Protocollo HTTP

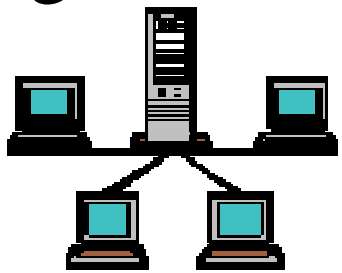
Mario Bravetti

Protocolli di Internet

- IP (Internet Protocol) consente l'instradamento dei pacchetti del mittente al destinatario su reti di vasta scala.
- Protocollo TCP (Transmission Control Protocol) realizza un "canale virtuale": garantisce che i pacchetti giungano dal mittente al destinatario e siano nel corretto ordine.
- Protocolli SMTP (posta elettronica), FTP (trasmissione di files), NNTP (news), HTTP (pagine web) realizzano servizi comunemente usati su Internet.

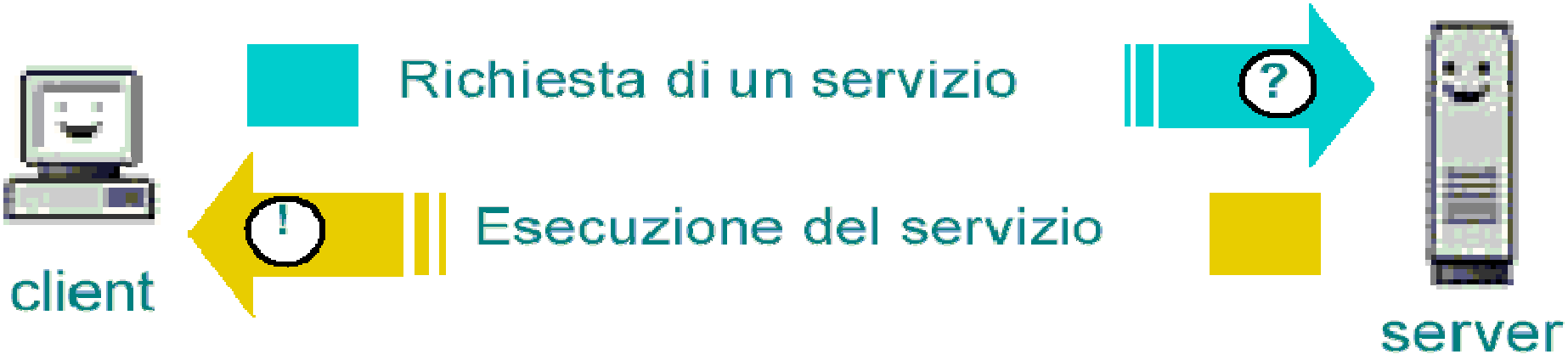
Modelli architetturali di rete

- Modello Client-Server: un programma (client) genera una richiesta e un altro programma (server) che gli risponde (esempio: web)

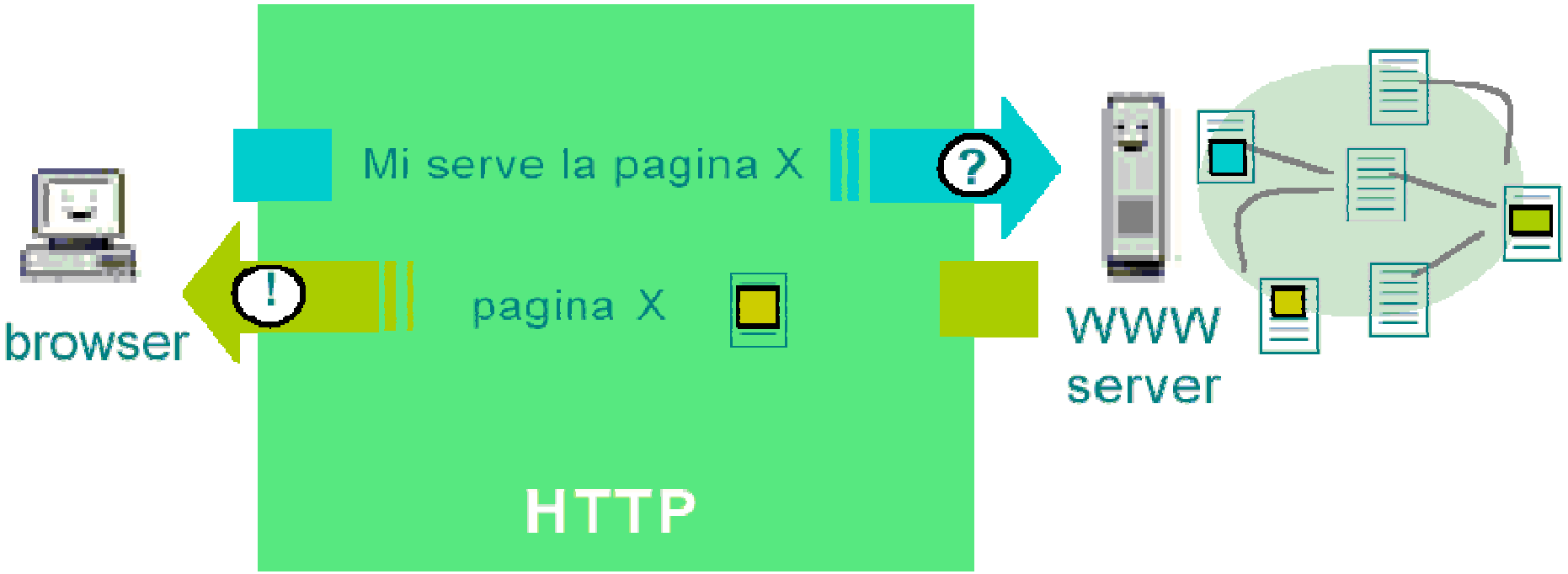


- Modello Peer-to-Peer (P2P): non esiste sistema fornitore di servizi "centrale" ma ogni partner nella comunicazione ha ruoli e funzioni equivalenti e per questo motivo è detto pari
  (esempio: videoconferenza, torrent)

Architettura Client-Server

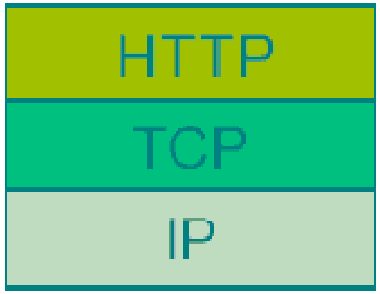


World Wide Web



Protocollo per WWW: HTTP

- HTTP (HyperText Transfer Protocol) è il protocollo che utilizzano client e server WWW per comunicare.
- Versioni: HTTP 1.0 (RFC 1945) e HTTP 1.1 (RFC 2068 con updates successivi in RFC 2616)



Il browser WWW: l'HTTP client

- Esempi: Google Chrome, Mozilla Firefox, Microsoft Internet Explorer, Apple Safari, …
- Principali compiti del browser:
  - Trasmettere al server le richieste di reperimento dati (pagine web ed eventuali allegati)
  - Ricevere dal server le informazioni richieste
  - Visualizzare il contenuto delle informazioni ricevute (eventualmente eseguendo codice lato client ricevuto insieme alla pagina, es. javascript e applet Java)
  - Gestire operazioni locali sulle informazioni ricevute (es. salvare, stampare)

Browser Plug-ins

- Tipicamente il browser è in grado di visualizzare solo alcuni formati di file tra cui l'HTML, i formati GIF e JPEG per le immagini, il testo, ecc.
- Altri formati possono essere visualizzati/gestiti installando sul browser appositi programmi detti plug-in (es. il plug-in per applet Java)

HTTP server

- Esempi: Apache, Microsoft Internet Information Services (IIS), …
- Principali compiti dell'HTTP server:
  - Rimanere in ascolto di richieste da parte dei client
  - Soddisfare le richieste inviando al client il documento richiesto (eventualmente generato dinamicamente tramite esecuzione di codice lato server)
- Il server deve:
  - Rispondere alla richiesta nel modo più efficiente possibile
  - Gestire contemporaneamente più richieste

HTTP Server Plug-ins

- Anche il server ha plug-in, es.
  - Tomcat plug-in per eseguire Servlet/Java Server Pages)
  - PHP plug-in
  - ….

Protocollo HTTP

Protocollo HTTP (2)

- Fasi dell'interazione fra client e server sono:
  - apertura di una connessione TCP
  - invio di una richiesta da parte del client, che specifica la risorsa desiderata;
  - invio di una risposta da parte del server che contiene la risorsa richiesta;
  - chiusura della connessione TCP

HTTP: la richiesta

- La richiesta del client contiene le seguenti informazioni (nell'header):
  - il metodo, cioè il comando, che si chiede al server di eseguire (tipicamente richiesta di un oggetto);
  - il numero di versione del protocollo HTTP, URL dell'oggetto al quale applicare il comando, eventuali parametri (dati) passati all'oggetto;
  - altre informazioni, fra le quali il tipo di client e i tipi di dati che il client accetta.

HTTP: la richiesta (2)

- I metodi definiti in HTTP sono:
  - GET: richiesta di ricevere un oggetto (metodo più usato) passando eventuali parametri all'interno dell'indirizzo, ad esempio:
    `nunky.csr.unibo.it/~mbravett/p.sh.cgi?nome=mario&cognome=bravetti&indirizzo=via+Sacchi`
  - POST: richiesta di ricevere un oggetto passando parametri come dati veri e propri all'interno del body della richiesta HTTP
  - HEAD: richiesta di ricevere la sola intestazione di una pagina
  - DELETE, LINK e UNLINK, PUT: aggiornamento da remoto del sito

GET vs POST

- La differenza primaria tra GET e POST sta nel passaggio dei valori.
- Concettualmente si dovrebbe usare:
  - GET quando i parametri passati al programma lato server sono semplici stringhe
  - POST quando i parametri passati sono un vero e proprio file (es. immagine)
- GET è di default e va bene in tutte le query che non richiedono "data encoding". POST per:
  - dati di input "codificati" (es: mail)
  - uso di caratteri non ASCII
  - stringa di input molto lunga

HTTP: la risposta

- La risposta del server è composta da:
  - una riga di stato, che specifica l'esito della richiesta e indica:
    - la versione del protocollo HTTP
    - un codice numerico di stato (ES: 404 not found)
    - la specifica testuale dello stato
  - alcune meta-informazioni sul tipo di dato in cui è rappresentato l'oggetto richiesto
  - l'oggetto richiesto

HTTP: la risposta (2)

- Alcune meta-informazioni sono le seguenti:
  - **`server`**: tipo di server
  - **`date`**: data e ora della risposta
  - **`content-type`**: tipo dell'oggetto
  - **`content-length`**: numero di byte dell'oggetto
  - **`content-language`**: Linguaggio delle informazioni
  - **`content-encoding`**: tipo di codifica dell'oggetto (es. compressione)

MIME

- MIME (Multipurpose Internet Mail Extensions) fu sviluppato come un sistema di specifica e descrizione del contenuto di un messaggio di posta elettronica
- Con il tempo questo modello si è esteso ad altri usi, in particolare al WWW divenendo un sistema di specifica e descrizione del contenuto di un file inviato da un server HTTP al browser (o con POST in richiesta)

MIME (2)

- Le caratteristiche di MIME sono state definite inizialmente tramite l'RFC 1341.
- In seguito sono state aggiornate le RFC 1521, 1522 e 1523.
- Attualmente sono di riferimento l'RFC 1896, e le RFC dalla 2045 alla 2049.

MIME (3)

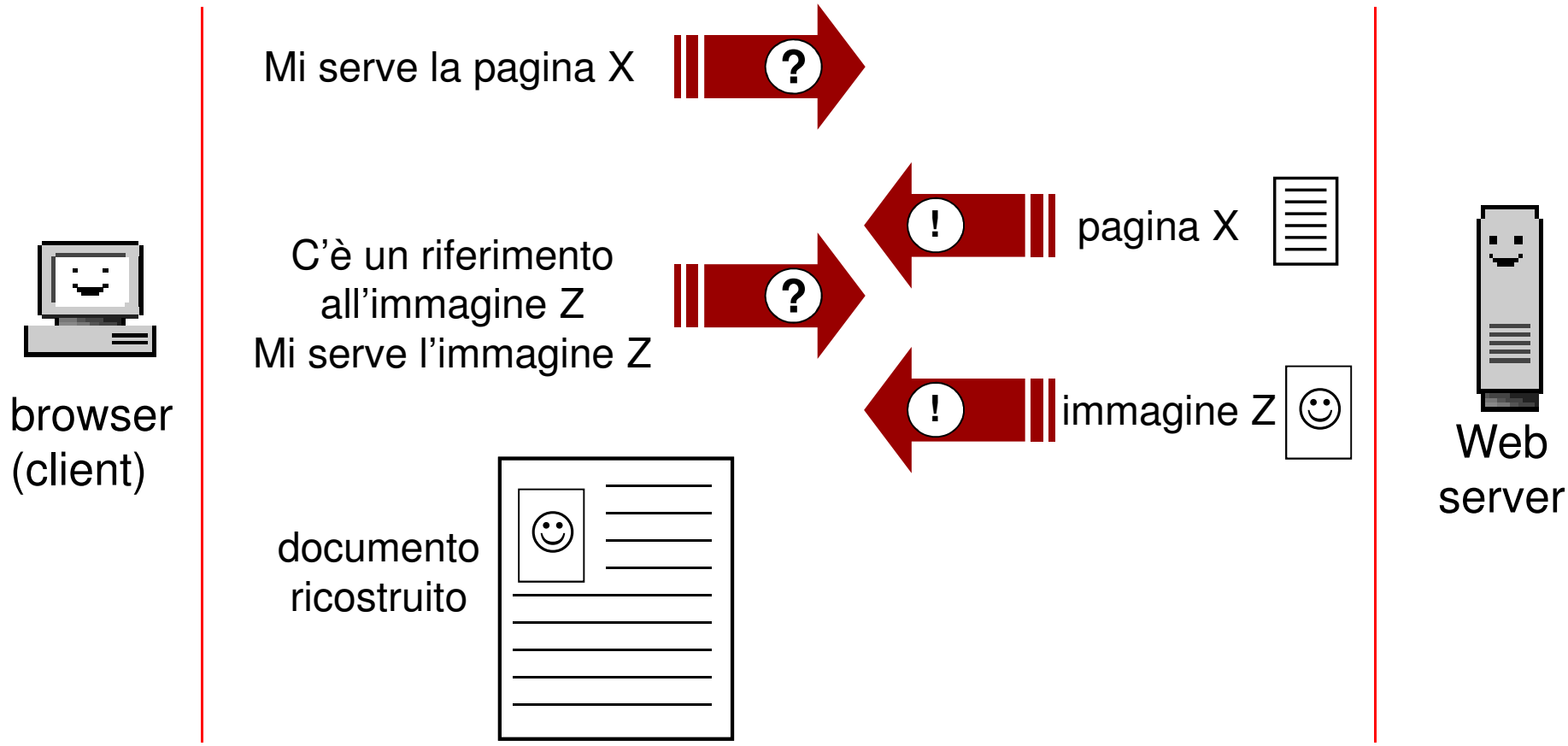
- Il tipo MIME di un documento è indicato nel campo *Content-Type:* in modo che il destinatario possa individuare il tipo di file trasmesso.

- I tipi MIME sono nella forma *tipo/sottotipo*

MIME (4)

- Alcuni esempi sono:
  - tipo MIME *application*: application/msword, application/pdf, application/postscript, application/zip.
  - tipo MIME *audio*: audio/x-midi, audio/x-wav, audio/mpeg
  - tipo MIME *image*: image/gif, image/jpeg, image/png, image/tiff.
  - tipo MIME *text*: text/html; text/plain (per il formato txt),, text/sgml; text/richtext (per il formato rtf), text/xml;
  - tipo MIME *video*: video/mpeg, video/quicktime, video/x-msvideo (per il formato avi)
  - tipo MIME *multipart* utilizzato nei messaggi di posta elettronica contenenti allegati multipli
  - tipo MIME *multipart/form-data*: per l'upload di files (mulitpli) con una form (richiesta POST)

Ricostruire un documento



Linguaggio pagine web: HTML

- HTML (HyperText Markup Language) è il linguaggio con cui vengono descritte le pagine web trasmesse dal server HTTP al browser.
- HTML è un linguaggio di markup: i documenti ipertestuali HTML sono file di testo che contengono appositi "marcatori" (detti tag) che definiscono il formato di layout del testo e i link.

<I> ciccio </I> *ciccio*

uso esteso di HTTP: cookies

- Cookies (informazioni salvate nel client su ordine del server) gestite tramite comandi nell'header della richiesta/risposta HTTP:
  - Set-Cookie HTTP Response Header: il server spedisce al client il comando di assegnare dei cookie.
  - Cookie HTTP Request Header: quando effettua una richiesta il client spedisce al server il valore attuale di tutti cookie di pertinenza.

uso esteso di HTTP: servizi web

- Richieste HTTP effettuate da un programma anziché da una pagina HTML
  - come se il sever contenesse oggetti remoti e si invocassero metodi su tali oggetti (RPC)
- Formato di dati inviati in richiesta e ricevuti in risposta interoperabile (es. XML)
- Uso "libero" di tipo mime per i files inviati con una richiesta POST (es. text/xml)
  - invece per richieste da HTML (form) deve essere multipart-form/data

uso esteso di HTTP: servizi web

- Ma uso anche di altri metodi HTTP solitamente non utilizzati es. PUT, DELETE.
- Servizi come gestori di risorse identificate da URL
  - PUT url: scrive la risorse url inviandone il contenuto
  - DELETE url: cancella la risorsa url
  - POST url: modifica risorsa url o crea una sua sottorisorsa (identificata da "url/nomerisorsa")
  - GET url: legge la risorsa url e ne restituisce il contenuto (cache se nessuna PUT, DELETE o POST precedente!)
- Contenuto chiamato rappresentazione (es. in XML)
  - servizi REST (Representational State Transfer)