC2109

HTML HyperText Markup Language

HTML è un linguaggio di specifica delle informazioni che deriva da SGML (Standard Generalized Markup Language). E' un **markup language** (TeX, RTF)

I linguaggi markup usano dei **tag** definiti **funzionalmente** per caratterizzare il testo incluso.

tag HTML

testo di tipo header 1: `<H1>testo</H1>`
testo in grassetto: `<STRONG>testo</STRONG>` oppure `<B>testo</B>`

link: `<A HREF = "destinazione"> descrizione </A>`

immagini: `<IMG SRC = "myimage.gif">`

applet Java:

```
<APPLET CODE="Hello.class" WIDTH=100 HEIGHT=80>
```

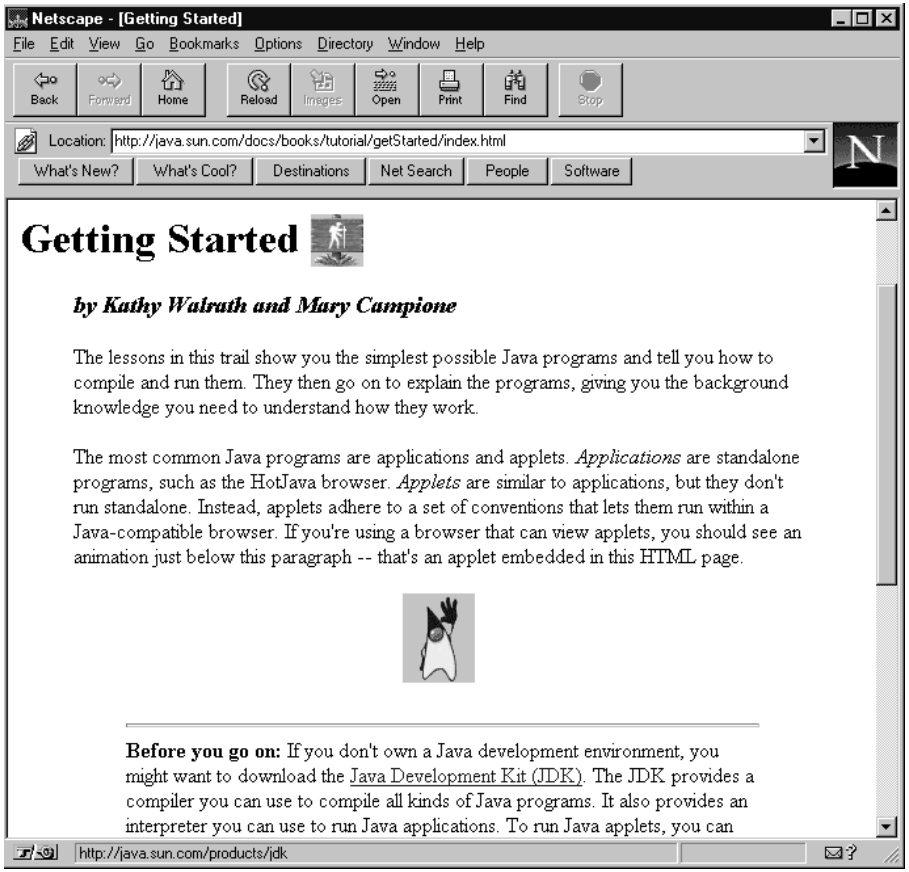
HTML **molto semplice** per non complicare il cliente
Visualizzazione dipendente dal browser

versione	browser	proprietà
1.0	storico	header, liste, enfasi
2.0	Mosaic	Inline Image, form
2.1	Netscape/Microsoft	tabelle, allineamento
3.2	Netscape/Microsoft	frame, ...

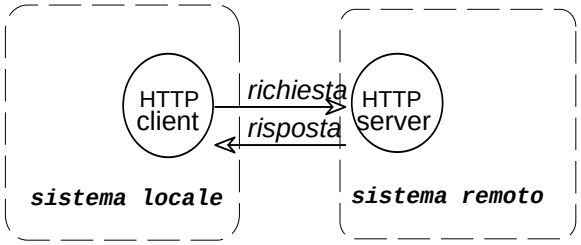
Esempio pagina HTML (codice)

```
<head> <title>Getting Started</title> </head>

<body>
<h1> Getting Started <img src=../images/Start.gif
height=40 width=40 align=top>
</h1>
<p>
<h3><em>by Kathy Walrath and Mary
Campione</em></h3>
<p>
The lessons in this trail show you the simplest
possible Java programs and tell you how to compile
and run them. They then go on to explain the
programs, giving you the background knowledge you
need to understand how they work.
<p>

...........

<p align=center>
<center>
<applet code=Animator.class codebase="../example"
width=55 height=68>
        <param name=endimage value=10>
        <param name=pauses value="2500|100">
</applet>
</center>
</p>

<hr>
<strong>Before you go on:</strong> If you don't
own a Java development environment, you might want
to download the
<a href="http://java.sun.com/products/jdk"> Java
Development Kit (JDK)</a>. The JDK provides a
compiler you can use to compile all kinds of Java
programs. It also provides an interpreter you can
use to run Java applications. To run Java applets,
you can ............
```


Esempio pagina HTML (visualizzazione)



Programmazione client/server in WWW



Possibilità di avere ***risposta*** con informazioni dinamiche

Che tipo di elaborazione delle informazioni e
dove viene eseguita

richiesta	risposta	tipo di elaborazione
Documento HTML	statica (la pagina è un file, non modificabile)	semplice trasferimento file dal server
CGI	dinamica	qualunque elaborazione sul nodo server
Java applet	statica	codice dal server non modificabile, esecuzione sul client
Java applicazione	dinamica	server elabora dinamicamente il codice (in base alla richiesta), esecuzione dinamica sul client

Modelli di Programmazione e mobilità

Dimensioni di classificazione del modello Client/Server:

- trasferimento del **C**odice
- trasferimento dei **Dati**
- trasferimento della **E**xecution **U**nit (lo stato di esecuzione di un processo, senza ripartire dall'inizio)

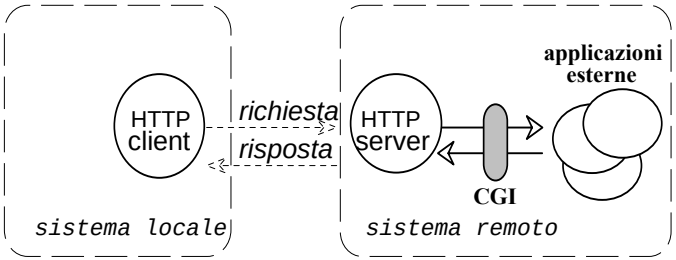
modello	trasferimento codice	trasferimento dati	trasferimento EU
RPC	NO	SI	NO
COD	SI	NO	NO
REV	SI	SI	NO
MA	SI	SI	SI

RPC: Remote Procedure Call
COD: Code On Demand (Applets java)
REV: Remote Evaluation (remote execution UNIX)
MA: Mobile Agents

Mobilità **strong** e mobilità **weak**

Common Gateway Interface (CGI)

CGI --> **standard per la interazione** con le **risorse del server** aldifuori del Web



CGI è uno **standard** per la interfaccia con applicazioni esterne (residenti sulla macchina server)

CGI fornisce all'utente la capacità di eseguire una applicazione sulla macchina server remota

Uso di un direttorio specifico per contenere i programmi
/cgi-bin/

URL da eseguire del programma
http://host/cgi-bin/program

Strumento tipicamente **non interattivo**
Collo di bottiglia

richiesta	*risposta*	*tipo di elaborazione*
CGI	dinamica	qualunque, sul nodo server

Programmazione CGI

Una applicazione CGI permette agli utenti di eseguire una applicazione sul nodo dove risiede il server www.

Applicazioni CGI possono essere scritte in:

- C/C++
- Fortran
- PERL
- TCL
- Any Unix shell
- Visual Basic
- AppleScript

normale attivazione di programma Unix, modello **filtro** con variabili di ambiente predefinite

Devono rispettare l'interfaccia CGI con il server

Interfaccia tra **server Web** e la applicazione **CGI**

- **variabili di ambiente**
- **linea di comando**
- **standard input**
- **standard output**

Variabili d'ambiente
utilizzate dal server per dare informazioni alla CGI:

- `SERVER_NAME`, `SERVER_PORT`,
  - nome nodo server o suo indirizzo
- `SERVER_SOFTWARE`,
  - nome e versione del server HTTP
- `PATH_NAME`,
  - cammino per il programma CGI
- `GATEWAY_INTERFACE`,
  - versione interfaccia CGI cui il server aderisce
- `REQUEST_METHOD`,
  - metodo invocato nella richiesta GET/POST
- `QUERY_STRING`,
  - parametri passati nella GET
- `REMOTE_HOST, REMOTE_ADDR, AUTH_TYPE, REMOTE_USER, CONTENT_TYPE, CONTENT_LENGTH,`

Linea di comando
richieste di tipo ISINDEX, per ricerche di testo nei documenti
Le parole da ricercare sono inserite dal server sulla linea di comando della applicazione CGI (compatibilità)

Standard input
il server ridirige sull'ingresso della applicazione CGI i dati ricevuti dal client.
numero di byte variabile d'ambiente `CONTENT_LENGTH`
tipo dei dati MIME nella `CONTENT_TYPE`

Standard output
l'applicazione CGI manda il risultato elaborato sullo standard output al server, che prepara i dati (HTML) e li spedisce al client

Client HTTP → server HTTP → CGI

Tipicamente, uso di **form**

```
<TITLE>Esempio di Form </TITLE>
<H1>Esempio di Form </H1>

<FORM METHOD="POST" ACTION="http://www-
lia.deis.unibo.it/cgi-bin/post-query">

Inserisci del testo: <INPUT NAME="entry">

e premi per invio: <INPUT TYPE="submit"
                         VALUE="Invio">
</FORM>
```

Visualizzazione **form**

Client HTTP → server HTTP → CGI

Attributi del **form tag**

```
<TITLE>Esempio di Form </TITLE>
<H1>Esempio di Form </H1>

<FORM METHOD="POST" ACTION="http://www-
lia.deis.unibo.it/cgi-bin/post-query">

Inserisci del testo: <INPUT NAME="entry">

e premi per invio: <INPUT TYPE="submit"
                    VALUE="Invio">
</FORM>
```

Dove:

ACTION URL di chi processa la query

METHOD metodo usato per sottomettere il form:

- **POST** il form con i dati è spedito come data body (metodo consigliato)
- **GET** il form con i dati è spedito attaccato all'URL (`action?name=value&name=value`)

caso GET

```
http://www-lia.deis.unibo.it
                /cgi-bin/get-query?entry=testo
```

caso POST

```
http://www-lia.deis.unibo.it
                /cgi-bin/post-query
```

e come data body:

```
                entry=testo
```


server HTTP → CGI

la CGI deve decodificare i dati in arrivo
arrivati con **POST o GET**

- gli spazi ("") sono rimpiazzati da "+"
- gli elementi del form sono coppie chiave - valore
  tipo nome=antonio
- ogni coppia è separata da &
- caratteri non numerici sono introdotti oda un carattrere di escape "!"-> %21

caso GET
leggiamo i dati come stringhe dall'environment
var -> `QUERY_STRING`

caso POST
leggiamo i dati dallo standard input

I passi della CGI

- **decodificare i parametri**
- **eseguire il programma**
- **riportare il risultato al Web server usando lo standard output**

(pagina HTML o statica o dinamicamente costruita)

Applicazione CGI → server HTTP

Applicazione CGI usa **standard output** per mandare al server i dati
I dati devono essere preceduti da un header

Tipi di dati forniti:

- **full document** con il corrispondente **MIME** type (text/html, text/plain per testo ASCII, etc.)

  Esempio: per spedire una pagina HTML

```
Content-type: text/html

<HTML><HEAD>
<TITLE>output di HTML da script CGI</TITLE>
</HEAD><BODY>
<H1>titolo</H1>
semplice <STRONG>prova</STRONG>
</BODY></HTML>
```

- **reference** a un altro documento
  Esempio: riferisco un documento gopher

```
Content-type: text/html
Location: gopher://httprules.foobar.org/0

<HTML><HEAD>
<TITLE > Sorry ... it moved </TITLE>
</HEAD><BODY>
<H1>go to gopher </H1>
Now available at
<A HREF="gopher://httprules.foobar.org/0">
         new location</A> of gopher server.
</BODY></HTML>
```


Applicazione CGI

Esempio: generazione della pagina di risposta (caso full document)

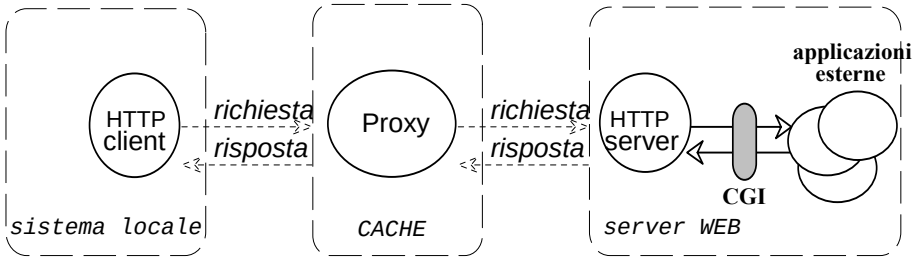
```
#include <stdio.h>
..............


main(int argc, char *argv[]) {
  int cl;
```

generazione di un full document in risposta

```
  printf("Content-type: text/html");

  cl = atoi(getenv("CONTENT_LENGTH"));

  for(x=0;cl && (!feof(stdin));x++) {
    ...
    elaborazione dell'input (stdin)
    ...
  }

  printf("<H1>Query Results</H1>");
  printf("You submitted ...");

  for(x=0; x <= m; x++)
    printf("...........", ... , ....);
}
```

Ambiente Complessivo di Esecuzione

I sistemi Web cominciano a delineare un nuovo modello di operazione
cliente
intermedi (**proxy, cache**)
servitore
con accesso alle **risorse** del servitore



Integrazione con **risorse di calcolo accedute via server**

Sistemi a diversi livelli (multitier)
in cui la elaborazione risultante viene fornita
in modo legacy (**Web compatibile**) ma
ottenuta tramite la interazione di molte entità in gioco

Il sistema comincia ad avere **stato** e a diventare il punto di accesso ad una **infrastruttura**

INTERNET: PRINCIPALI PROBLEMI

Problema del traffico
alcuni formati di informazioni possono richiedere un **eccesso di banda**
necessità di adeguare le infrastrutture

Problema della sicurezza
la trasparenza richiede la integrazione con i sistemi locali di esecuzione/visualizzazione: intrusioni o virus da controllare
riservatezza delle transazioni e autenticazione clienti

Problema della visualizzazione
alcuni formati richiedono una lunga visualizzazione: tempi di accesso molto rallentati

sfruttare la possibile concorrenza/asincronismo

Uso di clienti con strategie adatte a
abbreviare il **tempo di risposta**
favorire il **browsing** delle informazioni
(**web navigation e ricerche mediante robot, spider, worm,** ecc.)

evoluzione dei protocolli

per garantire una apertura alle esigenze man mano determinate

La ricerca di informazioni sul WEB

Web come ipertesto (grafo) con milioni di nodi → problema di reperire le informazioni attraverso i link

Esistono **indici del web** (detti anche cataloghi o directories), realizzati per facilitare la ricerca di informazioni su Internet

Organizzazione **indici**:

- alfabetica
- per argomento (gerarchici)
- per area geografica (gerarchici)
- con possibilità di ricerca (parole chiave)

Gli indici sono costruiti utilizzando dei programmi che esplorano i site web presenti in rete.

Programmi più diffusi:

- **search engine**
- **spider**
- **crawler**
- **worm**
- **knowbots** (**know**ledge ro**bots**)

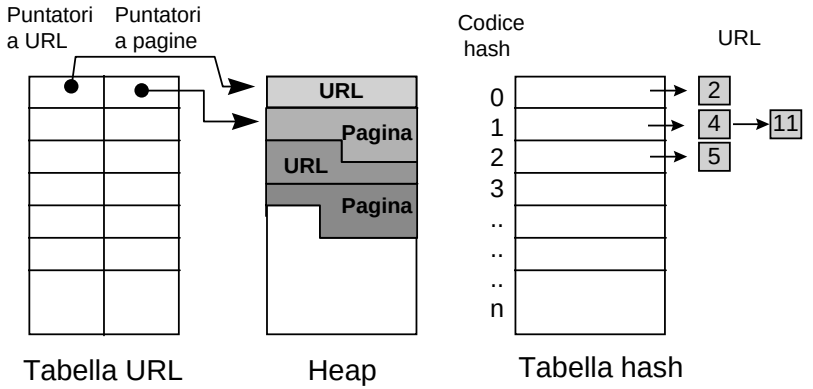
Esempio:
AltaVista (http://altavista.digital.com) Web index:

- 30 milioni pagine da 275,600 servers
- 4 milioni articoli da 14,000 Usenet news groups.

E' acceduto più di 21 milioni di volte al giorno
(dati di Ottobre 96)

I motori di ricerca

Struttura di supporto tipica:



Due fasi: **ricerca** e **indicizzazione**

Passi della **ricerca** (algoritmi **breadth-first**, **depth-first**):

- prelevare un URL
- eseguire hash URL
- Se hash URL è in Tabella hash allora STOP

altrimenti

- aggiungere hash URL in Tabella hash
- aggiungere Puntatori a URL e a pagina in Tabella
- aggiungere URL e Pagina (o titolo) in Heap
- ripetere tutti i passi per ogni link della pagina

Problemi:

- **dimensioni** del grafo web
- **punto di partenza** della ricerca
- **peso** dei link (anche autorità e centralità)
- tipo di ricerca: **depth-first** → stack overflow
  breadth-first → dimensioni heap
- come trattare i link presenti nelle **active map** (CGI)
- **URL obsoleti** e **macchine non** raggiungibili

I motori di ricerca

Fase di **indicizzazione**

La procedura di indexing estrae le parole chiave da ogni pagina (o titolo) web memorizzati nell'heap nella fase di ricerca (sintesi delle pagine)

Per trovare le **parole chiave**:
- si scartano le parole poco significative (articoli, etc.)
- si scelgono parole che nella pagina hanno la frequenza maggiore (es. Lycos)

Per ogni parola ottenuta si memorizza in una tabella la parola e l'URL che la contiene.

Alla fine della indicizzazione si ordina la tabella sulle parole e si salva su file che verrà consultato per le ricerche da parte degli utenti

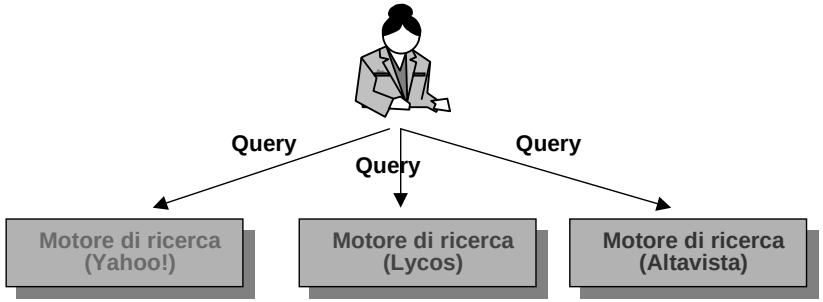
problemi:
- titoli pagine spesso poco significativi
- analisi intere pagine costosa
- pagine solo video o audio, oppure active map

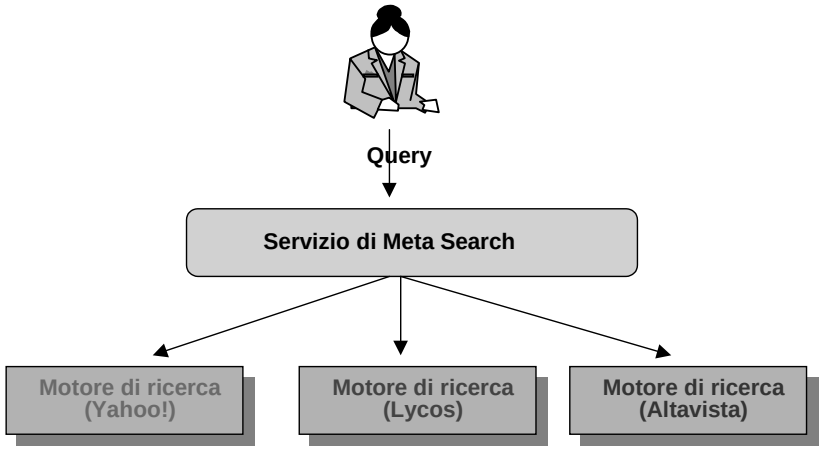
Ricerche e indicizzazioni cooperative
Harvest è un motore di ricerca che richiede a tutti i server www di eseguire una applicazione per indicizzare localmente la macchina
Un motore centrale raccoglie tutti i risultati
Problema della sicurezza

I meta-searcher

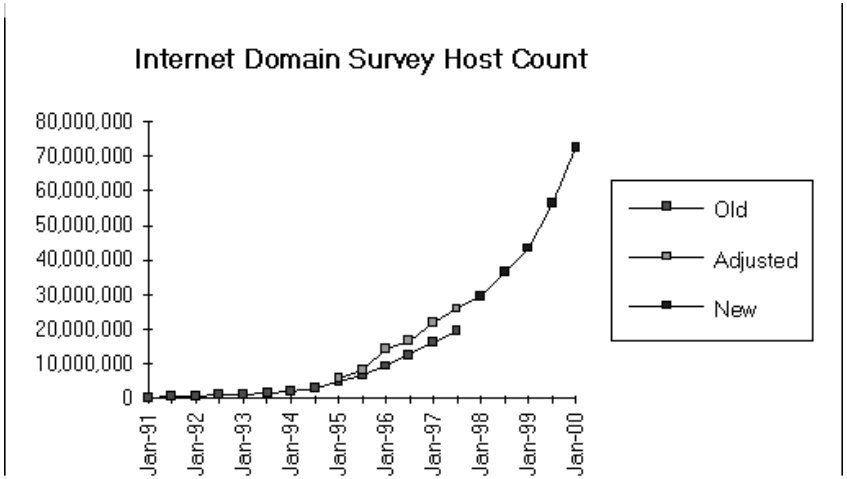
Effettuano la ricerca delle informazioni su più indici del web contemporanemente.

Un meta-searcher invia la stessa query contemporaneamente a più indici del web e NON contiene un database





Dati recenti su Web



Source: Internet Software Consortium (www.isc.org)

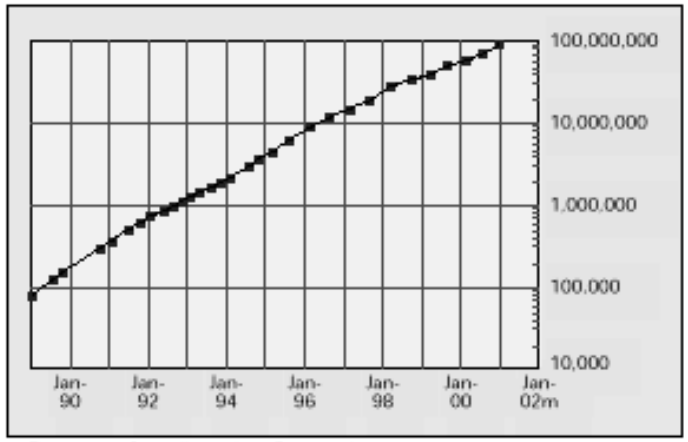


Figure 1. *Overall trends in Internet hosts*