Livello delle Applicazioni

Luigi Vetrano

Anno Accademico 2013-2014

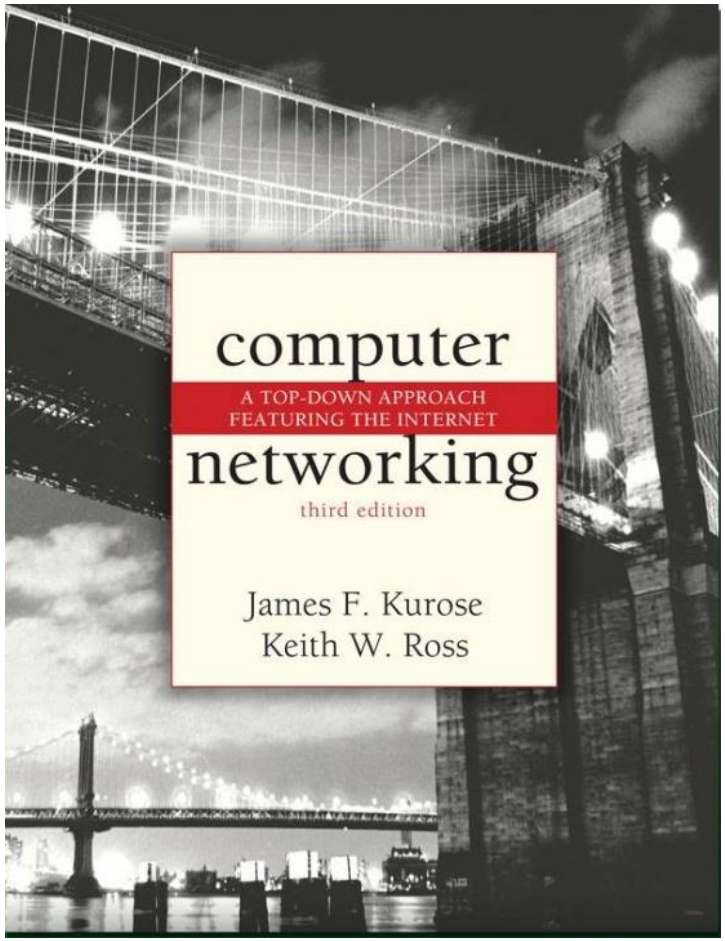
Livello di Applicazione

Queste slides sono liberamente adattate dalle slides del libro:

Computer Networking

A Top Down Approach Featuring the Internet

Jim Kurose - Keith Ross
Addison-Wesley, 2005-2008.



Livello di applicazione

- Principi delle applicazioni di rete
- Web e HTTP
- FTP
- Posta elettronica
  - SMTP, POP3, IMAP
- DNS

Alcune diffuse applicazioni di rete

- Posta elettronica
- Web
- Messaggistica istantanea
- Autenticazione in un calcolatore remoto (Telnet e SSH)
- Condivisione di file P2P
- Giochi multiutente via rete
- Streaming di video-clip memorizzati
- Telefonia via Internet
- Videoconferenza in tempo reale

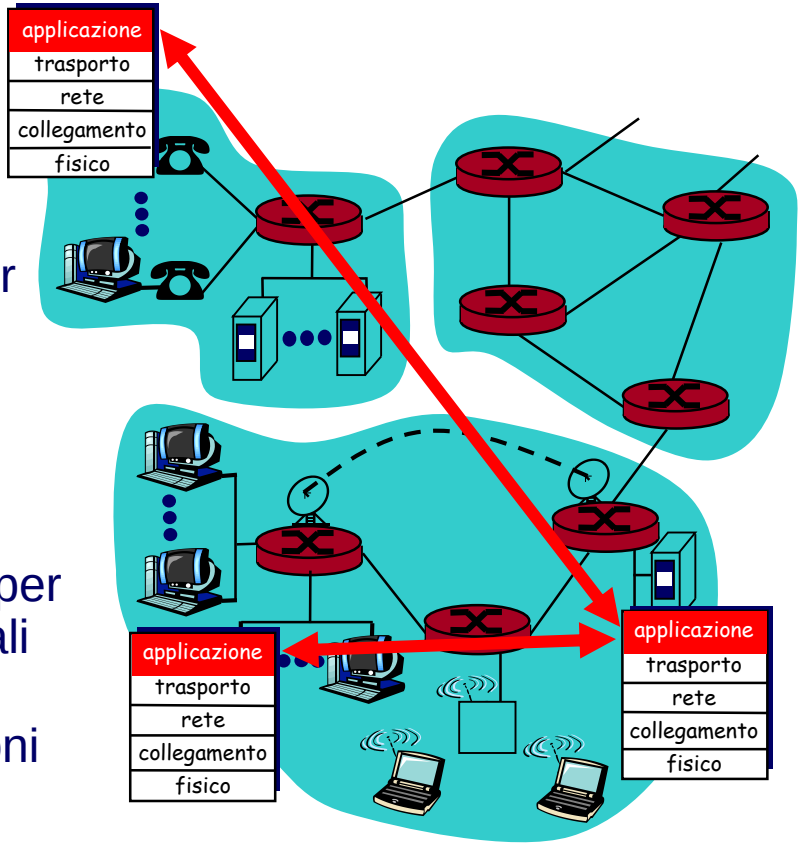
Creare un'applicazione di rete

Scrivere programmi che

- girano su sistemi terminali diversi
- comunicano attraverso la rete
- Ad es. il Web: il software di un server Web comunica con il software di un browser

software in grado di funzionare su più macchine

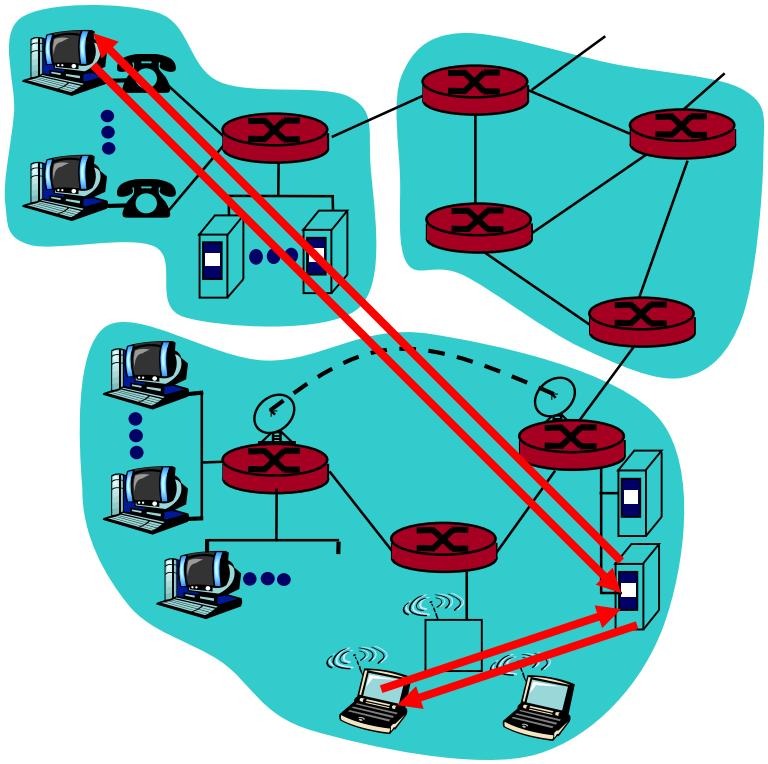
- non occorre predisporre programmi per i dispositivi del nucleo della rete, quali router o commutatori Ethernet
- Anche volendo sviluppare applicazioni per i network-core devices, non è possibile (perché ?)



Architettura dell' applicazione

- Quando si progetta una applicazione di rete bisogna decidere l'Architettura Applicativa
- L'architettura dell'applicazione è diversa dall'architettura di Rete vista prima (e.g. i 4-5 livelli del TCP o i 7 livelli ISO/OSI).
- Essa fissa come l'applicazione è organizzata sui vari end-systems
  - Client-server
  - Peer-to-peer (P2P)
  - Architetture ibride (client-server e P2P)

Architettura client-server



server:

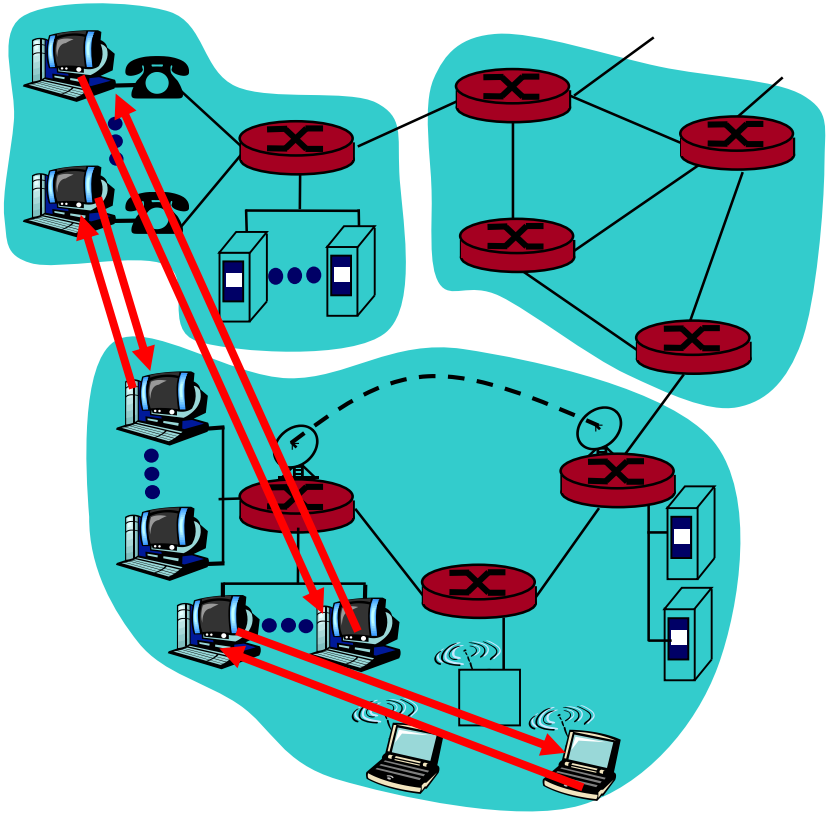
- host sempre attivo (always on)
- indirizzo IP pubblico e fisso
- server farm per creare un potente server virtuale

client:

- comunica con il server
- può contattare il server in qualunque momento
- Always-on o Sometimes-on
- può avere indirizzi IP dinamici
- non comunica direttamente con gli altri client

Architettura P2P pura

- Non c'è un server sempre attivo
- coppie arbitrarie di host (peer) comunicano direttamente tra loro
- i peer non devono necessariamente essere sempre attivi, e cambiano indirizzo IP
- Un esempio: Gnutella
- Facilmente scalabile
- Difficile da gestire



Ibridi (client-server e P2P)

- Napster
  - Scambio di file secondo la logica P2P
  - Ricerca di file centralizzata secondo la logica Client-Server:
    - i peer registrano il loro contenuto presso un server centrale
    - i peer chiedono allo stesso server centrale di localizzare il contenuto
- Messaggistica istantanea
  - La chat tra due utenti è del tipo P2P
  - Individuazione della presenza/location centralizzata:
    - l'utente registra il suo indirizzo IP sul server centrale quando è disponibile online
    - l'utente contatta il server centrale per conoscere gli indirizzi IP dei suoi amici

Processi e indirizzamento

- Affinché un processo su un host invii un messaggio a un processo su un altro host, il mittente deve identificare il processo destinatario.
- Un host A ha un indirizzo IP univoco a 32 bit
  - (in realtà l'IP individua l'interfaccia di rete)
- D: È sufficiente conoscere l'indirizzo IP dell'host su cui è in esecuzione il processo per identificare il processo stesso?
- R: No, sullo stesso host possono essere in esecuzione molti processi.
- L'identificatore comprende sia l'indirizzo IP che i numeri di porta associati al processo in esecuzione su un host.
- Esempi di numeri di porta:
  - HTTP server: 80
  - Mail server: 25
- I numeri di porta dei processi base sono well-known e sono stati assegnati dallo IANA
- Quando create una nuova applicazione, dovete assegnarle un nuovo numero di porta

Numeri di porta riservati: well-known port (1)

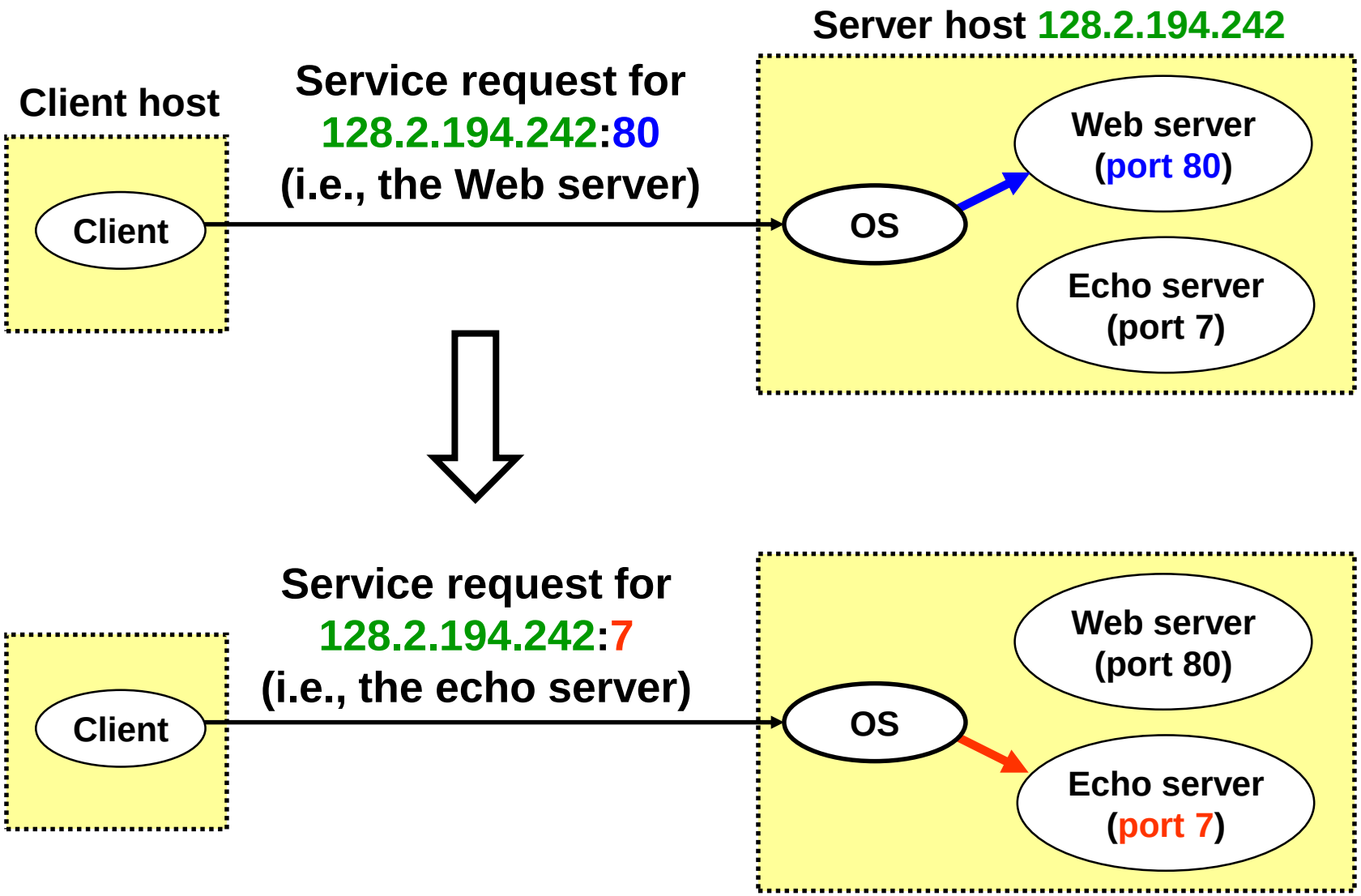
- I numeri di porta nell'intervallo [0:1023] sono riservati. Gli altri sono assegnabili dal S.O. ai servizi erogati dai programmi utente
- Alcuni numeri riservati sono assegnati a specifici servizi (well-know port number)
- Servizi UDP/TCP con well-know port number:
  - Porta 007 – UDP/TCP – echo – Servizio di echo
  - Porta 009 – UDP/TCP – discard – Ignorare il pacchetto
  - Porta 013 – UDP/TCP – daytime – Ora del giorno
  - Porta 067 – UDP/TCP – bootps – Server BOOTP/DHCP
  - Porta 088 – UDP/TCP – kerberos – Servizio di sicurezza
  - Porta 111 – UDP/TCP – rpc – Remote Procedure Call
  - Porta 123 – UDP/TCP – ntp – Network Time Protocol

Numeri di porta riservati: well-known port (2)

- Servizi solo UDP con well-know port number:
  - Porta 069 – UDP – tftp –trivial file transfer protocol
- Servizi solo TCP con well-know port number:
  - Porta 020 – TCP – ftp-data – file transfer protocol (dati)
  - Porta 021 – TCP – ftp – file transfer protocol (comandi)
  - Porta 022 – TCP – ssh – secure shell
  - Porta 023 – TCP – telnet – terminale remoto (o virtuale)
  - Porta 025 – TCP – smtp – simple mail transfer protocol
  - Porta 080 – TCP – www – world wide web
  - Porta 110 – TCP – pop3 – post office protocol (ver. 3)
  - Porta 161 – TCP – snmp – simple network mgmt prot

http://www.iana.org/assignments/port-numbers

Usare le Porte per Identificare i Servizi



Requisiti del servizio di trasporto di alcune applicazioni comuni

Application	Data Loss	Bandwidth	Time Sensitive
ftp	No	Variable	No
E-mail	No	Variable	No
Web	No	Variable	No
RT Audio/Video	Tolerant	au: 5kbps-1Mbps vi:10kbps-5Mbps	yes, 100's msec
Stored audio/video	Tolerant	Same as above	yes, few secs
Interactive games	Tolerant	Few kbps up	yes, 100's msec
Instant messaging	No	Variable	yes and no

Servizi dei protocolli di trasporto Internet

Servizio di TCP:

- *orientato alla connessione (CO) :* è richiesto un setup fra i processi client e server
- *trasporto affidabile* fra i processi d'invio e di ricezione
- *controllo di flusso:* il mittente non vuole sovraccaricare il destinatario
- *controllo della congestione:* "strozza" il processo d'invio quando le rete è sovraccaricata
- *non offre:* temporizzazione, ampiezza di banda minima

Servizio di UDP:

- *NON orientato alla connessione (CL):* trasferimento dati inaffidabile fra i processi d'invio e di ricezione
- non offre: setup della connessione, affidabilità, controllo di flusso, controllo della congestione, temporizzazione né ampiezza di banda minima

D: Perché preoccuparsi? Perché esiste UDP?

Applicazioni e protocolli di trasporto

Applicazione	Protocollo applicativo	Protocollo di trasporto
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	proprietary (e.g. RealNetworks)	TCP o UDP
Internet telephony	proprietary (e.g., Dialpad, Skype)	UDP

Web e HTTP

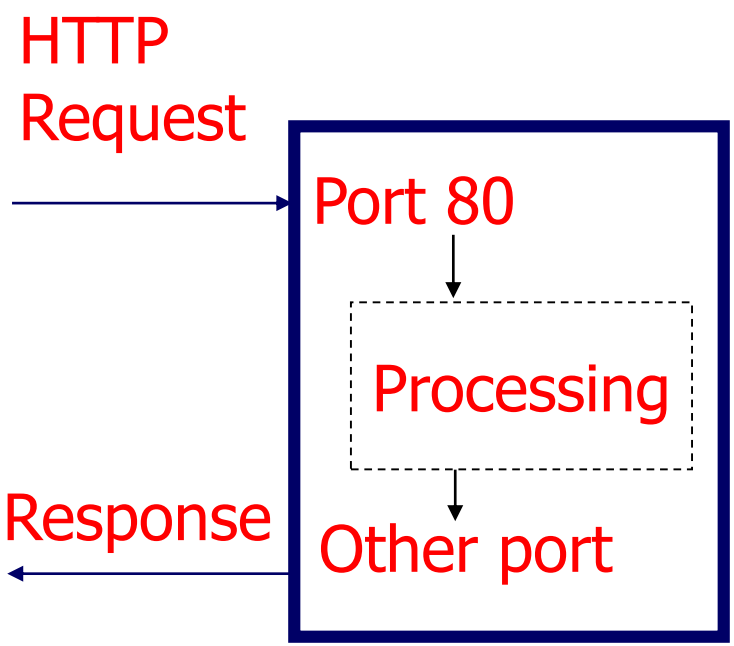
Terminologia

- Una pagina web è costituita da oggetti
- Un oggetto può essere un file HTML, un'immagine JPEG, un'applet Java, un file audio, …
- Una pagina web è formata da un file base HTML che include diversi oggetti referenziati
- Ogni oggetto è referenziato da un URL
- Esempio di URL (Universal Resource Locator):

`http://www.someschool.edu/someDept/pic.gif`

protocollo — nome dell'host — nome del percorso

Web (HTTP) Server



- Processa le richieste HTTP, generalmente sulla porta 80 TCP
  - La Risposta usa un'altra porta
- Può riguardare:
  - Restituire un documento, con il suo MIME-type-info
    - e.g., HTML document, TXT document
  - Invocare un programma o un modulo, restituendo il suo output
  - Sottomettere una form ad un programma o modulo, restituendo il suo output
- Le Risorse sono descritte usando l' URL

URL

Protocol :// Host : **Port** / Path

http://www.hiperlab.org:80/public/RadioMobile.html#content

http://www.hiperlab.org:8080/private/gps/

ftp://luigi:test@ftp.hiperlab.org/public/ThisSlides.ppt

file:///D:/ity10250/Desktop/DOCS.HTM

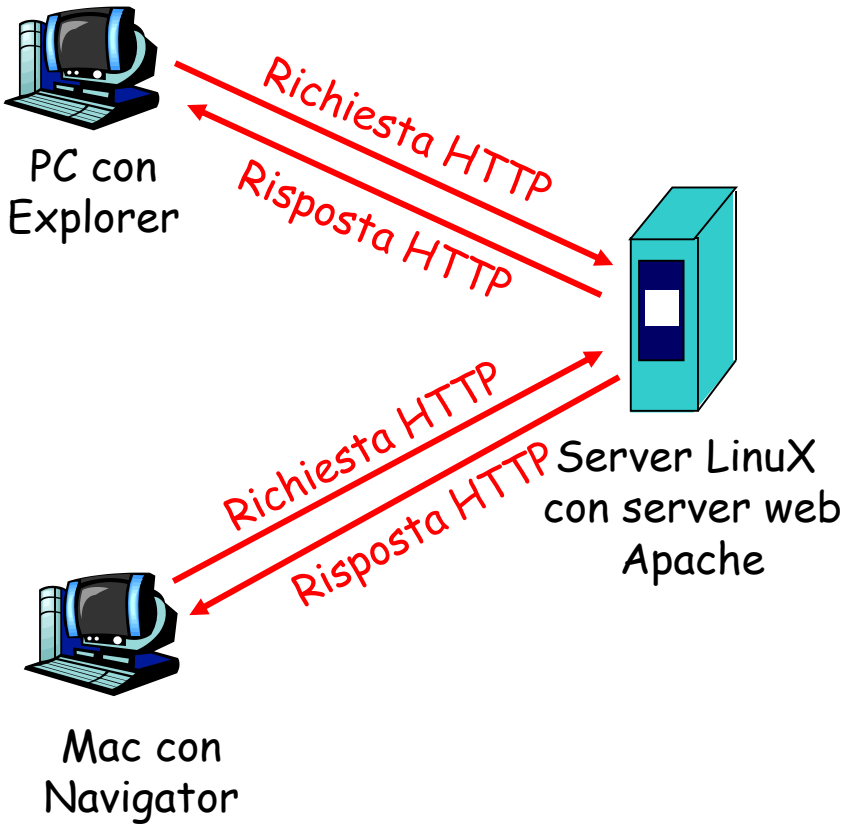
URL

- URL: Uniform Resource Locator
  - Un modo di codificare informazioni di protocollo, login, DNS (o IP) address, di path in una singola stringa
  - E' un caso speciale di Uniform Resource Identifer (URI)
    - URL è un URI per un sito da cui possono essere ricavate informazioni varie
      URN è un URI per un nome
- Sintassi generale:
  - {partition/protocol}://{userid}:{password}@{domain:port}/{path}
  - http://me:too@my.com:1234/index.html#anchor
  - news://nntp.upenn.edu
  - imap://email:me@my.com/folder1
  - file:///D:/ity10250/Desktop/DOCS.HTM

Panoramica su HTTP

HTTP: hypertext transfer protocol

- Protocollo a livello di applicazione del Web
- Modello client/server
  - *client:* il browser (user agent) che richiede, riceve, "visualizza" gli oggetti del Web
  - *server:* il server web invia oggetti in risposta a una richiesta
- HTTP 1.0: RFC 1945
- HTTP 1.1: RFC 2068



Panoramica su HTTP (cont.)

Usa TCP:

- Il client inizializza la connessione TCP (crea una socket) con il server, la porta 80
- Il server accetta la connessione TCP dal client
- Messaggi HTTP scambiati fra browser (client HTTP) e server web (server HTTP)
- Connessione TCP chiusa

HTTP è un protocollo "senza stato" (stateless)

- Il server non mantiene informazioni sulle richieste fatte dal client

nota

I protocolli che mantengono lo "stato" sono complessi!

- La storia passata (stato) deve essere memorizzata
- Se il server e/o il client si bloccano, le loro viste dello "stato" potrebbero essere contrastanti e dovrebbero essere riconciliate

Connessioni HTTP

Connessioni non persistenti

- Almeno un oggetto viene trasmesso su una connessione TCP
- HTTP/1.0 usa connessioni non persistenti

Connessioni persistenti

- Più oggetti possono essere trasmessi su una singola connessione TCP tra client e server
- HTTP/1.1 usa connessioni persistenti nella modalità di default

Connessioni non persistenti

Supponiamo che l'utente immetta l'URL
www.univaq.it/informatica/index.html

contiene testo e riferimenti a 10 immagini jpeg

Client

1a. Il client HTTP inizializza una connessione TCP con il server HTTP (processo) a **www.univaq.it** sulla porta **80**

Server

1b. Il server HTTP all'host **www.univaq.it** in attesa di una connessione TCP alla porta 80 "accetta" la connessione e avvisa il client

2. Il client HTTP trasmette un *messaggio di richiesta* (con l'URL) nella socket della connessione TCP.
Il messaggio indica che il client vuole l'oggetto **informatica/index.html**

3. Il server HTTP riceve il messaggio di richiesta, forma il *messaggio di risposta* che contiene l'oggetto richiesto e invia il messaggio nella sua socket

tempo

Connessioni non persistenti (cont.)

4. Il server HTTP chiede al TCP di chiudere la connessione

5. Il client HTTP riceve il messaggio di risposta che contiene il file html e visualizza il documento html. Esamina il file html, trova i riferimenti a 10 oggetti jpeg

6. I passi 1-5 sono ripetuti per ciascuno dei 10 oggetti jpeg

tempo

Due browser diversi possono interpretare quanto ricevuto in modo diverso: HTTP non ha nulla a che vedere con come il client interpreta la pagina HTML

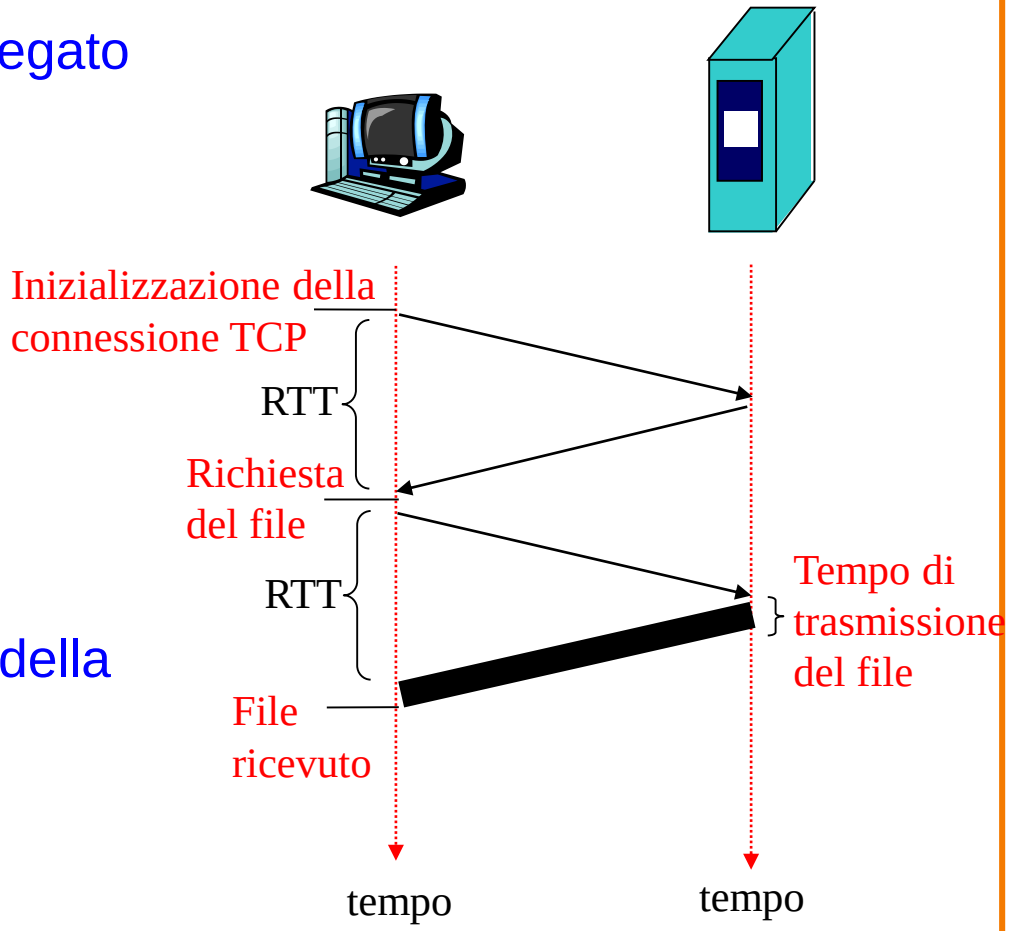
Le pagine Web

Schema del tempo di risposta

Definizione di RTT: tempo impiegato da un piccolo pacchetto per andare dal client al server e ritornare al client.

Tempo di risposta:

- un RTT per inizializzare la connessione TCP
- un RTT perché ritornino la richiesta HTTP e i primi byte della risposta HTTP
- tempo di trasmissione del file



totale = 2RTT + tempo di trasmissione

Connessioni persistenti

Svantaggi delle connessioni non persistenti:

- richiede 2 RTT per oggetto
- overhead del sistema operativo per *ogni* connessione TCP
- i browser spesso aprono connessioni TCP parallele per caricare gli oggetti referenziati

Connessioni persistenti

- il server lascia la connessione TCP aperta dopo l'invio di una risposta
- i successivi messaggi tra gli stessi client/server vengono trasmessi sulla connessione aperta

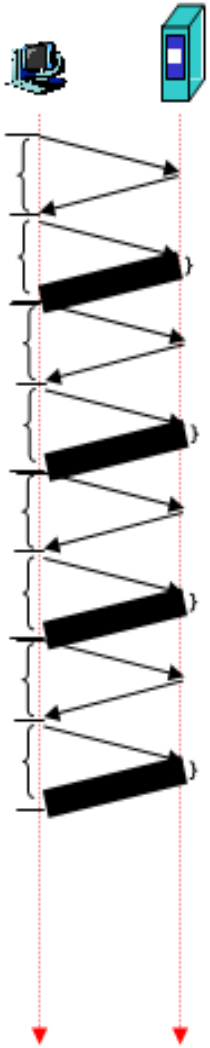
Connessione persistente ***senza*** pipelining:

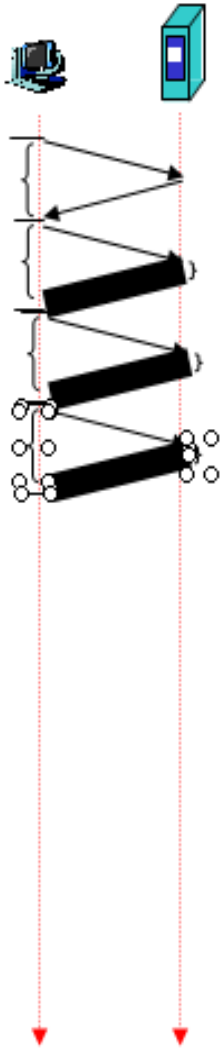
- il client invia una nuova richiesta solo quando ha ricevuto la risposta precedente
- un RTT per ogni oggetto referenziato

Connessione persistente ***con*** pipelining:

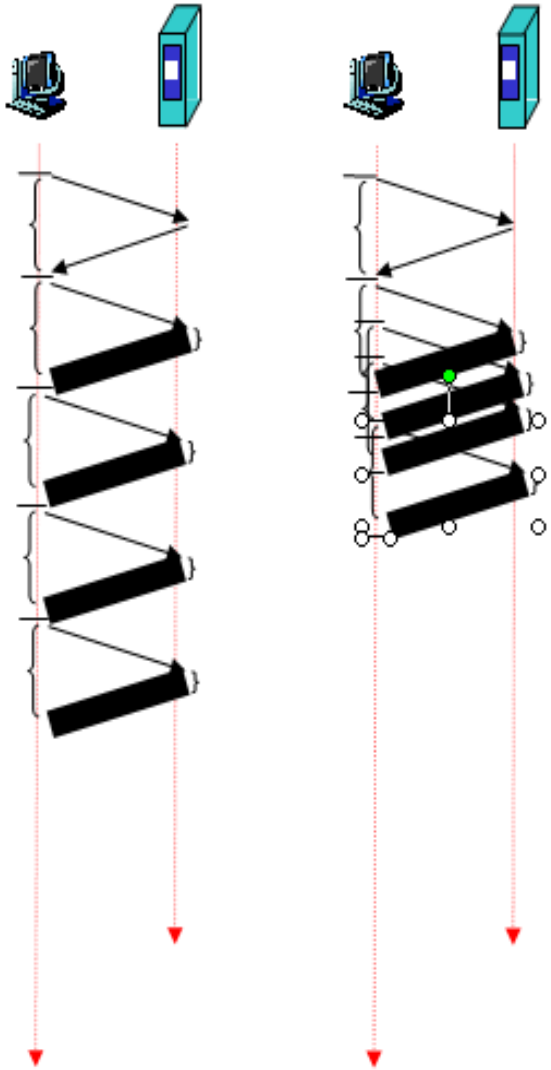
- è la modalità di default in HTTP/1.1
- il client invia le richieste non appena incontra un oggetto referenziato
- un solo RTT per tutti gli oggetti referenziati

Non Persistente o Persistente

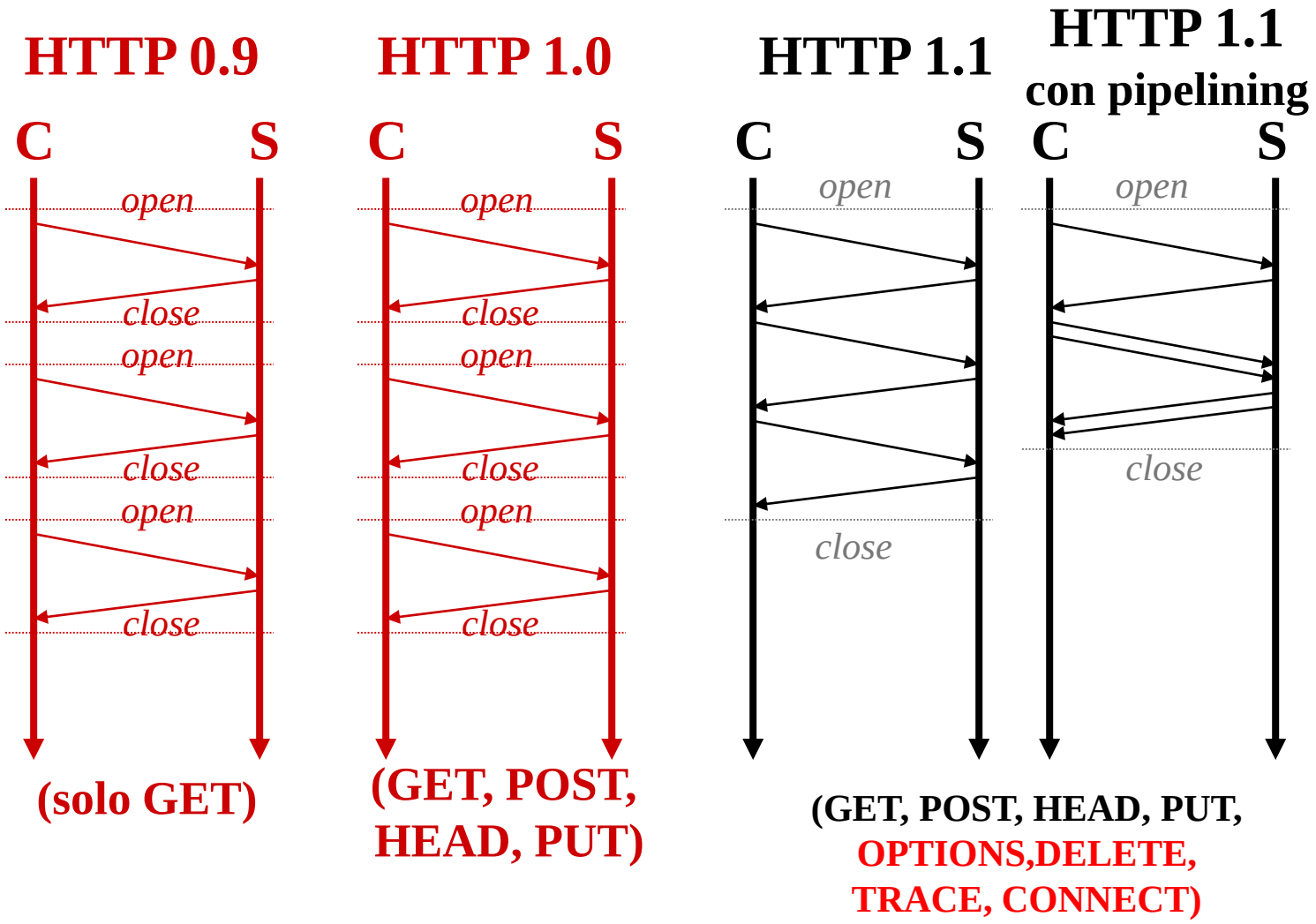




Persistente: no pipeling o pipeling



Versioni di HTTP



Messaggi HTTP

- due tipi di messaggi HTTP: *richiesta*, *risposta*
- Messaggio di richiesta HTTP:
  - ASCII (formato leggibile dall'utente)

Riga di richiesta (comandi GET, POST, HEAD)

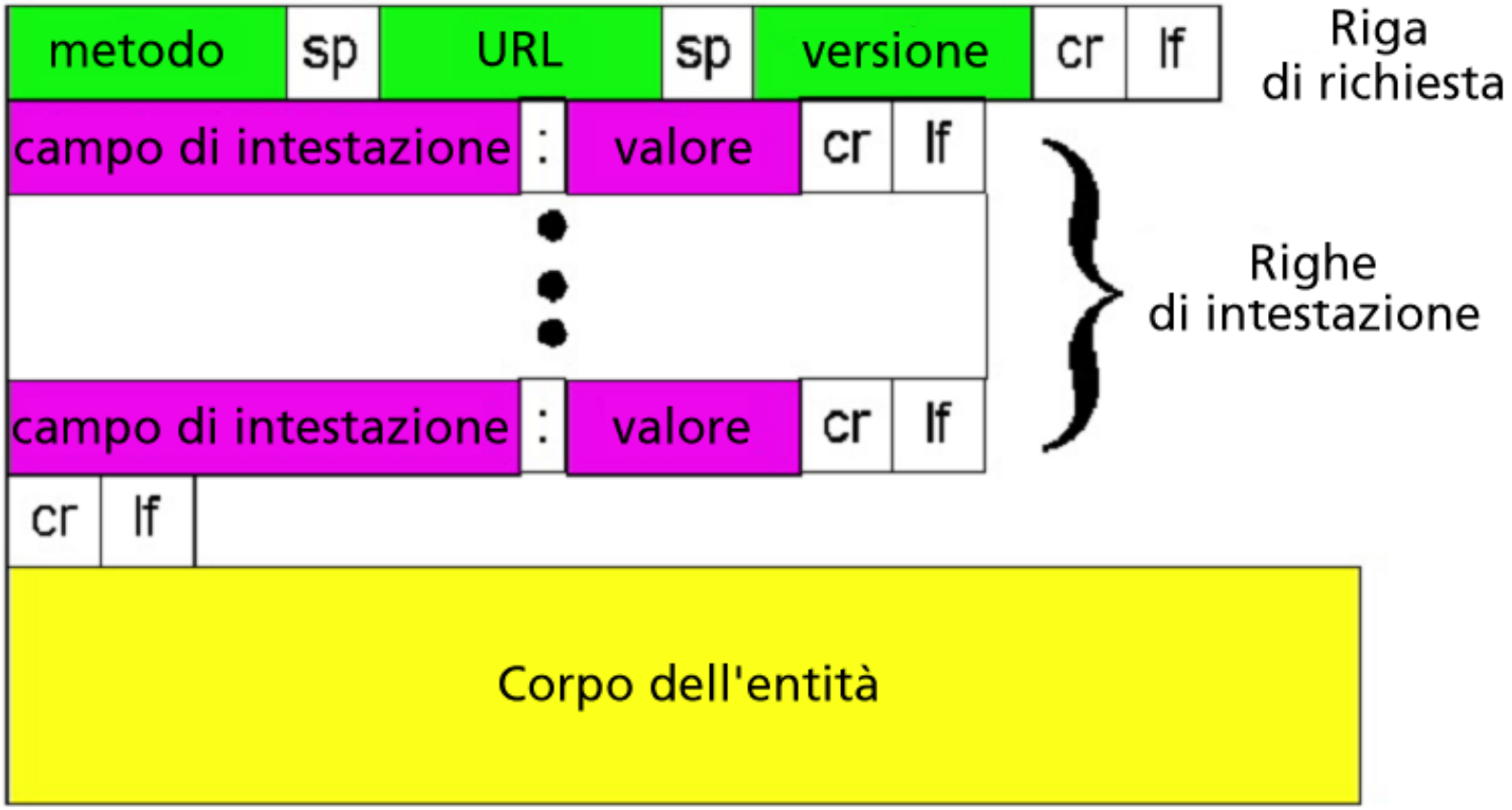
Righe di intestazione

Un carriage return e un line feed indicano la fine del messaggio

```
GET /somedir/page.html HTTP/1.1<crlf>
Host: www.someschool.edu<crlf>
User-agent: Mozilla/4.0<crlf>
Connection: close<crlf>
Accept-language:fr<crlf>
<crlf>
```

(carriage return e line feed extra)

Messaggio di richiesta HTTP

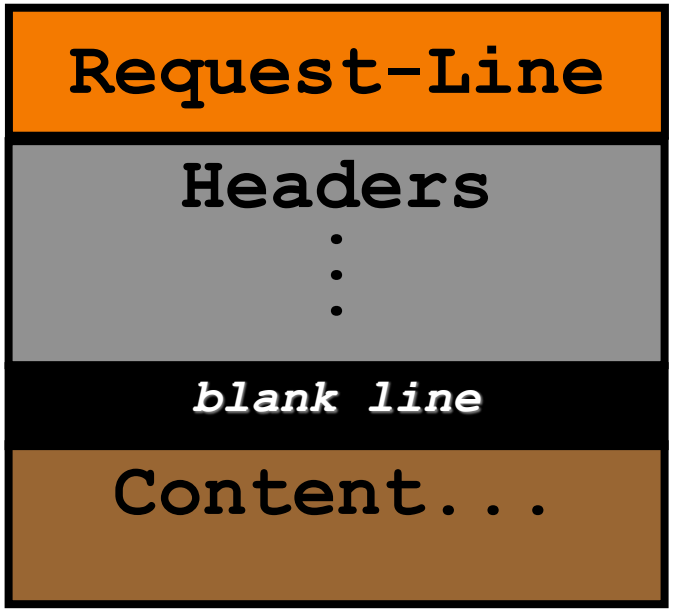


Un esempio di richiesta

```
GET /myfile.html HTTP/1.1
Referer: http://www.delta.com/sigma.html
Connection: Keep-Alive
User-Agent: Mozilla/4.61 (Macintosh; I; PPC)
Host: www.alpha.com:80
Accept: image/gif, image/jpeg, image/png, */*
Accept-Encoding: gzip
Accept-Language: en, fr
Accept-Charset: iso-8859-1,*,utf-8
<cr><lf>
```

Richieste HTTP ≥ 1.0

- Linee di testo (ASCII).
- Linee terminano con CRLF "**\r\n**"
- La prima linea è la "Request-Line"



Request Line

Method URI HTTP-Version\r\n

- La ‘request line’ contiene 3 *tokens* (words).
- Il carattere ‘space’ “ “ separa i tokens.
- <CR><LF> (\r\n) termina la richiesta

Request Method

- Il ‘Request Method’ può essere:

GET	**HEAD**	**PUT**
POST	**DELETE**	**TRACE**
OPTIONS		

Metodi

- GET: recupera le informazioni identificate dall' URI.
- HEAD: recupera meta-information sull' URI.
- POST: invia informazioni ad un URI e riceve i risultati.
- PUT: Memorizza informazioni nella locazione indicata dall' URI.
- DELETE: rimuove l'entity identificata dall' URI.

Altri Metodi

- TRACE: usato per tracciare l' HTTP forwarding attraverso proxies, tunnels, ecc.

- OPTIONS: usato per determinare le capacità del server, o le caratteristiche di una data risorsa.

URI: Universal Resource Identifier

- URI definiti in RFC 2396.

- Absolute URI:
  - scheme://hostname[:port]/path
  - http://www.cs.rpi.edu:80/blah/foo

- Relative URI:
  - /path
  - /blah/foo

Non viene menzionato il server

Le ‘Header Lines’

- Dopo la *Request-Line* ci sono un certo numero (possibilmente zero) di HTTP *header lines*.

- Ogni header line contiene un nome di un attributo seguito da “:” seguito da uno spazio e da un valore.

Name e Value sono semplice testo.

Fine degli Headers

- Ogni header finisce con CRLF (\r\n)
- La fine della ‘header section’ è identificata da una linea bianca.
  - Solo CRLF
- Per i metodi GET e HEAD, la fine degli headers corrisponde con la fine della richiesta !

POST

- Il metodo POST include dei *contenuti (dati)* dopo gli headers (dopo la linea bianca).

- Non c'è un formato prestabilito pei i dati (raw bytes).

- Il metodo POST deve includere una linea Content-Length nella sezione degli headers:

```
Content-length: 267
```

Upload dell'input di un form

Metodo Post:

- La pagina web spesso include un form per l'input dell'utente
- L'input arriva al server nel corpo dell'entità

Metodo URL:

- Usa il metodo GET
- L'input arriva al server nel campo URL della riga di richiesta:

`www.somesite.com/animalsearch?monkeys&banana`

Es: una richiesta POST

```
POST /~vetrano/No-change.cgi HTTP/1.1
Accept: */*
Host: www.univaq.it
User-Agent: Secure_Agent V2.3
Content-Length: 37
Referer: http://costanti.univaq.it/blog

St_uid=6660182722&item=test1&grade=27
```

GET vs POST

- Si ricorda che le ‘query strings’ (ovvero le coppie ‘name=value’) vengono trasferite
  - nell’ URL delle richieste GET:
    - GET /blog.php name1=value1&name2=value2 HTTP/1.1
      Host: hiperlab.org
  - e nel body delle richieste POST:
    - POST /blog.php HTTP/1.1
      Host: hiperlab.org
      content-length=##

      name1=value1&name2=value2

GET o POST ?

- R #1: Usare GET in ambiente sicuro e POST in ambiente insicuro anche se GET è più flessibile:
  - Le richieste GET possono essere ‘cached’
  - Una GET rimane nel browser history
  - Ad una GET si può associare un bookmark
  - Una GET può essere condivisa con altri
  - Una GET può essere editata
- R #2: Usare POST con dati sensibili.
- R #3: Usare POST con richieste molto lunghe
  - Internet Explorer forza un URL a non superare 2048 caratteri.
- R #4: Usare GET in ambiente AJAX
  - Quando si usa XMLHttpRequest, i browsers implementano la POST comew un processo a due step (prima si spediscono le header-lines e poi i dati). Questo significa che una GET è più veloce – cosa gradita in ambiente AJAX.

Tipi di metodi

HTTP/1.0

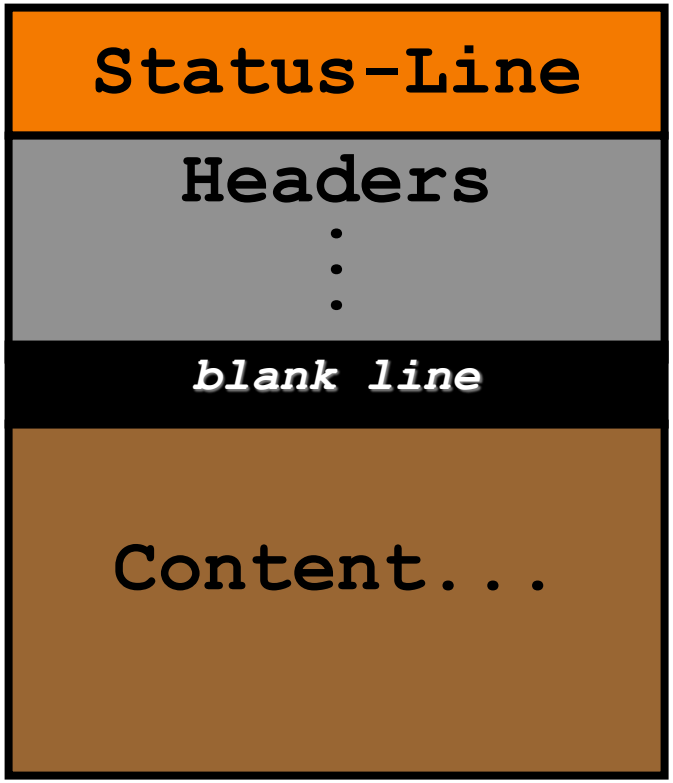
- GET
- POST
- HEAD
  - chiede al server di escludere l'oggetto richiesto dalla risposta

HTTP/1.1

- GET, POST, HEAD
- PUT
  - include il file nel corpo dell'entità e lo invia al percorso specificato nel campo URL
- DELETE
  - cancella il file specificato nel campo URL

Risposta HTTP

- ASCII Status Line
- Headers Section
- Content può essere di tutto (non solo testo)
  - Tipicamente un documento HTML o una immagine.



Messaggio di risposta HTTP

Riga di stato (protocollo codice di stato espressione di stato)

Righe di intestazione

```
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 ...
Content-Length: 6821
Content-Type: text/html

dati dati dati dati dati ...
```

dati, ad esempio il file HTML richiesto

Status code

- Lo status code è un numero di tre cifre, di cui la prima indica la classe della risposta, e le altre due la risposta specifica. Esistono le seguenti classi:
  - 1xx: Informational. Una risposta temporanea alla richiesta, durante il suo svolgimento.
  - 2xx: Successful. Il server ha ricevuto, capito e accettato la richiesta.
  - 3xx: Redirection. Il server ha ricevuto e capito la richiesta, ma sono necessarie altre azioni da parte del client per portare a termine la richiesta.
  - 4xx: Client error. La richiesta del client non può essere soddisfatta per un errore da parte del client (errore sintattico o richiesta non autorizzata).
  - 5xx: Server error. La richiesta può anche essere corretta, ma il server non è in grado di soddisfare la richiesta per un problema interno (suo o di applicazioni CGI).

Esempi di status code

100 **Continue** (se il client non ha ancora mandato il body)

200 **Ok** (GET con successo)

201 **Created** (PUT con successo)

301 **Moved permanently** (URL non valida, il server conosce la nuova posizione

400 **Bad request** (errore sintattico nella richiesta)

401 **Unauthorized** (manca l'autorizzazione)

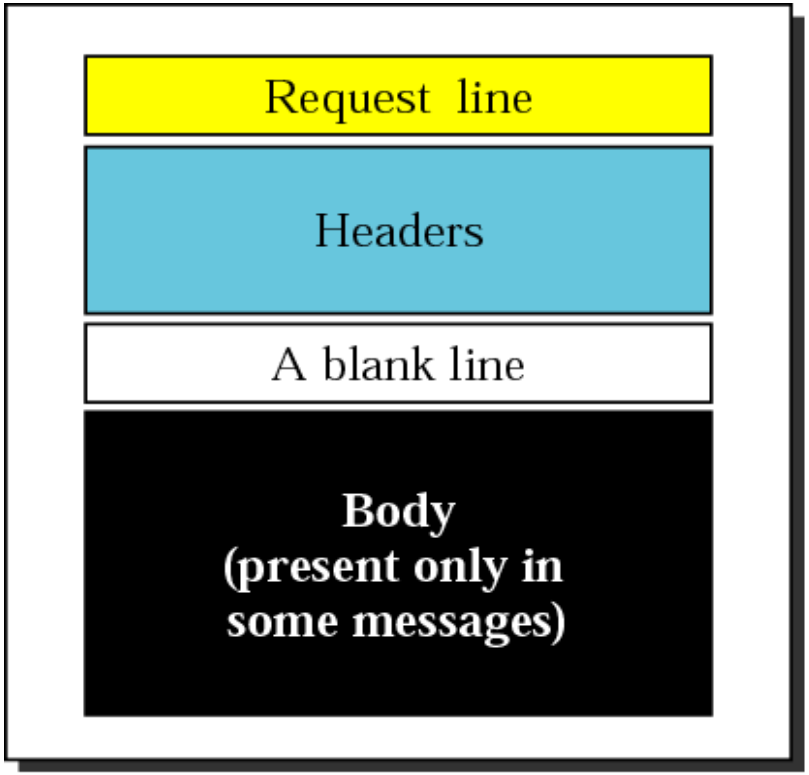
403 **Forbidden** (richiesta non autorizzabile)

404 **Not found** (URL errato)

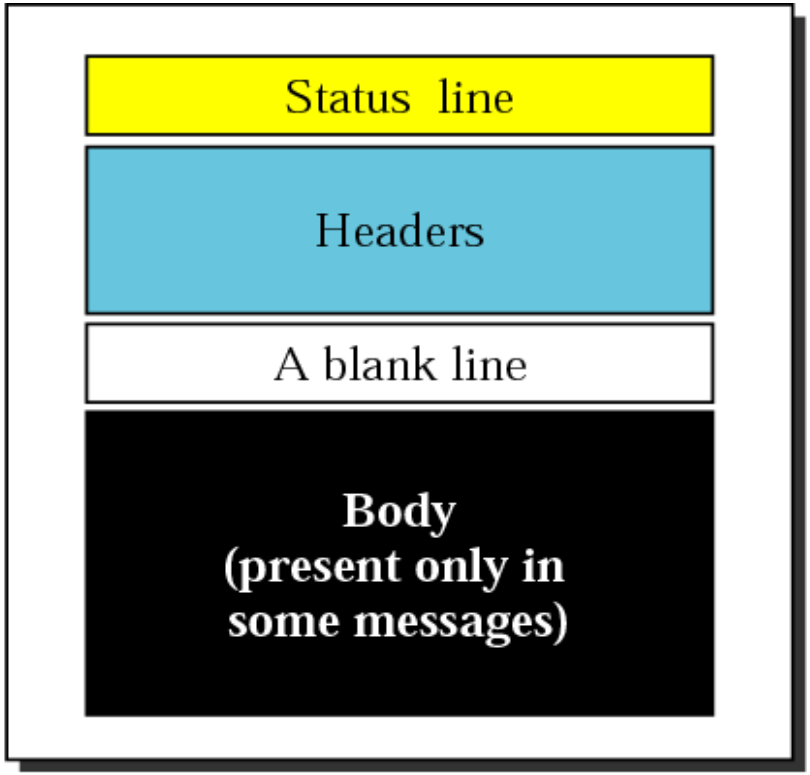
500 **Internal server error** (tipicamente un CGI mal fatto)

501 **Not implemented** (metodo non conosciuto dal server)

Request-Response

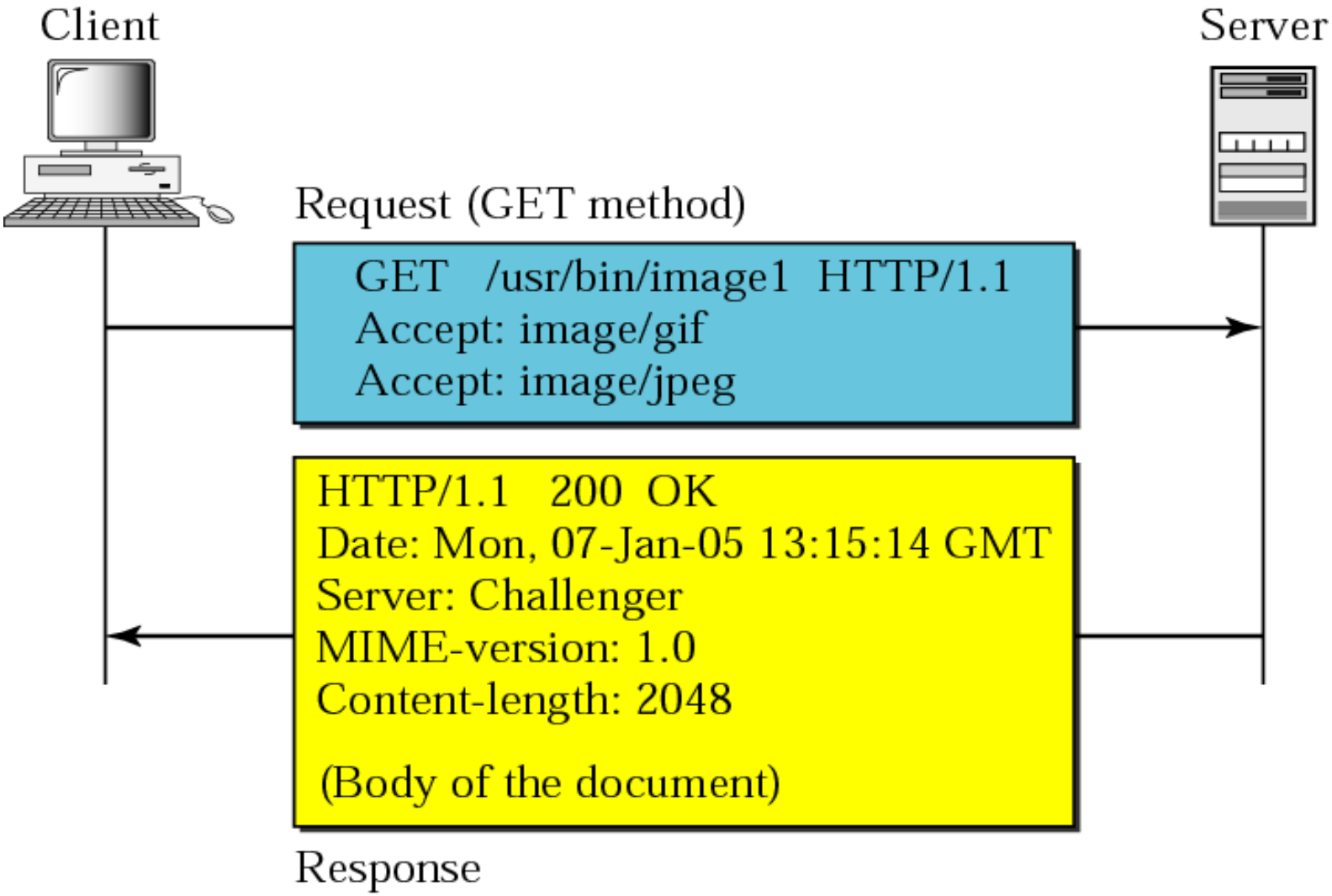


Request message



Response message

Request-Response



Provate HTTP (lato client)

1. Collegatevi via Telnet al vostro server web preferito:

```
telnet cis.poly.edu 80
```

Apre una connessione TCP alla porta 80 (porta di default per un server HTTP) dell'host cis.poly.edu.
Tutto ciò che digitate viene trasmesso alla porta 80 di cis.poly.edu

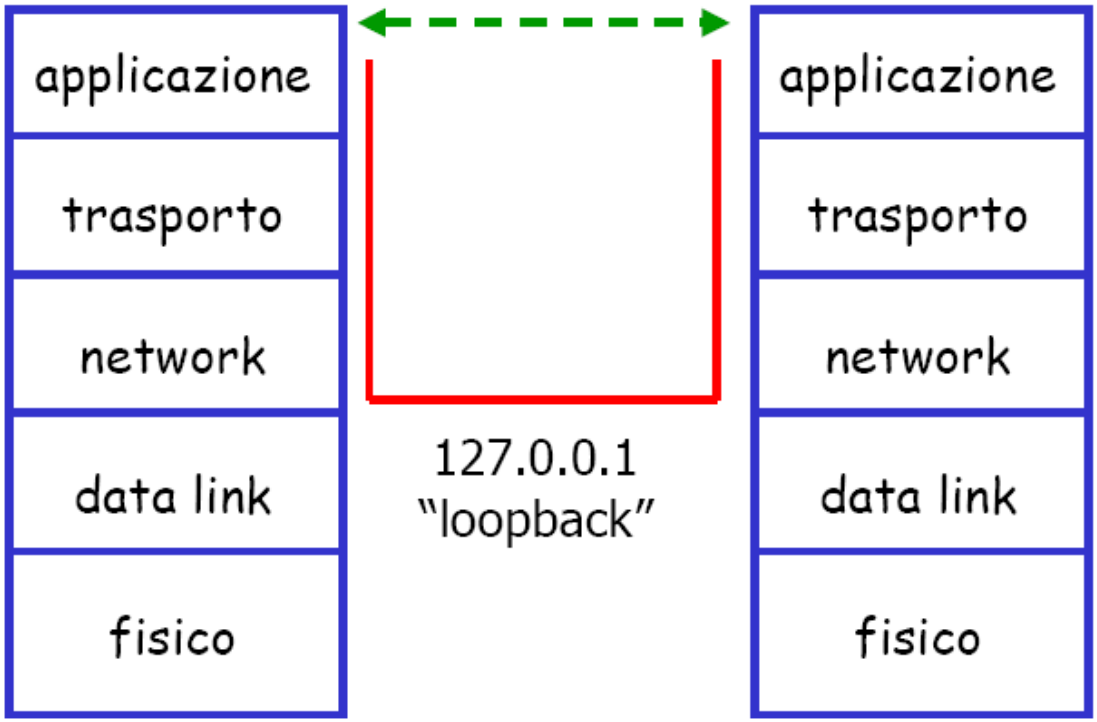
2. Digitate una richiesta GET:

```
GET /~ross/ HTTP/1.1
Host: cis.poly.edu
```

Digitando questo (premete due volte il tasto Invio), trasmettete una richiesta GET minima (ma completa) al server HTTP

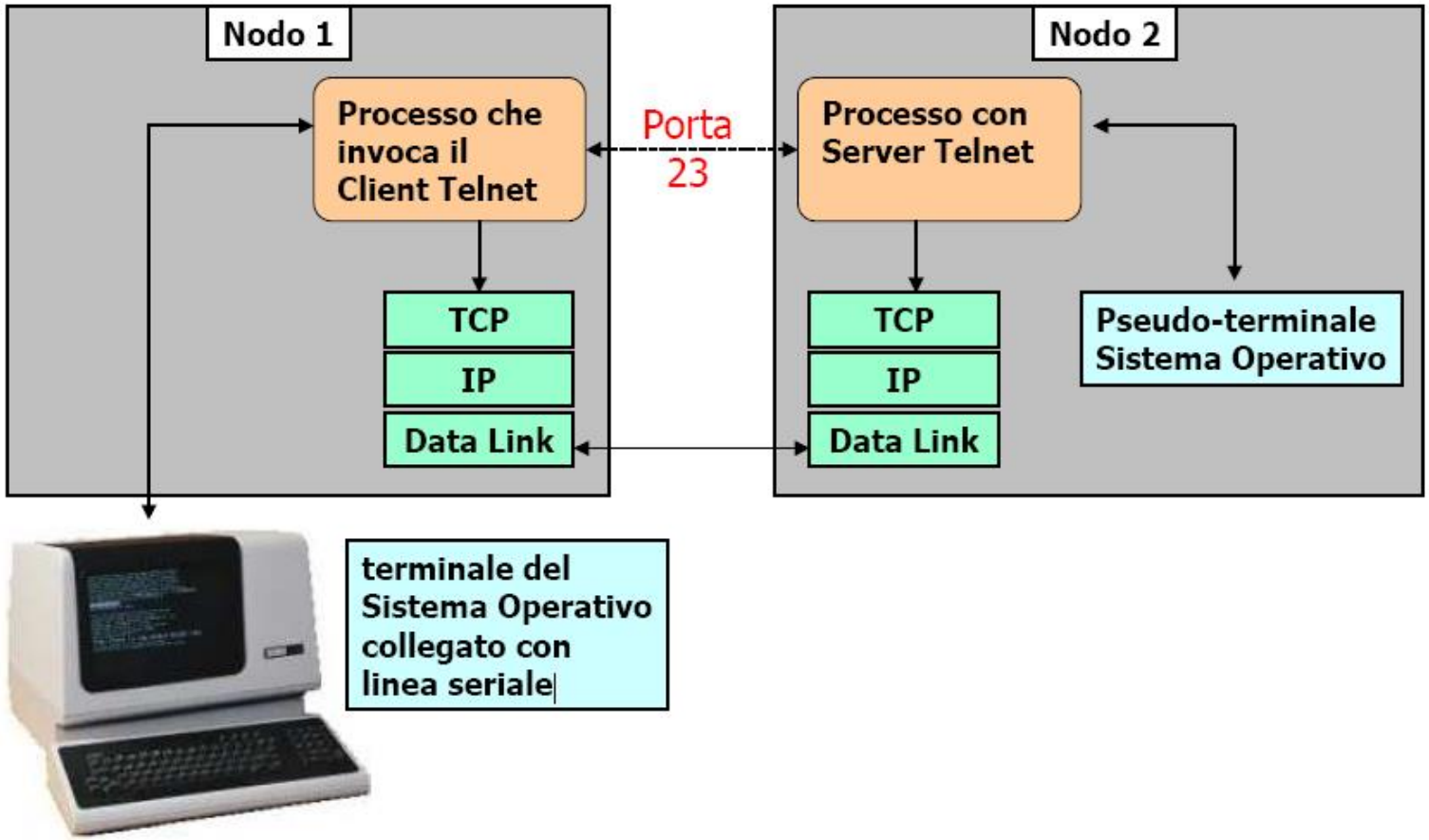
3. Guardate il messaggio di risposta trasmesso dal server HTTP!

Internet fra processi nello stesso nodo



Vantaggio di astrarsi dalle specifiche del particolare S.O.

Telnet


Nodo 1
Processo che invoca il Client Telnet
TCP
IP
Data Link
Porta 23
Nodo 2
Processo con Server Telnet
TCP
IP
Data Link
Pseudo-terminale Sistema Operativo
terminale del Sistema Operativo collegato con linea seriale

Telnet sulla porta 80

```
$ telnet www.hiperlab.org 80
Trying 62.149.131.80...
Connected to www.hiperlab.org (62..149.131.80).
Escape character is '^]'.
GET /univaq  HTTP/1.1
From: info@pippo.edu

HTTP/1.1 200 OK
Date: Thu, 28 Oct 2007 16:27:46 GMT
Server: Apache/1.3.9 (Unix) ApacheJServ/1.1.2 PHP/4.1.2
PHP/3.0.18
MIME-version:1.0
Content-Type: text/html
Last-modified: Friday, 15-Oct-07 02:11:31 GMT
Content-length: 12340

Connection closed by foreign host.
```

Osserviamo HTTP in azione

- Esempio Telnet
- Esempio Ethereal

Lato Server (index.html)

UltraEdit-32 - [E:\pers\index.html]

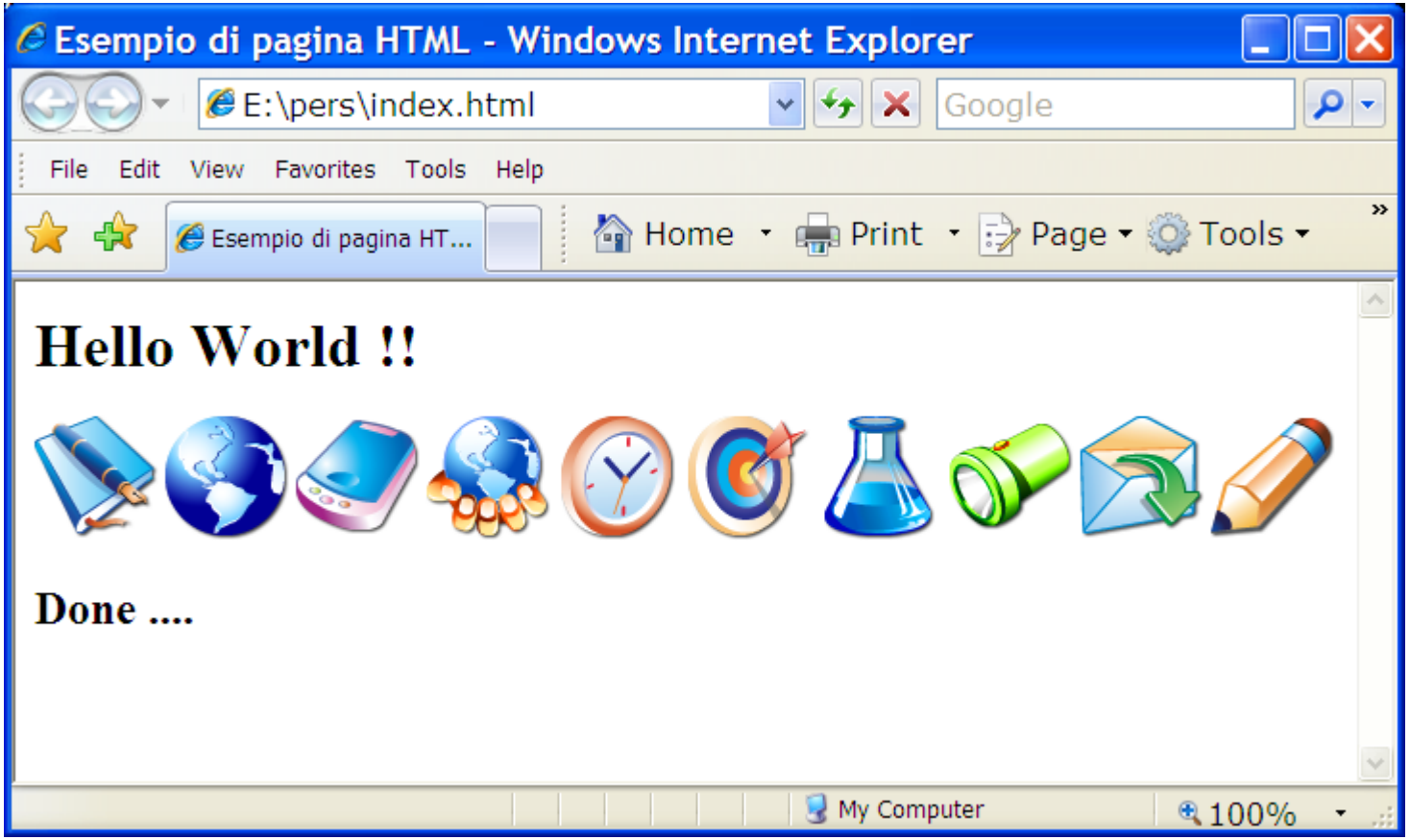
File Edit Search Project View Format Column Macro Advanced Window Help

index.html

```
1 <HTML>
2 <HEAD>
3 <!-- Test di Persistenza [By L. Vetrano] -->
4 <TITLE>Esempio di pagina HTML</TITLE>
5 </HEAD>
6 <BODY>
7 <H1>Hello World !!</H1>
8 <img src="icons/1.png">
9 <img src="icons/2.png">
10 <img src="icons/3.png">
11 <img src="icons/4.png">
12 <img src="icons/5.png">
13 <img src="icons/6.png">
14 <img src="icons/7.png">
15 <img src="icons/8.png">
16 <img src="icons/9.png">
17 <img src="icons/10.png">
18 <H2>Done ....</H2>
19 </BODY>
20
```

Ln 21, Col. 8, C0 | DOS | Mod: 21/10/2008 6.11.48 | File Size: 432 | INS

Lato client (Browser)



arp-dns-http.pcap - Ethereal

File Edit View Go Capture Analyze Statistics Help

Filter: http Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Info
8	0.078855	192.168.0.2	64.28.67.150	HTTP	GET / HTTP/1.1
33	0.559092	192.168.0.2	64.28.67.114	HTTP	GET /Slashdot/pc.gif?index,10099327
36	0.577636	192.168.0.2	64.28.67.57	HTTP	GET /banner/tkgk0172en.gif?10099327
47	0.652638	64.28.67.114	192.168.0.2	HTTP	HTTP/1.1 200 OK (GIF89a)
82	0.801814	64.28.67.150	192.168.0.2	HTTP	HTTP/1.1 200 OK (text/html)
93	0.862869	64.28.67.57	192.168.0.2	HTTP	HTTP/1.1 200 OK (GIF89a)

```
⊞ Frame 8 (706 bytes on wire, 706 bytes captured)
⊞ Ethernet II, Src: 192.168.0.2 (00:20:af:1b:07:fa), Dst: netgear (00:e0:29:68:8b:fb)
⊞ Internet Protocol, Src: 192.168.0.2 (192.168.0.2), Dst: 64.28.67.150 (64.28.67.150)
⊞ Transmission Control Protocol, Src Port: 2741 (2741), Dst Port: http (80), Seq: 1, Ack: 1, Len: 640
⊟ Hypertext Transfer Protocol
  ⊟ GET / HTTP/1.1\r\n
      Request Method: GET
      Request URI: /
      Request Version: HTTP/1.1
    Host: slashdot.org\r\n
    User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:0.9.7) Gecko/20011221\r\n
    Accept: text/xml, application/xml, application/xhtml+xml, text/html;q=0.9, image/png, image/jpeg, image/g
    Accept-Language: en-us\r\n
    Accept-Encoding: gzip, deflate, compress;q=0.9\r\n
    Accept-Charset: ISO-8859-1, utf-8;q=0.66, *;q=0.66\r\n
    Keep-Alive: 300\r\n
    Connection: keep-alive\r\n
    Cookie: user=%2539%2536%2539%2536%253a%253a%2538%2531%2536%2535%2539%2564%2536%2534%2565%2564%2530%2533%2
    \r\n
```

```
0000  00 e0 29 68 8b fb 00 20  af 1b 07 fa 08 00 45 00   ..)h...  ......E.
0010  02 b4 7d 84 40 00 40 06  76 63 c0 a8 00 02 40 1c   ..}.@.@. vc....@.
0020  43 96 0a b5 00 50 74 49  43 bd 75 c8 67 f6 80 18   C....PtI C.u.g...
0030  7d 78 d2 32 00 00 01 01  08 0a 02 49 eb 5e 10 dc   }x.2.... ...I.^..
0040  f9 dc 47 45 54 20 2f 20  48 54 54 50 2f 31 2e 31   ..GET /  HTTP/1.1
0050  0d 0a 48 6f 73 74 3a 20  73 6c 61 73 68 64 6f 74   ..Host:  slashdot
0060  2e 6f 72 67 0d 0a 55 73  65 72 2d 41 67 65 6e 74   .org..Us er-Agent
```

File: "D:\Recovered-01-09-06\Lezioni\ar P: 274 D: 6 M: 0

EffeTech HTTP Sniffer - Unregistered Version

File View Sniffer Help

No.	Time	Client(IP:PORT)	Server(IP:PORT)	URL	File Le...	Status	Information
0	Apr 17,...	192.168.0.50 :59880	www.hiperlab.org	/		304, FIN	txt
1	Apr 17,...	192.168.0.50 :59914	www.hiperlab.org	/css/style.css		304, FIN	css
2	Apr 17,...	192.168.0.50 :59915	www.hiperlab.org	/js/mootools-yui-compressed.js		206, FIN	txt
3	Apr 17,...	192.168.0.50 :59911	www.hiperlab.org	/lighter/Lighter.js	14053	200, FIN	txt
4	Apr 17,...	192.168.0.50 :59880	www.hiperlab.org	/lighter/Fuel.css.js		304, FIN	js
5	Apr 17,...	192.168.0.50 :59880	www.hiperlab.org	/lighter/Fuel.css.js		304, FIN	js
6	Apr 17,...	192.168.0.50 :59914	www.hiperlab.org	/css/style.css		304, FIN	css
7	Apr 17,...	192.168.0.50 :59880	www.hiperlab.org	/css/imageMenu.css		304, FIN	css
8	Apr 17,...	192.168.0.50 :59880	www.hiperlab.org	/js/fx.elements.js		404, FIN	htm
9	Apr 17,...	192.168.0.50 :59880	www.hiperlab.org	/js/fx.elements.js	12827	200, Responded	jpg
10	Apr 17,...	192.168.0.50 :59914	www.hiperlab.org	/js/imageMenu.js	13378	200, FIN	jpg
11	Apr 17,...	192.168.0.50 :59914	www.hiperlab.org	/js/imageMenu.js		Requested	
12	Apr 17,...	192.168.0.50 :59996	www.hiperlab.org	/lighter/Fuel.js.js		304, FIN	js
13	Apr 17,...	192.168.0.50 :59995	www.hiperlab.org	/lighter/Fuel.html.js		304, FIN	js
14	Apr 17,...	192.168.0.50 :59996	www.hiperlab.org	/img/hlab.png	15869	200, FIN	png

HTTP Request Header

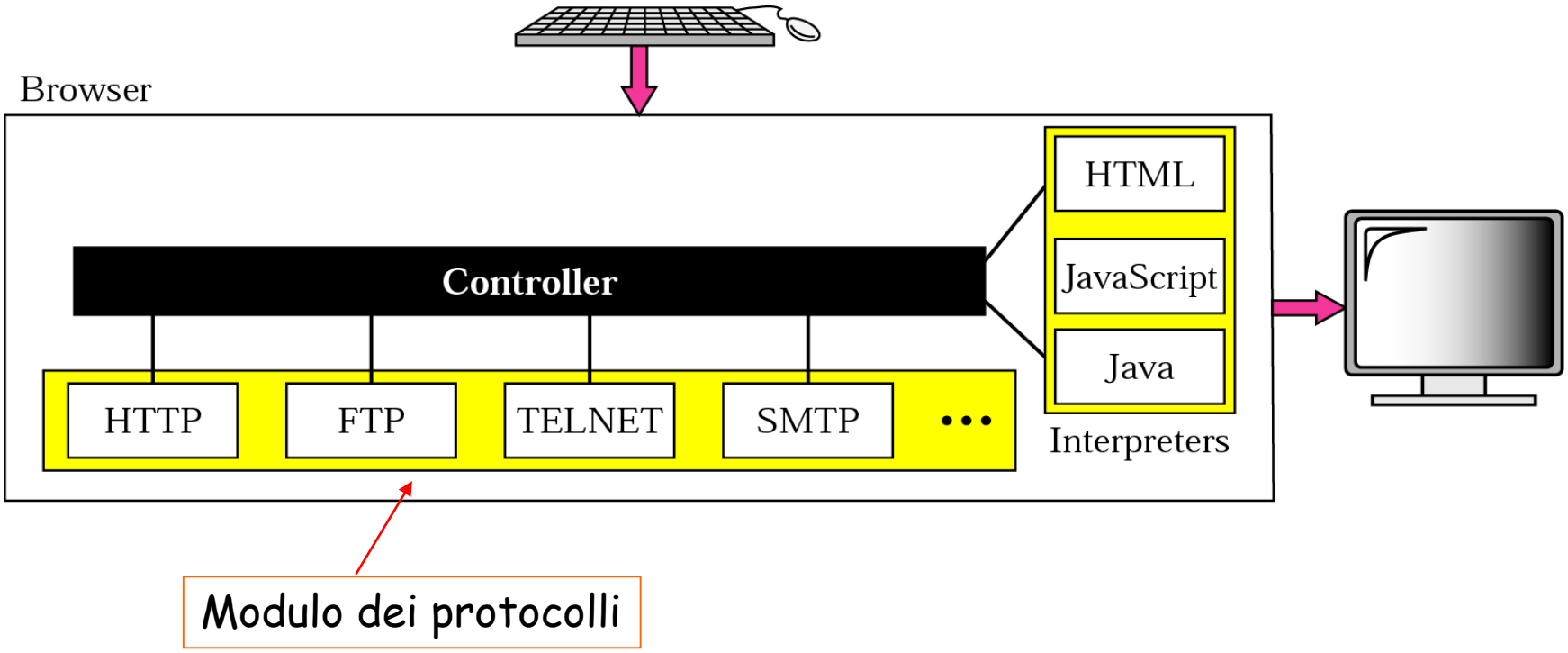
```
GET / HTTP/1.1
Host: www.hiperlab.org
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; it;
rv:1.9.2) Gecko/20100115 Firefox/3.6
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,
*/*;q=0.8
Accept-Language: it-it,it;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
If-Modified-Since: Sun, 28 Mar 2010 12:21:09 GMT
If-None-Match: "80688b2571ceca1:271ac5"
Cache-Control: max-age=0
```

HTTP Response Header

```
HTTP/1.1 304 Not Modified
Date: Sat, 17 Apr 2010 08:20:56 GMT
Content-Location: http://www.hiperlab.org/index.htm
Last-Modified: Sun, 28 Mar 2010 12:21:09 GMT
Accept-Ranges: bytes
ETag: "80688b2571ceca1:271ac5"
Server: Microsoft-IIS/6.0
MicrosoftOfficeWebServer: 5.0_Pub
X-Powered-By: ASP.NET
```

Ready | Buffer: 1% | URLs: 21 | Packets: 156

Browser


Browser
Controller
HTTP
FTP
TELNET
SMTP
...
HTML
JavaScript
Java
Interpreters
Modulo dei protocolli

Interazione client-server

cookies

Molti dei più importanti siti web usano i cookies

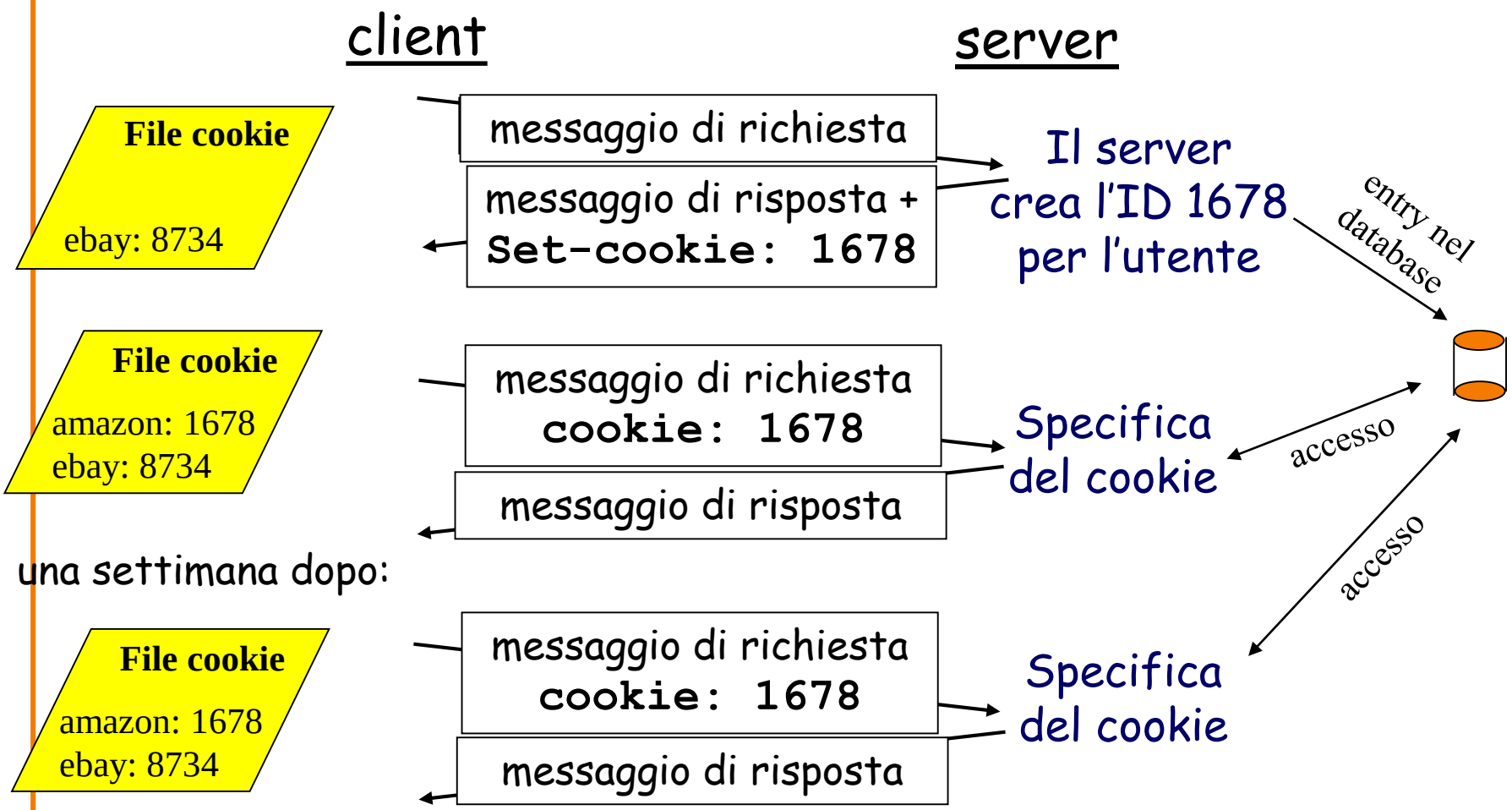
Quattro componenti:

1. Una riga di intestazione nel messaggio di *risposta* HTTP
2. Una riga di intestazione nel messaggio di *richiesta* HTTP
3. Un file cookie mantenuto sul sistema terminale dell'utente e gestito dal browser dell'utente
4. Un database sul sito

Esempio:

- Susan accede sempre a Internet dallo stesso PC
- Visita per la prima volta un particolare sito di commercio elettronico
- Quando la richiesta HTTP iniziale giunge al sito, il sito crea un identificativo unico (ID) e una *entry* nel database per ID

Cookies (continua)



Cookies (continua)

Cosa possono contenere i cookie:

- autorizzazione
- carta per acquisti
- raccomandazioni
- stato della sessione dell'utente (e-mail)

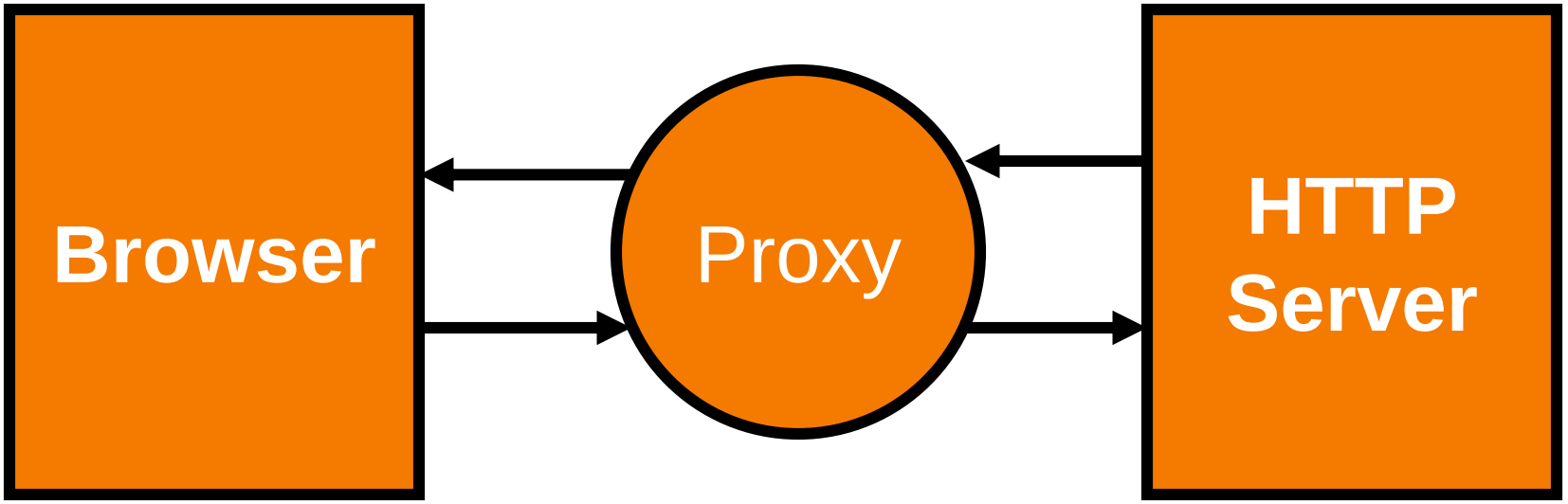
Si noti che i contenuti dei cookies non sono mai letti dal browser o notificati all'utente: il cookie è preparato e consumato dal server.

nota

Cookie e privacy:

- i cookie permettono ai siti di imparare molte cose sugli utenti
- l'utente può fornire al sito il nome e l'indirizzo e-mail
- i motori di ricerca usano il reindirizzamento e i cookie per sapere ancora di più
- le agenzie pubblicitarie ottengono informazioni dai siti

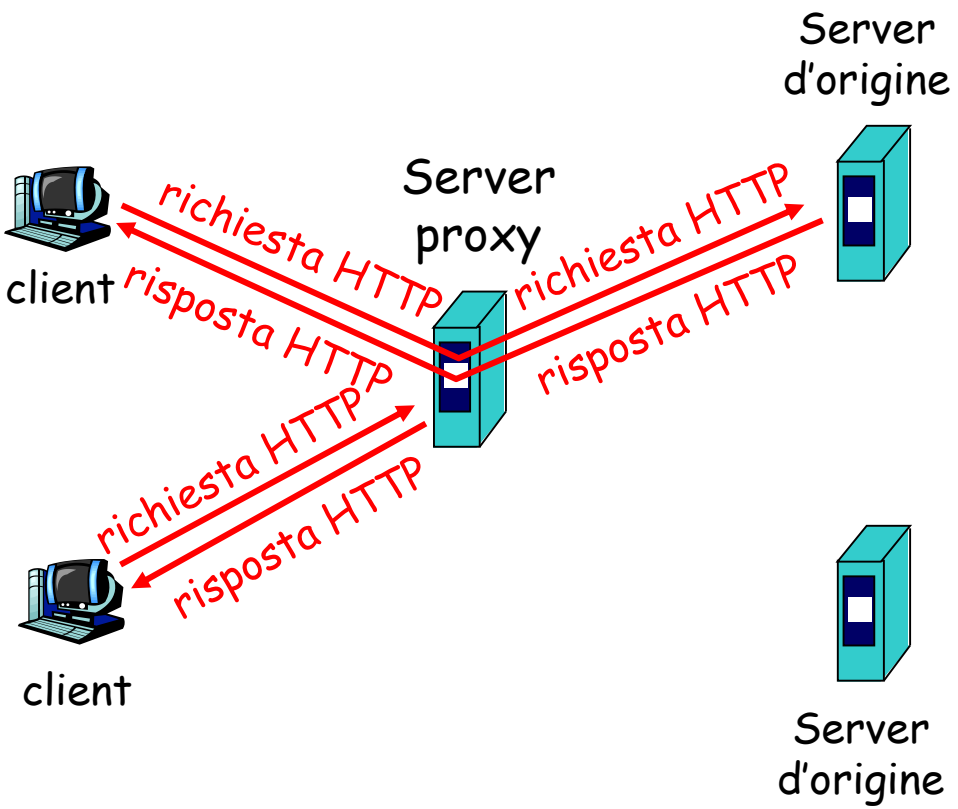
HTTP Proxy Server



Cache web (server proxy)

Obiettivo: soddisfare la richiesta del client senza coinvolgere il server d'origine

- L'utente configura il browser: accesso al Web tramite la cache
- Il browser trasmette tutte le richieste HTTP alla cache
  - oggetto nella cache: la cache fornisce l'oggetto
  - altrimenti la cache richiede l'oggetto al server d'origine e poi lo inoltra al client



Cache web (continua)

- La cache opera come client e come server
- Tipicamente la cache è installata da un ISP (università, aziende o ISP residenziali)

Perché il caching web?

- Riduce i tempi di risposta alle richieste dei client.
- Riduce il traffico sul collegamento di accesso a Internet.
- Internet arricchita di cache consente ai provider “scadenti” di fornire dati con efficacia (ma così fa la condivisione di file P2P)

GET condizionale

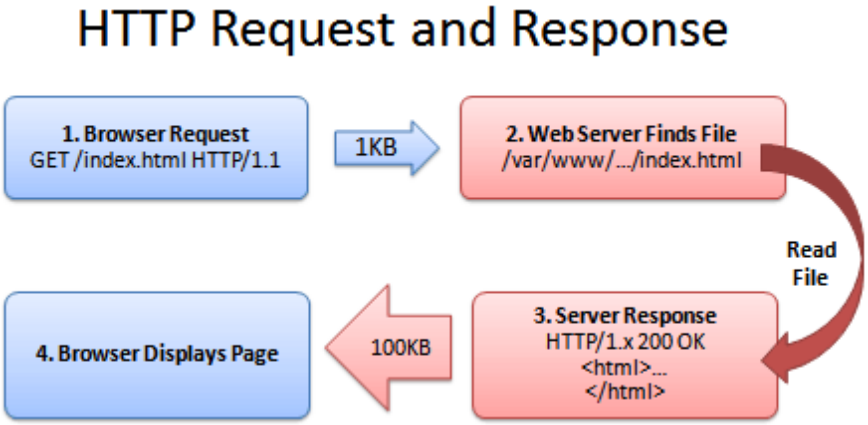
Obiettivo: non inviare un oggetto se la cache ha una copia aggiornata dell'oggetto

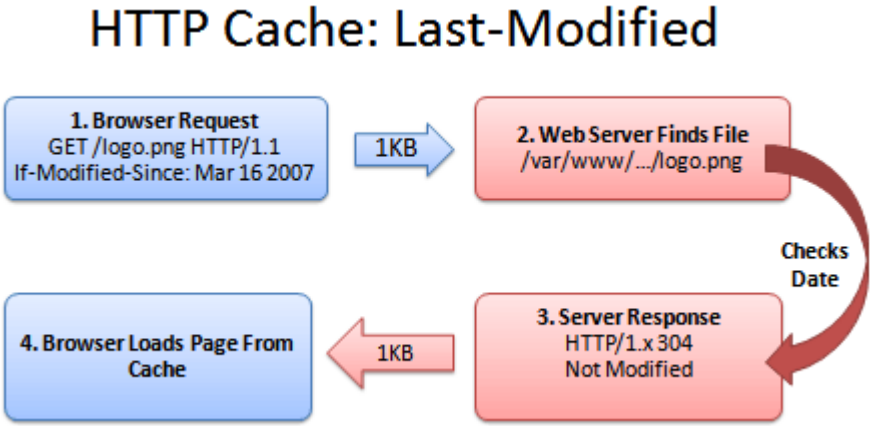
cache: specifica la data della copia dell'oggetto nella richiesta HTTP

```
If-modified-since: <data>
```

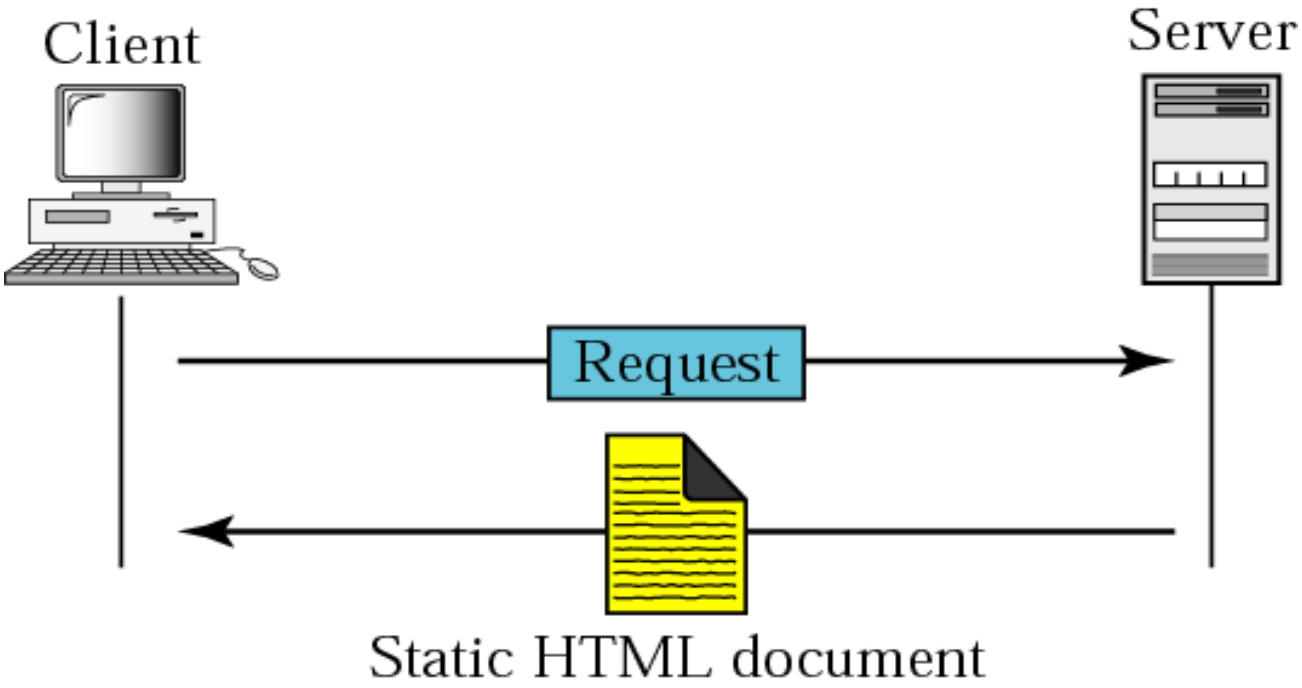
server: la risposta non contiene l'oggetto se la copia nella cache è aggiornata:

```
HTTP/1.0 304 Not Modified
```

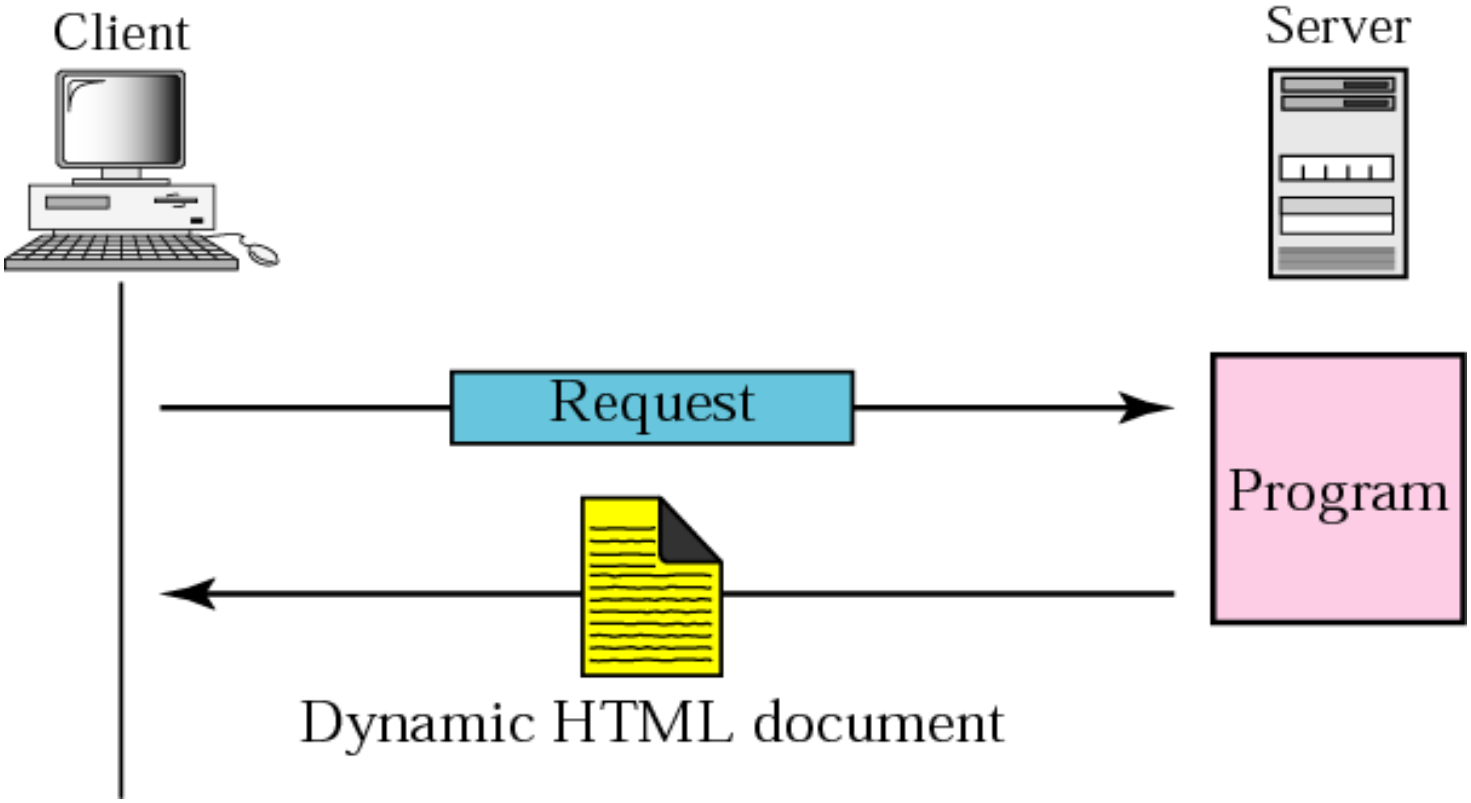




Documenti statici

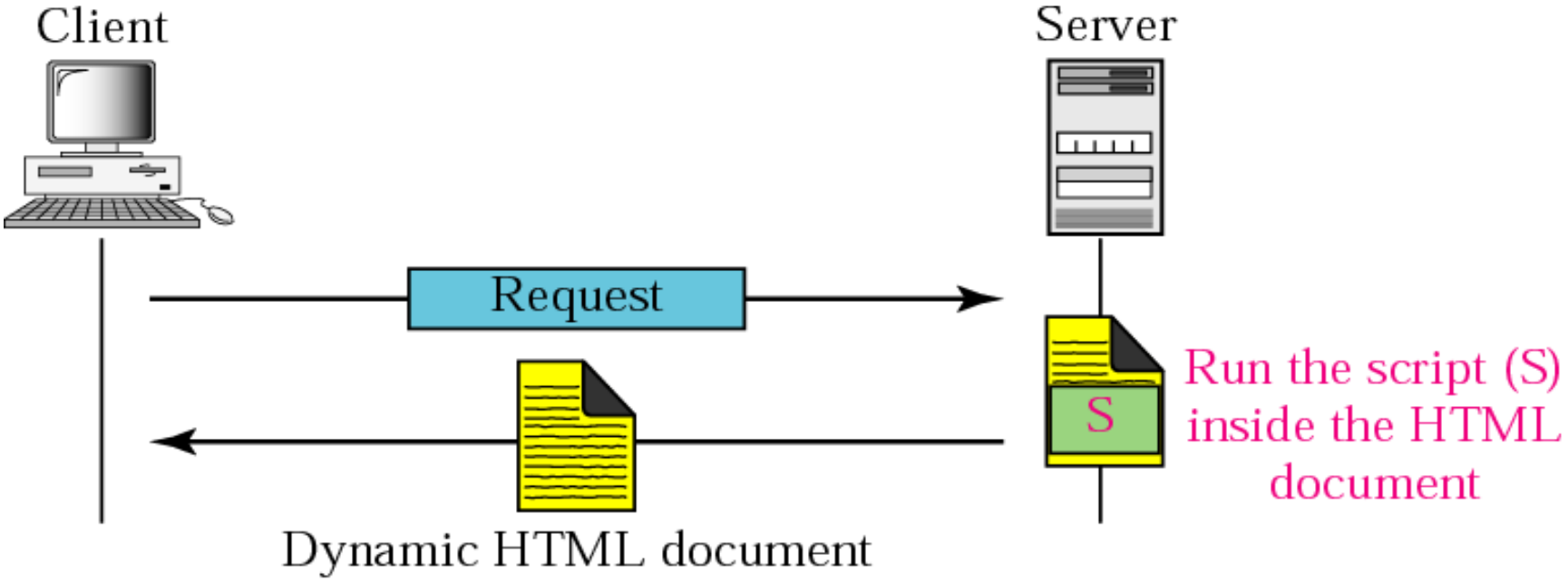


Documenti dinamici CGI

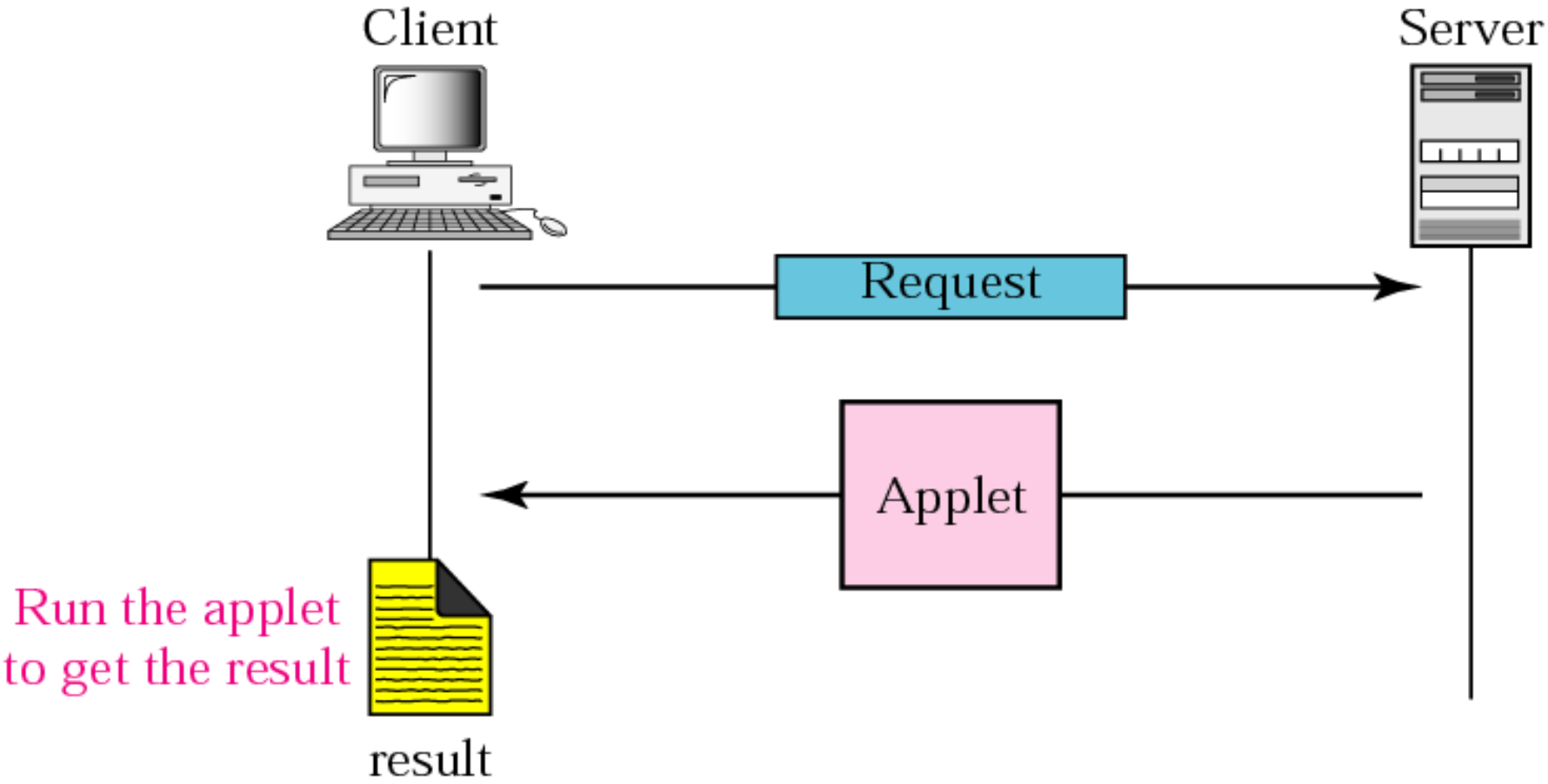


Documenti dinamici SSI

```
<!-- #include file="prova.html" -->
  <!-- #echo var="DATE_LOCAL" -->
```

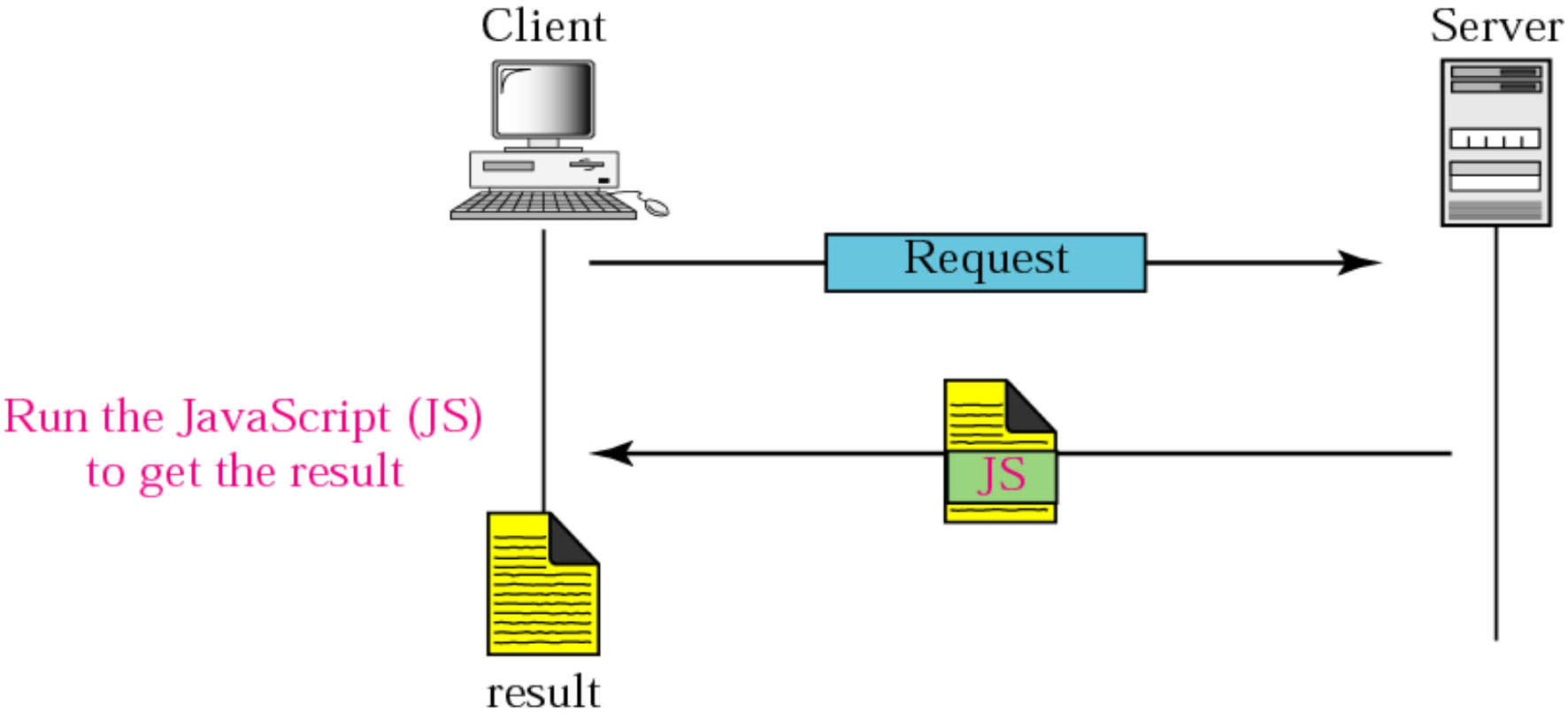


Documenti attivi (Java)



I documenti attivi a volte vengono detti documenti dinamici lato client

Documenti attivi (Javascript)



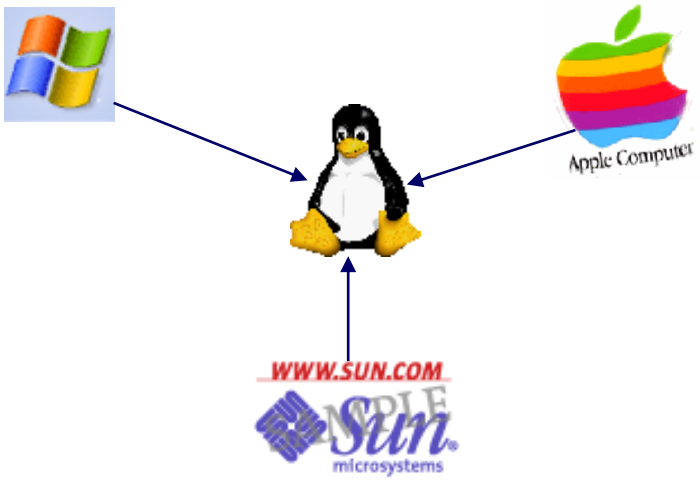
```
<html>
<head>
<script type="text/javascript">
function displayDate()
{
document.getElementById("demo").innerHTML=Date();
}
</script>
</head>
<body>

<h1>My First Web Page</h1>
<p id="demo">This is a paragraph.</p>

<button type="button" onclick="displayDate()">Display
Date</button>

</body>
</html>
```

Perchè un servizio FTP ?

- Scopo: Trasferire files tra due computers
  - Promuovere il file sharing (programmi e/o dati)
  - Incoraggiare l’uso indiretto/implicito di computers remoti
  - Trasferire i dati in modo affidabile e efficiente

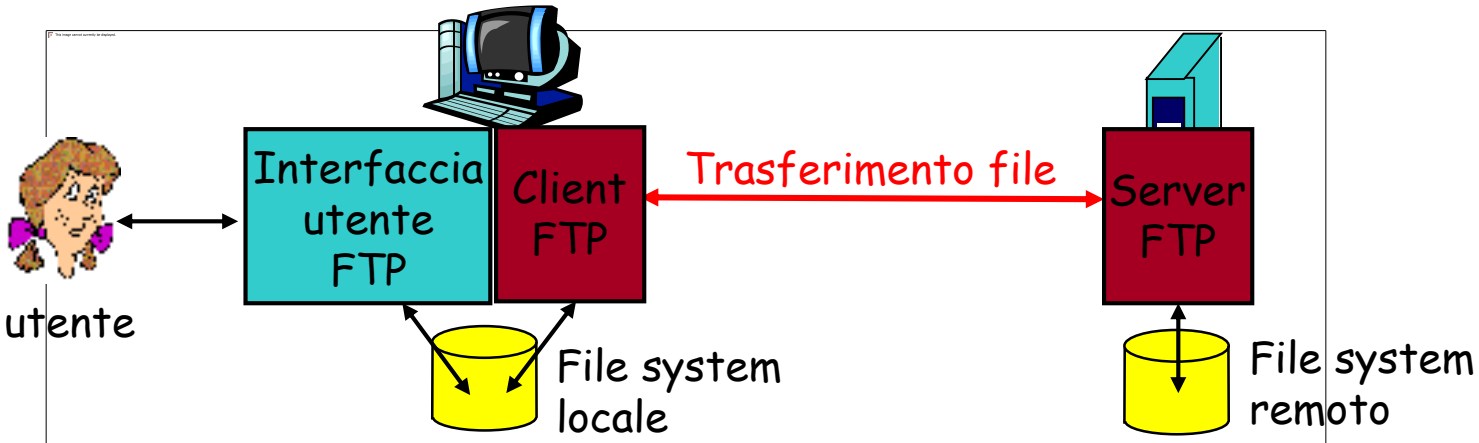
Problemi del File Transfer

- Sistemi eterogenei usano differenti:
  - Operating Systems
  - Character Sets
  - Naming Conventions
  - Directory Structures
  - File Structures and Formats
  - Data representation



- FTP deve indirizzare e risolvere questi problemi

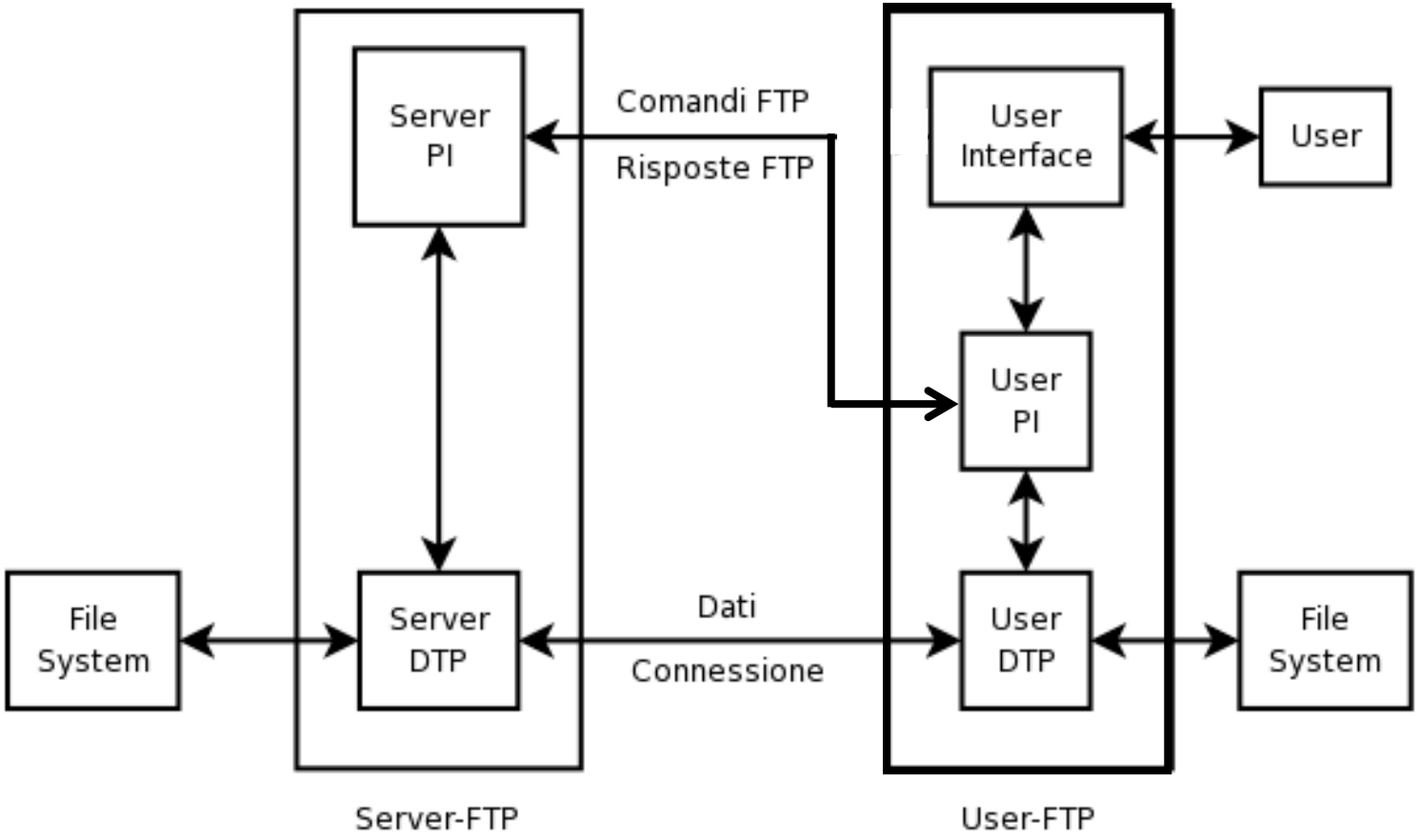
FTP: file transfer protocol



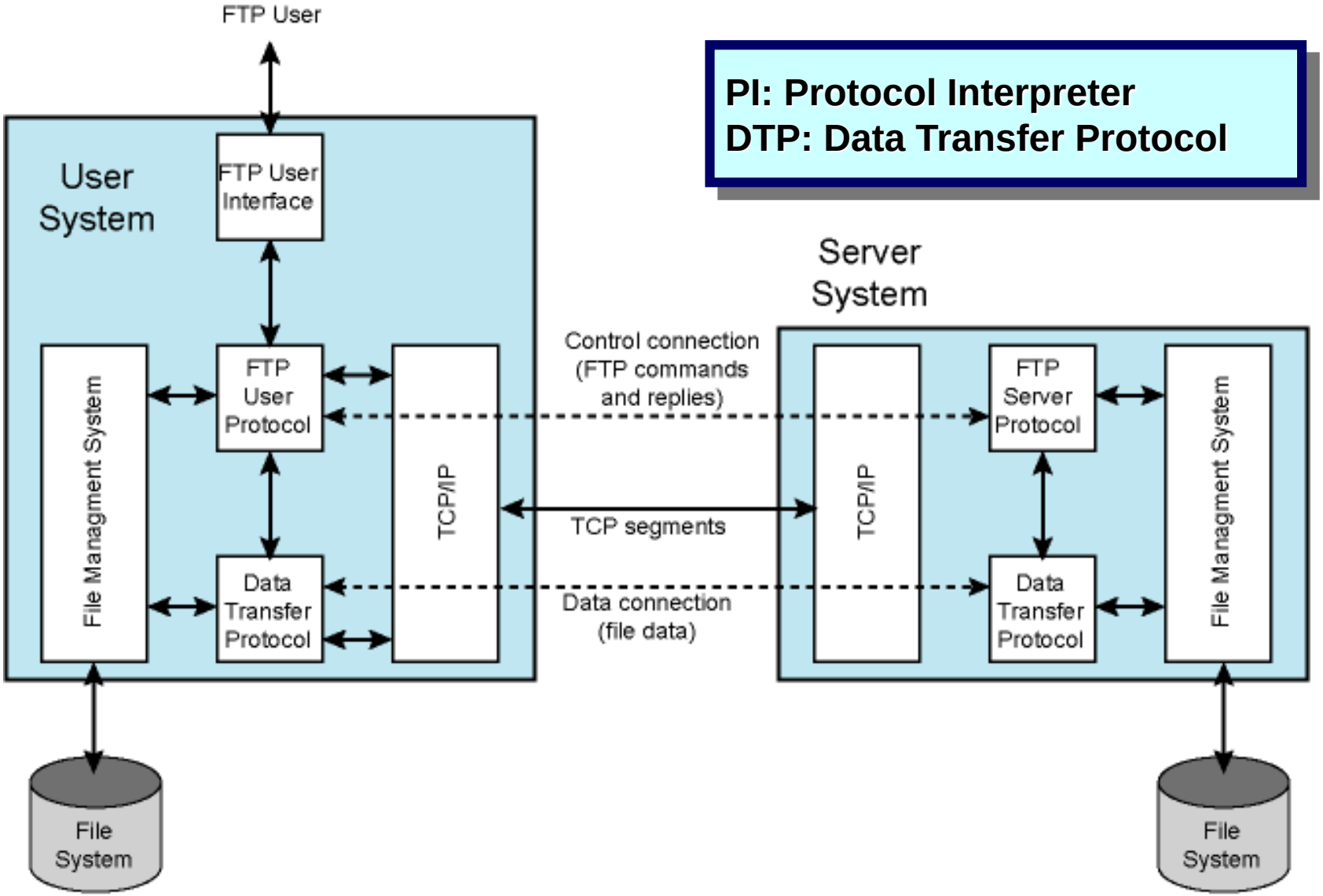
- Trasferimento file a/da un host remoto
- Modello client/server
  - *client:* il lato che inizia il trasferimento (a/da un host remoto)
  - *server:* host remoto
- ftp: RFC 959
- server ftp: porta 21

Il modello FTP

PI: Protocol Interpreter
DTP: Data Transfer Protocol



Il modello FTP



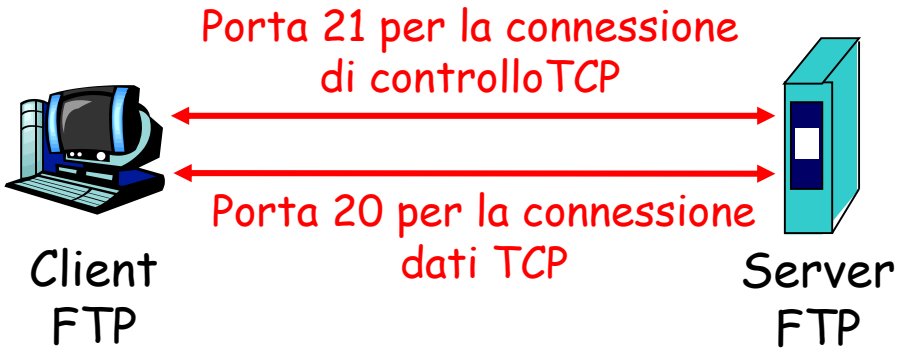
PI: Protocol Interpreter
DTP: Data Transfer Protocol

Connessione di Controllo

- La connessione di controllo è il servizio “well known” sulla porta 21.
- Usa il protocollo a caratteri TELNET per la comunicazione.
- Comandi e risposte sono linee di testo in chiaro (default ASCII-7bit).

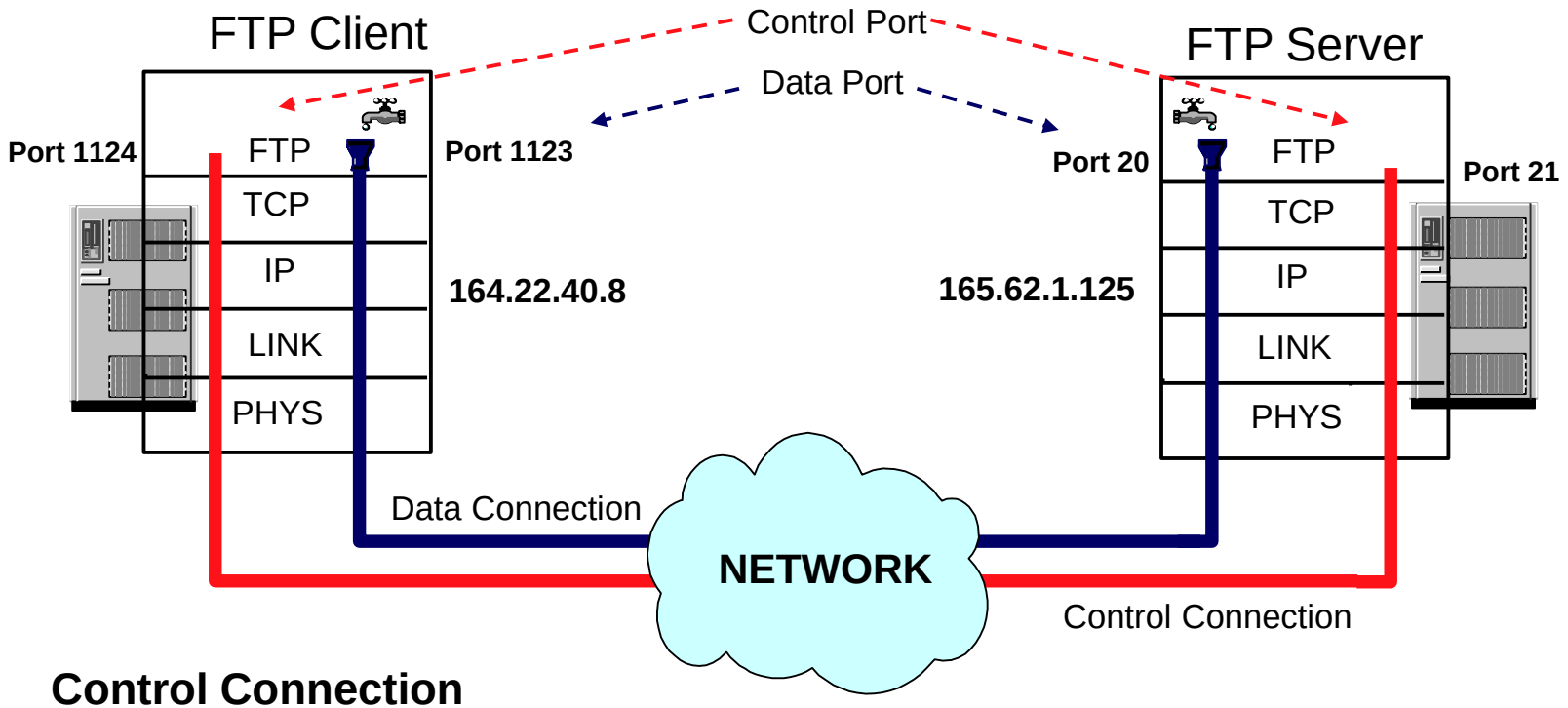
FTP: connessione di controllo, connessione dati

- Il client FTP contatta il server FTP sulla porta 21 (connessione di controllo), specificando TCP come protocollo di trasporto
- Il client ottiene l'autorizzazione sulla connessione di controllo
- Il client cambia la directory remota inviando i comandi sulla connessione di controllo
- Quando il **server** riceve un comando per trasferire un file, **apre** una connessione dati TCP con il client sulla porta 20
- Dopo il trasferimento di un file, il server chiude la connessione dati (**NON** quella di controllo)



- Il server apre una seconda connessione dati TCP per trasferire un altro file.
- Connessione di controllo: "fuori banda" (*out of band*)
- Il server FTP mantiene lo "stato": directory corrente, autenticazione precedente

FTP MODEL

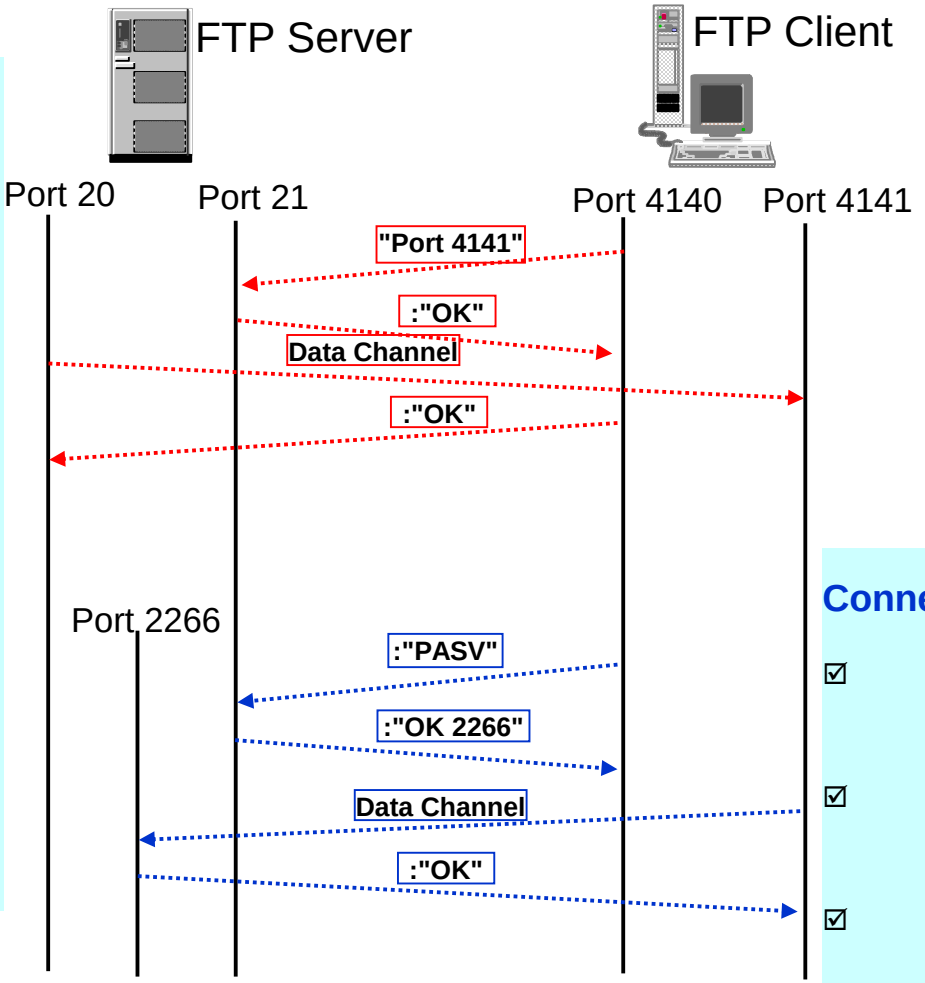


1. **Control Connection**
   1. Uses Well known **port 21** on FTP Server and a locally assigned port on FTP client.
   2. The connection is **created** when the client establishes a connection to the server.
   3. This connection is used for **FTP commands and replies** to control the data transfer
2. **Data Transfer Connection**
   1. Uses well known **port 20** on FTP Server and a locally assigned port on FTP client.
   2. The connection is **created on demand** for each file transfer and **destroyed at the end** of each file transfer.

Esempio di Connessione FTP

Connessione FTP Attiva (normale)

- ☑ Il Cliente apre un canale FTP di controllo verso il server (Porta 21) e dice al server il numero di porta su cui si aspetta i dati (Porta 4141).
- ☑ Il server fa l'ACK (acknowledge) della richiesta.
- ☑ Il server apre il canale dati (Porta 20) verso quello corrispondente del cliente (Porta 4141).
- ☑ Il cliente fa l'ACK di questa connessione.



Connessione FTP Passiva

- ☑ Il Client apre un canale di controllo verso il server FTP (Porta 21) in passive mode.
- ☑ Il server fa l'ACK del passive mode e alloca la Porta 2266 come canale dati per il client.
- ☑ Il client apre il canale dati dal suo (Port 4141) a quello del server (Port 2266).
- ☑ Il server fa l'ACK della connessione dati.

Comandi e risposte FTP (Provateli!!)

Comandi comuni:

- Inviati come testo ASCII-7 bits- sulla connessione di controllo
- **`USER`** ***`username`***
- **`PASS`** ***`password`***
- **`LIST`**
  elenca i file della directory corrente
- **`RETR`** ***`filename`***
  recupera (*get*) un file dalla directory corrente
- **`STOR`** ***`filename`*** memorizza (*put*) un file nell'host remoto

Codici di ritorno comuni:

- Codice di stato ed espressione (come in HTTP)
- **`331 Username OK, password required`**
- **`125 data connection already open; transfer starting`**
- **`425 Can't open data connection`**
- **`452 Error writing file`**

Access Control Commands

USER	*specify user*
PASS	*specify password*
CWD	*change directory*
CDUP	*change directory to parent*
QUIT	*logout*

Transfer Parameter Commands

- PORT publish local data port
- PASV server should listen
- TYPE establish data representation
- MODE establish transfer mode
- STRU establish file structure

Service Commands

RETR	*retrieve file*
STOR	*send file*
STOU	*send file and save as unique*
APPE	*send file and append*
ABOR	*abort prev. service command*
PWD	*print working directory*
LIST	*transfer list of files over data link*

MODE OF TRANSFER

- There are two modes of transfer in FTP: ascii and binary.
  - ASCII is used only for files saved in ASCII format (this includes Postcript files)
  - Binary is used for files that are formatted and saved using a wordprocessing software like Word (.doc), spreadsheets (.xls), images (.jpg, .gif), and many executable programs (.exe) and videos (.avi).

Data Transfer

- User data transfer protocol should "listen" on specified data port
  - using the command PORT a1,a2,a3,a4,p1,p2
  - a1 .. a4 are 4 octets of the user's IP address
  - p1 and p2 is for the port that the user should listen
    - actually calculated as (p1*256+p2)
  - Server initiates data connection and data transfer
- An alternative is Passive Mode
  - by just sending command PASV
  - server listens to a specific port and client should access that port
  - http://www.freefire.org/articles/ftpexample.php

FTP PI interpretation

User Command	FTP Command	Result
CD mystuff	CWD mystuff	Change to a directory called mystuff
GET resume.doc	RETR resume.doc	Download resume.doc to the FTP client
PUT timecard.doc	STOR timecard.doc	Upload timecard.doc to the FTP server
DIR	NLST	Obtain directory list from server

FTP commands

Command	Full Name	Description
USER *username*	User	Identifies the user accessing the file system. Typically, this is the first command sent after the underlying connection is established (the TCP handshake completes successfully).
PASS *password*	Password	Sends the user's password (this information passes across the wire unencrypted by default)
CWD *pathname*	Change Working Directory	Changes the current directory
QUIT	Quit	Terminates the command connection

FTP commands (continued)

Command	Full Name	Description
PORT *h1,h2,h3,h4,p1,p2*	Port	Sets up a data connection port number (*h* indicates the host's IP address bytes; *p* indicates the port number)
PASV	Passive	Asks the server to listen on a non-default data port and wait for the client to establish a new connection on that port number
TYPE *code*	Type	Indicates the data representation type to be used in transfer (for example, ASCII, EBCDIC, and Image [binary] are three possible data types)
RETR *pathname/filename*	Retrieve	Gets a file from the server DTP
STOR *pathname/filename*	Store	Sends a file to the server DTP
DELE *pathname/filename*	Delete	Deletes a file
RMD *pathname*	Remove	Deletes a directory on the FTP server
MKD *pathname*	Make	Creates a directory on the FTP server

```
220-Welcome to Pure-FTPd.
220-You are user number 5 of 30 allowed.
220 You will be disconnected after 15 minutes of inactivity.
USER anonymous
230 Anonymous user logged in
SYST
215 UNIX Type: L8
FEAT
211-Extensions supported:
 EPRT
 IDLE
 MDTM
 SIZE
 REST STREAM
 MLST type*;size*;sizd*;modify*;UNIX.mode*;UNIX.uid*;UNIX.gid*;u
 MLSD
 TVFS
 ESTP
 PASV
 EPSV
 SPSV
 ESTA
 AUTH TLS
 PBSZ
 PROT
211 End.
PWD
257 "/" is your current location
TYPE A
200 TYPE is now ASCII
PORT 192,168,0,100,4,106
200 PORT command successful
LIST -a
150 Connecting to port 25294
226-Options: -a -l
226 31 matches total
CWD img
```

Dialogo tipico FTP

Why Out-of-Band Control?

- Avoids need to mark the end of the data transfer
  - Data transfer ends by closing of data connection
  - Yet, the control connection stays up
- Aborting a data transfer
  - Can abort a transfer without killing the control connection
  - … which avoids requiring the user to log in again
  - Done with an ABOR on the control connection
- Third-party file transfer between two hosts
  - Data connection could go to a different hosts
  - … by sending a different client IP address to the server
  - E.g., user coordinates transfer between two servers

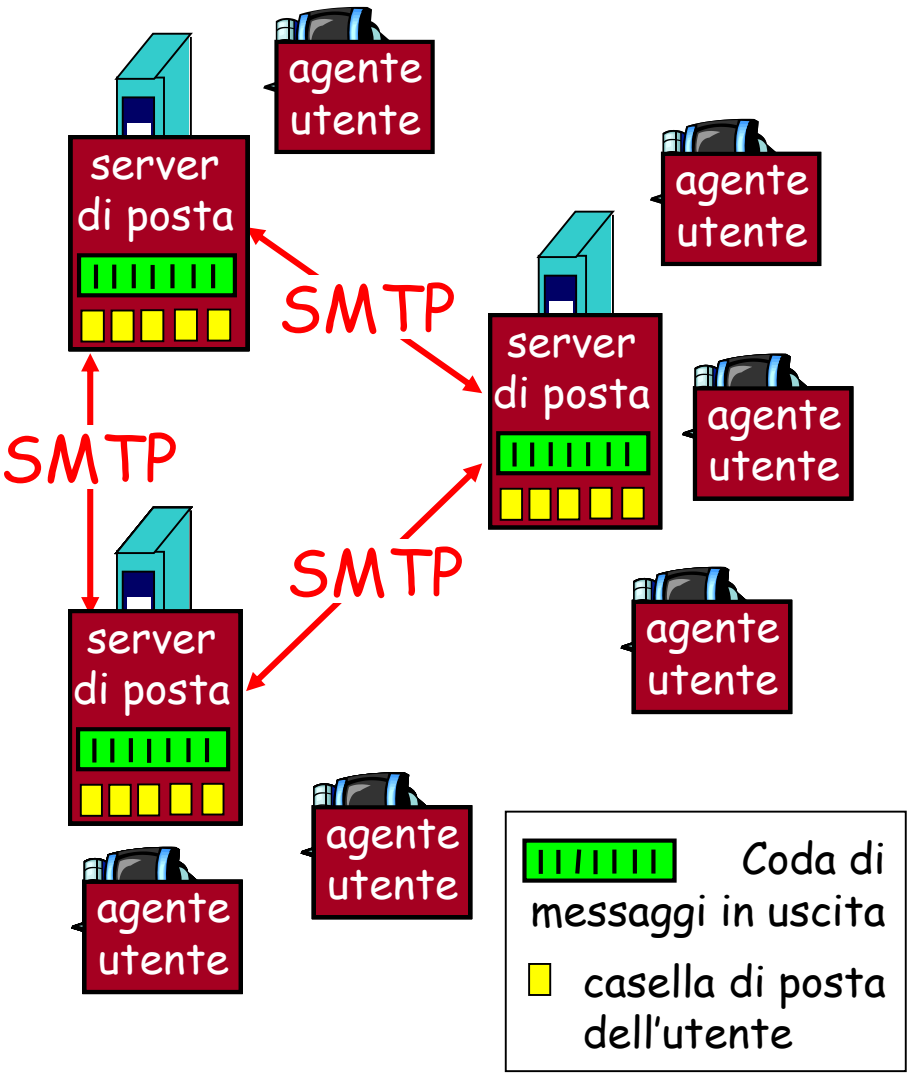
Posta elettronica

Tre componenti principali:

- agente utente
- server di posta
- protocollo: SMTP

Agente utente

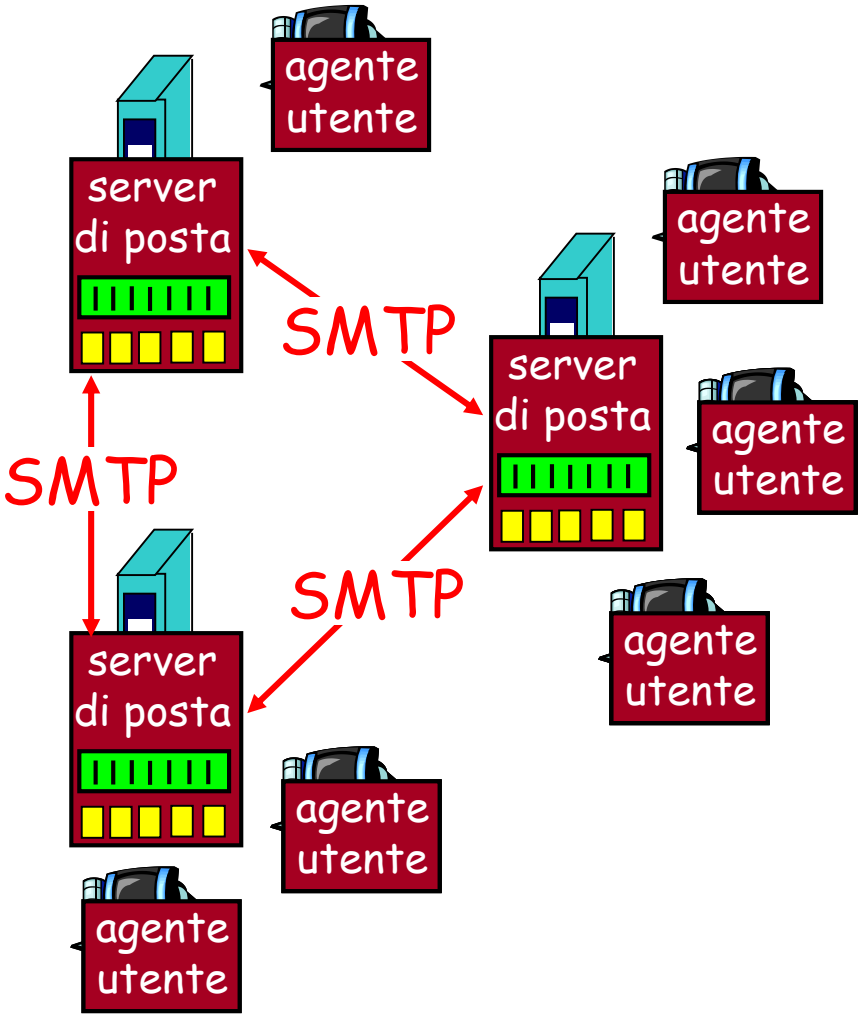
- detto anche “mail reader”
- composizione, editing, lettura dei messaggi di posta elettronica
- esempi: Eudora, Outlook, elm, Netscape Messenger
- i messaggi in uscita o in arrivo sono memorizzati sul server di posta



Posta elettronica: server di posta

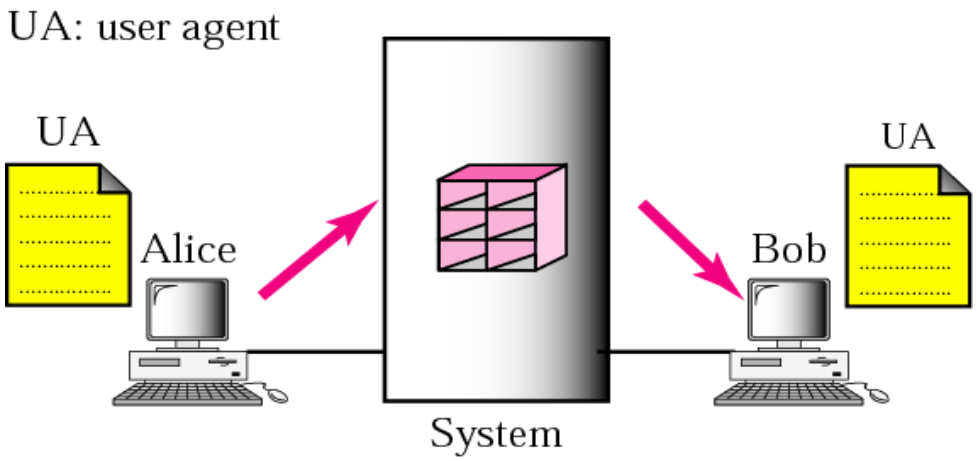
Server di posta

- Casella di posta (*mailbox*) contiene i messaggi in arrivo per l'utente
- Coda di messaggi da trasmettere
- Protocollo SMTP tra server di posta per inviare messaggi di posta elettronica
  - client: server di posta trasmittente
  - "server": server di posta ricevente



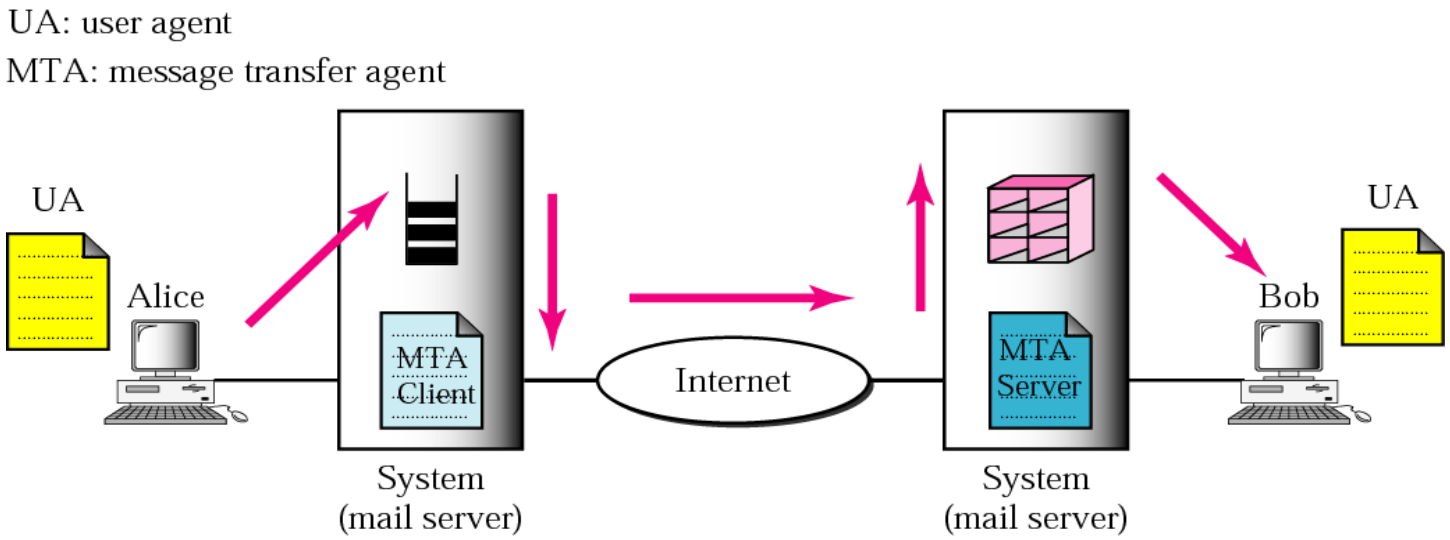
Scenario 1

When the sender and the receiver of an email are on the same system, we need only two user agents.

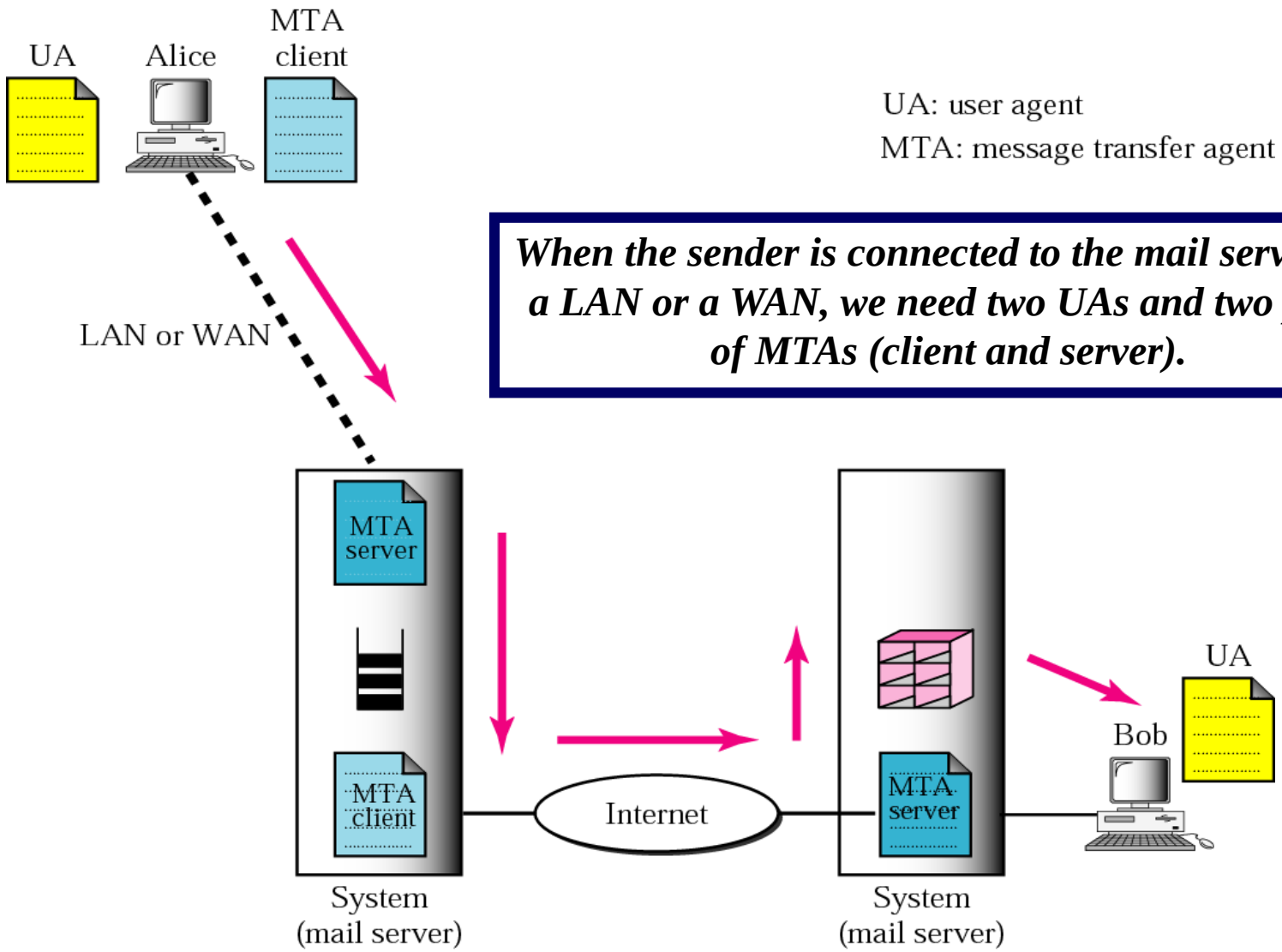


Scenario 2

When the sender and the receiver of an email are on different systems, we need two UAs and a pair of MTAs (client and server).

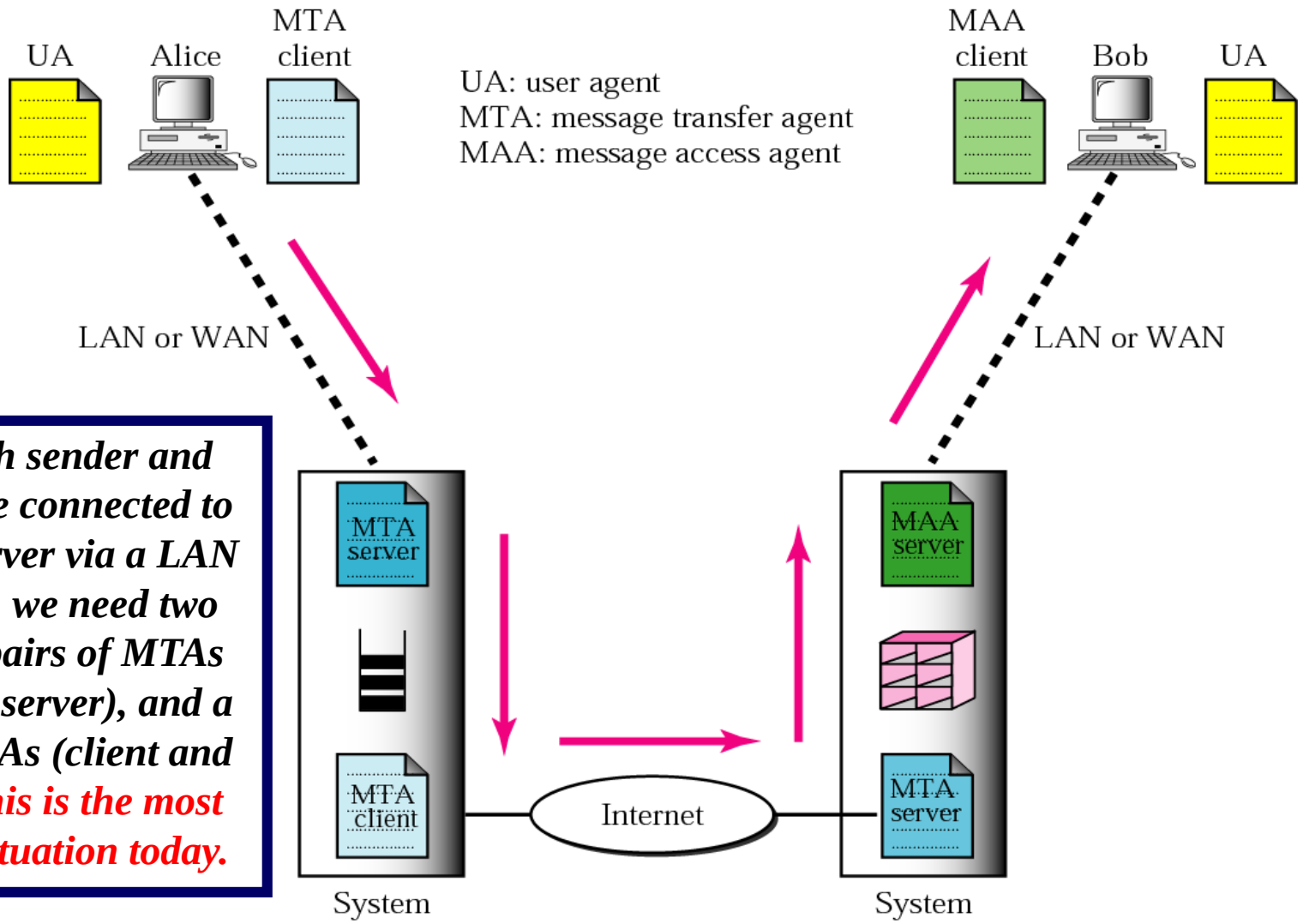


Scenario 3



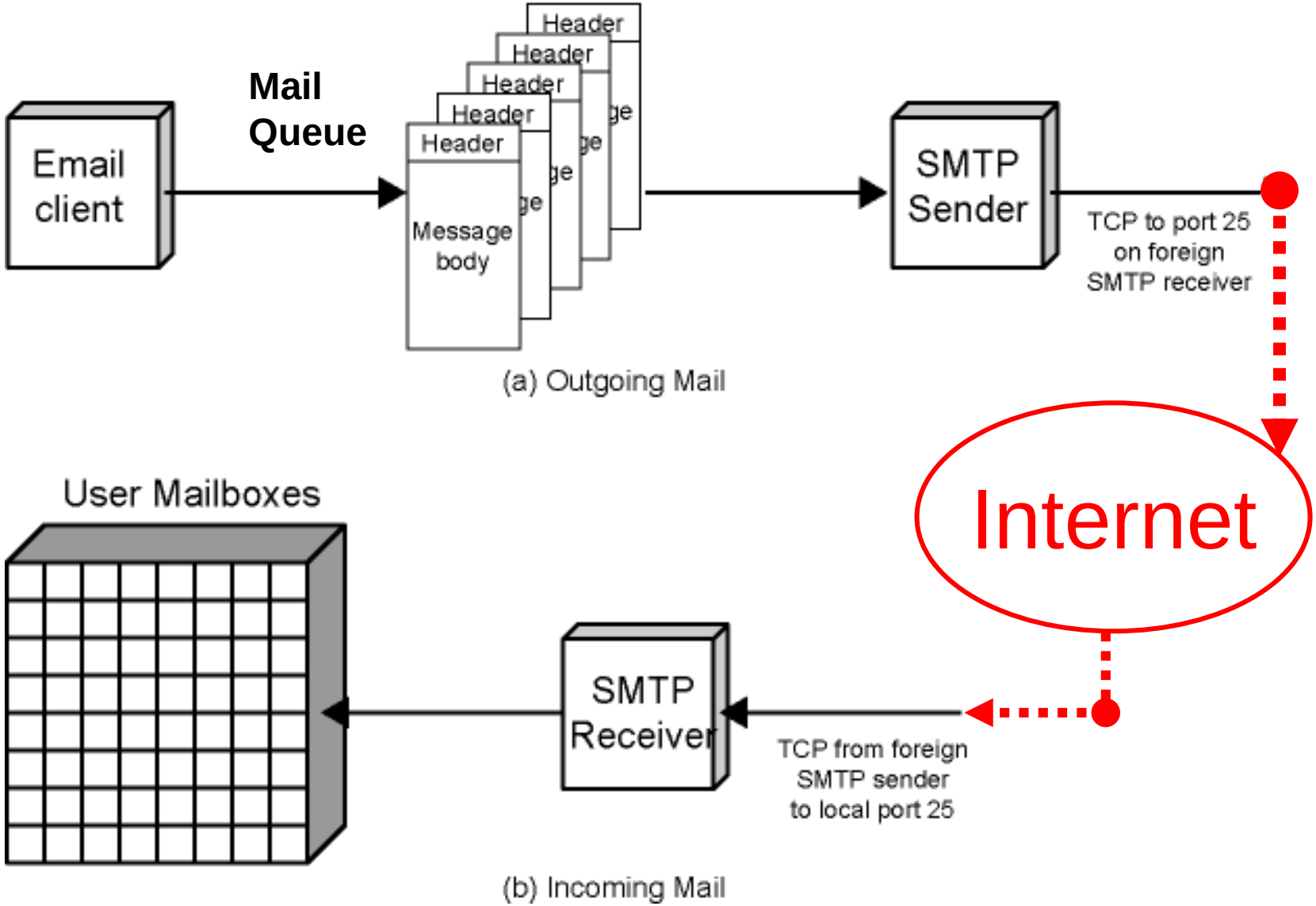
When the sender is connected to the mail server via a LAN or a WAN, we need two UAs and two pairs of MTAs (client and server).

Scenario 4

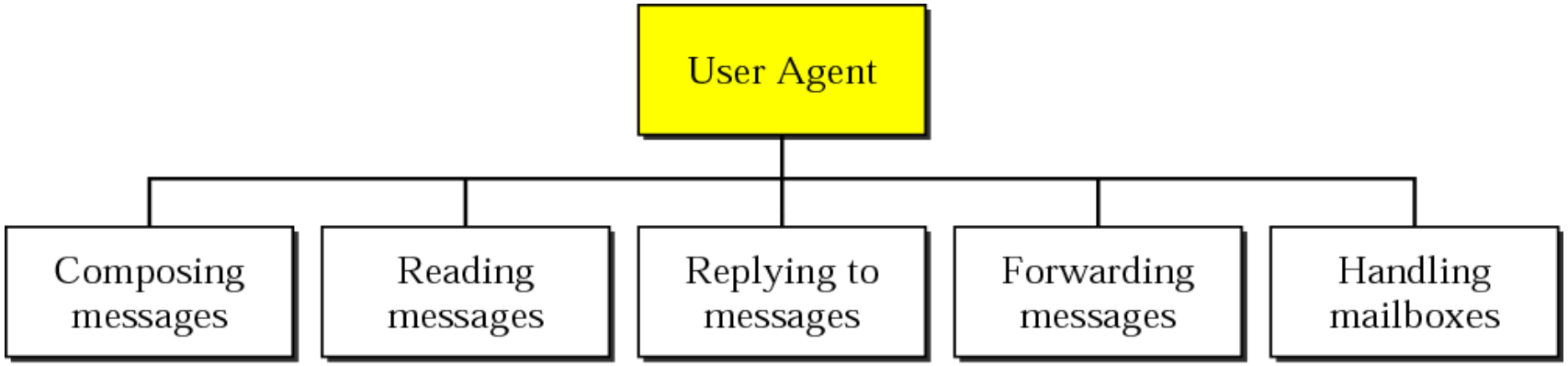


When both sender and receiver are connected to the mail server via a LAN or a WAN, we need two UAs, two pairs of MTAs (client and server), and a pair of MAAs (client and server). This is the most common situation today.

SMTP Mail Flow



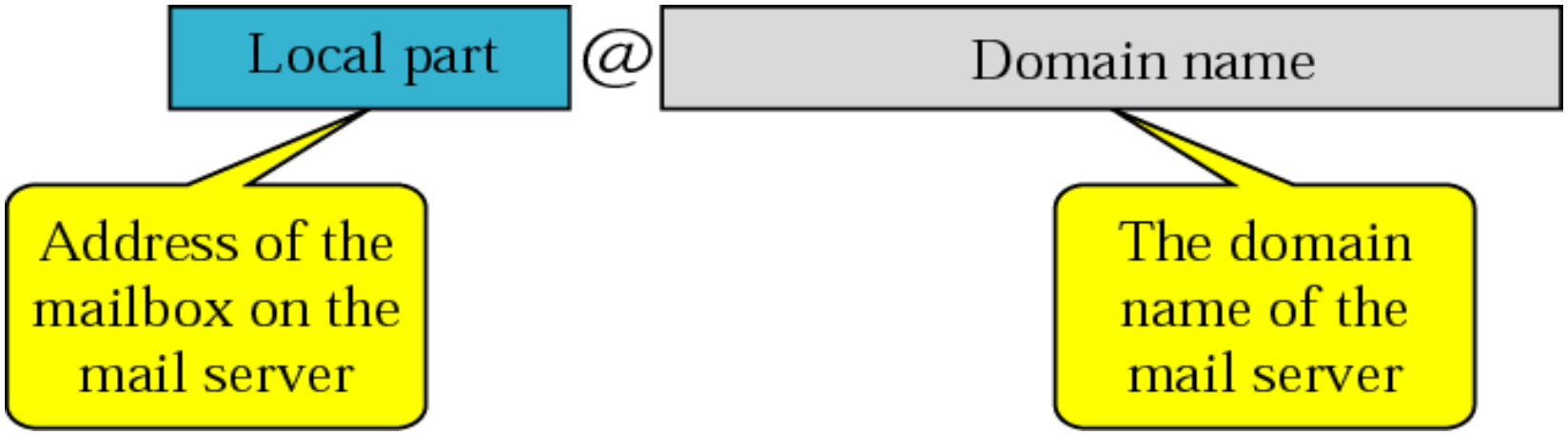
User agent



Some examples of command-driven user agents are mail, pine, and elm

Some examples of GUI-based user agents are Eudora, Outlook, and Netscape.

Email address



Relaying

- In SMTP terms, *relaying* means asking an SMTP sender to deliver an email on behalf of:
  - another SMTP server, or
  - an email client
- Relaying is quite dangerous since it is one of the main enablers of spam
  - sending SMTP servers should enable relaying only for local senders
    - Can be checked via domain name control
    - May require authentication

ESMTP and Authentication

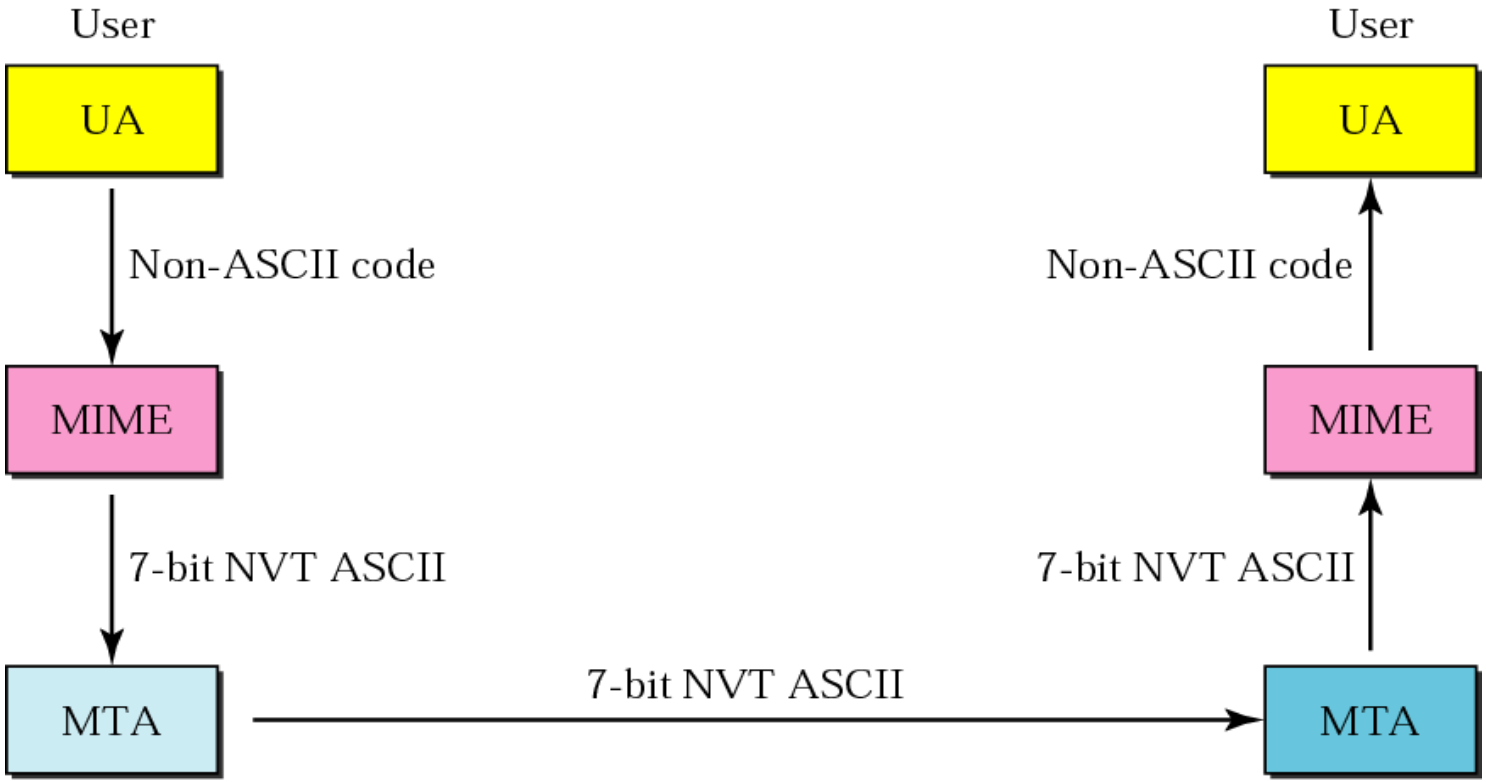
- SMTP Service Extensions
  - defined in some RFCs after RFC 821
    - EHLO (Extended HELO)
      - Server returns supported extensions and SMTP features
    - Some new parameters for existing SMTP commands
  - RFC 2821 published to cover core SMTP + extensions

- RFC 2554 added authentication feature to SMTP
  - AUTH command

Multipurpose Internet Mail Extension (MIME)

- Extension to RFC 822
- SMTP is only for 7-bit ASCII text messages, can not transmit executables
  - uuencode and other schemes are available
    - Not standardized
  - Can not transmit text including international characters (e.g. ö, ç, ğ, â, å, ä, è, é, ê, ë)
- MIME is intended to solve these problems
  - to be used over SMTP
  - compatible with RFC 822
- MIME is actually a framework to handle attachments

MIME


User
UA
Non-ASCII code
MIME
7-bit NVT ASCII
MTA
7-bit NVT ASCII
MTA
7-bit NVT ASCII
MIME
Non-ASCII code
UA
User

Overview of MIME

- New message header fields (to be included in RFC 822 header)
  - MIME version
  - Content type
    - description for the data (text, audio, video, image, etc..)
  - Content transfer encoding
    - Data should be encoded such that SMTP can carry
  - Content Description
    - plain text description for the object in the body
    - optional, used when needed

Content Types (some of them)

- Text body (unformatted plain text)
  - ASCII or ISO 8859 charset
  - a different charset may be defined at content-type header field
- Multipart
  - multiple independent parts, each may be of different type
  - separated by a *boundary* (a random-like string) for which value is defined at content-type header field
  - Four subtypes: Mixed, Parallel, Alternative, Digest
  - Multipart/mixed — order of different parts is important
  - Multipart/parallel — order is not important
  - Multipart/alternative — same content but alternative representations
- Message/RFC822
  - the content is an entire message (including header and body)
  - despite its name, the embedded message can be of any MIME type
  - what is the use of this content type?

Content Types (some of them)

- Image
  - jpeg, gif, etc.
- Video
  - mpeg
- Audio
- Application
  - binary data to be processed by an external application
    - attachments of any type
  - application name is a subpart
    - msword, postscript, etc.

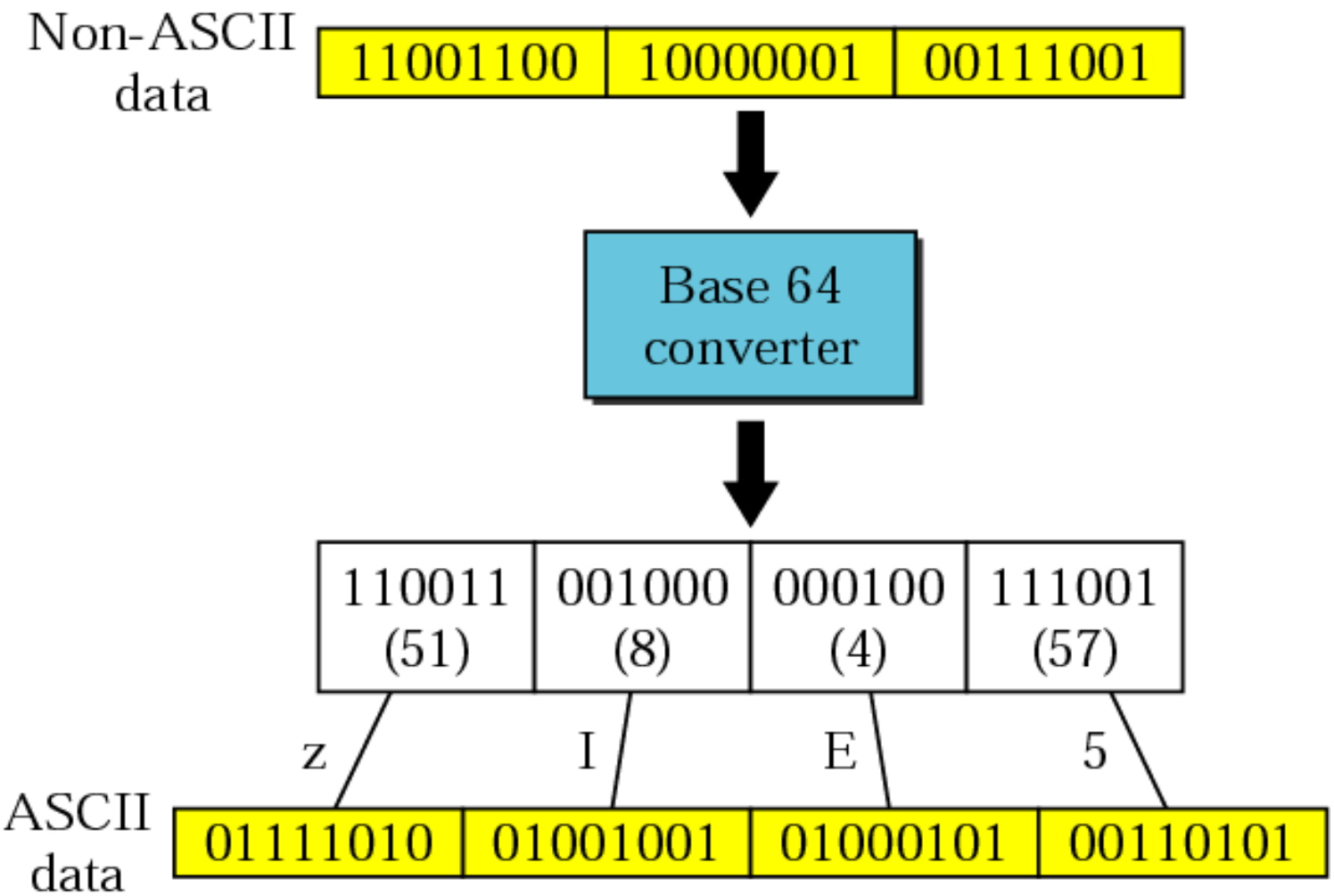
MIME Transfer Encoding

- Reliable delivery across various environments
- Content-transfer-encoding field
  - Six alternative methods
  - For three of them (7bit, 8bit, binary), no encoding done
    - Only 7-bit is safe for SMTP
- X-token
  - Named nonstandard encoding
  - vendor or application specific
- Quoted-printable
  - Useful when data are mostly printable ASCII characters
  - Non-printable characters represented by hex code
  - See the rules in the book
- Base64 (Radix-64)
  - Maps arbitrary binary input onto printable output (33% overhead)

Posta elettronica: SMTP [RFC 2821]

- usa TCP per trasferire in modo affidabile i messaggi di posta elettronica dal client al server, porta 25
- trasferimento diretto: il server trasmittente al server ricevente
- tre espressioni per il trasferimento
  - handshaking (saluto)
  - trasferimento di messaggi
  - chiusura
- interazione comando/risposta
  - comandi: testo ASCII
  - risposta: codice di stato ed espressione
- i messaggi devono essere nel formato ASCII a 7 bit

Base64


Non-ASCII data
11001100
10000001
00111001
Base 64 converter
110011 (51)
001000 (8)
000100 (4)
111001 (57)
z
I
E
5
ASCII data
01111010
01001001
01000101
00110101

Alfabeto Base64

Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
0	A	17	R	34	I	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	-
13	N	30	e	47	v		
14	O	31	f	48	w	(pad)	=
15	P	32	g	49	x		
16	Q	33	h	50	y		

UUencoder/UUdecoder

Codifica

```
Read 3 bytes: A,B,C
   b0=0x20 + (( A >> 2 ) & 0x3F)
   b1=0x20 + (((A << 4) | ((B >> 4) & 0xF)) & 0x3F)
   b2=0x20 + (((B << 2) | ((C >> 6) & 0x3)) & 0x3F)
   b3=0x20 + (( C ) & 0x3F)
Write b0,b1,b2,b3
```

Decodifica

```
Read 4 bytes: b0, b1, b2, b3
   A = (((b0 - 0x20) & 0x3F) << 2 & 0xFC) | (((b1 - 0x20) & 0x3F) >> 4 & 0x03);
   B = (((b1 - 0x20) & 0x3F) << 4 & 0xF0) | (((b2 - 0x20) & 0x3F) >> 2 & 0x0F);
   C = (((b2 - 0x20) & 0x3F) << 6 & 0xC0) | ((b3 - 0x20) & 0x3F);
Write A,B,C
```

Esempio di Codifica UUE

Carattere Originale	C								a								t							
ASCII decimale	67								97								116							
ASCII binario	0	1	0	0	0	0	1	1	0	1	1	0	0	0	0	1	0	1	1	1	0	1	0	0
Decimale	16								54				5				52							
+32	48								86				37				84							
Uuencoded	Q								2				F				0							

```
begin 644 cat.txt
#Q2F0
`

end
```

Un messaggio in Base64

$ base64 -E CERTIFICATE thawte-personal-freemail-2000.8.30.cer

-----BEGIN CERTIFICATE

MIIDKTCCApKgAwIBAgIBDDANBgkqhkiG9w0BAQQFADCB0TELMAkGA1UEBhMCWkEx
FTATBgNVBAgTDFdlc3Rlcm4gQ2FwZTESMBAGA1UEBxMJQ2FwZSBUb3duMRowGAYD
VQQKExFUaGF3dGUgQ29uc3VsdGluZzEoMCYGA1UECxMfQ2VydGlmaWNhdGlvbiBT
ZXJ2aWNlcyBEaXZpc2lvbjEkMCIGA1UEAxMbVGhhd3RlIFBlcnNvbmFsIEZyZWVt
YWlsIENBMSswKQYJKoZIhvcNAQkBFhxwZXJzb25hbC1mcmVlbWFpbEB0aGF3dGUu
Y29tMB4XDTAwMDgzMDAwMDAwMFoXDTAyMDgyOTIzNTk1OVowgZIxCzAJBgNVBAYT
AlpBMRUwEwYDVQQIEwxXZXN0ZXJuIENhcGUxEjAQBgNVBAcTCUNhcGUgVG93bjEP
MA0GA1UEChMGVGhhd3RlMR0wGwYDVQQLExRDZXJ0aWZpY2F0ZSBTZXJ2aWNlczEo
MCYGA1UEAxMfUGVyc29uYWwgRnJlZW1haWwgUlNBIDIwMDAuOC4zMDCBnzANBgkq
hkiG9w0BAQEFAAOBjQAwgYkCgYEA3jMypmPHCSVFPtJueCdngcXaiBmClw7jRCmK
YzUqbXA8+tyu9+50bzC8M5B/+TRxoKNtmPHDT6Jl2w36S/HW3WGl+YXNVZo1Gp2S
dagnrthy+boC9tewkd4c6avgGAOofENCUFGHgzzwObSbVIoTh/+zm51JZgAtCYns
lGvpoWkCAwEAAaNOMEwwKQYDVR0RBCIwIKQeMBwxGjAYBgNVBAMTEVByaXZhdGVM
YWJlbDEtMjk3MBIGA1UdEwEB/wQIMAYBAf8CAQAwCwYDVR0PBAQDAgEGMA0GCSqG
SIb3DQEBBAUAA4GBAHMbbyZli/8VNEtZYortRL5Jx+gNu4+5DWomKmKEH7iHY3Qc
bbfPGlORS+HN5jjZ7VD0Omw0kqzmkpxuwSMBwgmn70uuct0GZ/VQby5YuLYLwVBX
tewc1+8XttWIm7eiiBrtOVs5fTT8tpYYJU1q9J3Fw5EvqZa4BTxS/N3pYgNI

-----END CERTIFICATE-----

Scenario: Alice → Bob

1) Alice usa il suo agente utente per comporre il messaggio da inviare a **bob@someschool.edu**

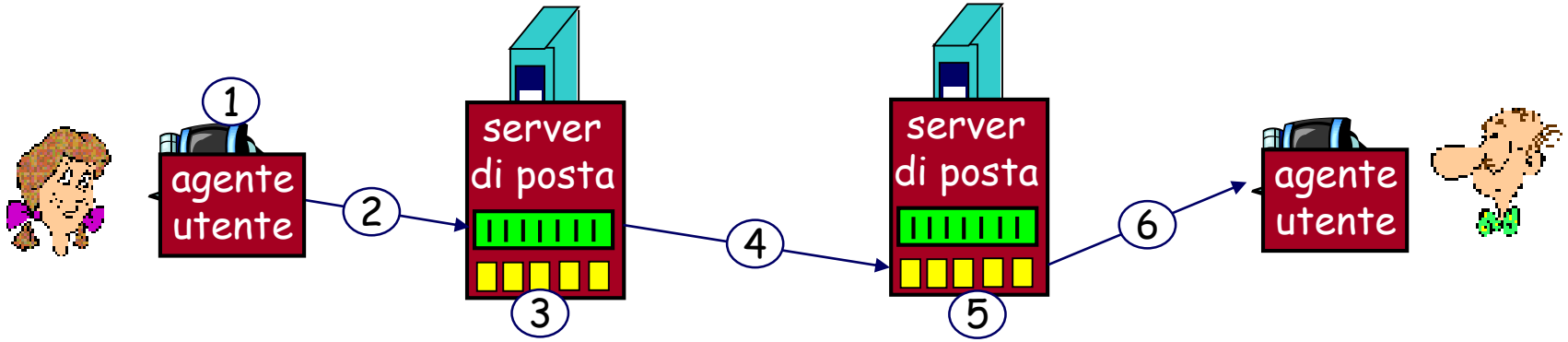
2) L'agente utente di Alice invia un messaggio al server di posta di Alice; il messaggio è posto nella coda di messaggi

3) Il lato client di SMTP apre una connessione TCP con il server di posta di Bob

4) Il client SMTP invia il messaggio di Alice sulla connessione TCP

5) Il server di posta di Bob pone il messaggio nella casella di posta di Bob

6) Bob invoca il suo agente utente per leggere il messaggio



Esempio di interazione 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
```

Provate un’interazione SMTP:

- telnet servername 25
  - Riceverete la risposta 220 dal server
- Digitate i comandi (nell’ordine)
  - HELO,
  - MAIL FROM,
  - RCPT TO,
  - DATA,
  - QUIT
- Questo vi consente di inviare messaggi di posta elettronica senza usare il client di posta (outlook)

SMTP: note finali

- SMTP usa connessioni persistenti
- SMTP richiede che il messaggio (intestazione e corpo) sia nel formato ASCII a 7 bit
- Il server SMTP usa `CRLF.CRLF` per determinare la fine del messaggio

Confronto con HTTP:

- HTTP: pull
- SMTP: push
- Entrambi hanno un'interazione comando/risposta in ASCII, codici di stato
- HTTP: ogni oggetto è incapsulato nel suo messaggio di risposta
- SMTP: più oggetti vengono trasmessi in un unico messaggio

```
nslookup -q=mx technolabs.it
```

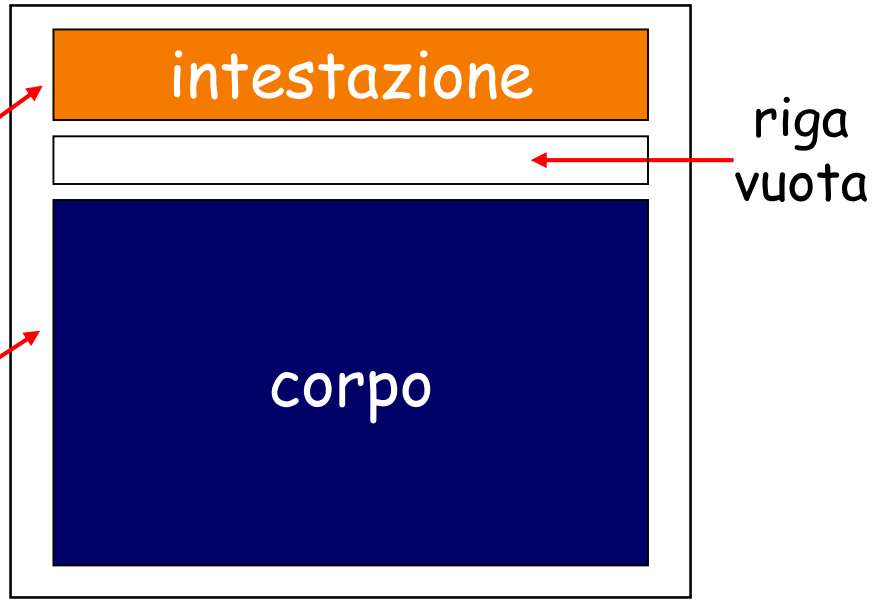
Formato dei messaggi di posta elettronica

SMTP: protocollo per scambiare messaggi di posta elettronica

RFC 822: standard per il formato dei messaggi di testo:

- Righe di intestazione, per esempio
  - To:
  - From:
  - Subject:

  differenti dai comandi SMTP !
- Corpo
  - il “messaggio”, soltanto caratteri ASCII



Formato del messaggio

- MIME: estensioni di messaggi di posta multimediali, RFC 2045, 2056
- Alcune righe aggiuntive nell'intestazione dei messaggi dichiarano il tipo di contenuto MIME

Versione MIME

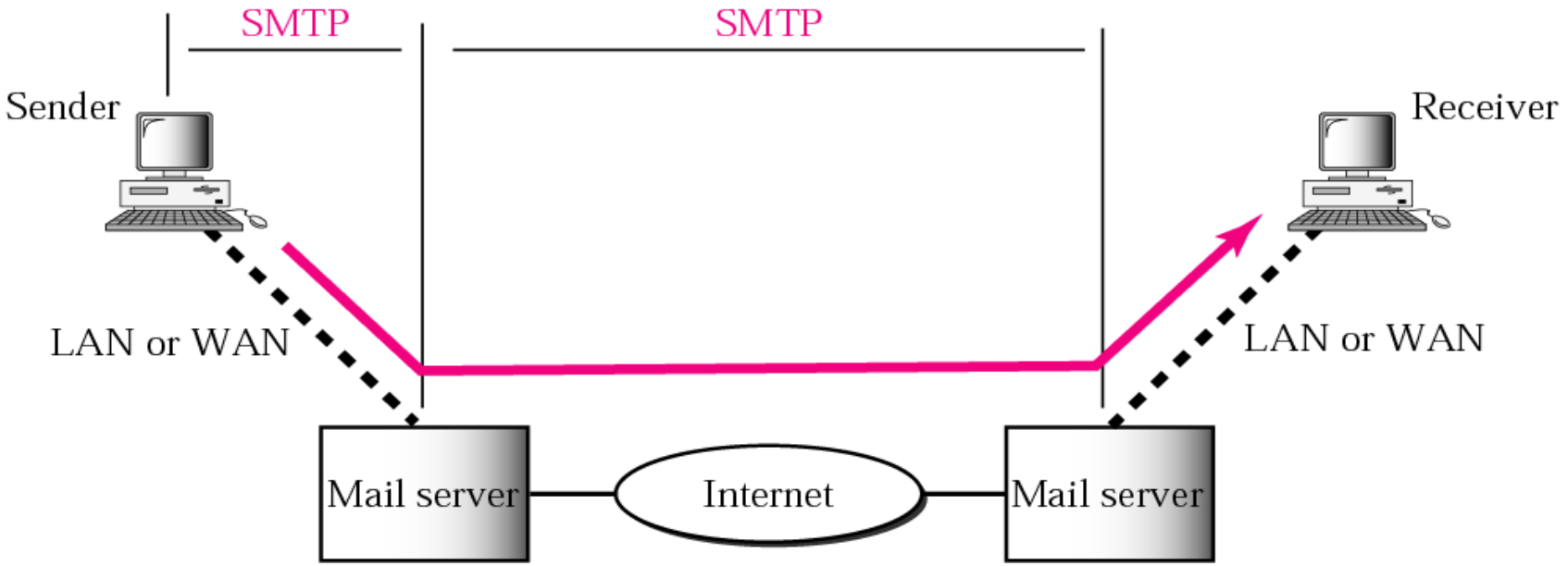
Metodo usato per codificare i dati

Tipo di dati multimediali

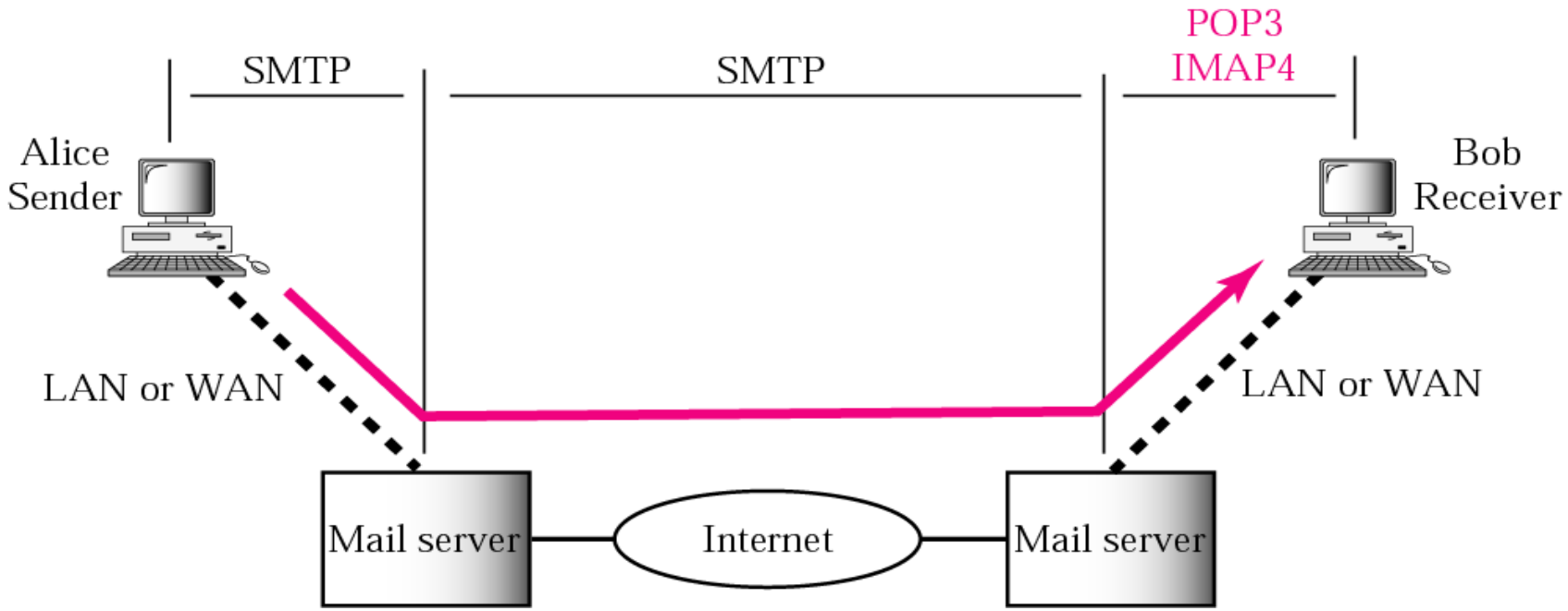
Dati codificati

```
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
```

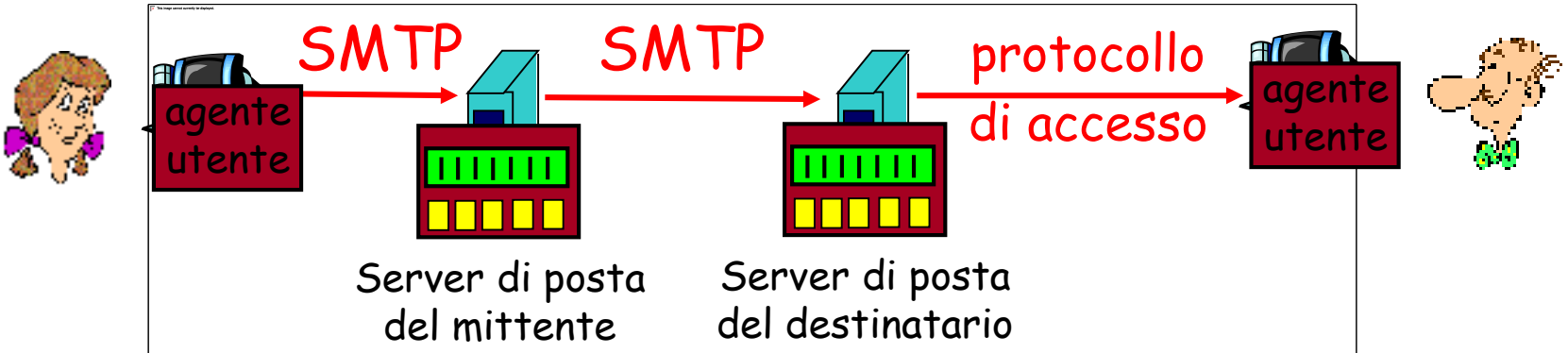
SMTP range


SMTP
SMTP
Sender
Receiver
LAN or WAN
LAN or WAN
Mail server
Internet
Mail server

POP3 and IMAP4



Protocolli di accesso alla posta



- SMTP: consegna/memorizzazione sul server del destinatario
- Protocollo di accesso alla posta: ottenere i messaggi dal server
  - POP: Post Office Protocol [RFC 1939]
    - autorizzazione (agente <--> server) e download
  - IMAP: Internet Mail Access Protocol [RFC 1730]
    - più funzioni (più complesse)
    - manipolazione di messaggi memorizzati sul server
  - HTTP WebMail: Hotmail , Yahoo! Mail, ecc.

La spedizione: l’SMTP

- Per inviare un messaggio si utilizza un programma in grado di utilizzare il protocollo SMTP.
- Quando il server SMTP riceve il messaggio, che include il mittente (autenticato) ed il destinatario inizia la ricerca del percorso su cui instradare il messaggio.
- Per far ciò analizza la parte relativa al dominio del destinatario (la stringa che segue il simbolo @) e ricava l’indirizzo IP. Instradando tramite i vari DNS il messaggio verso l’appropriato percorso, il messaggio arriva a destinazione.

La ricezione: il POP3

- Sul server del destinatario esistono varie caselle postali: una per ogni account attivo, create in seguito alla registrazione degli utenti del servizio di posta elettronica gestito dal dominio.
- Su questo server un software di gestione dei messaggi ricevuti smista la corrispondenza memorizzando in ciascuna casella i dati ricevuti e li mantiene fino a quando l'utente non si collega tramite un client di lettura di e-mail.
- I software client più diffusi sono Outlook express, Outlook, Eudora, FireBird e Messenger. Questi programmi, oltre ai parametri di spedizione, contengono anche le configurazioni per la ricezione dei messaggi tramite protocollo POP3 o Imap4.

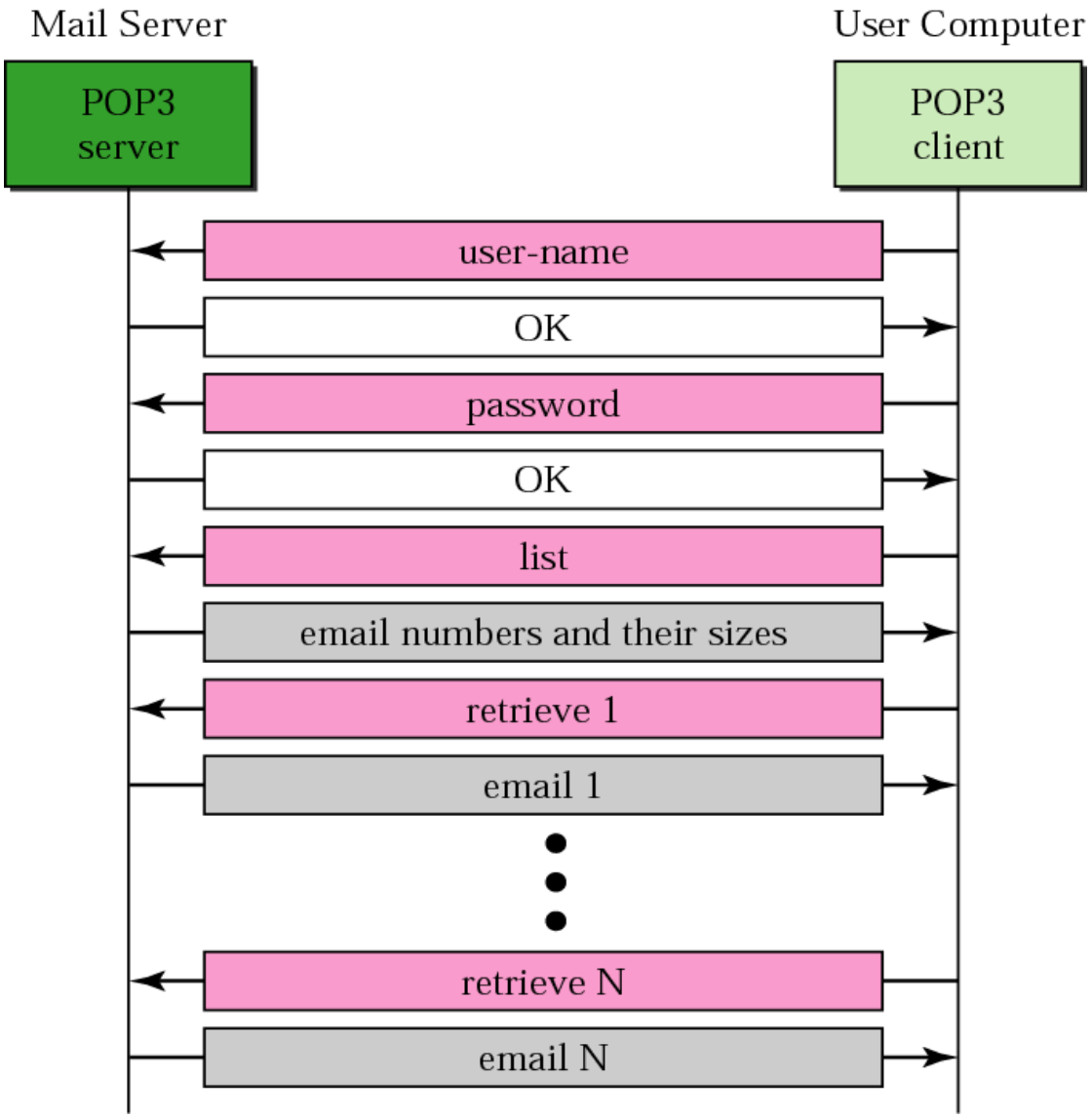
POP3 vs IMAP4

- La differenza fra i due è che nel primo caso, più diffuso, i messaggi vengono scaricati dal server al client, e anche se possono a richiesta restare memorizzati sul server, vengono poi comunque gestiti sulla macchina locale.
- Nel secondo caso, invece, i messaggi rimangono sempre sul server e vengono soltanto letti a distanza da un client autenticato.
  - Così facendo è possibile gestire la posta da più computer differenti: i messaggi saranno sempre in linea tutti insieme, a differenza del caso del POP3. Sarà però appesantita la situazione sul server, perché mantenendo sempre tutti i messaggi di tutti gli utenti e dovendo gestire le transazioni remote sarà sottoposto ad un carico di memorizzazione e di funzionamento maggiore.

Non solo testo: gli allegati

- Con la posta elettronica non viaggiano solo i testi.
  - Un’evoluzione naturale del sistema ha consentito di allegare dati di qualsiasi tipo ai messaggi.
- Utilizzando un codice di trasformazione dei dati in una serie di caratteri inviabili è possibile fondere nel corpo di una email anche immagini, file audio, documenti, fogli di calcolo, presentazioni, archivi compressi e tutto quanto sia memorizzabile in un computer.

POP3


Mail Server
User Computer
POP3 server
POP3 client
user-name
OK
password
OK
list
email numbers and their sizes
retrieve 1
email 1
retrieve N
email N

Protocollo POP3

Fase di autorizzazione

- Comandi del client:
  - **`user:`** dichiara nome dell'utente
  - **`pass:`** password
- Risposte del server
  - **`+OK`**
  - **`-ERR`**

```
S: +OK POP3 server ready
C: user bob
S: +OK
C: pass hungry
S: +OK user successfully logged on
```

Fase di transazione, client:

- **`list:`** elenca i numeri dei messaggi
- **`retr:`** ottiene i messaggi per numero
- **`dele:`** cancella
- **`quit`**

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

POP3 (altro) e IMAP

Ancora su POP3

- Il precedente esempio usa la modalità “scarica e cancella”
- Bob non può rileggere le e-mail se cambia client
- Modalità “scarica e mantieni”: copia i messaggi su più client
- POP3 è un protocollo senza stato tra le varie sessioni

IMAP

- Mantiene tutti i messaggi in un unico posto: il server
- Consente all’utente di organizzare i messaggi in cartelle
- IMAP conserva lo stato dell’utente tra le varie sessioni:
  - I nomi delle cartelle e l’associazione tra identificatori dei messaggi e nomi delle cartelle

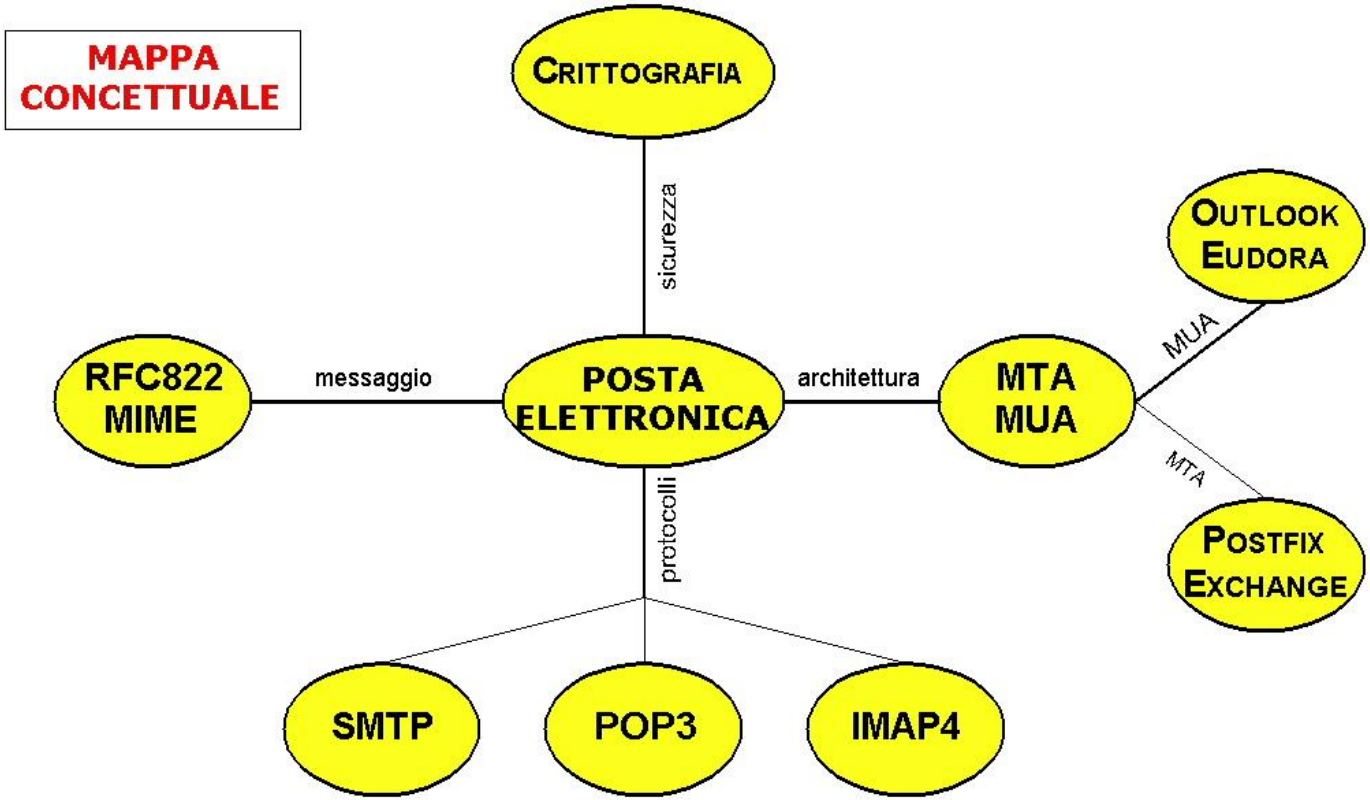
La sicurezza: crittografia e firma digitale

- Pur essendo uno strumento efficace e veloce non può essere considerata uno strumento assolutamente attendibile nei recapiti.
- Una e-mail non può cioè costituire prova legale di invio nè di autenticità del mittente o del contenuto, se non supportato da criteri di autenticazione, relativi ad aspetti delicatissimi di sicurezza legati allo sviluppo della crittografia o alla firma digitale.
- Solo allora una e-mail ha valore quanto la firma di un notaio.

Autenticazione e SMTP

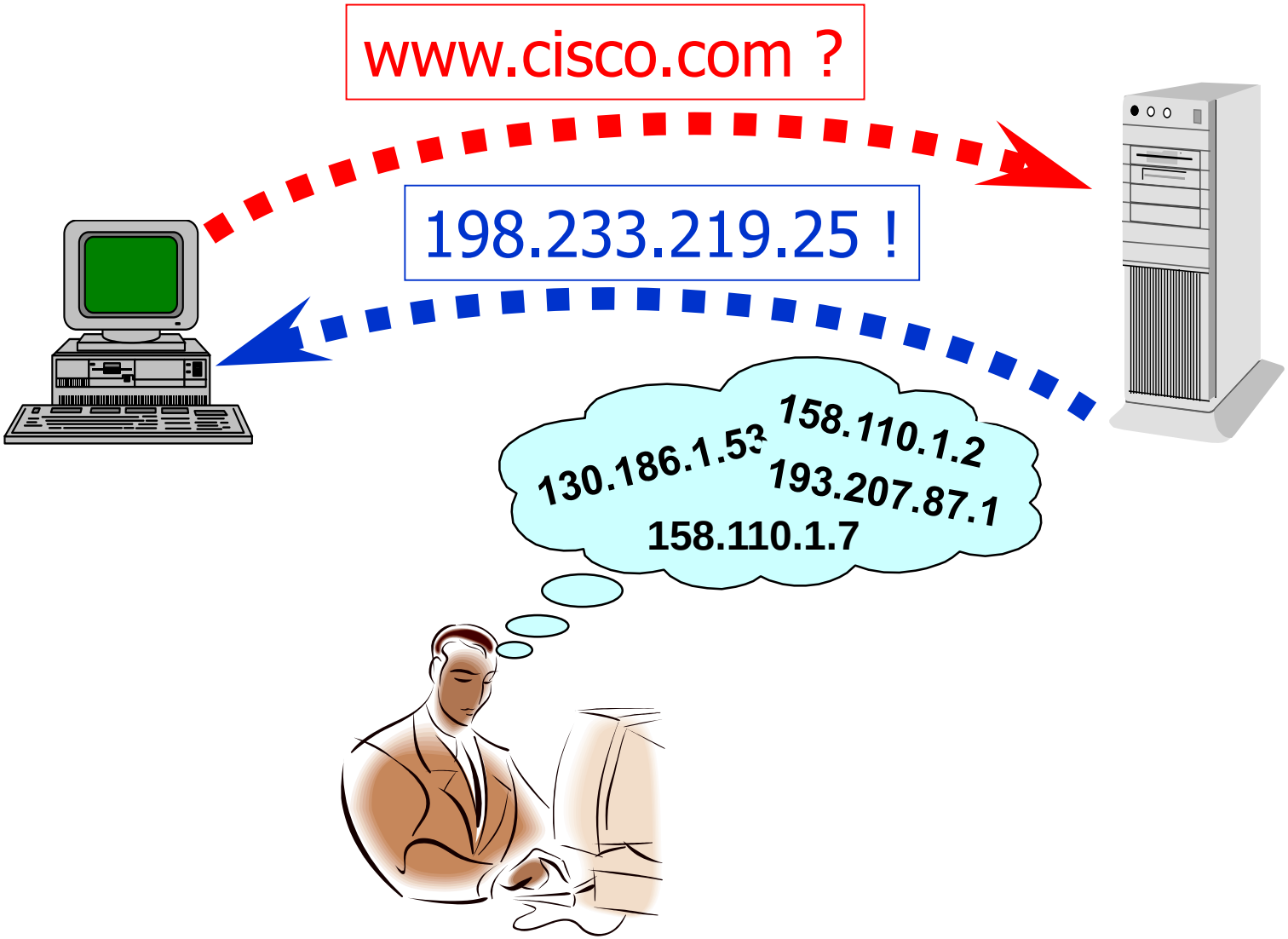
- Attualmente i server SMTP richiedono l’autenticazione dei client per evitare il fenomeno di spamming abilitando opportunamente la SMTP Authentication
- I server che non abilitano una tale restrizione vengono chiamati server open relay.

Mappa concettuale



Mail User Agent (MUA)
Mail Transport Agent (MTA)

Domain Name System (DNS)



DNS: le funzioni

- Ad ogni risorsa TCP/IP può essere assegnato un nome simbolico
- Sono necessari:
  - un metodo per associare il nome simbolico di una macchina all'indirizzo (o agli indirizzi) IP: risoluzione diretta
  - un metodo per associare ad un indirizzo IP il nome simbolico della macchina: risoluzione inversa
- Domain Name System (DNS)
  - definito presso ISI - USC 1984
  - RFC 882, RFC 883, RFC 973 (obsolete)
  - RFC 1034, RFC 1035, RFC 1123, RFC 1537, RFC 1912

Un pò di storia

- Ai tempi di ARPANET esisteva in ogni sistema operativo un unico file, hosts.txt, che elencava tutti gli host e i loro indirizzi IP. Ogni notte tutti gli host della rete lo copiavano dal sito in cui era manutenuto
- Quando la rete comprendeva solo qualche centinaio di grosse macchine questo approccio funzionava bene; quando la rete crebbe venne inventato il servizio DNS (23 giugno 1983) da Paul Mockapetris, Jon Postel e Craig Partrige; le specifiche originali sono descritte nello standard RFC 882.
- Nel 1987 vennero pubblicati commenti allo standard RFC del DNS, con i nomi RFC 1034 e RFC 1035, rendendo obsolete le specifiche precedenti.

Domain Name System (DNS)

- È un servizio utilizzato per la risoluzione di nomi di host in indirizzi IP e viceversa. Il servizio è realizzato tramite un database distribuito, costituito dai server DNS.
- Il nome DNS denota
  - il protocollo che regola il funzionamento del servizio,
  - i programmi che lo implementano,
  - i server su cui questi girano,
  - l'insieme di questi server che cooperano per fornire il servizio.
- L'operazione di convertire un nome in un indirizzo è detta **risoluzione** DNS
- Convertire un indirizzo IP in nome è detto **risoluzione inversa**.

DNS: Motivazioni ed utilizzi

- Gli esseri umani trovano più facile ricordare nomi testuali
- È possibile attribuire più nomi allo stesso indirizzo IP (o viceversa) per rappresentare diversi servizi o funzioni forniti da uno stesso host (o più host che erogano lo stesso servizio. Questa flessibilità risulta utile in molti casi:
  - Un utilizzo molto popolare di questa possibilità è il cosiddetto virtual hosting basato sui nomi, una tecnica per cui un web server dotato di una singola interfaccia di rete e di singolo indirizzo IP può ospitare più siti web, usando l'indirizzo alfanumerico trasmesso nell'header HTTP per identificare il sito per cui viene fatta la richiesta.
  - Utilizzando nomi diversi per riferirsi ai diversi servizi erogati da un host, è possibile spostare una parte dei servizi su un altro host, e spostare i client su questo nuovo host modificando il suo record nel DNS.
  - Facendo corrispondere più indirizzi IP a un nome, il carico dei client viene distribuito su diversi server, ottenendo un aumento delle prestazioni complessive del servizio e una tolleranza ai guasti (ma è necessario assicurarsi che i diversi server siano sempre allineati, ovvero offrano esattamente lo stesso servizio ai client).

DNS: caratteristiche principali

- database distribuito
- basato sul modello client/server
- tre componenti principali:
  - Resource Record : spazio dei nomi e informazioni associate
  - Nameserver : application server che mantiene i dati
  - Resolver : client per l'interrogazione del nameserver
- accesso veloce ai dati: database in memoria centrale e meccanismo di caching diffuso

Esempio

- Hosts

  `cheltenham.cs.princeton.edu → 192.12.69.17`

  `192.12.69.17 → 80:23:A8:33:5B:9F`

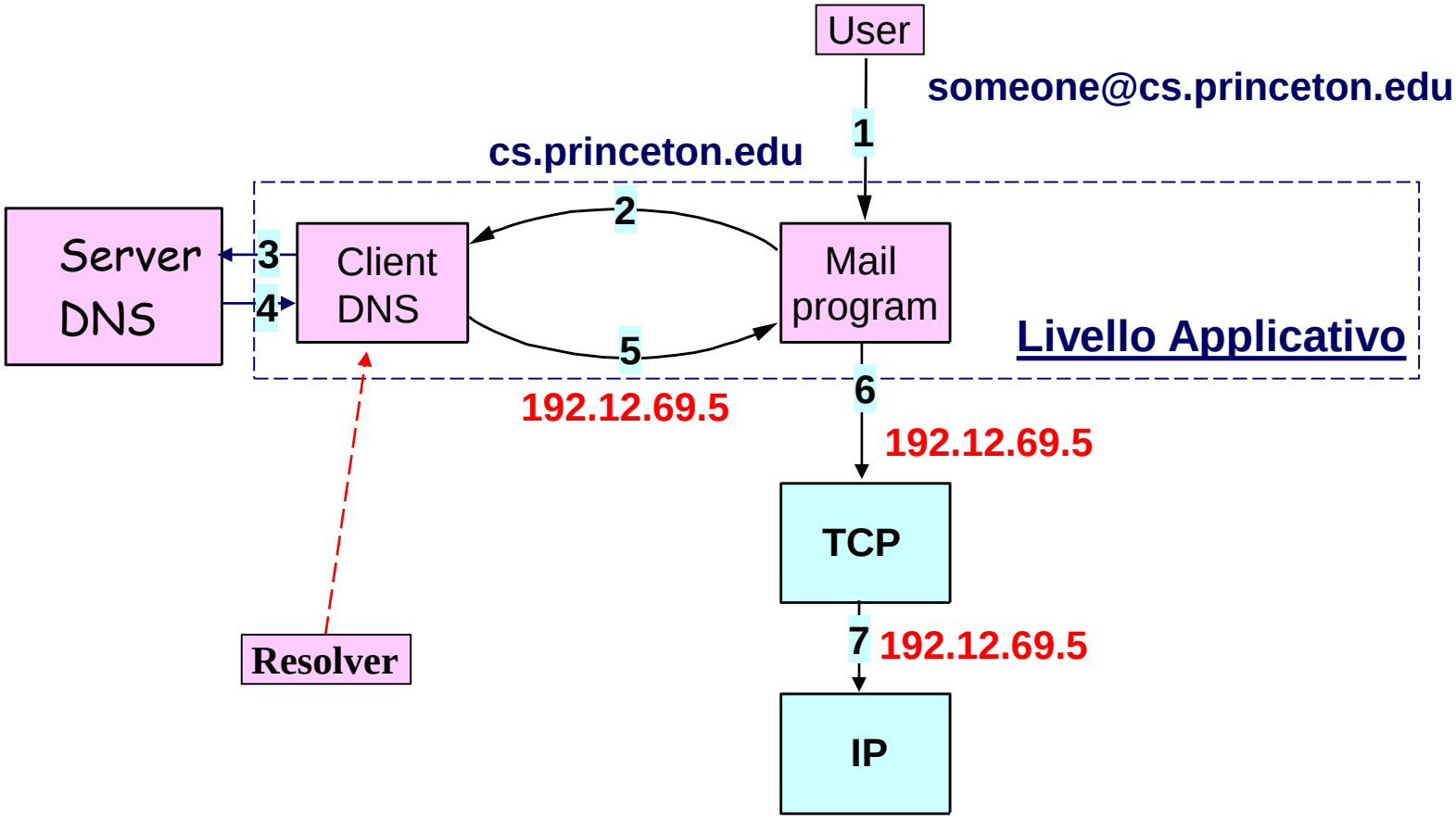
- Files

  `/usr/llp/tmp/foo → (server, fileid)`

- Users

  `Luigi Vetrano → luigi.vetrano@technolabs.it`

Esempio: Risoluzione di una email



DNS: Domain Name System

Persone: molti identificatori:
- nome, codice fiscale, carta d'identità

Host e router di Internet:
- indirizzo IP (32 bit) - usato per indirizzare i datagrammi
- "nome", ad esempio, www.yahoo.com – usato dagli esseri umani
- "MAC address", ad es. 00:11:22:33:44:55 – usato a livello DL

D: Come associare un indirizzo IP a un nome ?

Domain Name System:

- *Database distribuito* implementato in una gerarchia di *server DNS*
- *Protocollo a livello di applicazione* che consente agli host, ai router e ai server DNS di comunicare per *risolvere* i nomi (tradurre indirizzi/nomi)
  - nota: funzioni critiche di Internet implementate come protocollo a livello di applicazione
  - complessità nelle parti periferiche della rete

DNS

Servizi DNS

- Traduzione degli hostname in indirizzi IP
- Host aliasing
  - un host può avere più nomi
- Mail server aliasing
- Distribuzione locale
  - server web replicati: insieme di indirizzi IP per un nome canonico

Perché non centralizzare DNS?

- singolo punto di guasto
- volume di traffico
- database centralizzato distante
- manutenzione

Un database centralizzato su un singolo server DNS non è *scalabile* !

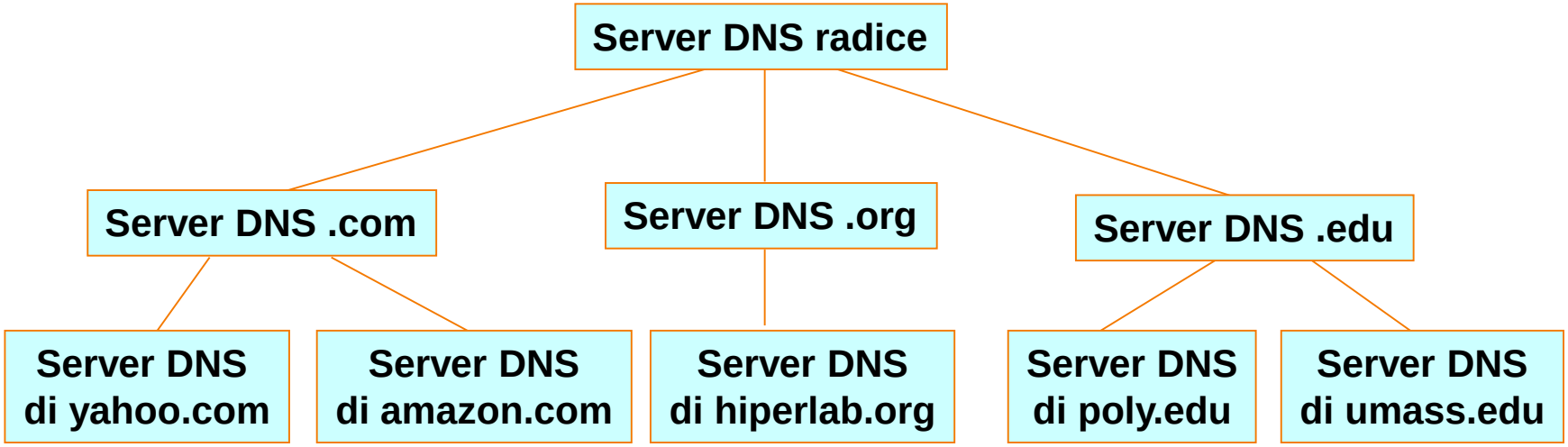
Nomi DNS

- Un nome o dominio è costituito da una serie di etichette separate da punti,
  - ad esempio it.wikipedia.org.
- A differenza degli indirizzi IP, dove la parte più importante del numero è la prima partendo da sinistra, in un nome DNS la parte più importante è la prima partendo da destra
  - Questa è detta dominio di primo livello (o TLD, Top Level Domain), per esempio .org o .it.
- Un dominio di secondo livello consiste in due parti, per esempio wikipedia.org, e così via. Ogni ulteriore elemento specifica una ulteriore suddivisione.
- Quando un dominio di secondo livello viene registrato all'assegnatario, questo è autorizzato a usare i nomi di dominio relativi ai successivi livelli come it.wikipedia.org (dominio di terzo livello) e altri come some.other.stuff.wikipedia.org (dominio di quinto livello) e così via.

FQDN (Fully Qualified Domain Name)

- Quando un nome di dominio termina con il carattere punto [.] il nome viene detto completo (FQDN)
  - ftp.univaq.it.
- Solo un nome completo può corrispondere a un indirizzo IP
- Un nome di dominio non completo è detto parziale (PQDN)
  - informatica
- I nomi parziali non corrispondono ad IP ma vengono utilizzati in relazione ad un dominio per essere trasformati in FQDN e quindi in IP
  - informatica + univaq.it = informatica.univaq.it. → 193.204.130.2

Database distribuiti e gerarchici

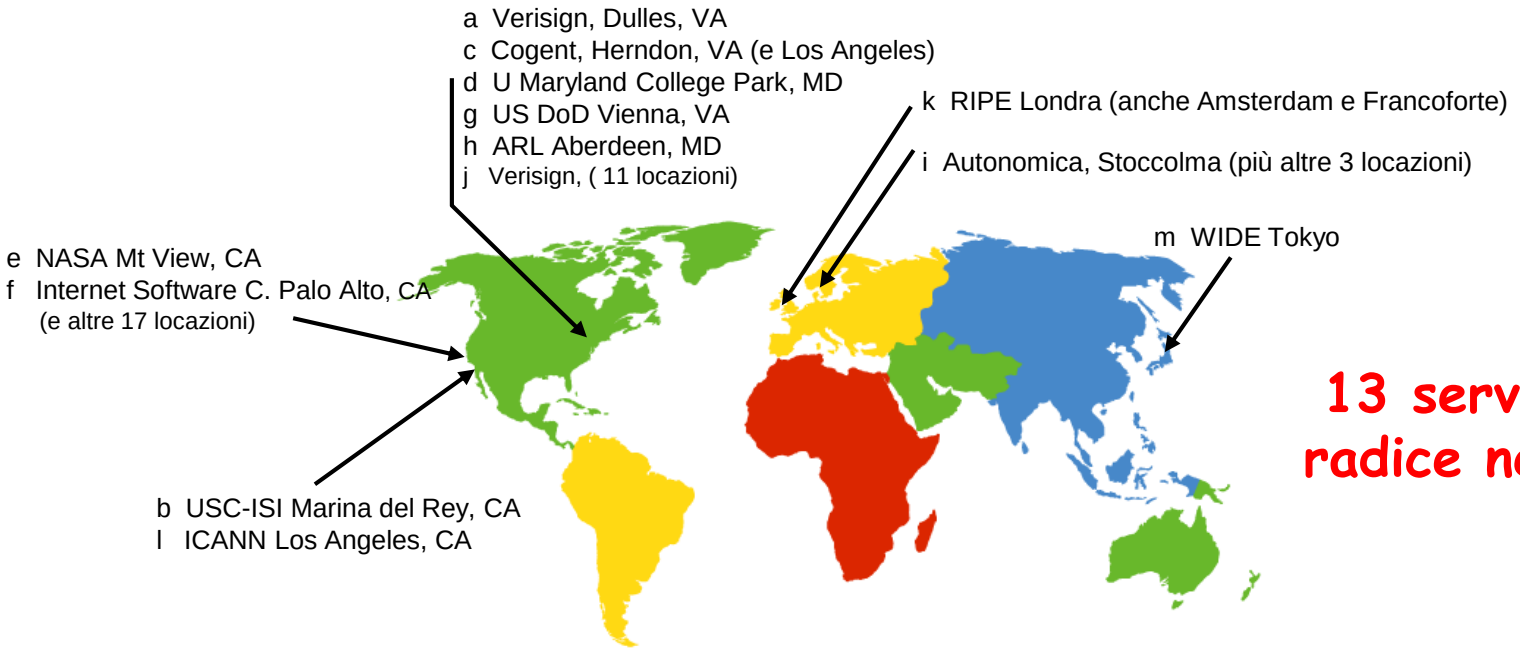


Il client vuole l'IP di www.amazon.com; 1ª approssimazione:

- Il client interroga il server radice per trovare il server DNS **com**
- Il client interroga il server DNS **com** per ottenere il server DNS amazon.com
- Il client interroga il server DNS **amazon.com** per ottenere l'indirizzo IP di www.amazon.com

DNS: server DNS radice

- Contattato da un server DNS locale che non può tradurre il nome
- server DNS radice:
  - contatta un server DNS autorizzato se non conosce la mappatura
  - ottiene la mappatura
  - restituisce la mappatura al server DNS locale



13 server DNS radice nel mondo

13 root server in tutto il mondo

- 10 negli Stati Uniti, 2 in Europa (Londra e Stoccolma) ed uno a Tokio.
- Ecco la lista dei server, con relativo indirizzo IP:

1. A.ROOT-SERVERS.NET. IP=198.41.0.4
2. B.ROOT-SERVERS.NET. IP=128.9.0.107
3. C.ROOT-SERVERS.NET. IP=192.33.4.12
4. D.ROOT-SERVERS.NET. IP=128.8.10.90
5. E.ROOT-SERVERS.NET. IP=192.203.230.10
6. F.ROOT-SERVERS.NET. IP=192.5.5.241
7. G.ROOT-SERVERS.NET. IP=192.112.36.4
8. H.ROOT-SERVERS.NET. IP=128.63.2.53
9. I.ROOT-SERVERS.NET. IP=192.36.148.17
10. J.ROOT-SERVERS.NET. IP=198.41.0.10
11. K.ROOT-SERVERS.NET. IP=193.0.14.129
12. L.ROOT-SERVERS.NET. IP=198.32.64.12
13. M.ROOT-SERVERS.NET. IP=202.12.27.33

Server TLD e server di competenza

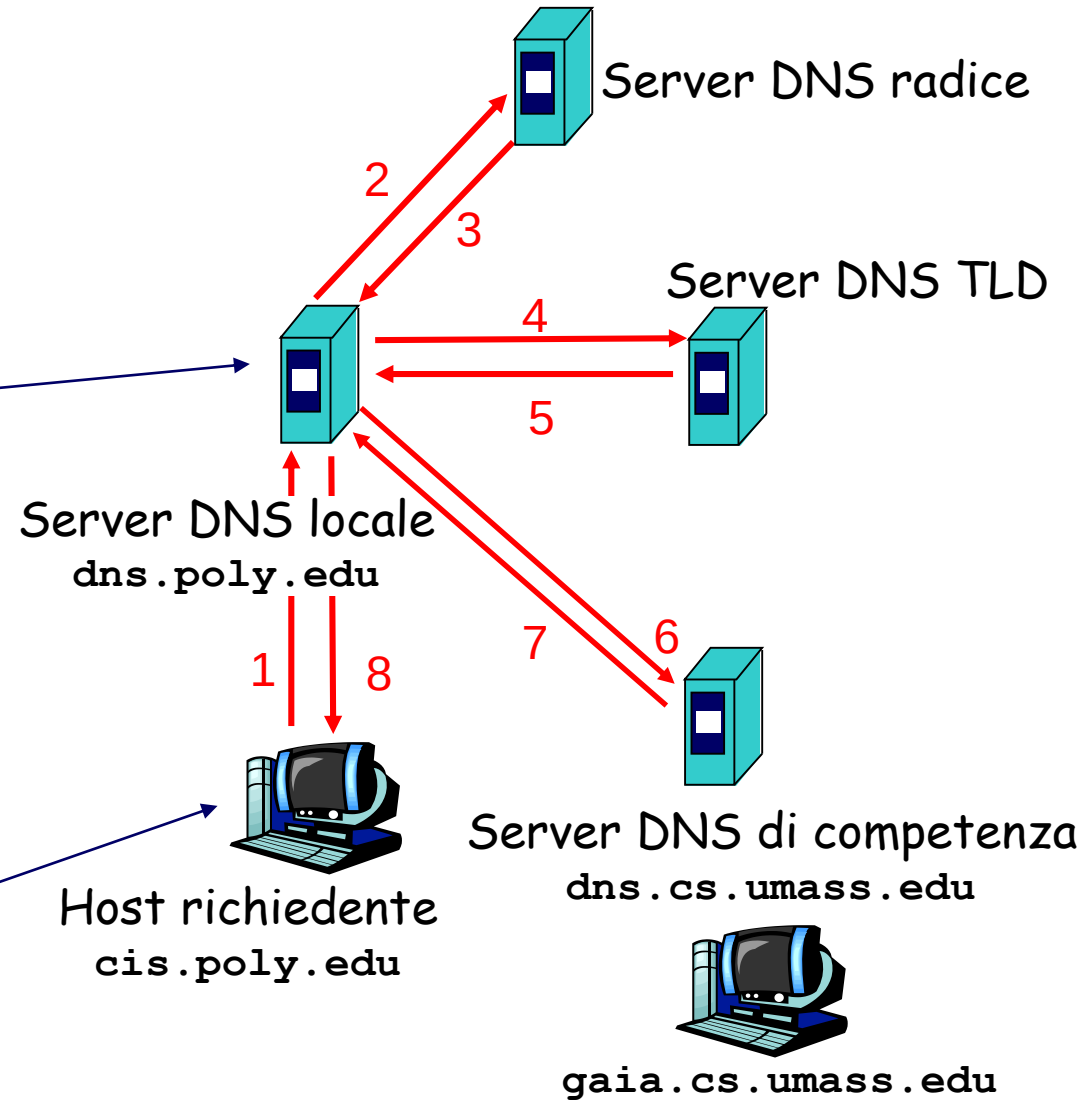
- Server TLD (top-level domain): si occupano dei domini com, org, net, edu, ecc. e di tutti i domini locali di alto livello, quali uk, fr, ca, jp e it.
  - Network Solutions gestisce i server TLD per il dominio com
  - Educause gestisce quelli per il dominio edu
- Server di competenza (authoritative server): ogni organizzazione dotata di host Internet pubblicamente accessibili (quali i server web e i server di posta) deve fornire i record DNS di pubblico dominio che mappano i nomi di tali host in indirizzi IP.
  - possono essere manutenuti dall'organizzazione o dal service provider

Server DNS locale

- Non appartiene strettamente alla gerarchia dei server
- Ciascun ISP (università, società, ISP residenziale) ha un server DNS locale.
  - detto anche "**default name server**"
- Quando un host effettua una richiesta DNS, la query viene inviata al suo server DNS locale
  - il server DNS locale opera da proxy e inoltra la query in una gerarchia di server DNS

Esempio

- L'host cis.poly.edu vuole l'indirizzo IP di gaia.cs.umass.edu



Query iterativa:

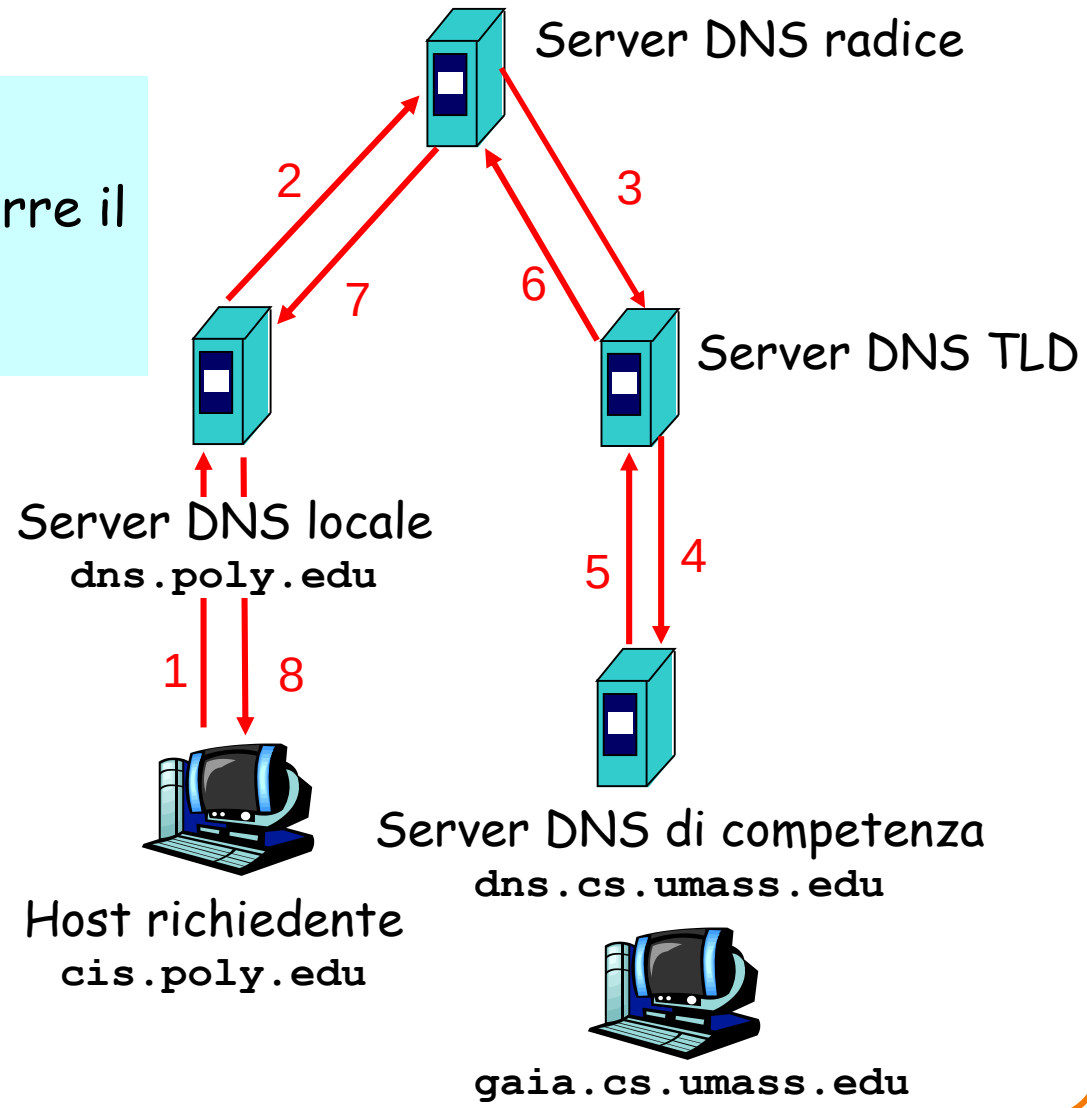
- Il server contattato risponde con il nome del server da contattare
- "Non conosco questo nome, ma chiedi a questo server"

Query ricorsiva:

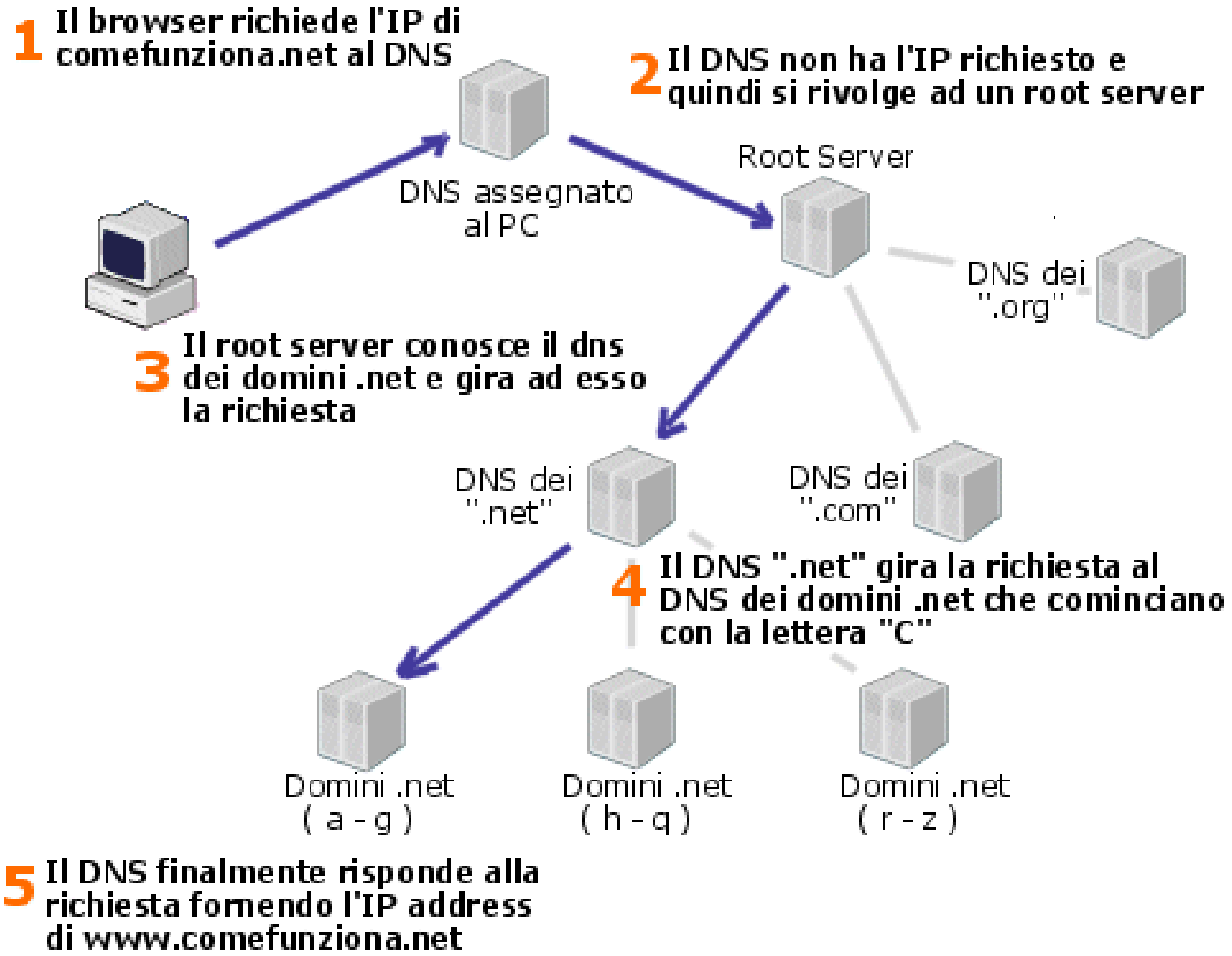
Query ricorsive

Query ricorsiva:

Affida il compito di tradurre il nome al server DNS contattato



Esempio DNS


1 Il browser richiede l'IP di comefunziona.net al DNS
2 Il DNS non ha l'IP richiesto e quindi si rivolge ad un root server
DNS assegnato al PC
Root Server
DNS dei ".org"
3 Il root server conosce il dns dei domini .net e gira ad esso la richiesta
DNS dei ".net"
DNS dei ".com"
4 Il DNS ".net" gira la richiesta al DNS dei domini .net che cominciano con la lettera "C"
Domini .net (a - g)
Domini .net (h - q)
Domini .net (r - z)
5 Il DNS finalmente risponde alla richiesta fornendo l'IP address di www.comefunziona.net

Risoluzione & Load balancing

Dominio o IP: www.google.com

Risolvi

Nome canonico: www.l.google.com

Alias: www.google.com

Indirizzi: 173.194.35.147
173.194.35.148
173.194.35.144
173.194.35.145
173.194.35.146

Name server locale

- I resolver sono configurati con l'indirizzo IP di un Name Server locale (solitamente sulla stessa rete).
- Le richieste fatte dal resolver al NS locale sono ricorsive: egli si aspetta che il NS locale gli esaudisca la richiesta
- Il NS locale invece fa delle richieste iterative per cui si fa carico di contattare vari NS fino ad ottenere la risposta

DNS consigliati

- **Google DNS**
  - **Server DNS preferito:** 8.8.8.8
  - **Server DNS alternativo:** 8.8.4.4
- **OpenDNS**
  - **Server DNS preferito:** 208.67.222.222
  - **Server DNS alternativo:** 208.67.220.220

DNS: caching e aggiornamento dei record

- Una volta che un server DNS impara la mappatura, la mette nella *cache*
  - le informazioni nella cache vengono invalidate (spariscono) dopo un certo periodo di tempo (TTL)
  - tipicamente un server DNS locale memorizza nella cache gli indirizzi IP dei server TLD
    - quindi i server DNS radice non vengono visitati spesso
- I meccanismi di aggiornamento/notifica sono progettati da IETF
  - RFC 2136
  - http://www.ietf.org/html.charters/dnsind-charter.html

Record DNS

DNS: database distribuito che memorizza i record di risorsa (RR)

Formato RR: **(name, value, type, ttl)**

- Type=A
  - **name** è il nome dell'host
  - **value** è l'indirizzo IP
- Type=NS
  - **name** è il dominio (ad esempio foo.com)
  - **value** è il nome dell'host del server di competenza di questo dominio
- Type=CNAME
  - **name** è il nome alias di qualche nome "canonico" (nome vero) `www.ibm.com` è in realtà `servereast.backup2.ibm.com`
  - **value** è il nome canonico
- Type=MX
  - **value** è il nome del server di posta associato a **name**

Type

Type	*Mnemonic*	*Description*
1	A	Address. A 32-bit IPv4 address. It is used to convert a domain name to an IPv4 address.
2	NS	Name server. It identifies the authoritative servers for a zone.
5	CNAME	Canonical name. It defines an alias for the official name of a host.
6	SOA	Start of authority. It marks the beginning of a zone. It is usually the first record in a zone file.
11	WKS	Well-known services. It defines the network services that a host provides.
12	PTR	Pointer. It is used to convert an IP address to a domain name.
13	HINFO	Host information. It gives the description of the hardware and the operating system used by a host.
15	MX	Mail exchange. It redirects mail to a mail server.
28	AAAA	Address. An IPv6 address
252	AXFR	A request for the transfer of the entire zone.
255	ANY	A request for all records.

Esempio di record

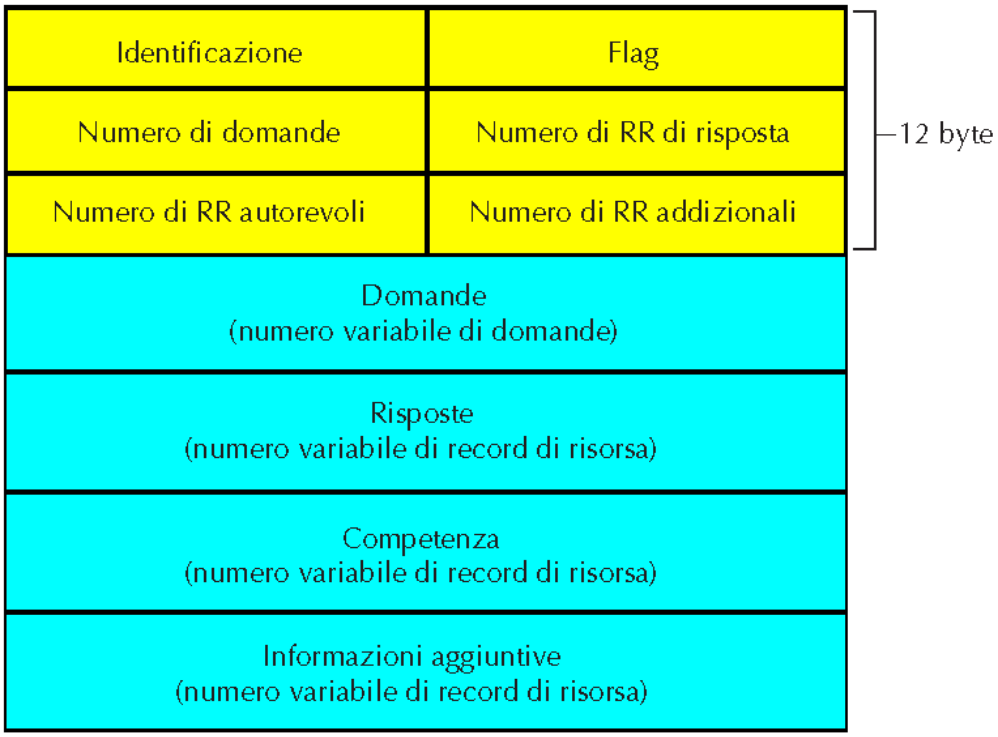
```
$TTL    43200
@               IN      SOA     ns.mesys.it.    hostmaster.mesys.it.
(
                        2002053101 ; serial
                        86400 ; refresh
                        3600 ; retry
                        604800 ; expire
                        86400 ; default_ttl
                        )
@               IN      MX      5     mail.mesys.it.
@               IN      NS      ns.mesys.it.
@               IN      NS      dns2.nic.it.
localhost       IN      A       127.0.0.1
ns              IN      A       151.4.83.2
ns1             IN      A       151.4.83.3
mail            IN      A       151.4.83.2
www             IN      CNAME   turtle.mesys.it.
ftp             IN      CNAME   dolphin.mesys.it.
```

Messaggi DNS

Protocollo DNS: domande (query) e messaggi di *risposta*, entrambi con lo stesso *formato*

Intestazione del messagggio

- Identificazione: numero di 16 bit per la domanda; la risposta alla domanda usa lo stesso numero
- Flag:
  - domanda o risposta
  - richiesta di ricorsione
  - ricorsione disponibile
  - risposta di competenza



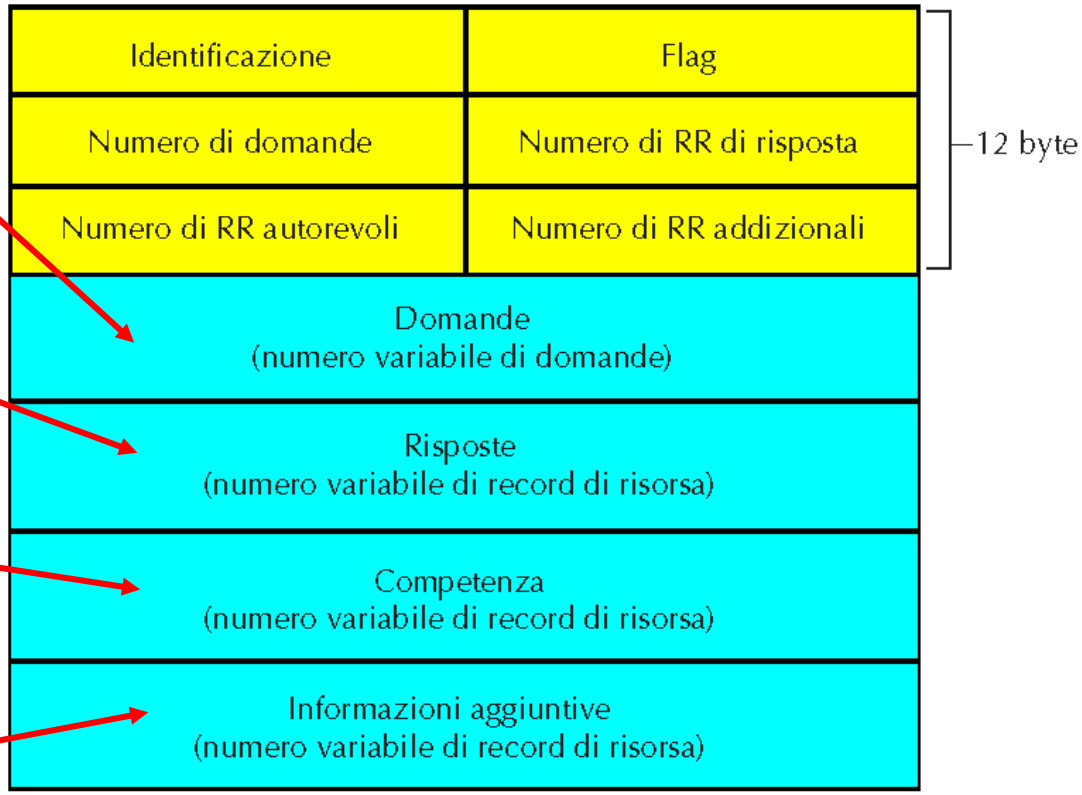
Messaggi DNS

Campi per
il nome richiesto
e il tipo di domanda

RR nella risposta
alla domanda

Record per
i server di competenza

Informazioni extra che
possono essere usate



Inserire record nel database DNS

- Esempio: abbiamo appena avviato la nuova società "Network Utopia"
- Registriamo il nome networkuptopia.com presso registrar (ad esempio, Network Solutions)
  - Forniamo a registrar i nomi e gli indirizzi IP dei server DNS di competenza (primario e secondario)
  - Registrar inserisce due RR nel server TLD com:

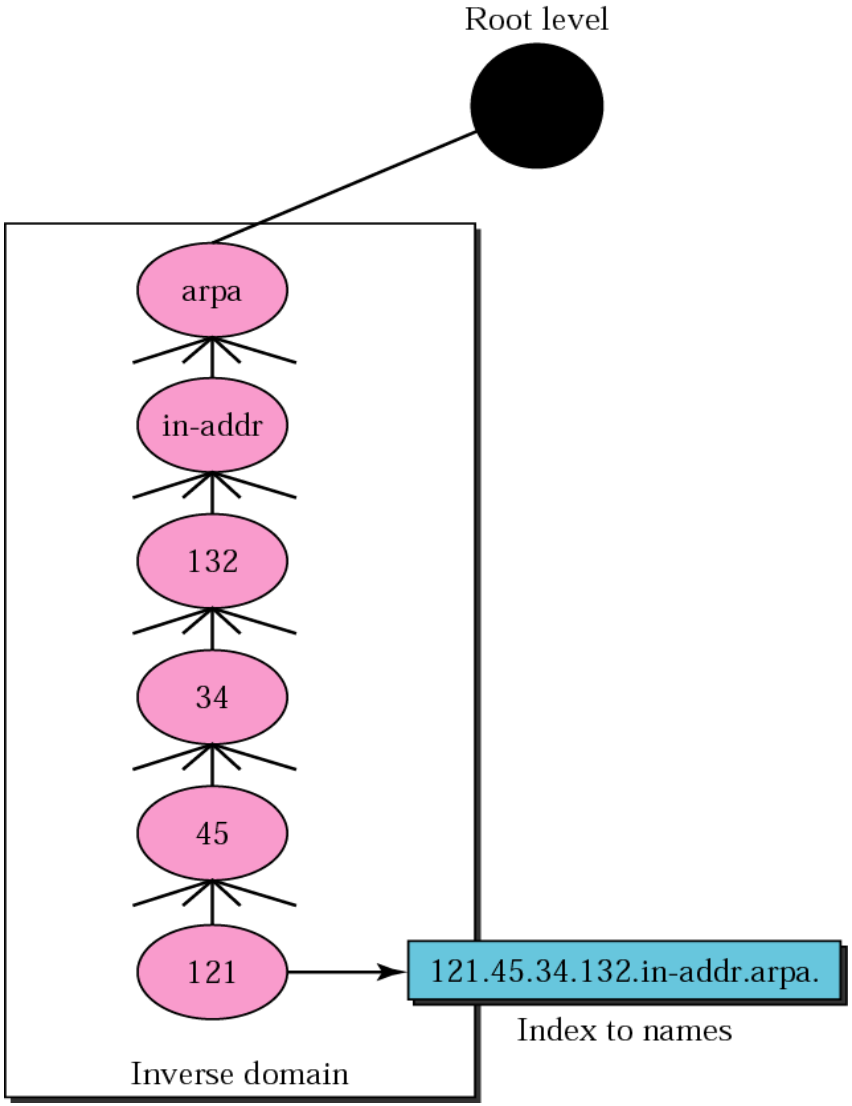
    (networkutopia.com, dns1.networkutopia.com, NS)
    (dns1.networkutopia.com, 212.212.212.1, A)

- Inseriamo nel server di competenza un record tipo A per www.networkuptopia.com e un record tipo MX per networkutopia.com

Risoluzione inversa

- Risalire al nome del dominio noto l'IP !
- L'indirizzo IP, es. 13.14.15.16, viene trasformato in un nome di dominio:
  - 16.15.14.13.in-addr.arpa
- E poi viene effettuata una normale risoluzione di questo nome di dominio.
- Questa risoluzione anziché fornire un IP fornisce un nome di dominio che è quello relativo all'IP di partenza.

DNS inverso


Root level
arpa
in-addr
132
34
45
121
121.45.34.132.in-addr.arpa.
Index to names
Inverse domain

Implementazione

- Il protocollo DNS è implementato da diversi software:
  - BIND (Berkeley Internet Name Domain), il nome del più comune demone DNS usato sui sistemi Unix.
  - DJBDNS (Dan J Bernstein's DNS implementation)
  - MaraDNS
  - NSD (Name Server Daemon)
  - PowerDNS
- Il DNS utilizza il protocollo di trasporto UDP e la porta 53 per soddisfare le richieste di risoluzione provenienti dagli host.
- I server DNS effettuano gli zone transfer usando il protocollo di trasporto TCP e la porta 53. Questa modalità viene usata anche quando una query ha una risposta molto lunga.
- “DNS uses UDP as the transport protocol when the size of the response message is less than 512 bytes. If the size of the response message is more than 512 bytes, a TCP connection is used”.

Virtual Hosts

```
<VirtualHost *:80>
ServerName www.romolo.it
DocumentRoot D:/xampp/htdocs/romolo
</VirtualHost>
```

Virtual Hosts

```
<VirtualHost *:80>
ServerName www.pippo.it
DocumentRoot D:/xampp/htdocs/pippo
</VirtualHost>
```