Università di Bergamo

Dipartimento di Ingegneria dell'Informazione e Metodi Matematici

5 - Protocolli Applicativi

Architetture e Protocolli per Internet

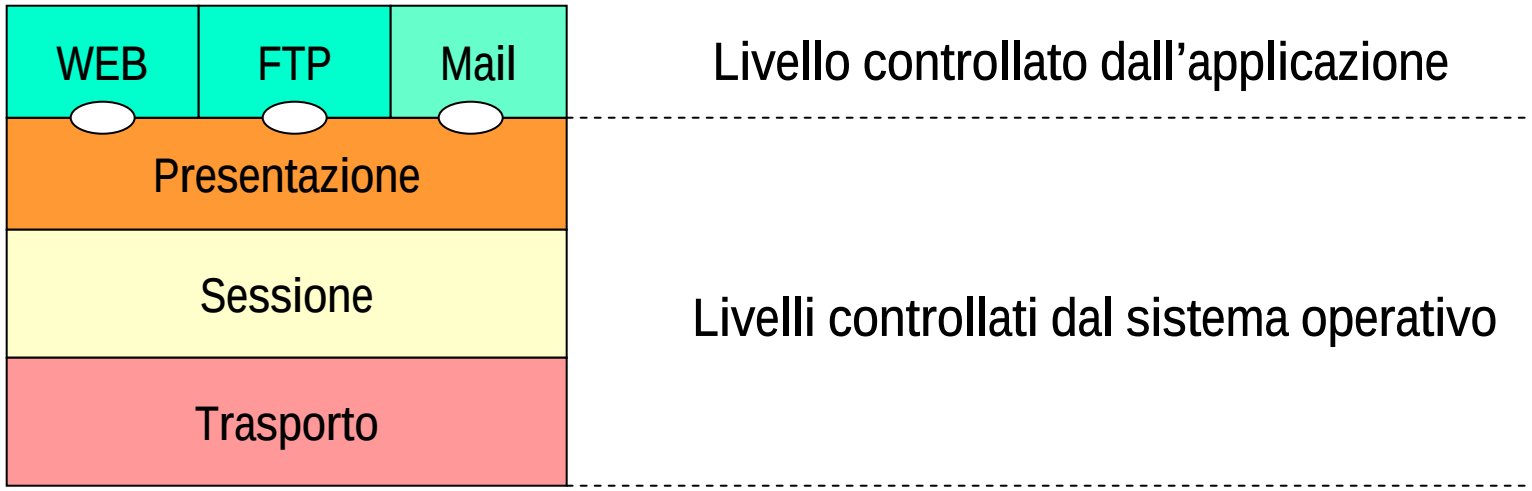
Processi e Protocolli

- **Processi in esecuzione su sistemi remoti possono scambiarsi informazioni e servizi mediante una rete**
- **L'interazione avviene mediante lo scambio di messaggi**
- **I protocolli applicativi sono le regole e i formati con i quali i processi costruiscono i messaggi e ne interpretano il significato**
- **I protocolli applicativi sono solo una parte delle applicazioni di rete:**

Applicazioni di rete	Protocolli di rete
Web (web server, browser, HTML)	HTTP
E-mail (mail server, mail client, MIME)	SMTP

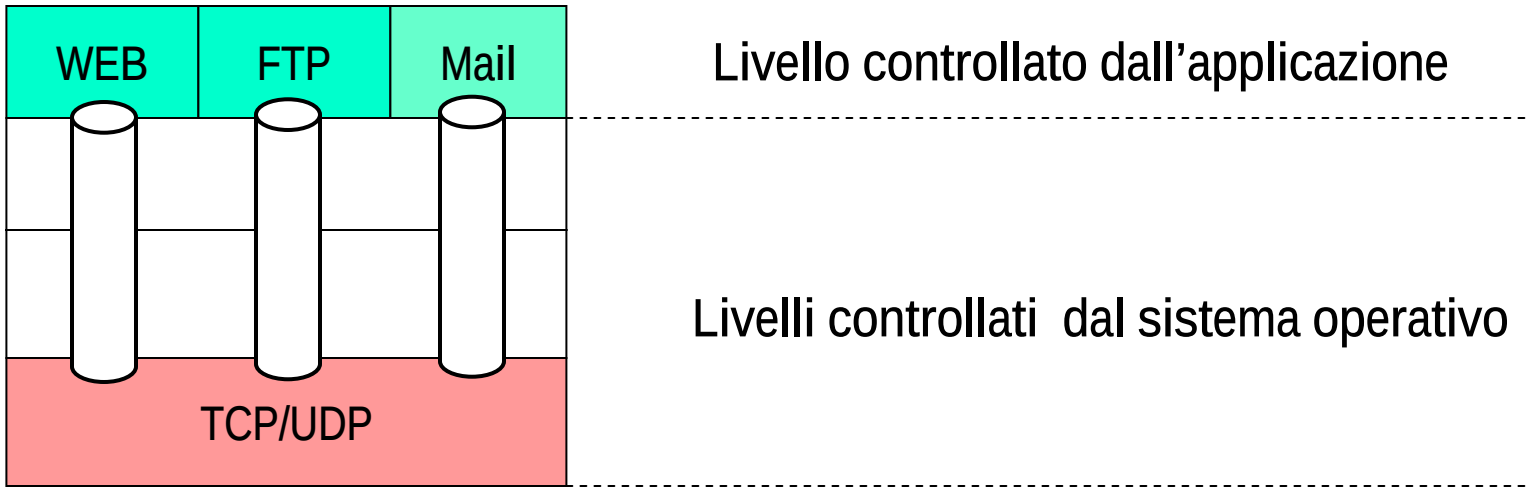
Interazione coi livelli inferiori: architettura OSI

- **Lo scambio di messaggi fra i processi applicativi avviene utilizzando i servizi dei livelli inferiori attraverso i SAP (Service Access Point)**
- **Ogni processo è associato ad un SAP**
- **Applicazioni nella pila OSI:**



Interazione coi livelli inferiori: architettura TCP/IP

- I protocolli applicativi che si appoggiano sull'Internet Protocol si appoggiano direttamente sul protocollo di trasporto (TCP oppure UDP)
- I SAP vengono chiamati *porte* e l'insieme porta TCP/UDP e indirizzo IP viene chiamato *socket*



Architettura Client-Server

- Ovviamente esiste una stretta relazione tra la struttura del programma software in esecuzione sugli host remoti ed il protocollo applicativo
- Lo scopo fondamentale del colloquio tra processi remoti è quello di fornire servizi. In particolare, due sono le funzioni che possono essere svolte da un processo applicativo
  - richiedere servizi
  - fornire servizi
- Se ogni processo applicativo svolge una sola delle due funzioni siamo in presenza di un'interazione di tipo client-server

Architettura Client-Server

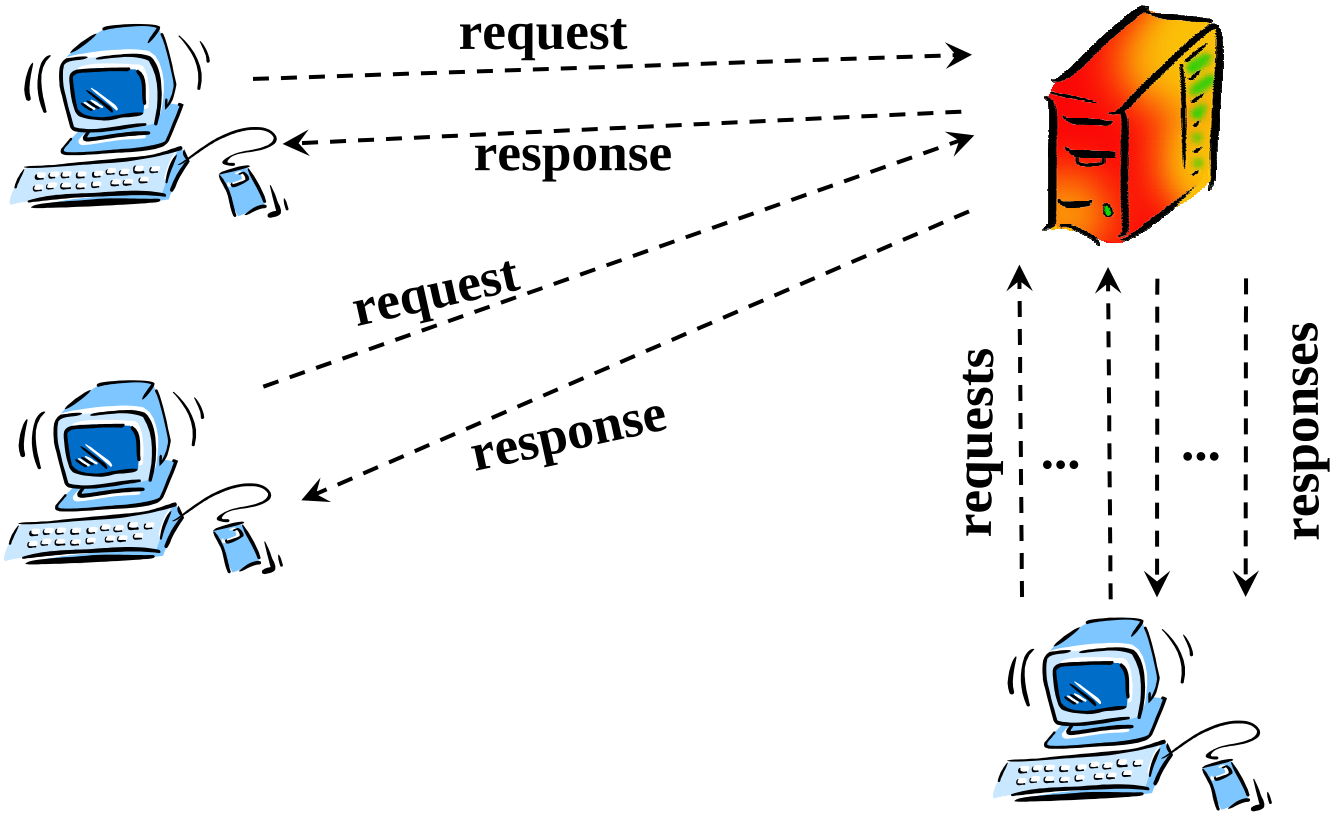
- Un processo client è solo in grado di fare richieste di servizio (informazioni) e di interpretare le risposte
- Un processo server ha solo il compito di interpretare le richieste e fornire le risposte
- Se è necessario nello stesso host sia fare richieste che fornire risposte vengono usati due processi, client e server.
- Un protocollo applicativo per un'architettura client-server rispecchia questa divisione di ruoli e prevede messaggi di richiesta (request) generati dal lato client e messaggi di risposta (response) generati dal server

Programmi Client e Server

- Distinguiamo tra programma (software) e processo, istanza del programma in esecuzione su un host
- Un processo server è in esecuzione a tempo illimitato sul proprio host (daemon) e viene attivato mediante una *passive open*
- Un processo client viene attivato solo al momento di fare le richieste e viene attivato mediante una *active open* su richiesta dell'utente o di altro processo applicativo
- La passive open del server fa sì che da quel momento il server accetti richieste dai client
- l'active open del client richiede l'indicazione dell'indirizzo e della porta del server

Programmi Client e Server

- Normalmente più client possono inviare richieste ad uno stesso server
- Un client può fare più richieste contemporanee



Programmi Client e Server

- **Un client può essere eseguito in modalità parallela o seriale**
  - **esempio: invia più richieste in parallelo per i file che compongono una pagina web**
- **Anche un server può essere eseguito in modalità parallela o seriale**
- **Normalmente gli applicativi che usano UDP sono gestiti in modo seriale:**
  - i pacchetti con le richieste arrivati al server vengono immagazzinati e attendono il loro turno; il server li esamina e genera le risposte

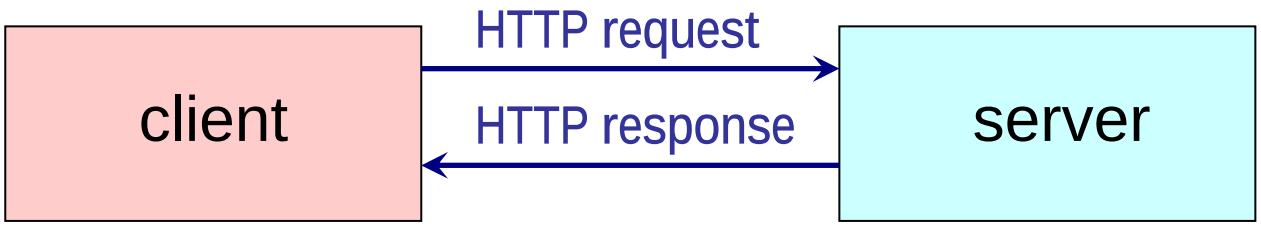
Programmi Client e Server

- Normalmente i server che usano TCP vengono eseguiti in modalità parallela e sono dunque in grado di rispondere a più richieste contemporaneamente
- Con ognuno dei client viene aperta una connessione TCP che viene mantenuta per il tempo necessario a scambiare richieste e risposte
- La gestione delle procedure per ciascun client collegato avviene mediante la generazione di processi figli
- Si parla di applicativi *multi-thread*
- La generazione di un figlio è detta *fork*

HyperText Transfer Protocol (HTTP)

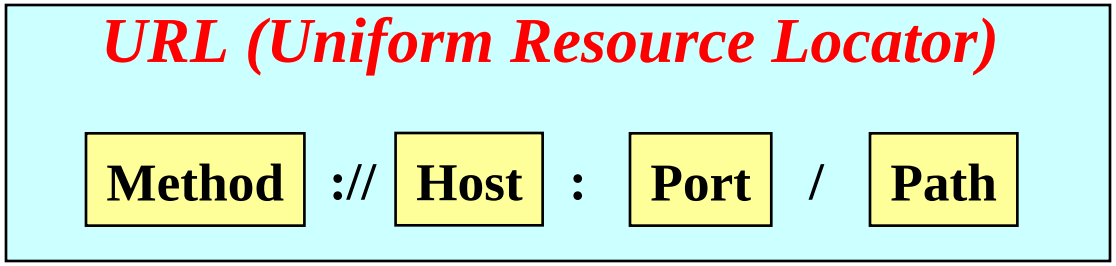
- "Hypertext Transfer Protocol -- HTTP/1.0," RFC 1945, May 1996.
- "Hypertext Transfer Protocol -- HTTP/1.1," RFC 2068, January 1997

- **architettura client-server**
- **i client richiedono oggetti (file) identificati da un URL al server**
- **i server restituiscono i file**
- **nessuna memoria sulle richieste viene mantenuta nei server (protocollo *stateless*)**



Trasporto dei messaggi

- **HTTP fa uso di TCP per il trasporto dei messaggi**
- **una web page è di solito composta da un documento base (HTML) e più oggetti collegati**
- **la richiesta di una web page fa uso di**



`http://www.unibg.it/index.html`

Indica il protocollo applicativo

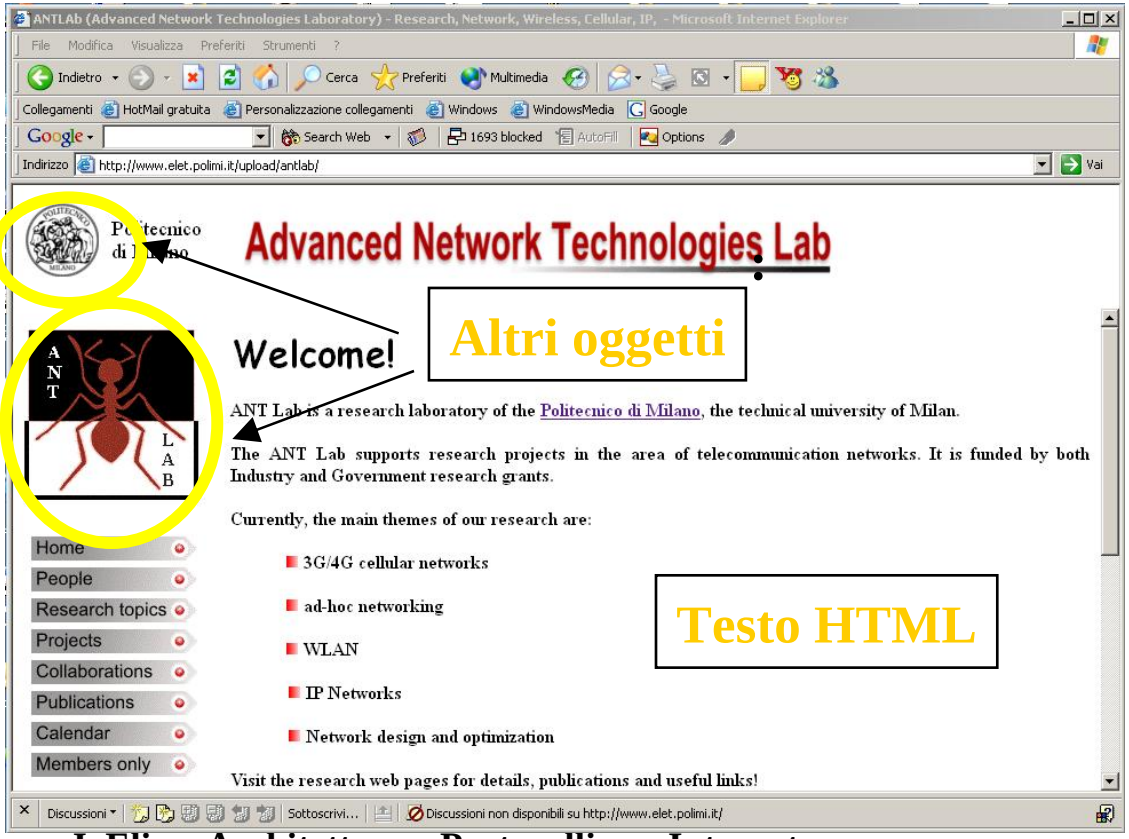
Indica l'indirizzo di rete (IP) del server

Indica la pagina web richieste

La porta TCP viene definita per default (80)

Trasporto dei messaggi

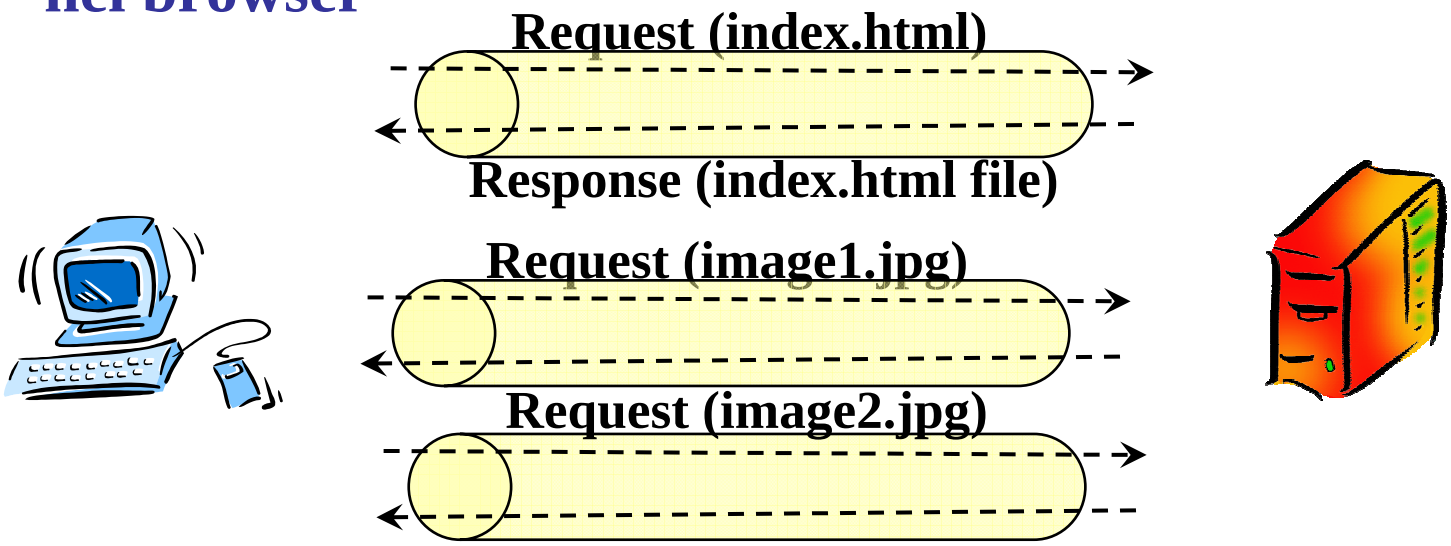
- **Supponiamo che un client richieda una pagina HTML di un server al cui interno sono contenuti i riferimenti ad altri oggetti (ad esempio 10 figure che compongono la pagina e che occorre visualizzare insieme al testo HTML).**



- **nel trasferimento dell'insieme di oggetti sono possibili 2 modalità**
- **Non-persistent connection (default mode di HTTP 1.0)**
- **Persistent connection (default mode di HTTP 1.1)**

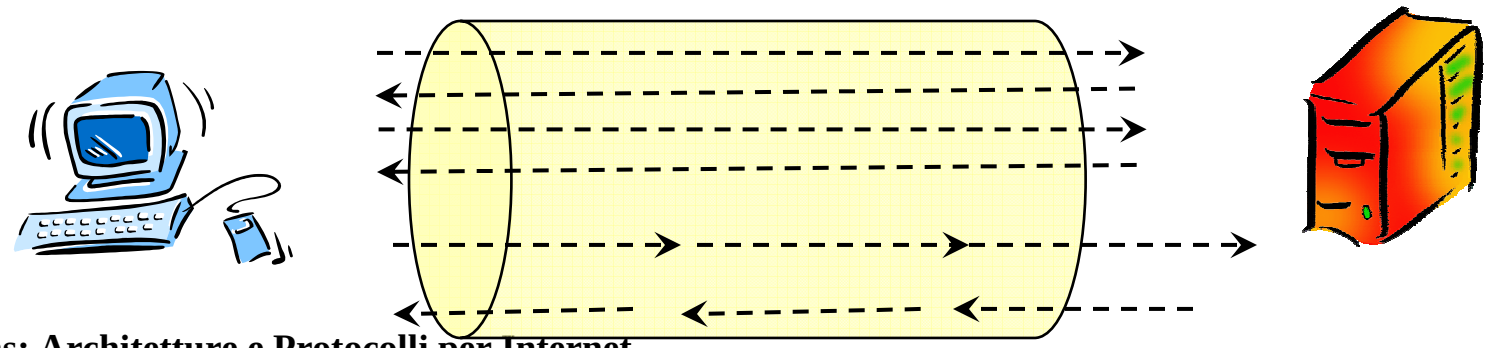
Non persistent

- Viene aperta una connessione per una sola request-response: inviata la pagina, il server chiude la connessione
- La procedura viene ripetuta per tutti i file collegati al documento HTML base
- Le connessioni TCP per più oggetti possono essere aperte in parallelo per minimizzare il ritardo
- Il numero massimo di connessioni è di solito configurabile nel browser



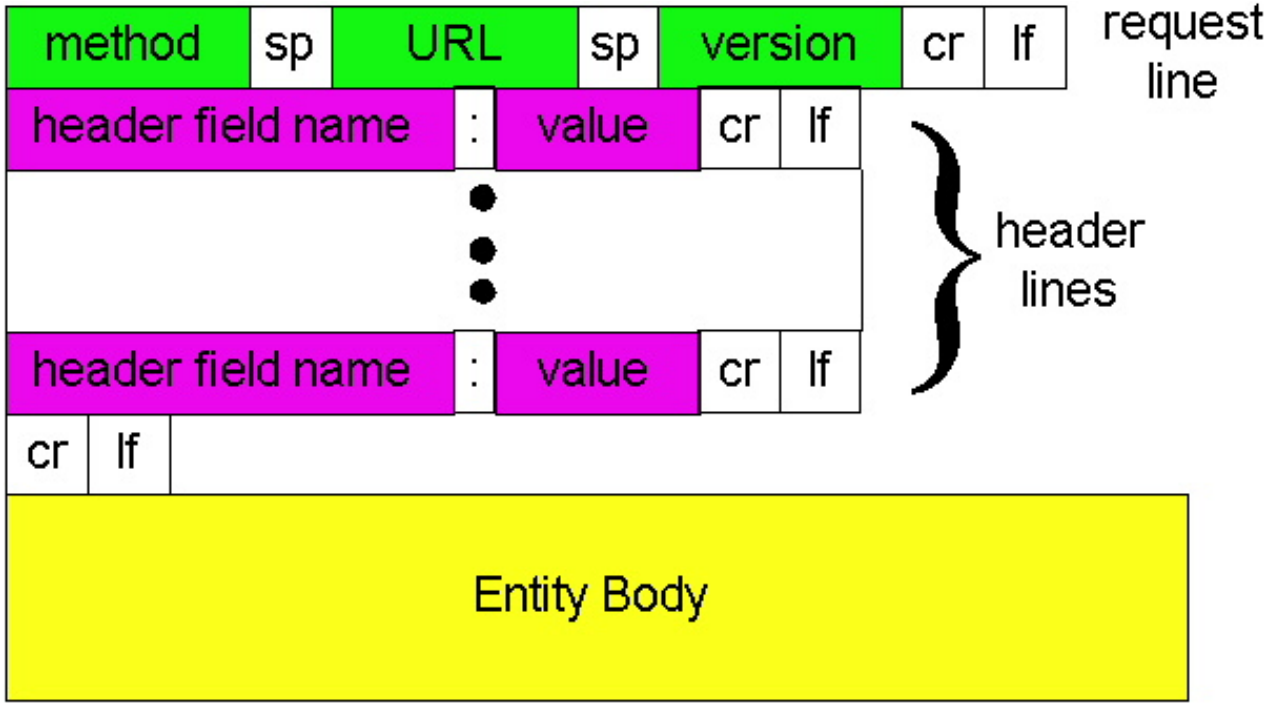
Persistent connection

- Nel caso *persistent* il server non chiude la connessione dopo l'invio dell'oggetto
- La connessione rimane aperta e può essere usata per trasferire altri oggetti della stessa pagina web o anche più pagine
- I server chiudono di solito la connessione sulla base di un *time-out*
  - *without pipelining:* il client invia una nuova richiesta solo dopo aver ricevuto la risposta per la precedente
  - *with pipelining:* più richieste vengono inviate consecutivamente dal client



Messaggi

Richieste




```
GET /ntw/index.html HTTP/1.1
Connection: close
User-agent: Mozilla/4.0
Accept: text/html, image/gif, image/jpeg
Accept-language:it
```

Messaggi

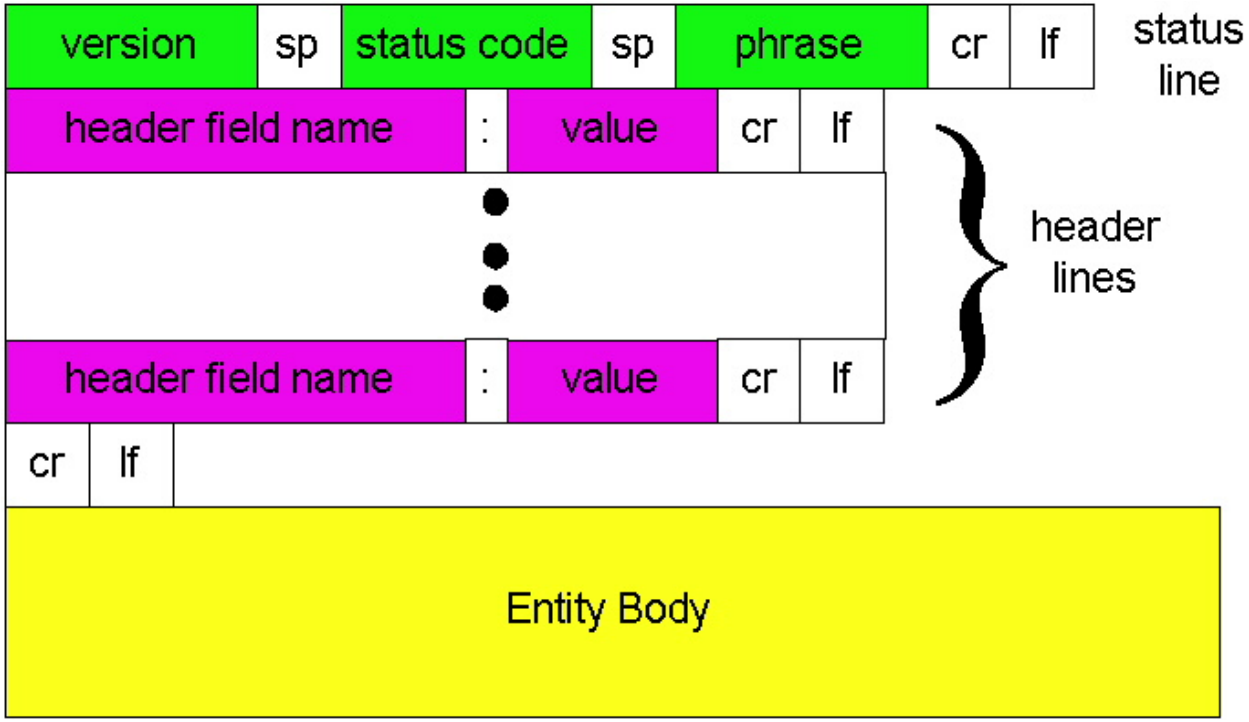
Methods:

GET	**E' usato quando il client vuole scaricare un documento dal server. Il documento richiesto è specificato nell'URL. Il server normalmente risponde con il documento richiesto nel corpo del messaggio di risposta.**
HEAD	**E' usato quando il client non vuole scaricare il documento ma solo alcune informazioni sul documento (come ad esempio la data dell'ultima modifica). Nella risposta il server non inserisce il documento ma solo degli header informativi.**
POST	**E' usato per fornire degli input al server da utilizzare per un particolare oggetto (di solito un applicativo) identificato nell'URL.**
PUT	**E' utilizzato per memorizzare un documento nel server. Il documento viene fornito nel corpo del messaggio e la posizione di memorizzazione nell'URL.**

- **Altri methods:**
  - **PATCH, COPY, MOVE, DELETE, LINK, UNLINK, OPTIONS.**

Messaggi

Risposte



Messaggi

Status codes:

- **1xx Informational**

`100 Continue:`	Prima parte della richiesta accettata.

- **2xx Success**

`200 OK:`	La richiesta ha avuto successo; l'informazione è inclusa

- **3xx Redirection**

`301 Moved Permanently:`	L'oggetto è stato spostato nell'URL indicato
`302 Found (Moved Temporarily):`	L'oggetto è stato spostato nell'URL indicato

- **4xx Client error**

`400 Bad Request:`	errore generico
`401 Unauthorized:`	Accesso senza necessari account e passwd
`404 Not Found:`	l'oggetto non esiste sul server

- **5xx Server error**

`500 Internal server error`	Errore o guasto nel server
`501 Not implemented`	Funzione non implementata
`503 Service unavailable`	Servizio non disponibile

Header

Header name : **Header value**

- Gli header servono per scambiare informazione di servizio aggiuntiva
- E' possibile inserire più linee di header per messaggio
- Esempi:

`Cache-control`	Informazione sulla cache
`Accept`	Formati accettati
`Accept-language`	Linguaggio accettato
`Authorization`	Mostra i permessi del client
`If-modified-since`	Invia il doc. solo se modificato
`User-agent`	Tipo di user agent

Messaggi

- **Esempio: di richiesta oggetto**

HTTP è testuale (ASCII)

```
GET /ntw/index.html HTTP/1.1
Connection: close
User-agent: Mozilla/4.0
Accept: text/html, image/gif, image/jpeg
Accept-language:it
```

- **Esempio: risposta**

```
HTTP/1.1 200 OK
Connection: close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 09:23:24 GMT
Content-Length: 6821
Content-Type: text/html
data data data data data ...
```

Cache locale: get condizionato

- E' possibile evitare di scaricare oggetti memorizzati nella memoria locale se non sono stati modificati

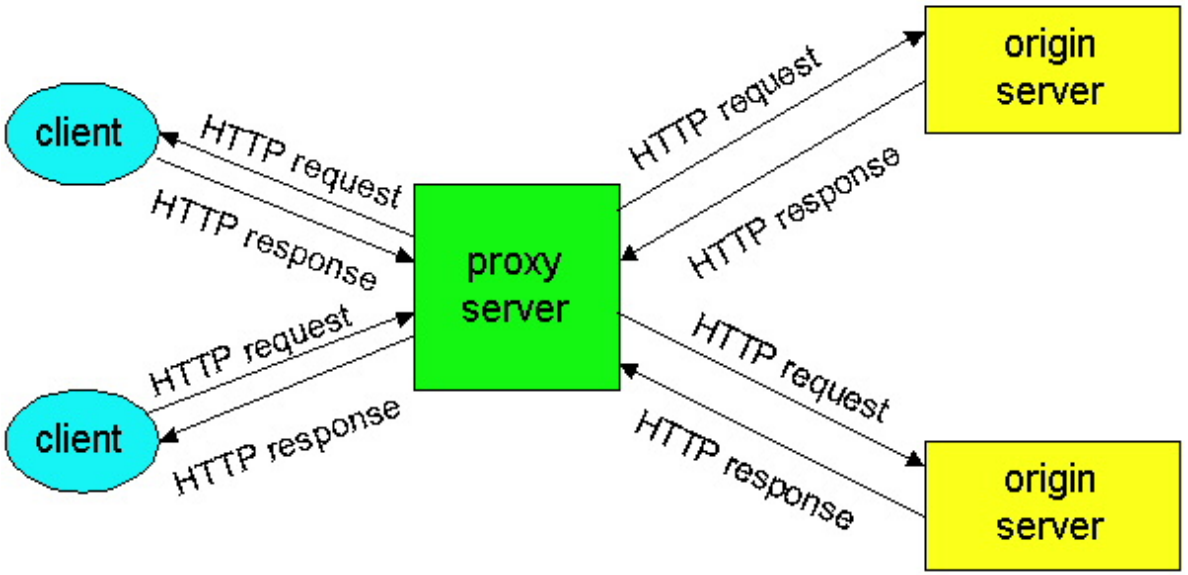
```
Client:
GET /fruit/kiwi.gif HTTP/1.0
User-agent: Mozilla/4.0
Accept: text/html, image/gif, image/jpeg
If-modified-since: Mon, 22 Jun 1998 09:23:24
```

```
Server:
HTTP/1.0 304 Not Modified
Date: Wed, 19 Aug 1998 15:39:29
Server: Apache/1.3.0 (Unix)
```
(empty entity body)

- E' possibile anche usare il metodo HEAD

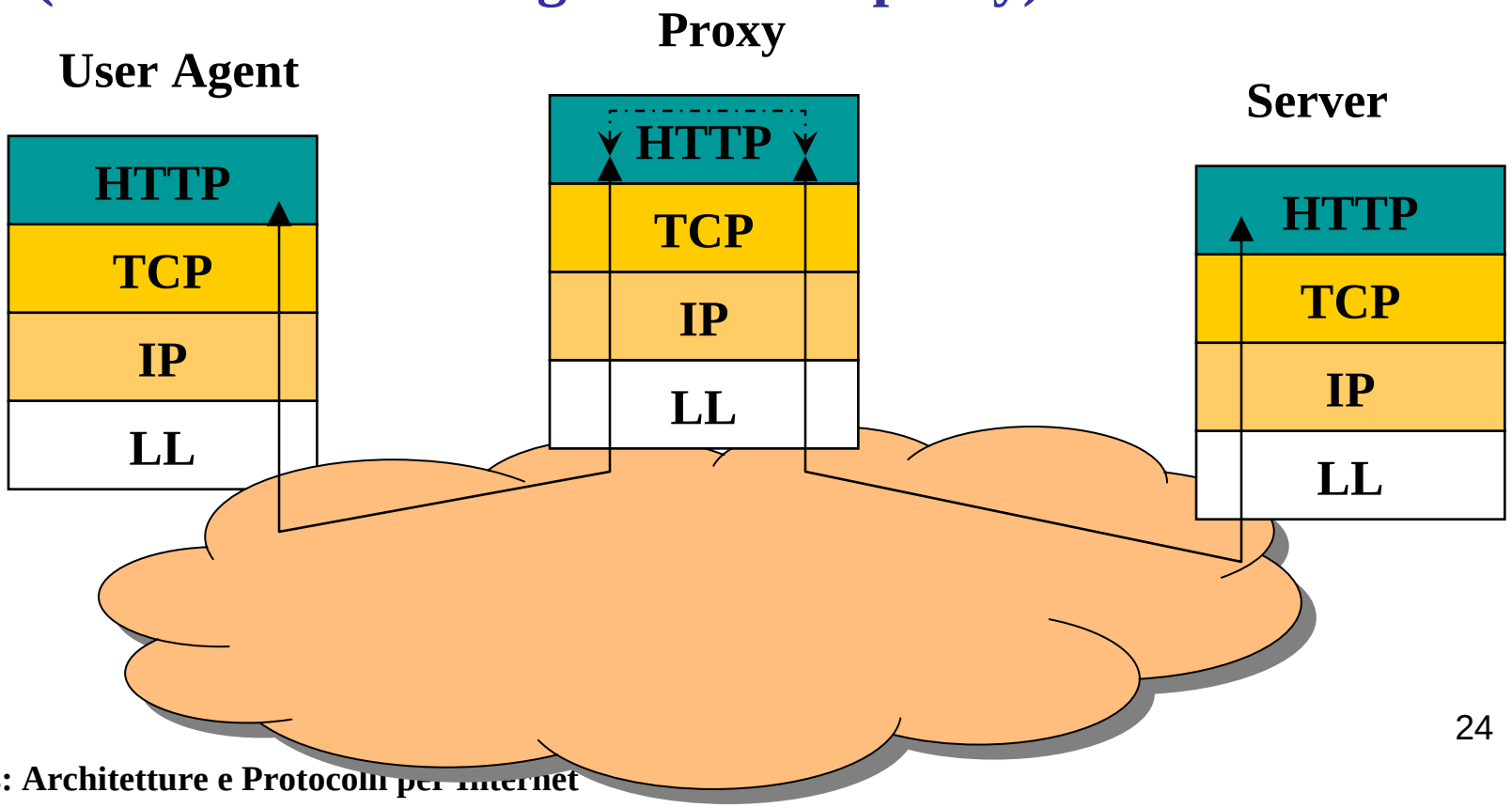
Cache di rete: uso dei proxy

- Compito principale dei proxy è fornire una grande memoria di cache
- Se un documento è contenuto nella cache viene scaricato più velocemente sul client



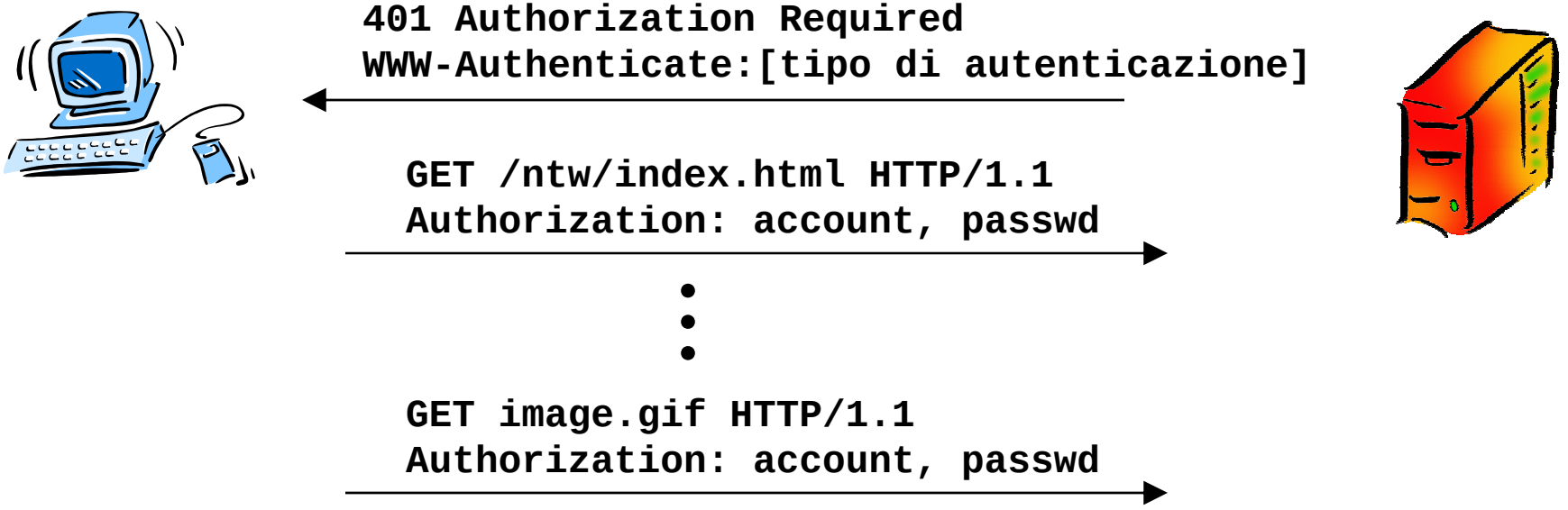
Proxy

- I proxy sono degli application gateway, ovvero degli instradatori di messaggi di livello applicativo
- Devono essere sia client che server
- Il server vede arrivare tutte le richieste dal proxy (mascheramento degli utenti del proxy)



Autenticazione

- HTTP è stateless e quindi non si possono riconoscere richieste successive dello stesso utente
- in HTTP esiste un elementare meccanismo di autenticazione (account e password) che serve a riconoscere gli utenti
- Normalmente il browser memorizza passwd e account in modo da non richiedere la digitazione ogni volta



Cookie

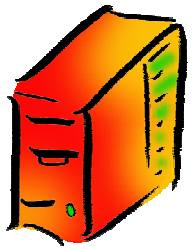
- Esiste anche un altro modo per riconoscere richieste successive di uno stesso utente che non richiede di ricordare password
- Il numero di cookie inviato dal server viene memorizzato in un opportuno file
- Cambiando il numero di cookie ad ogni operazioni (ad es. e-commerce) si può mantenere uno stato “virtuale” per ciascun utente.

```
GET /ntw/index.html HTTP/1.1
```

```
200 OK
Set-cookie:18988466
```

```
GET /ntw/carrello/index.html HTTP/1.1
Cookie: 18988466
```

```
GET image.gif HTTP/1.1
Cookie: 18988466
```

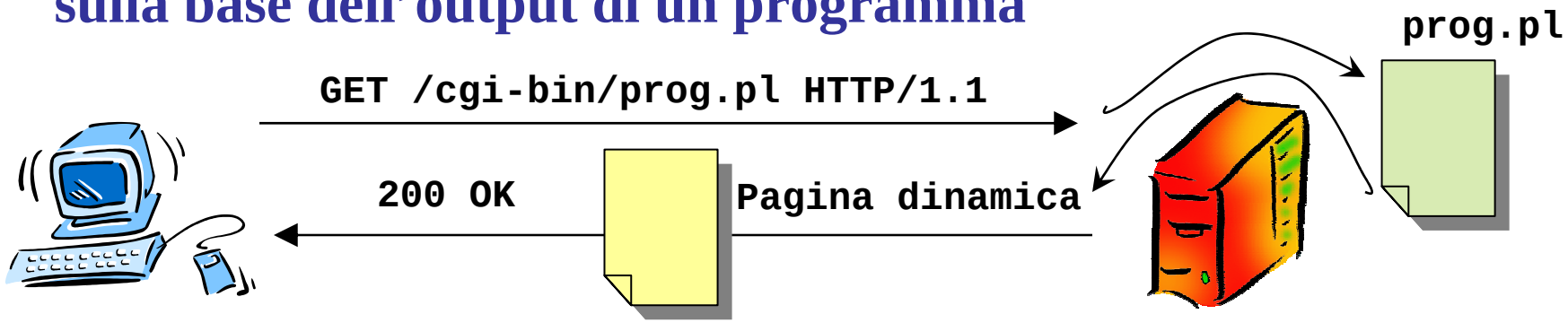


HTML
(HyperText Markup Language)

- HTTP trasferisce file e non si occupa della loro semantica
- Il funzionamento del WWW si basa sull'interpretazione di file e sulla loro visualizzazione
- Pagine di testo formattate sono trasferite come file ASCII mediante dei comandi di formattazione specificati nel linguaggio HTML
- Le pagine HTML possono contenere riferimenti ad altri oggetti che dal browser possono essere interpretati
  - come parte del documento da visualizzare (immagini)
  - come link ad altre pagine web
- Se una pagina HTML è memorizzata nel server e viene inviata su richieste è una **pagina statica**

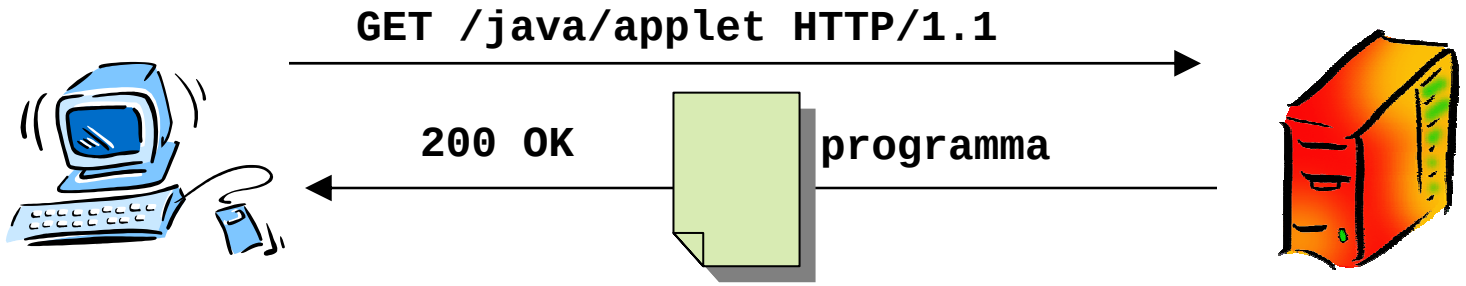
Pagine WEB dinamiche

- Se una pagina viene creata al momento della richiesta (e talora in base alle informazioni fornite del client) si parla di pagine dinamiche
- Se una richiesta si riferisce ad una pagina dinamica il server esamina la richiesta, esegue un programma associato a quella richiesta e genera la pagina di risposta sulla base dell'output di un programma



Pagine WEB attive

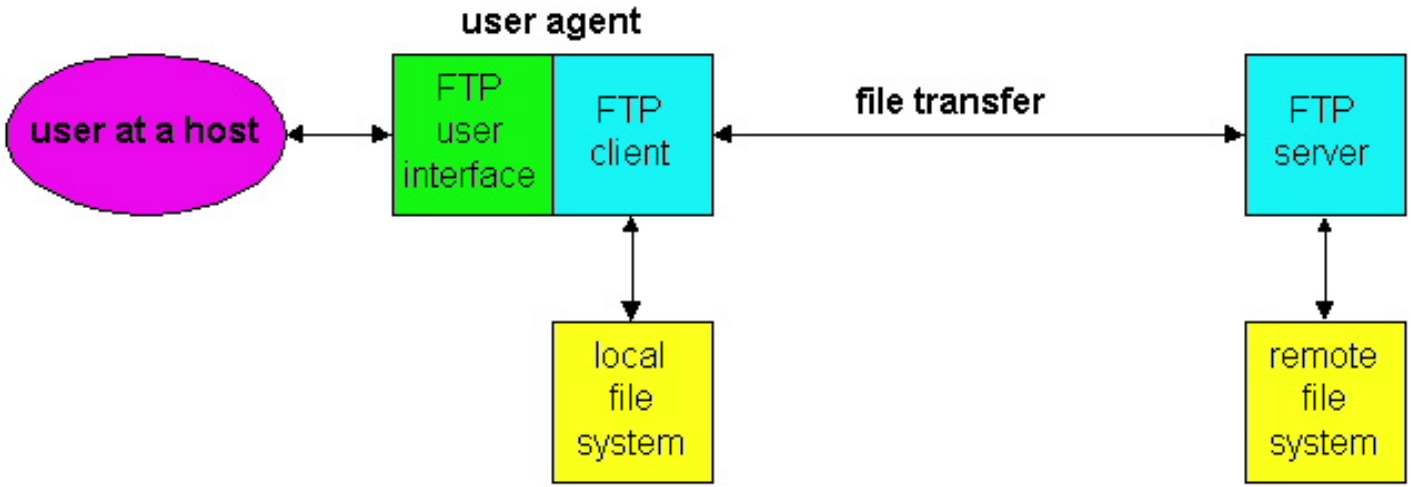
- Una pagina web può anche contenere un programma che deve essere eseguito dal client
- Il programma viene scaricato come un oggetto della pagina ed eseguito in locale sulla macchina del client
- Può essere utile per ottenere delle pagine in grado di interagire con l'utente, per grafici in movimento, ecc.



File Transfer Protocol (FTP)

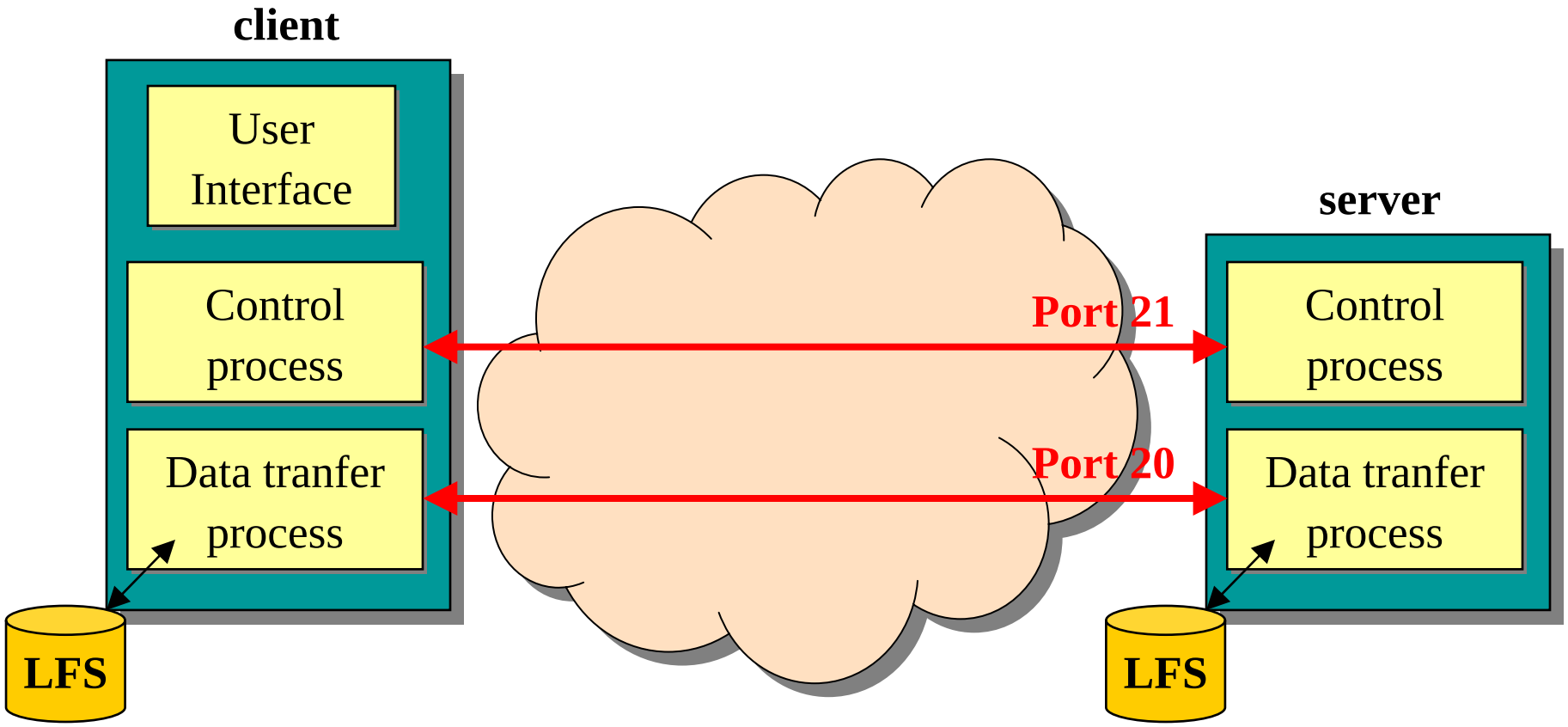
• "File Transfer Protocol", RFC 959, October 1985.

- **E' un protocollo usato per il trasferimento di file tra due host remoti**
- **Sia sul lato client che sul lato server l'applicazione opera direttamente sul file system della macchina**

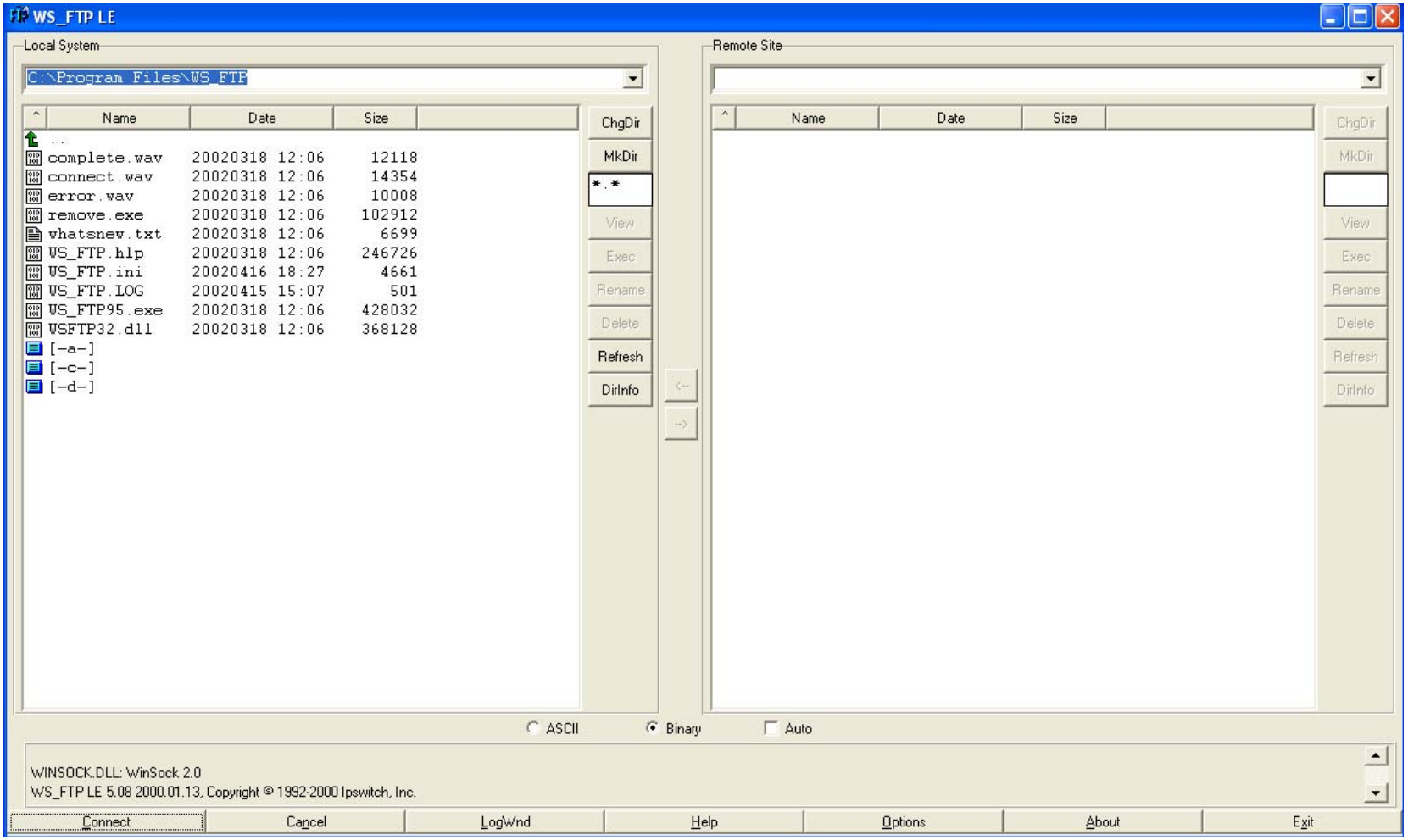


File Transfer Protocol (FTP)

- Fa uso di TCP per il trasporto
- Vengono aperte due connessioni TCP: una per i dati e una di controllo

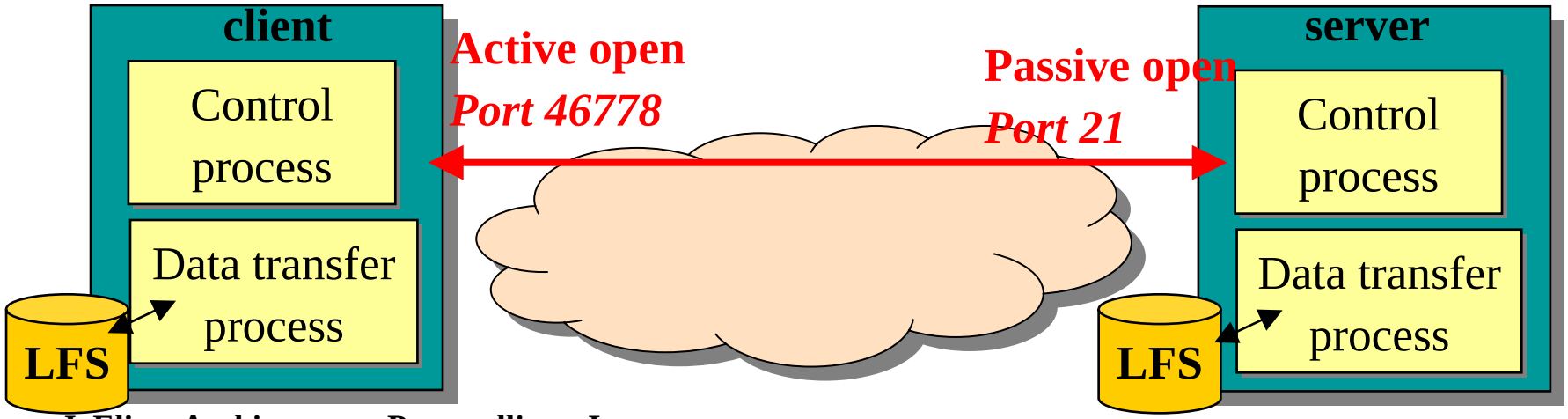


FTP: user interface



FTP: connessione di controllo

- La connessione di controllo viene aperta in modo simile alle altre applicazioni
  - Il server lancia una passive open sulla porta 21 e rimane in attesa di richieste di connessione
  - Il client lancia un active open ogni volta deve iniziare una sessione di trasferimento file e fa partire una richiesta di connessione TCP usando una porta dinamica
- La connessione di controllo è *persistent*, ovvero rimane aperta per tutta la durata della sessione di trasferimento e può essere usata per molti file da trasferire



FTP: connessione dati

- Le connessioni dati sono *non-persistent* ovvero sono aperte solo per trasferire un file o altre informazioni e poi sono immediatamente chiuse
- Per aprire una connessione dati:

  Metodo 1:
  - Il client che desidera iniziare un trasferimento dati effettua una passive open su una porta di sua scelta
  - Il client comunica la porta al server sulla connessione di controllo mediante il comando PORT
  - Il server fa un active open verso la porta del client usando il suo numero di porta noto 20

  Metodo 2:
  - Il client invia il comando di PASV al server
  - Il server sceglie un numero di porta, fa una passive open e comunica il numero di porta al client nella risposta
  - Il client fa un active open verso la porta comunicata dal server

FTP: connessione dati

- Il trasferimento di dati può avvenire con diverse modalità e formati:
- File type:
  - ASCII file: file di caratteri
  - Binary: formato generale per tutti i file non testuali
- Transmission mode:
  - Stream mode: il file viene trasferito al TCP come una sequenza non strutturata di byte
  - Block mode: il file viene trasferito come una serie di *data blocks*, ovvero in blocchi di cui i primi tre byte rappresentano l'header (*block header*)

FTP: comandi

- Il trasferimento di comandi di FTP è testuale (ASCII)

Comandi di accesso

```
USER username
PASS password
QUIT log out
```

Gestione file

```
CWD  change directory
DELE delete file
LIST list files
RETR retrive file
STOR store file
```

Modalità di trasferimento

```
TYPE file type
MODE transfer mode
```

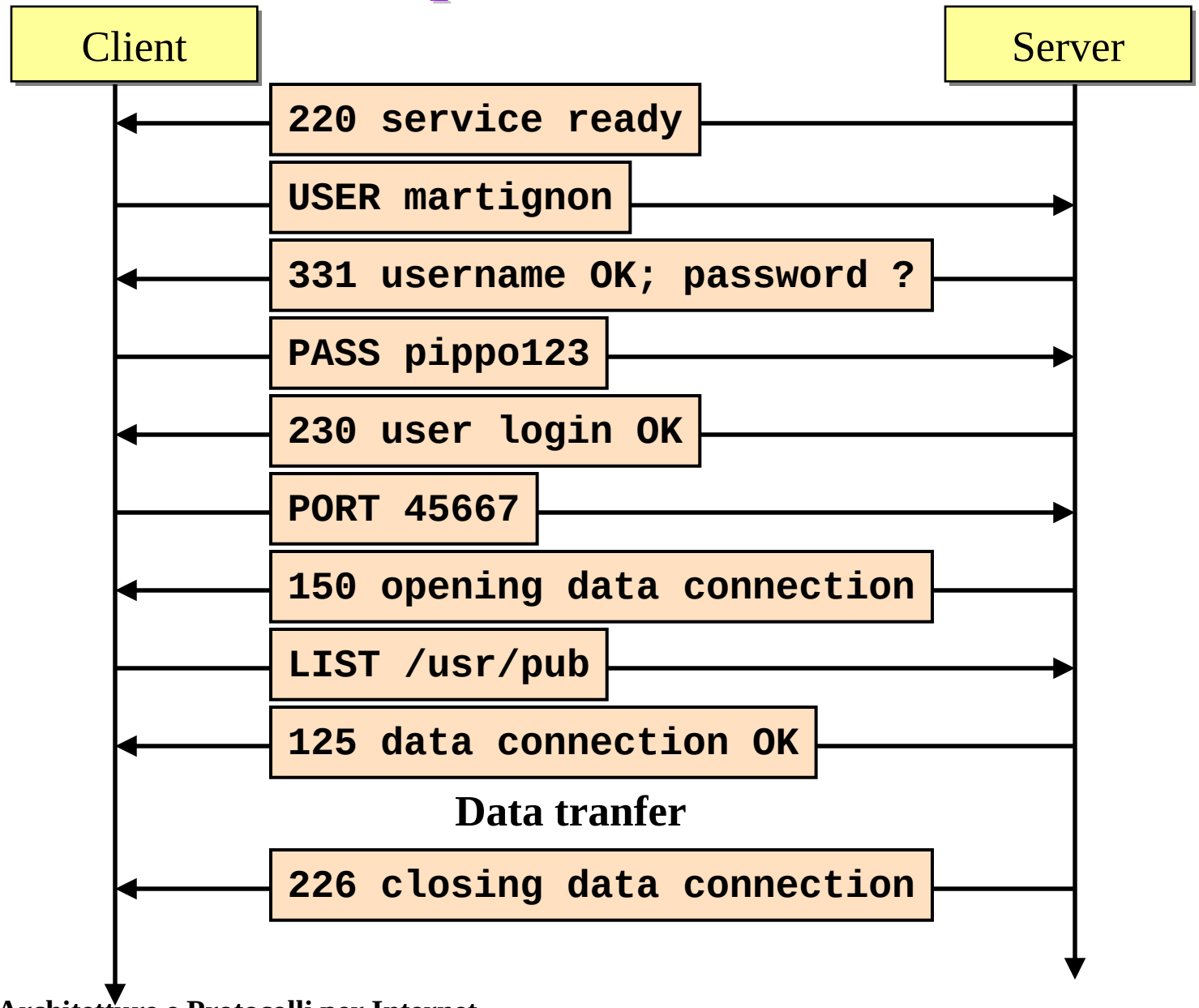
Gestione delle porte

```
PORT client port
PASV server choose port
```

FTP: risposte

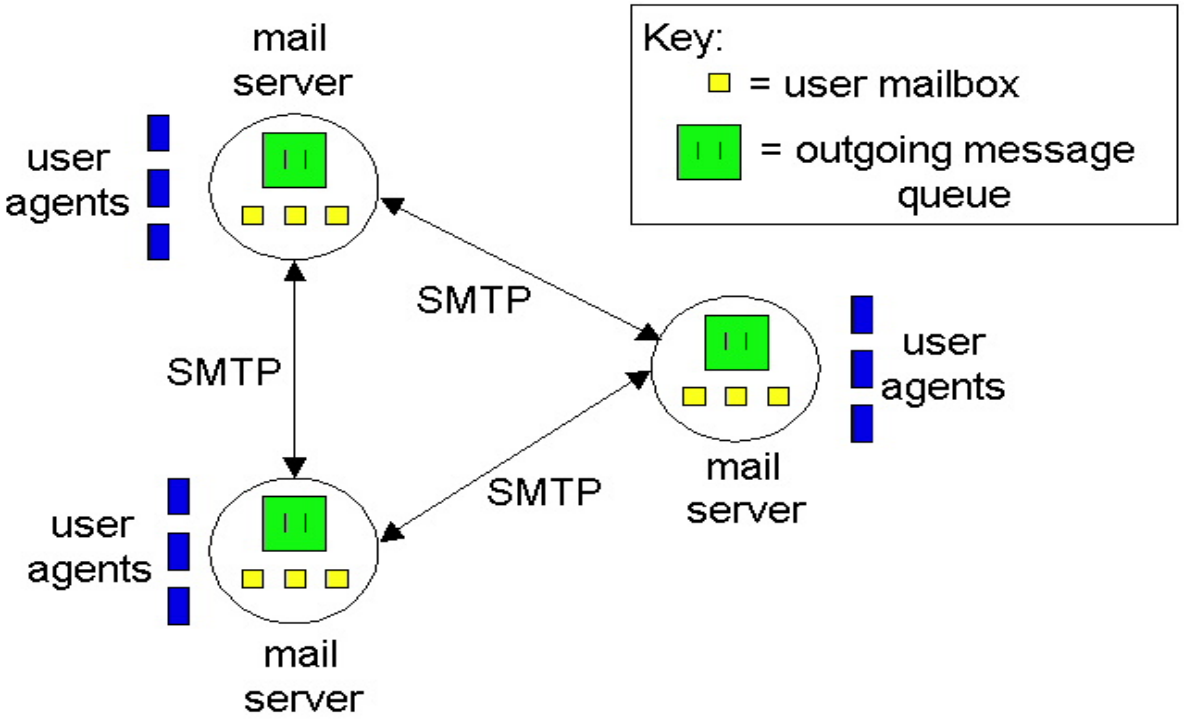
```
125 Data connection already open; transfer starting
200 Command OK
225 Data connection open
226 Closing data connection
227 Entering passive mode; srv. sends Ip_add.,port
230 User login OK
331 Username OK, password required
425 Can't open data connection
426 Connection closed; tranfer aborted
452 Error writing file
500 Syntax error; unrecognized command
501 Syntax error in parameters or arguments
502 Command not implemented
```

FTP: esempio trasferimento file



E-mail

- L'e-mail è un'applicazione che consente di inviare in modo asincrono messaggi testuali
- E' basata su una rete di server che comunica con il prot. appl. SMTP (Simple Mail Transfer Protocol)



SMTP

J.B. Postel, "Simple Mail Transfer Protocol," RFC 821, August 1982.

- **E’ un protocollo testuale**
- **richiede che anche il corpo dei messaggi sia ASCII**
  - **i documenti binari devono essere convertiti in ASCII**
- **quando un server riceve un messaggio da uno user agent**
  - **mette il messaggio in una coda**
  - **apre una connessione TCP con la porta 25 del server del destinatario**
  - **trasferisce il messaggio**

Colloquio tra client e server SMTP

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250 Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

Formato dei messaggi

D.H. Crocker, "Standard for the Format of ARPA Internet Text Messages," RFC 822, August 1982.

- **Il formato dei messaggi inviati (comando DATA) è specificato**
- **Alcuni header standard precedono il corpo del messaggio vero e proprio**

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Request of information
<black line>
<Body>
.
```

Multipurpose Internet Mail Extensions (MIME)

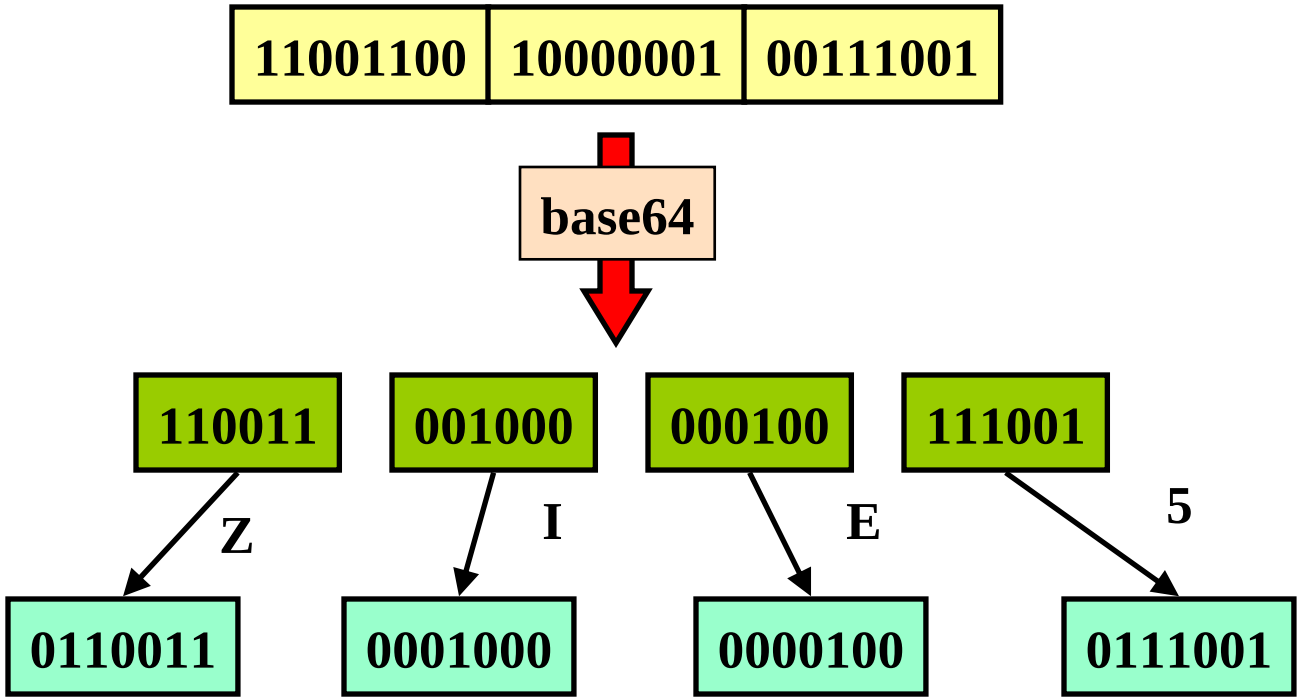
•"Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies," RFC 2045, Nov. 1996.
•"Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types," RFC 2046, Nov. 1996.

- **L'estensione MIME al RFC 822 ha come scopo principale quello di consentire il trasferimento di messaggi non ASCII**

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg
base64 encoded data .....
.....
.....base64 encoded data
.
```

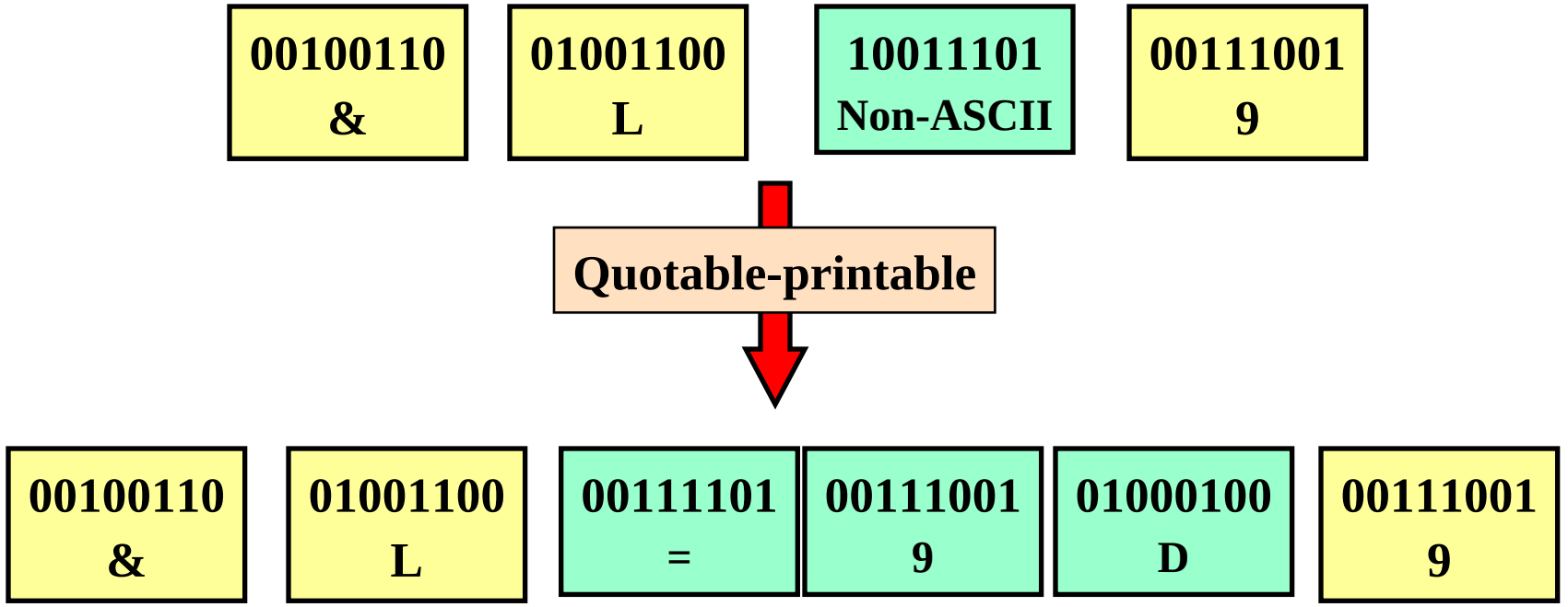
Multipurpose Internet Mail Extensions (MIME)

- **Codifiche:**
  - **Base64:**
    - ✓ Le sequenze di bit da trasferire sono divise in gruppi di 24 bit
    - ✓ Ogni gruppo è in diviso in 4 sotto-gruppi di 6 bit
    - ✓ Ad ogni gruppo viene aggiunto uno zero in testa e vengono trasferiti come caratteri ASCII



Multipurpose Internet Mail Extensions (MIME)

- **Quoted-printable**
  - ✓ Le sequenze di bit da trasferire sono divise in gruppi di 8 bit
  - ✓ Se una sequenza corrisponde ad un carattere ASCII è inviata così com'è
  - ✓ Altrimenti viene inviata come tre caratteri , "=" seguito dalla rappresentazione esadecimale del byte



Multipurpose Internet Mail Extensions (MIME)

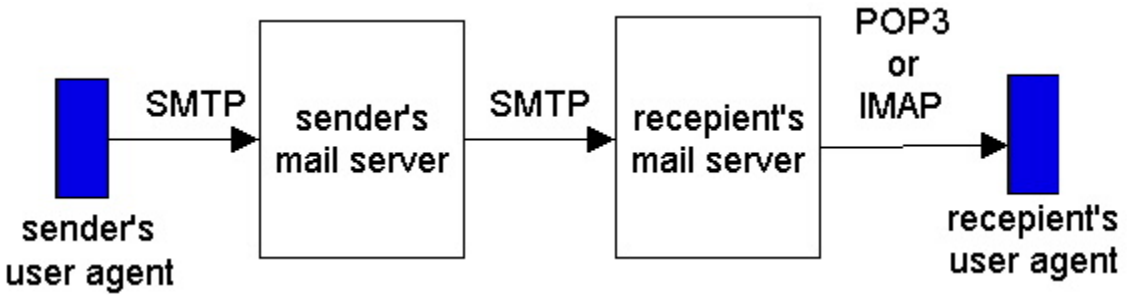
- **MIME consente anche il trasferimento di più oggetti come parti di uno stesso messaggio:**

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe with commentary
MIME-Version: 1.0
Content-Type: multipart/mixed; Boundary=StartOfNextPart
--StartOfNextPart
Dear Bob,
Please find a picture of an absolutely scrumptious crepe.

--StartOfNextPart
Content-Transfer-Encoding: base64
Content-Type: image/jpeg
base64 encoded data .....

--StartOfNextPart
Let me know if you would like the recipe.
.
```

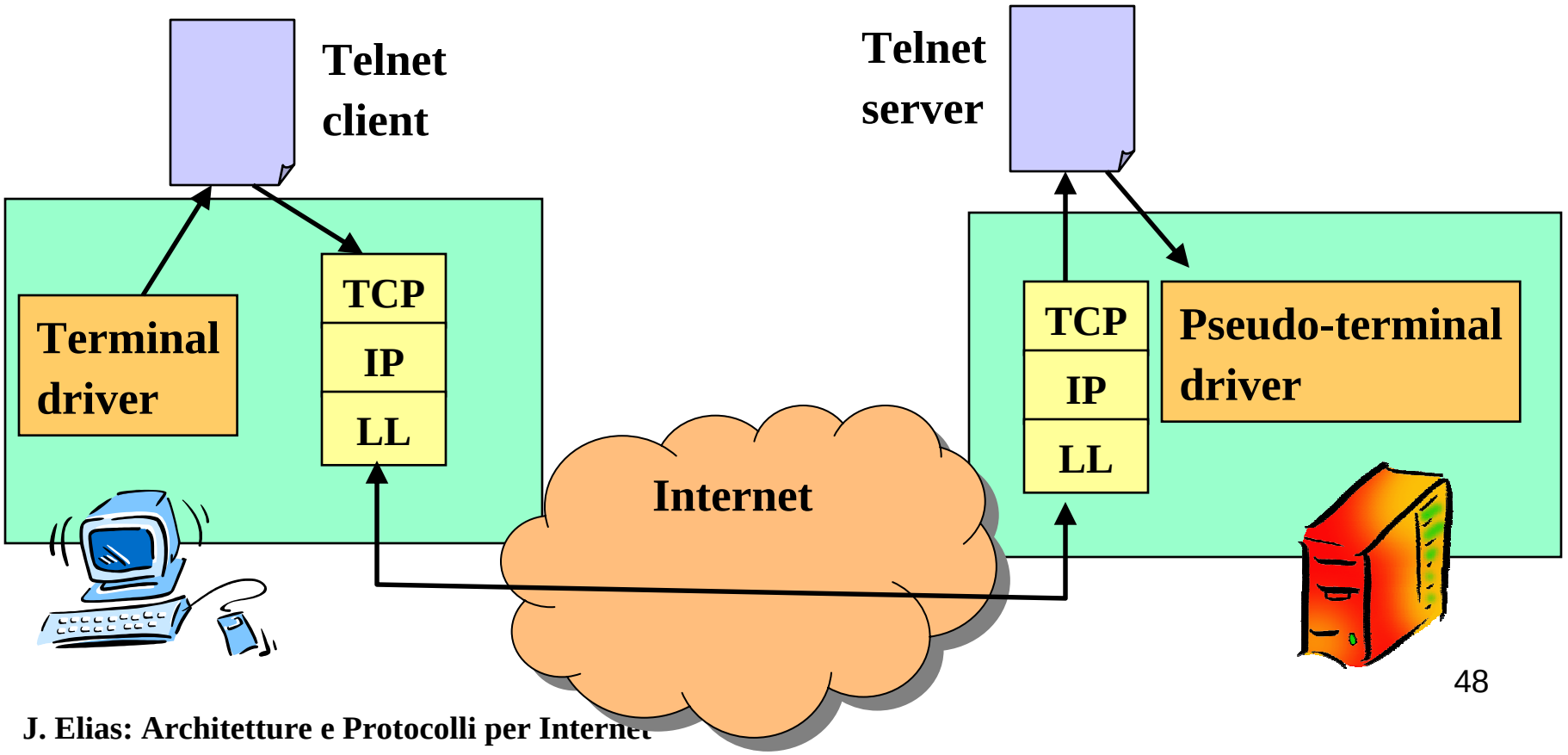
Protocolli di accesso al mailbox



- Diversi protocolli sono stati sviluppati per il colloquio tra user agent e server in fase di lettura dei messaggi presenti nel mailbox
  - POP3 (Post Office Protocol versione 3)
  - IMAP (Internet Mail Access Protocol)
  - HTTP

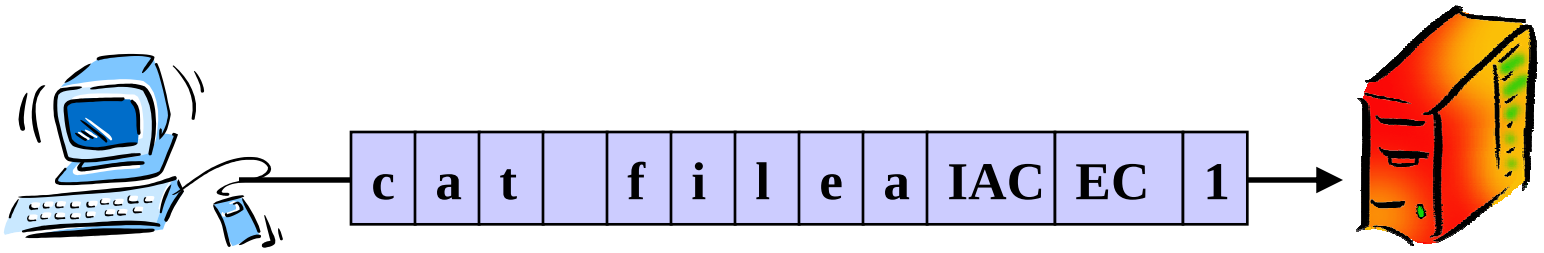
TELNET (TErminaL NETwork)

- E’ un semplice applicativo che consente di aprire un terminale remoto
- Al contrario di un terminale locale i comandi sono trasferiti su una connessione TCP

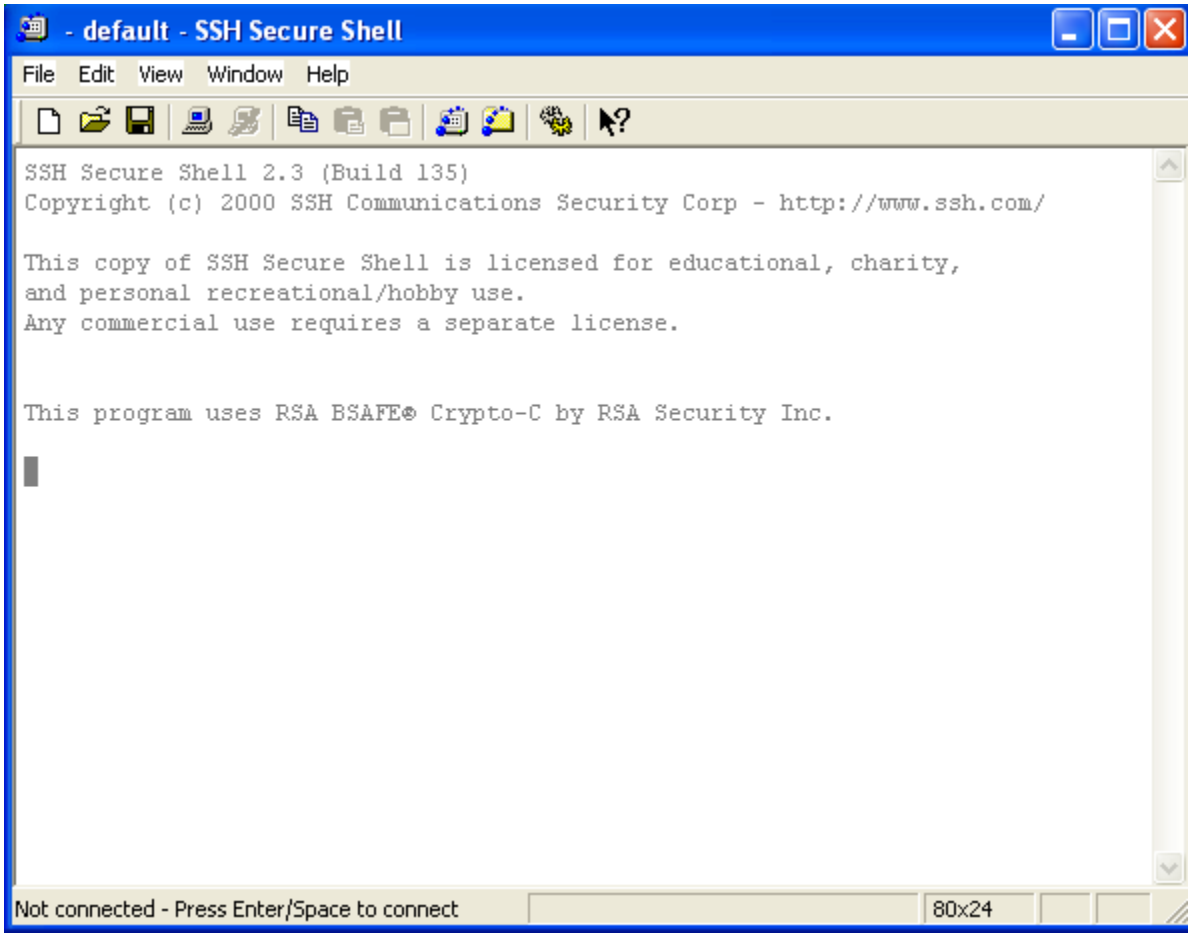


TELNET (TErminaL NETwork)

- TELNET trasferisce caratteri
  - Caratteri dati:
    - ✓ Sono caratteri ASCII con il primo bit pari a 0
  - Caratteri di controllo:
    - ✓ Sono comandi (Telnet Commands) codificati in sequenze di 8 bit con il primo pari ad 1
    - ✓ Tra questi
      - ✓ IAC (255): interpreta il prossimo come carattere di controllo (Interpret As Command)
      - ✓ EC (247): cancella carattere (Erase Character)

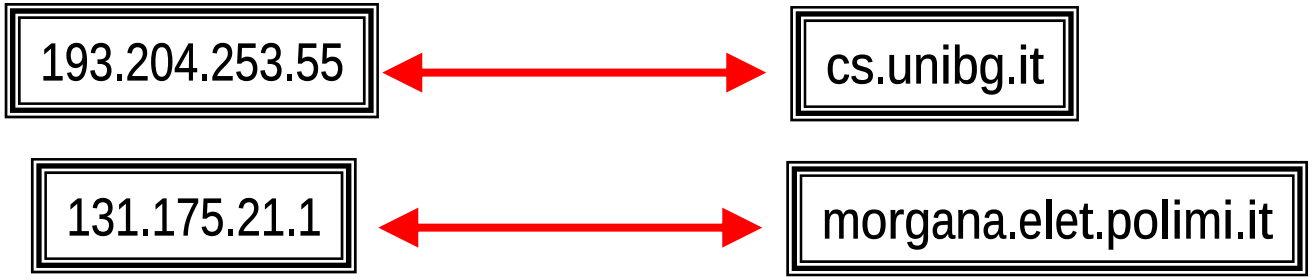


TELNET (TErminaL NETwork)



Domain Name System (DNS)

- Gli indirizzi IP sono poco adatti ad essere usati dagli applicativi.
- E’ più comodo utilizzare indirizzi simbolici
- gerarchici (via, città, stato)
- senza vincoli derivanti da esigenze di livello 3.
- Occorre una mappatura fra i due

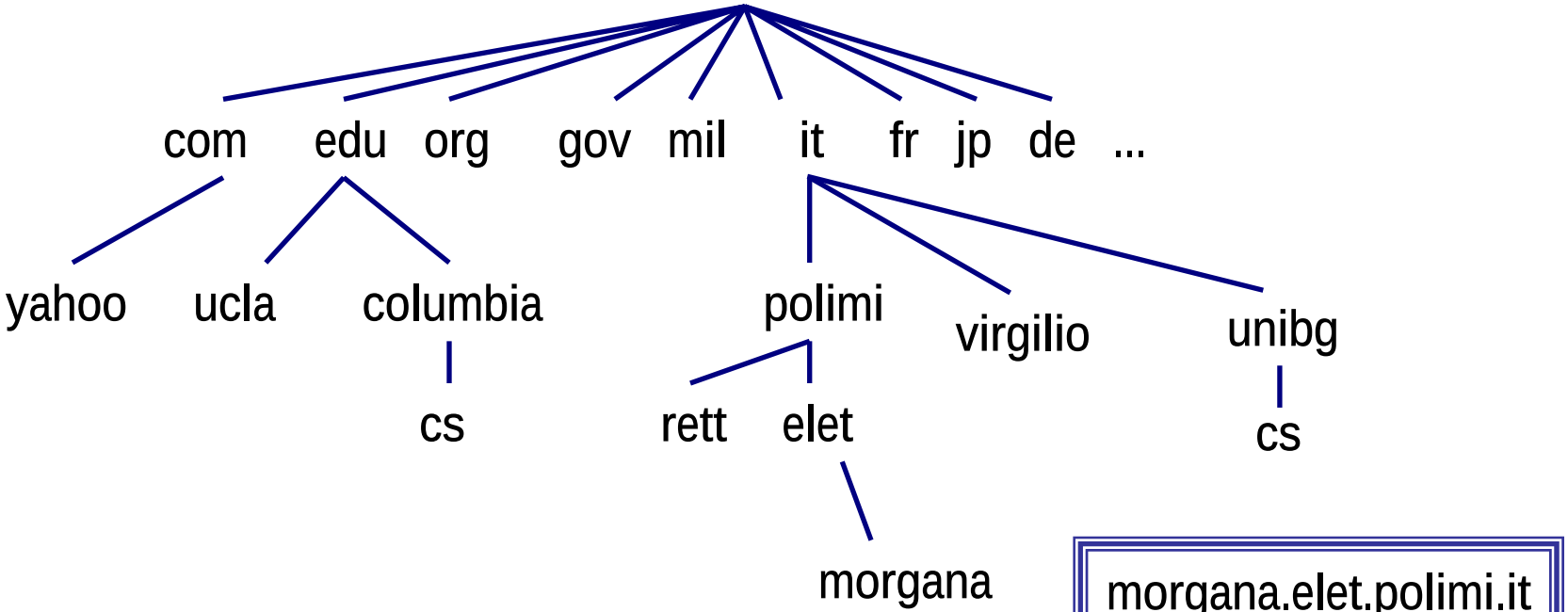


Domain Name System (DNS)

- "Domain Names - Concepts and Facilities," RFC 1034, Nov. 1987.
- "Domain Names - Implementation and Specification," RFC 1035, Nov. 1987.

- Le reti IP forniscono indirizzamento simbolico…
- …e un servizio di database distribuito che fornisce il servizio di mappaggio: DNS (Domain Name System)
- Il DNS è esso stesso un'applicazione IP che usa UDP/IP per trasferire i messaggi
- Il DNS viene usato anche per altri servizi:
  - Host aliasing
  - Mail server aliasing
  - Load distribution

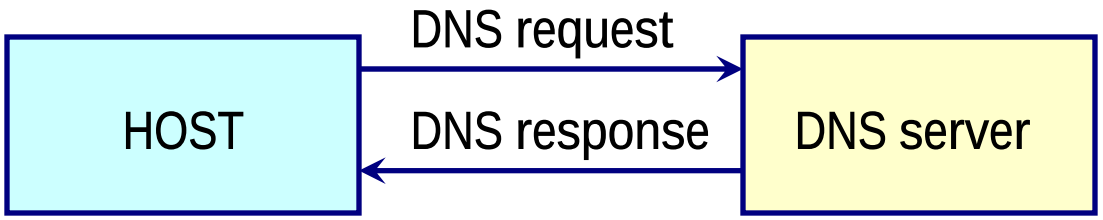
Indirizzamento simbolico



- **L'indirizzamento è di tipo gerarchico**
- **Ogni ramo è sotto il controllo di un'autorità**
- **Per ottenere un nuovo indirizzo occorre chiedere il permesso all'autorità competente**

Come ottenere un mappaggio

- Ogni host ha configurato l'indirizzo del server DNS (manualmente, vedi ad es. Pannello di controllo di WIN, o via DHCP)
- Le applicazioni che richiedono un mappaggio (browser, ftp, etc.) usano le funzioni del DNS
- Una richiesta viene inviata al server DNS usando UDP come protocollo di livello trasporto
- Il server reperisce l'informazione e restituisce la risposta



Informazioni memorizzate

Name, Value, Type, TTL

- **Type**
  - **A: (Address) Name è il nome di un host e Value è il suo indirizzo IP**

    (morgana.elet.polimi.it, 131.175.21.1, A, TTL)
  - **NS: (Name Server) Name è un domain e Value è il nome di un server che può ottenere le informazioni relative**

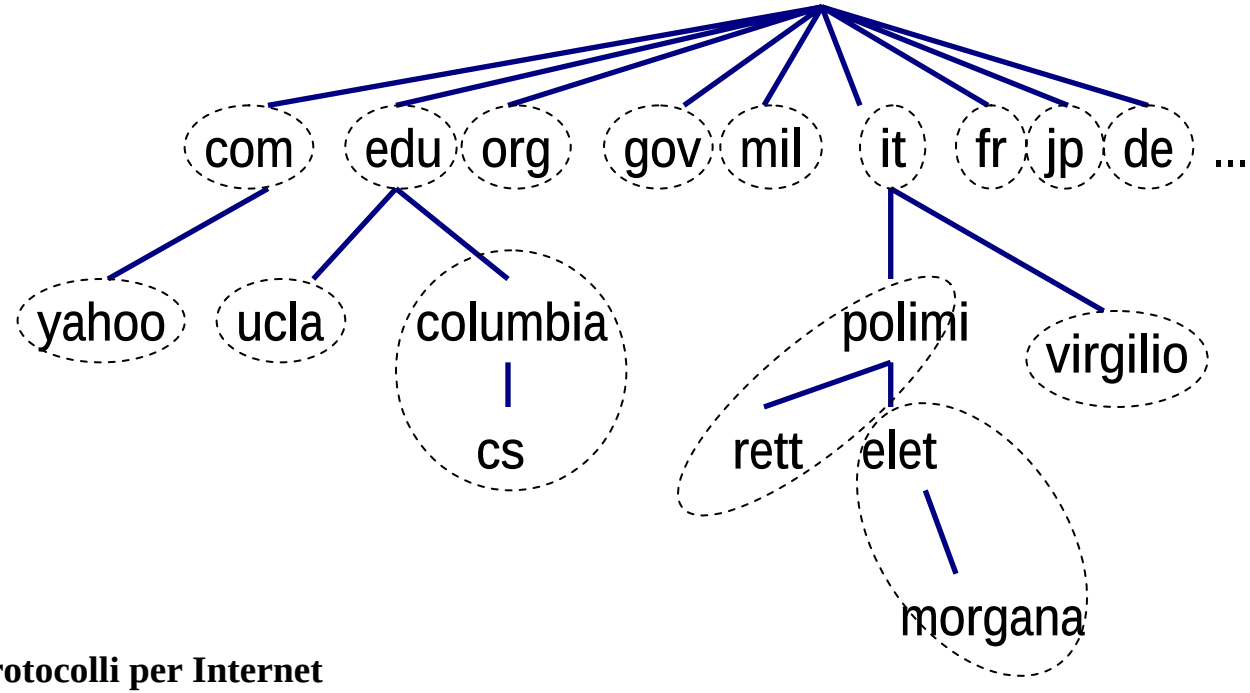
    (elet.polimi.it, morgana.elet.polimi.it, NS, TTL)
  - **CNAME: Name è un nome alternativo (alias) per un host il cui nome canonico è in Value**

    (www.polimi.it, zephyro.rett.polimi.it, CNAME, TTL)
  - **MX: (Mail eXchange) Name è dominio di mail o un alias di mail e Value è il nome del mail server**

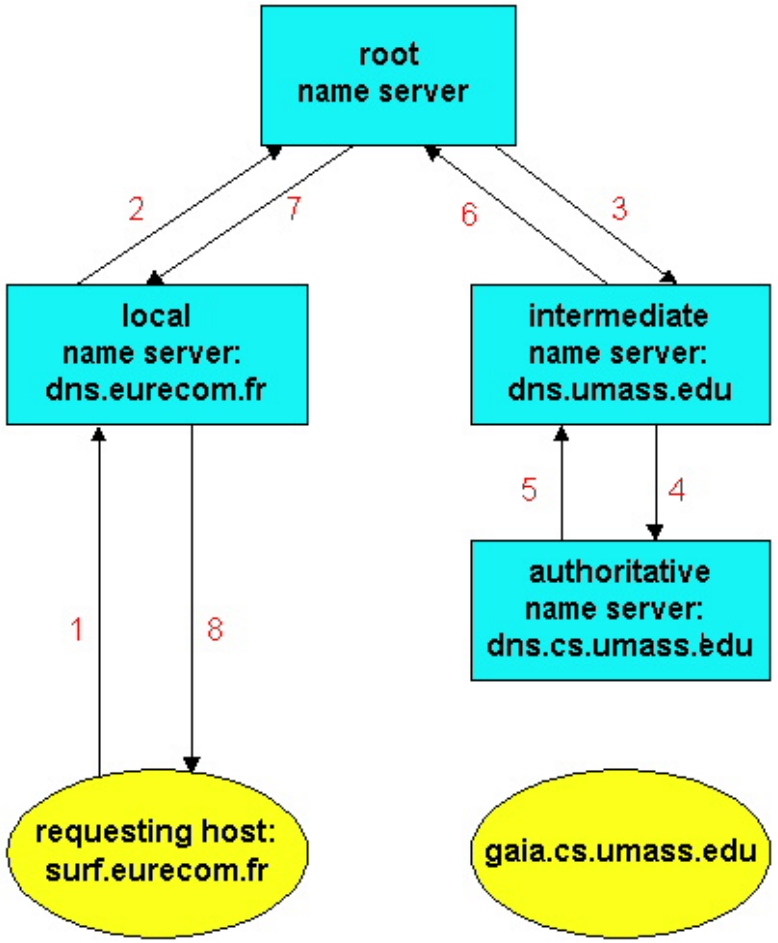
    (elet.polimi.it, mailserver.elet.polimi.it, MX,TTL)

Organizzazione del database

- I record in ARPANET erano contenuti in un name server centrale
- Per Internet la struttura del database è distribuita
- I rami sono partizionati in zone e un DNS server viene associato ad ogni zona
- Il server di una zona è responsabile per le informazioni di quella zona (authoritative)

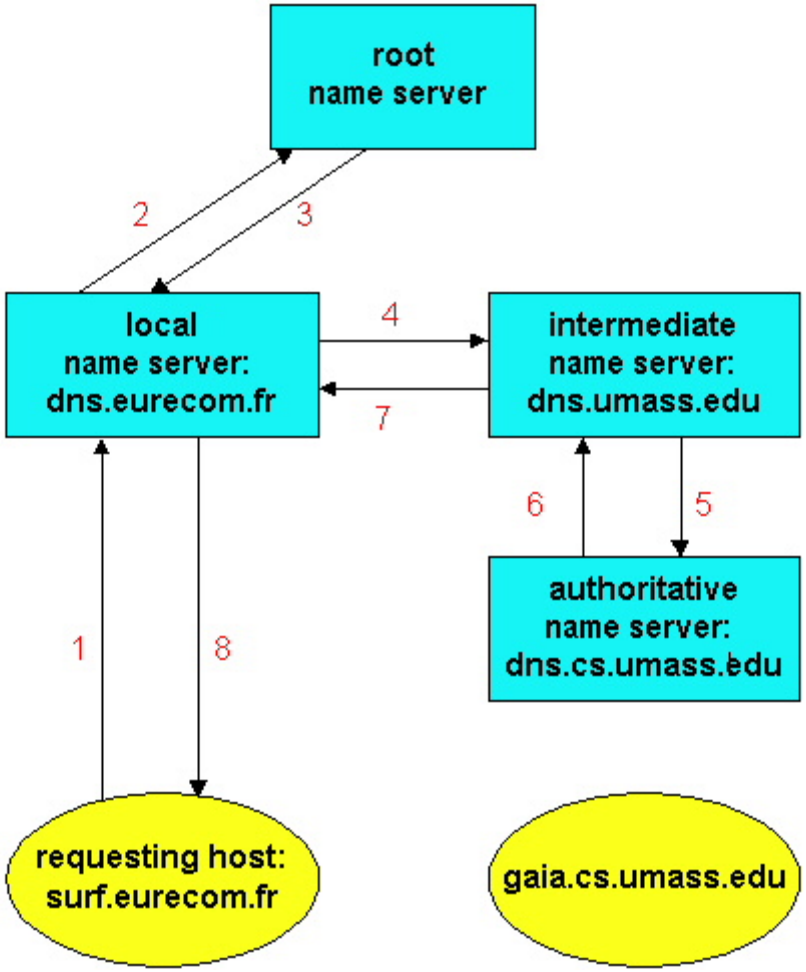


Reperire informazioni (1º modo)



- in modalità puramente ricorsiva
- la richiesta viene inoltrata seguendo la gerarchia
- la risposta segue la strada inversa

Reperire informazioni (2º modo)



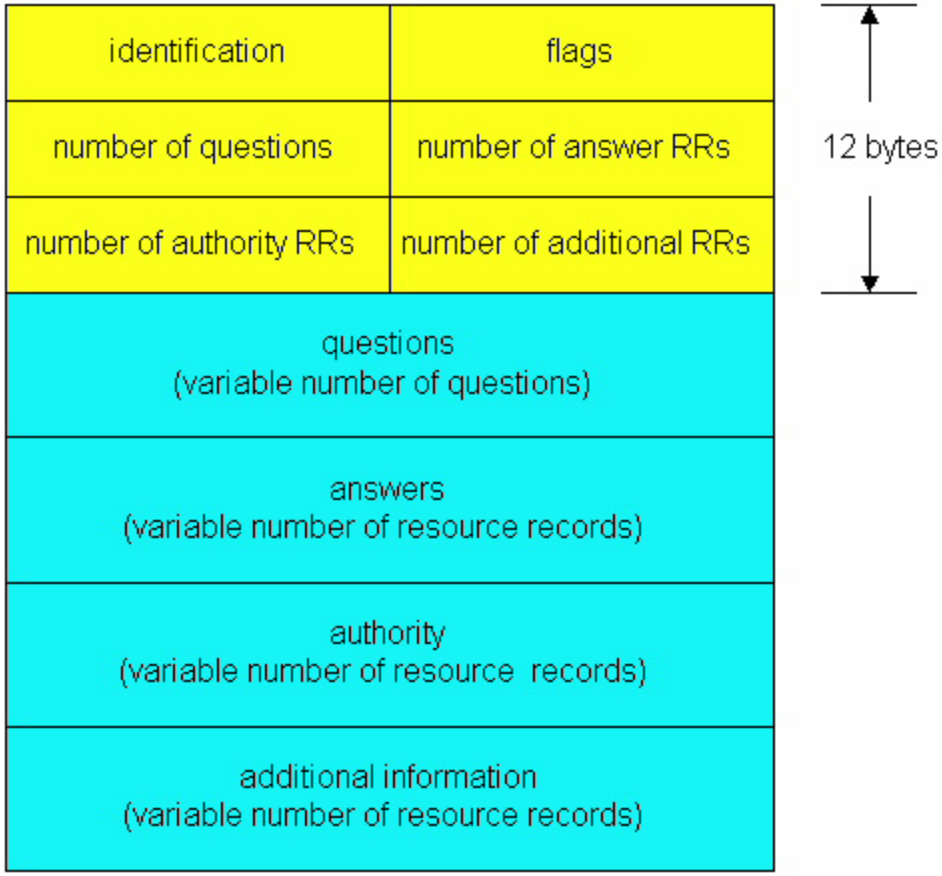
- con la modalità <u>iterativa</u>
- un server può rispondere ad una richiesta con il nome di un altro server dove reperire l'informazione

Caching

- Un server, dopo aver reperito un'informazione su cui non è authoritative può memorizzarla temporaneamente
- All'arrivo di una nuova richiesta può fornire l'informazione senza risalire sino al server authoritative
- Il TTL è settato dal server authoritative ed è un indice di quanto stabile nel tempo è l'informazione relativa
- I server non-authoritative usano il TTL per settare un time-out

Messaggi DNS

sono in binario (non ASCII)



- **identification: identificativo coppia richiesta/risposta**
- **flag: richiesta/risposta, authoritative/non auth., iterative/recursive**
- **number of: relativo al numero di campi nelle sez. successive**
- **questions: nome richiesto e tipo (di solito A o MX)**
- **answers: resourse records completi forniti in risposta**
- **authority: contiene altri record forniti da altri server**
- **additional infor.: informazione addizionale, ad es. il record con l'IP ADDR. per il MX fornito in answers**