

*Corso di*

*SISTEMI TELEMATICI*

*a.a. 2009-2010*

**Il livello di applicazione**

Prof. Floriano De Rango

# 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

# Livello di applicazione

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

# Principi delle applicazioni di rete (1)

- ❑ Sebbene le applicazioni siano diverse e possano avere molti componenti interattivi, quasi sempre il software ne costituisce il nucleo centrale.
- ❑ Ad esempio per il Web ci sono due parti di software che comunicano fra loro: il software del browser nell'host dell'utente e il software del server Web nel server stesso.
- ❑ Nel gergo dei sistemi operativi non sono pezzi di software (cioè, programmi) a comunicare ma i **processi**. Un processo può essere pensato come un programma che sta girando all'interno di un terminale.
- ❑ I processi su due diversi terminali comunicano fra loro scambiandosi **messaggi** attraverso la rete di calcolatori.
- ❑ I protocolli dello strato di applicazione definiscono non solo il formato e l'ordine dei messaggi scambiati fra i processi ma anche le azioni da intraprendere alla trasmissione o ricezione di tali messaggi.

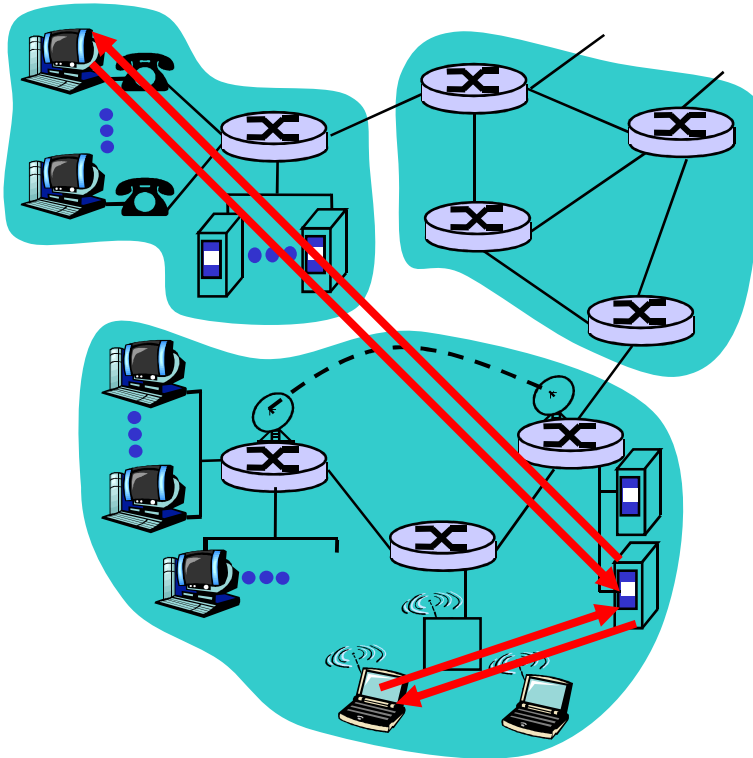
# Principi delle applicazioni di rete (2)

- ❑ E' importante fare una distinzione fra **applicazioni di rete** e **protocolli dello strato di applicazione**.
- ❑ Un protocollo dello strato di applicazione è una parte di un'applicazione di rete.
- ❑ Il Web, ad esempio, è un'applicazione di rete che permette agli utenti di ottenere a richiesta "documenti" dai server Web.
- ❑ L'applicazione Web è costituita da molti componenti:
  - ❖ gli standard per i formati dei documenti (HTML)
  - ❖ i browser del Web (es. Microsoft Internet Explorer)
  - ❖ i server del Web (es. i server Apache, Microsoft...)
  - ❖ un protocollo dello strato di applicazione.
- ❑ Il protocollo dello strato di applicazione Web, HTTP (HyperText Transfer Protocol [RFC 2616]) , definisce come i messaggi vengono passati fra browser e server Web.

# Architetture delle applicazioni di rete

- Quando si progetta una applicazione di rete bisogna decidere sull'architettura dell'applicazione.
- Essa definisce come l'applicazione deve essere 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 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

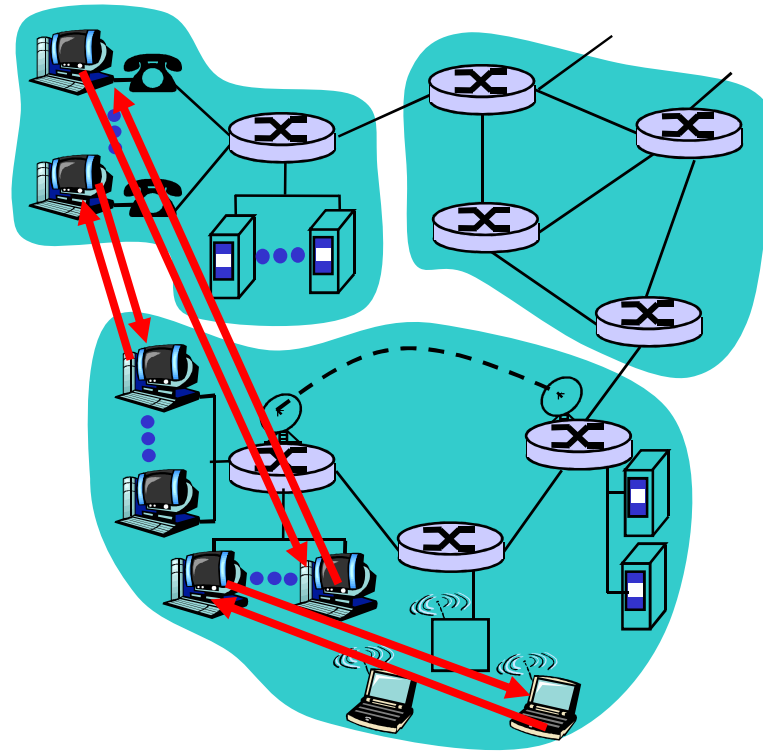


# Architettura P2P pura

- 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**:
  - 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 comunicanti

**Processo:** programma in esecuzione su di un host.

- All'interno dello stesso host, due processi comunicano utilizzando **schemi interprocesso** (definiti dal SO).
- Processi su host differenti comunicano attraverso lo scambio di **messaggi**.

**Processo client:** processo che dà inizio alla comunicazione

**Processo server :** processo che attende di essere contattato

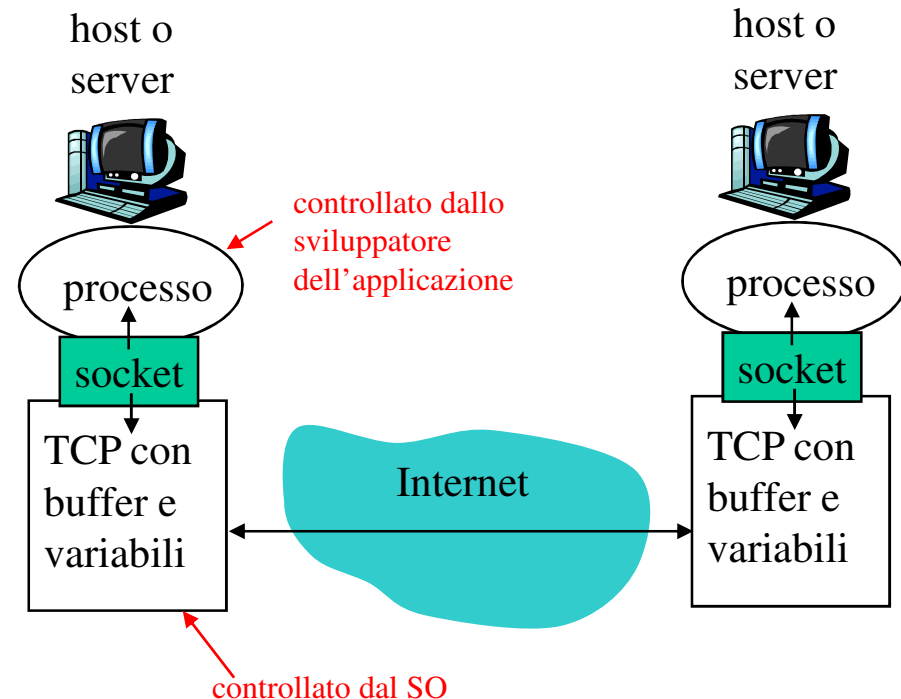
- Nota: le applicazioni con architetture P2P hanno processi client e processi server.
- Il peer che scarica è il client e quello che mette a disposizione il file è il server.

# Socket

Ogni messaggio generato da un processo deve essere inoltrato al livello di rete sottostante.

- Un processo invia/riceve messaggi al/dal suo **socket**.
- Un socket è analogo a una porta e il processo è la casa:
  - ❖ un processo che vuole inviare un messaggio, lo fa uscire dalla propria "porta" (socket)
  - ❖ il processo presuppone l'esistenza di un'infrastruttura esterna che trasporterà il messaggio attraverso la rete fino alla "porta" del processo di destinazione.

- I socket sono le API tra l'applicazione e la rete.



# Processi di 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.  
(si ricordi che l'indirizzo IP individua l'interfaccia di rete)
- ❑ Non è sufficiente conoscere l'indirizzo IP dell'host su cui è in esecuzione il processo per identificare il processo stesso poiché sullo stesso host possono essere in esecuzione molti processi.
- ❑ Per identificare un processo è necessario sia l'indirizzo IP dell'host su cui il processo è in esecuzione che il **numero di porta** associato all'applicazione.
- ❑ 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 (Internet Assigned Numbers Authority).
- ❑ Quando si desidera creare una nuova applicazione, bisogna assegnarle un nuovo numero di porta.

# Protocollo di livello applicazione

I processi comunicano inviando messaggi attraverso i socket; ma come sono fatti questi messaggi?

Un protocollo applicativo definisce:

- ❑ i **tipi** di messaggi scambiati, ad esempio messaggi di richiesta e di risposta
- ❑ la **sintassi** dei tipi di messaggio: quali sono i campi nel messaggio e come sono descritti
- ❑ la **semantica** dei campi, ovvero il significato delle informazioni nei campi
- ❑ le **regole** per determinare quando e come un processo invia e risponde ai messaggi.

## Protocolli di pubblico dominio:

- ❑ definiti nelle RFC
- ❑ sono una parte dell'applicazione; ad esempio:
  - ❑ HTTP [RFC 2616] → Web
  - ❑ SMTP [RFC 2821] → e-mail

## Protocolli proprietari:

- ❑ Il P2P fa uso essenzialmente di protocolli proprietari, ad esempio,
  - ❑ KaZaA, WinMx, Gnutella

# Quali servizi di trasporto richiede un'applicazione?

## Perdita di dati

- ❑ Alcune applicazioni (ad esempio, audio) possono tollerare qualche perdita.
- ❑ Altre applicazioni (ad esempio, trasferimento di file, telnet) richiedono un trasferimento dati affidabile al 100%.

## Temporizzazione

- ❑ Alcune applicazioni (ad esempio, telefonia Internet, giochi interattivi) per essere "realistiche" richiedono piccoli ritardi.

## Ampiezza di banda

- ❑ Alcune applicazioni (ad esempio, quelle multimediali) per essere "efficaci" richiedono un'ampiezza di banda minima.
- ❑ Altre applicazioni ("le applicazioni elastiche") utilizzano l'ampiezza di banda che è disponibile al momento della richiesta.

## 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 la rete è sovraccarica
- ❑ *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

## 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	proprietario (es. RealNetworks)	TCP o UDP
Internet telephony	proprietario (es. Dialpad, Skype)	UDP

# Livello di applicazione

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

# Web e HTTP

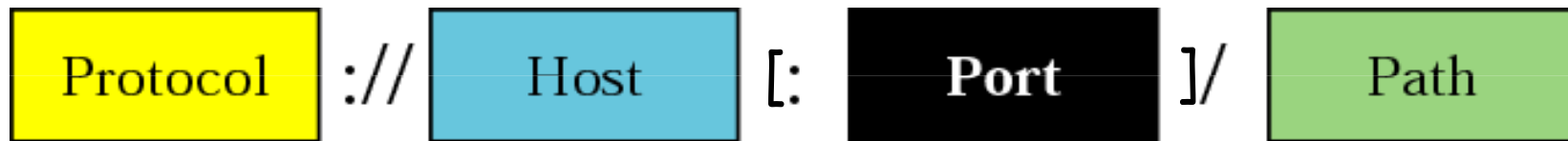
## Terminologia

- ❑ Una **pagina web** è costituita da **oggetti**.
- ❑ Un oggetto può essere un file HTML, un'immagine JPEG, un'applet Java, un file audio, o altro.
- ❑ 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 (Uniform Resource Locator):

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

  
protocollo                      nome dell'host                      nome del percorso

# URL

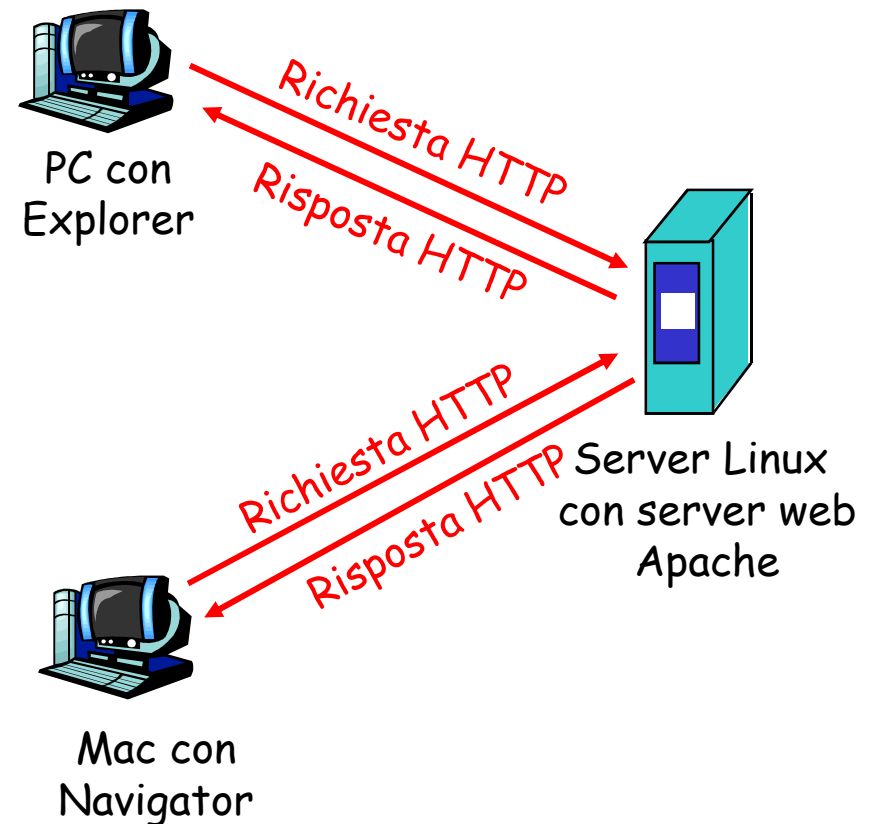


- L'URL specifica:
  - ❖ il metodo di accesso (protocollo e porta)
  - ❖ l'host su quale si trova l'oggetto ( IP o nome host)
  - ❖ il path dell'oggetto

# Panoramica su HTTP (1)

## HTTP: HyperText Transfer Protocol

- Protocollo di livello di applicazione del Web
- Modello client/server
  - ❖ *client*: il browser (user agent) richiede al server di "visualizzare" 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 (2)

## Utilizza come protocollo di trasporto il TCP:

- ❑ il client inizializza la connessione TCP (crea un socket) con il server sulla porta 80
- ❑ il server accetta la connessione TCP dal client
- ❑ vengono scambiati messaggi HTTP fra browser (client HTTP) e server web (server HTTP)
- ❑ viene chiusa la connessione TCP.

## 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 (1)

Supponiamo che l'utente immetta l'URL

`www.someSchool.edu/someDepartment/index.html`

(contiene testo,  
riferimenti a 10  
immagini jpeg)

## Client

## Server

1. Il client HTTP inizializza una connessione TCP con il server HTTP (processo) `www.someSchool.edu` sulla porta 80

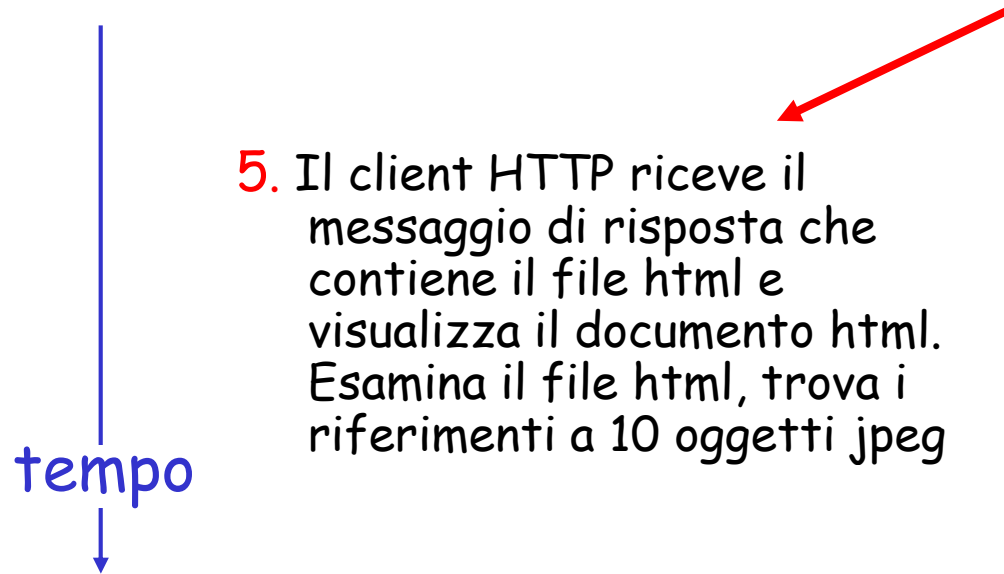
2. Il server HTTP all'host `www.someSchool.edu` in attesa di una connessione TCP alla porta 80 "accetta" la connessione e avvisa il client

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

4. Il server HTTP riceve il messaggio di richiesta, forma il *messaggio di risposta* che contiene l'oggetto richiesto e invia il messaggio nel suo socket e chiede al TCP di chiudere la connessione.

tempo  
↓

## Connessioni non persistenti (2)



Questa procedura viene ripetuta per ognuno dei 10 oggetti jpeg!

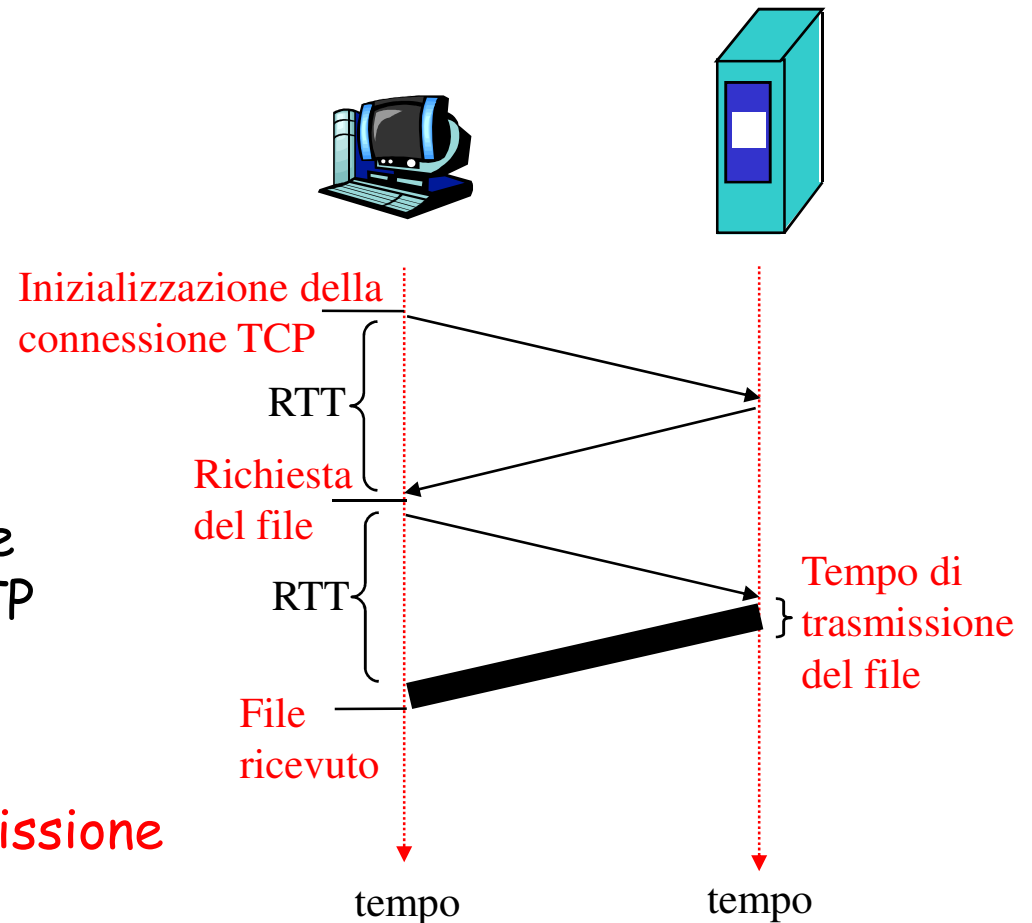
# Schema del tempo di risposta

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

## Tempo di risposta:

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

**totale =  $2RTT$  + tempo di trasmissione**



# Connessioni persistenti

## 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.

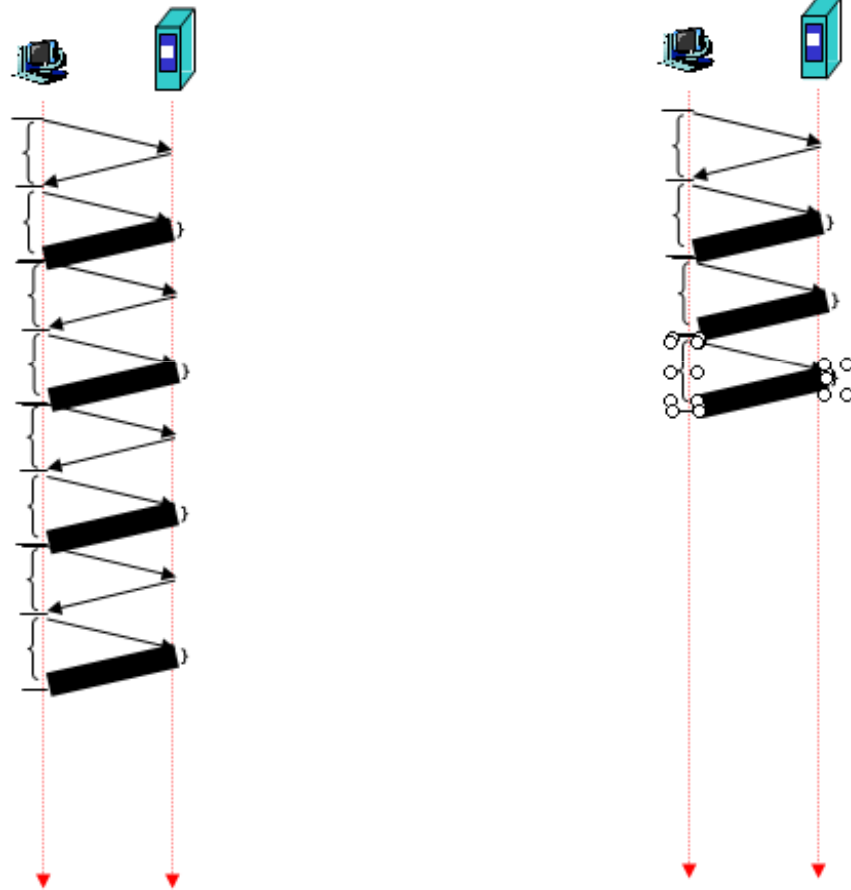
### Connessione persistente *senza* pipelining:

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

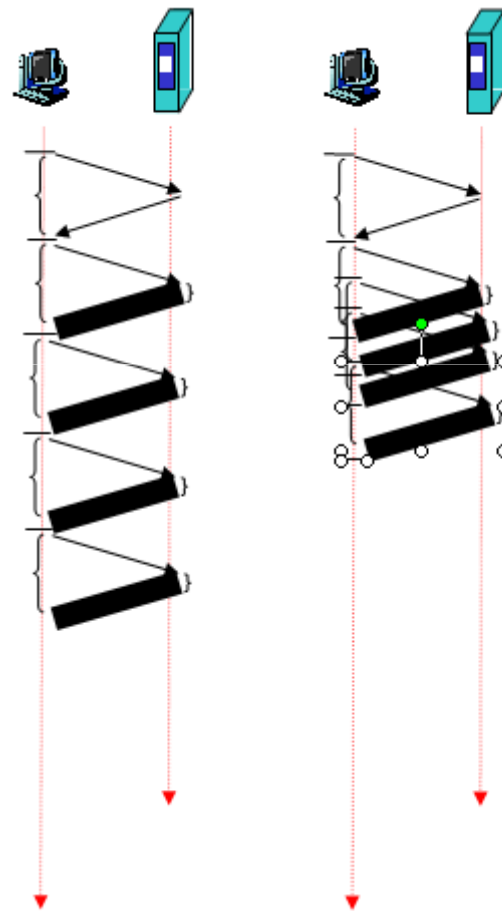
### Connessione persistente *con* pipelining:

- ❑ 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 vs persistente



# Persistente: non pipelining vs pipelining



# Messaggi HTTP

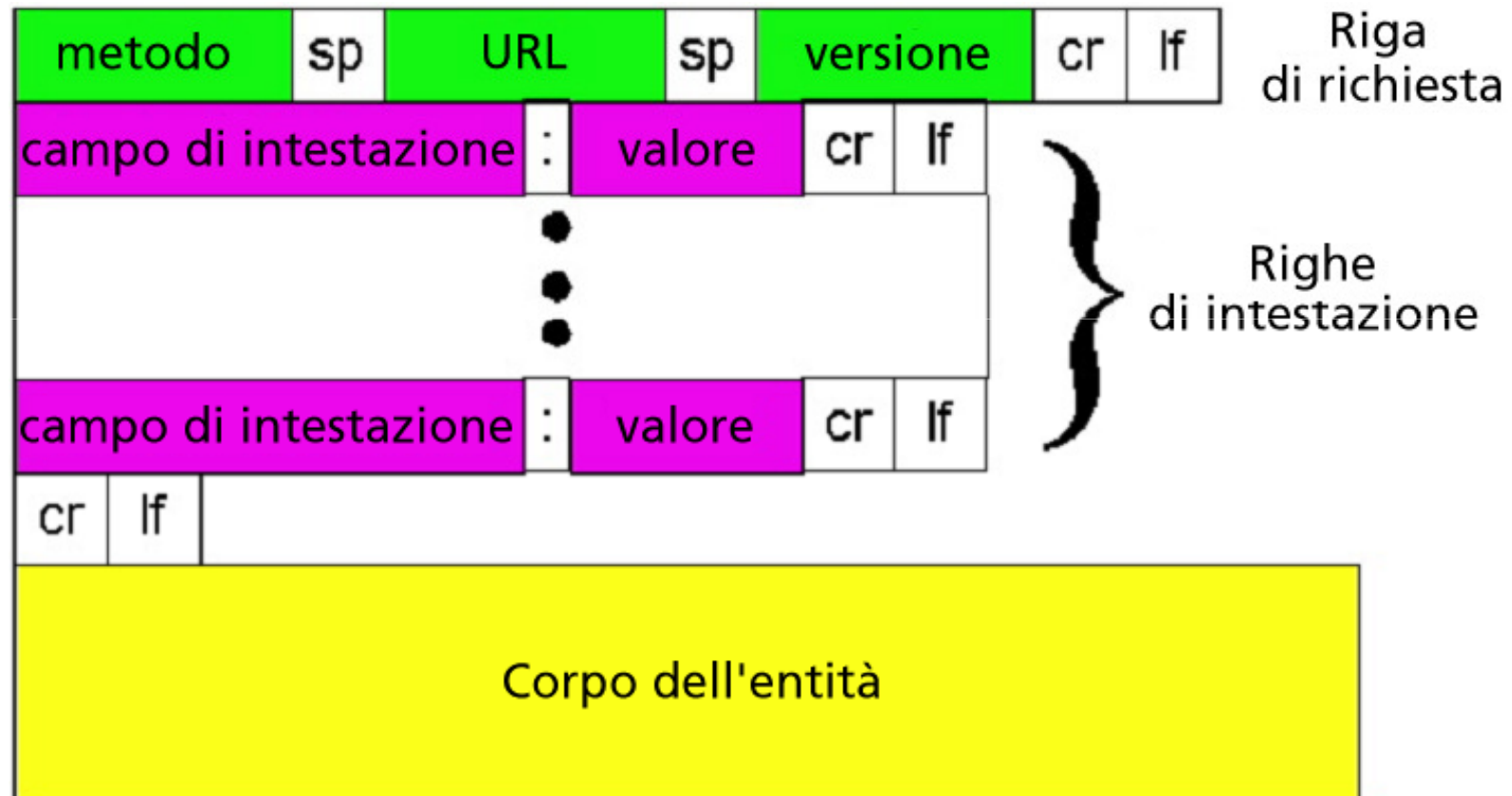
- Si hanno due tipi di messaggi HTTP: uno di *richiesta* e uno di *risposta*.
- **Messaggio di richiesta HTTP:**
  - ❖ ASCII (formato leggibile dall'utente)

Riga di richiesta → GET /somedir/page.html HTTP/1.1 <crLf>

Righe di intestazione  
[ Host: www.someschool.edu<crLf>  
User-agent: Mozilla/4.0<crLf>  
Connection: close<crLf>  
Accept-language: fr<crLf>

Un carriage return e un line feed indicano la fine del messaggio → <crLf>  
(carriage return e line feed extra)

## Messaggio di richiesta HTTP: formato generale





# Metodi

- ❑ GET: chiede la lettura della risorsa identificata dall' URL.
- ❑ HEAD: richiede solo l'intestazione della risposta. Serve per verificare le caratteristiche della risorsa, ad esempio la data dell'ultima modifica, senza trasferirla.
- ❑ POST: invia dati da elaborare al server (es. dati di una form). Il corpo della richiesta contiene i dati.
- ❑ PUT: memorizza informazioni nella locazione indicata dall' URL.
- ❑ DELETE: rimuove l'*entity* identificata dall' URL.
- ❑ TRACE: usato per tracciare l' HTTP forwarding attraverso proxy, tunnels, ecc.
- ❑ OPTIONS: usato per determinare le capacità del server, o le caratteristiche di una data risorsa.

# Tipi di metodi e versioni

## HTTP 1.0

- GET
- POST
- HEAD

## HTTP 1.1

- GET, POST, HEAD
- PUT
- DELETE
- TRACE
- OPTIONS

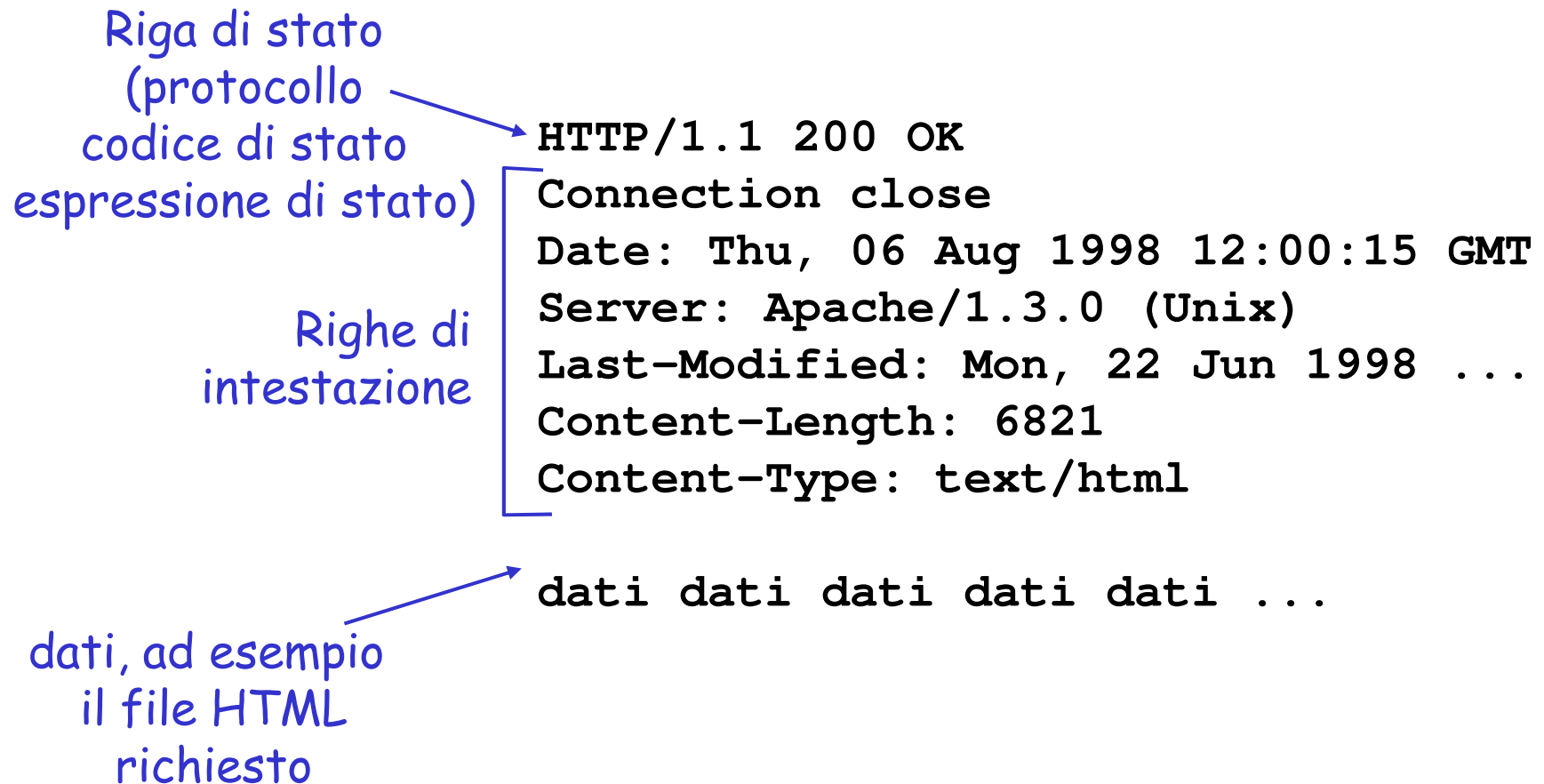
# Le righe di intestazione

- ❑ Dopo la *riga di richiesta* ci sono un certo numero di *HTTP header lines*.
- ❑ Ogni header line contiene un nome di un attributo seguito da ":", da uno spazio e da un valore.
- ❑ HTTP 1.0 definisce 16 header (tutti opzionali) mentre HTTP 1.1 46 (obbligatorio Host:).
- ❑ Alcuni header:
  - ❖ User Agent:            identifica il programma client che effettua la richiesta
  - ❖ Host:                    nome DNS del server
  - ❖ Accept:                 tipo di pagina che il client può gestire
  - ❖ Accept-Language:    lingue che il client può gestire
- ❖ Server:                 informazioni sul server
- ❖ Last-Modified         data ed ora di modifica della risorsa
- ❖ Content Type:         tipo MIME del corpo del messaggio
- ❖ Content Length:       lunghezza in byte del corpo del messaggio

Client

Server

# Messaggio di risposta HTTP



# Codici di stato della risposta HTTP

Alcuni codici di stato e relative espressioni:

## **200 OK**

- ❖ La richiesta ha avuto successo; l'oggetto richiesto viene inviato nella risposta.

## **301 Moved Permanently**

- ❖ L'oggetto richiesto è stato trasferito; la nuova posizione è specificata nell'intestazione `Location:` della risposta.

## **400 Bad Request**

- ❖ Il messaggio di richiesta non è stato compreso dal server.

## **404 Not Found**

- ❖ Il documento richiesto non si trova su questo server.

## **505 HTTP Version Not Supported**

- ❖ Il server non ha la versione di protocollo HTTP.

# Livello di applicazione

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

## Perchè abbiamo bisogno di un FTP Service?

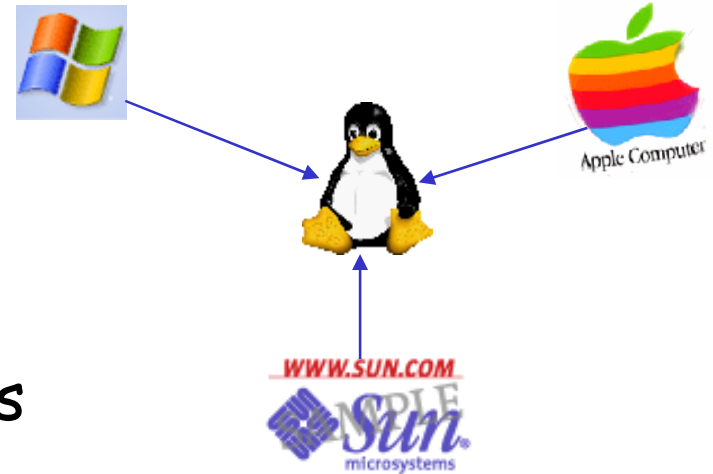
### □ **Scopo: Trasferire files tra due computers**

- ❖ Promuovere il file sharing (programmi e/o dati)
- ❖ Incoraggiare l'uso indiretto/implicito di computers remoti
- ❖ Trasferire i dati in modo affidabile ed efficiente

# Problemi del File Transfer

## □ Sistemi eterogenei usano differenti:

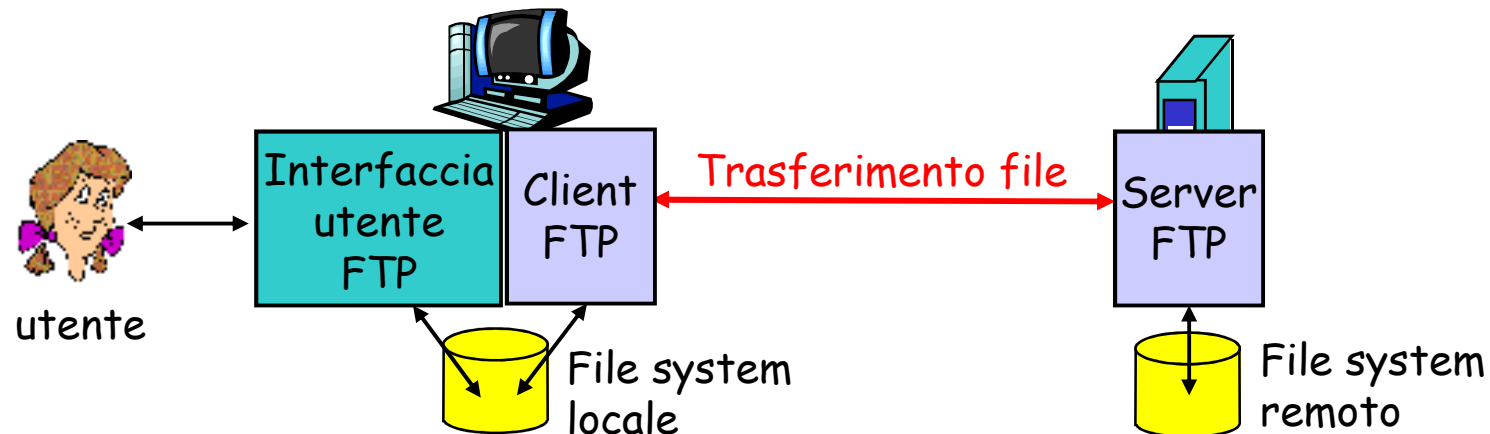
- ❖ Operating Systems
- ❖ Character Sets
- ❖ Naming Conventions
- ❖ Directory Structures
- ❖ File Structures and Formats
- ❖ **Data representation**



## □ FTP [RFC 959] deve indirizzare e risolvere questi problemi.



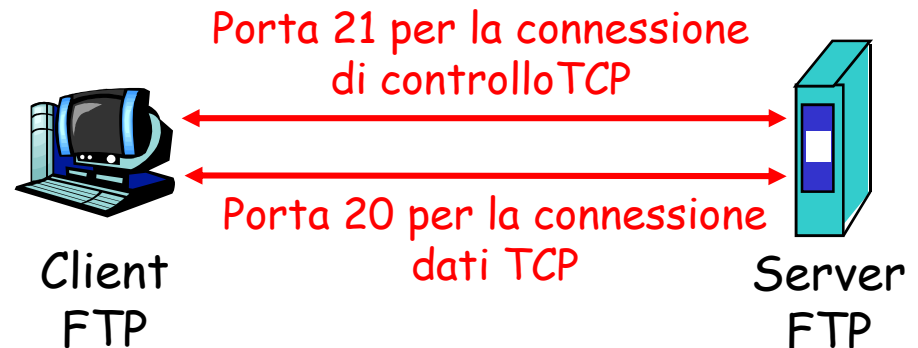
# FTP: file transfer protocol



- ❑ FTP è un protocollo utilizzato per il trasferimento di file a/da un host remoto.
- ❑ Per trasferire file con FTP è necessario installare sul proprio computer un programma FTP client (Windows e Linux includono un client FTP) che dialoga con un programma FTP server analogo, ma più sofisticato.
- ❑ Modello client/server:
  - ❖ *client*: il lato che inizia il trasferimento (a/da un host remoto)
  - ❖ *server*: host remoto

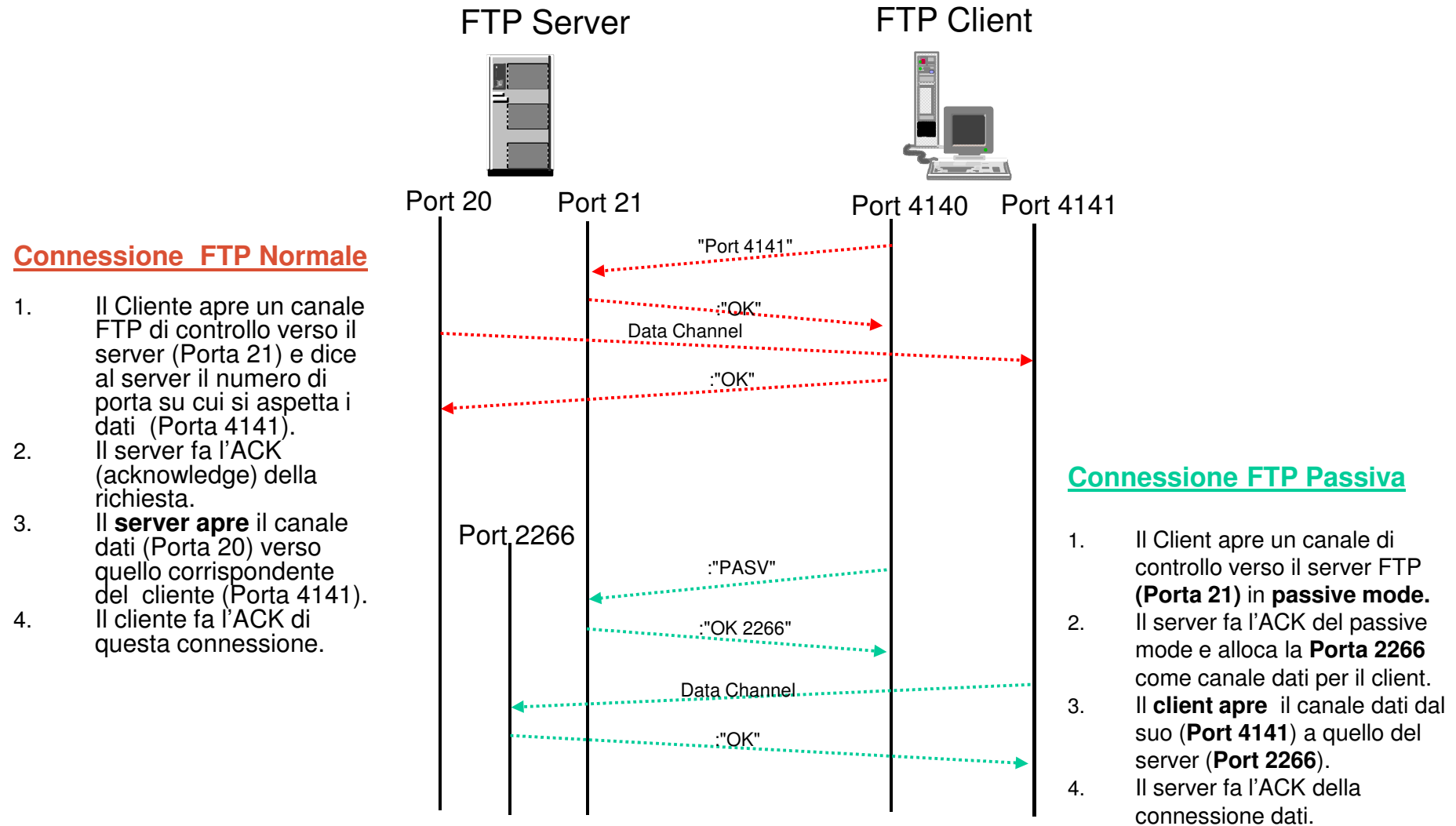
# FTP: connessione di controllo, connessione dati

- ❑ Il client FTP contatta il server FTP alla 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.

## Esempio di Connessione FTP



# Comandi e risposte FTP

## Comandi comuni:

- ❑ Inviati come testo ASCII 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

# Dialogo tipico FTP

ftp>**ftp.holdit.net**  
connected to ftp.holdit.net

220 sun ftp server ready

User:  
**joney**  
331 Guest login ok,  
Send ident as password

Password:  
**xxxxxxx**  
230 Oney login ok,  
access restrictions apply

ftp>**list hello.c myfile**  
200 PORT connect successful

150 opening ASCII mode  
data connection for hello.c

ftp>**QUIT**  
221 Goodbye

Parte l'FTP client e si connette all'host remoto.  
Il **client** riporta che la connessione è stata stabilita con successo.

Il **server replica** con un riscontro positivo.

Il client richiede all'utente di autenticarsi con il nome.  
L'utente digita il suo nome.  
Il server replica.

Il client richiede all'utente la password.  
L'utente digita la propria password.  
Il server replica.

L'utente richiede con il comando LIST il file hello.c  
Il client locale crea una seconda porta ed ha inviato il comando PORT al server.  
La connessione dati per il trasferimento è stata aperta.

L'utente chiude la connessione FTP.

# Livello di applicazione

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

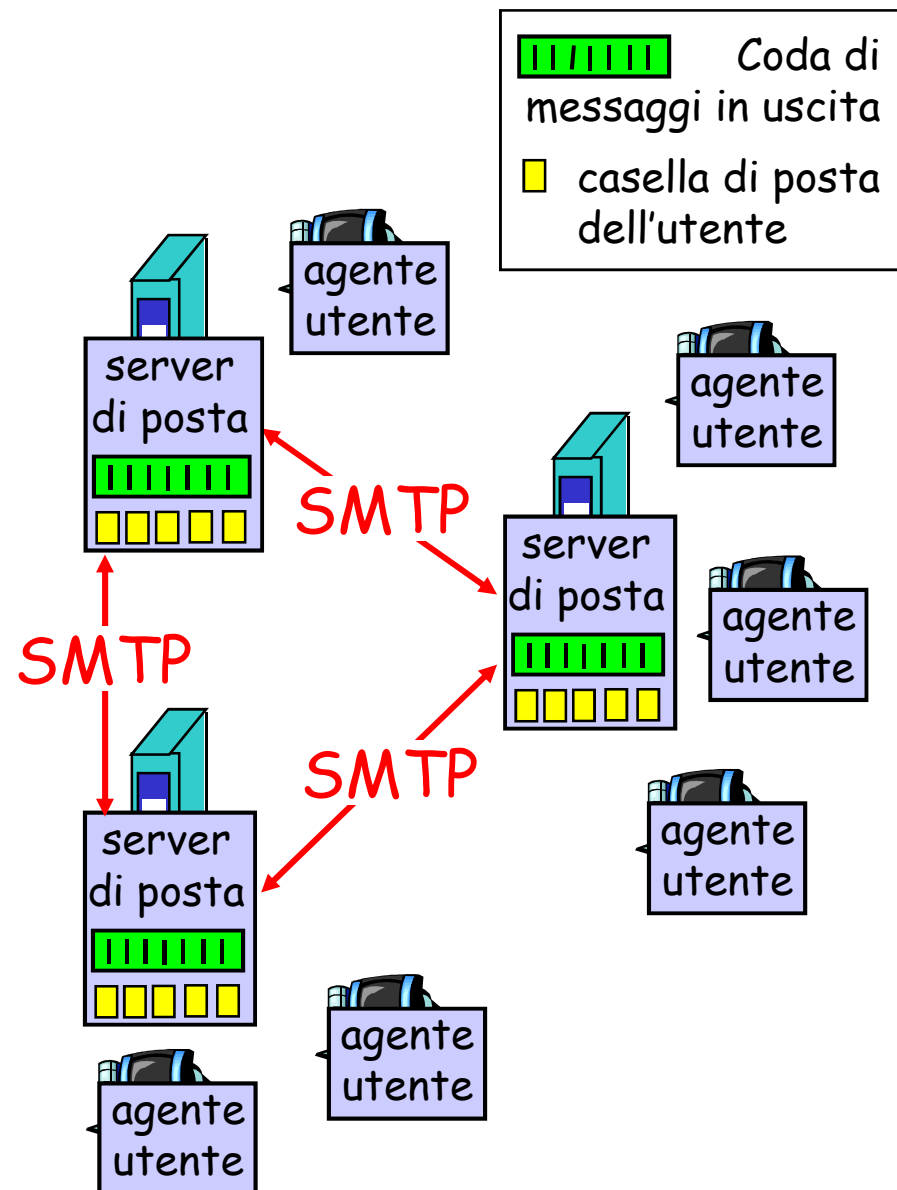
# Posta elettronica

## Tre componenti principali:

- ❑ agente utente
- ❑ server di posta
- ❑ Simple Mail Transfer Protocol: SMTP

### Agente utente:

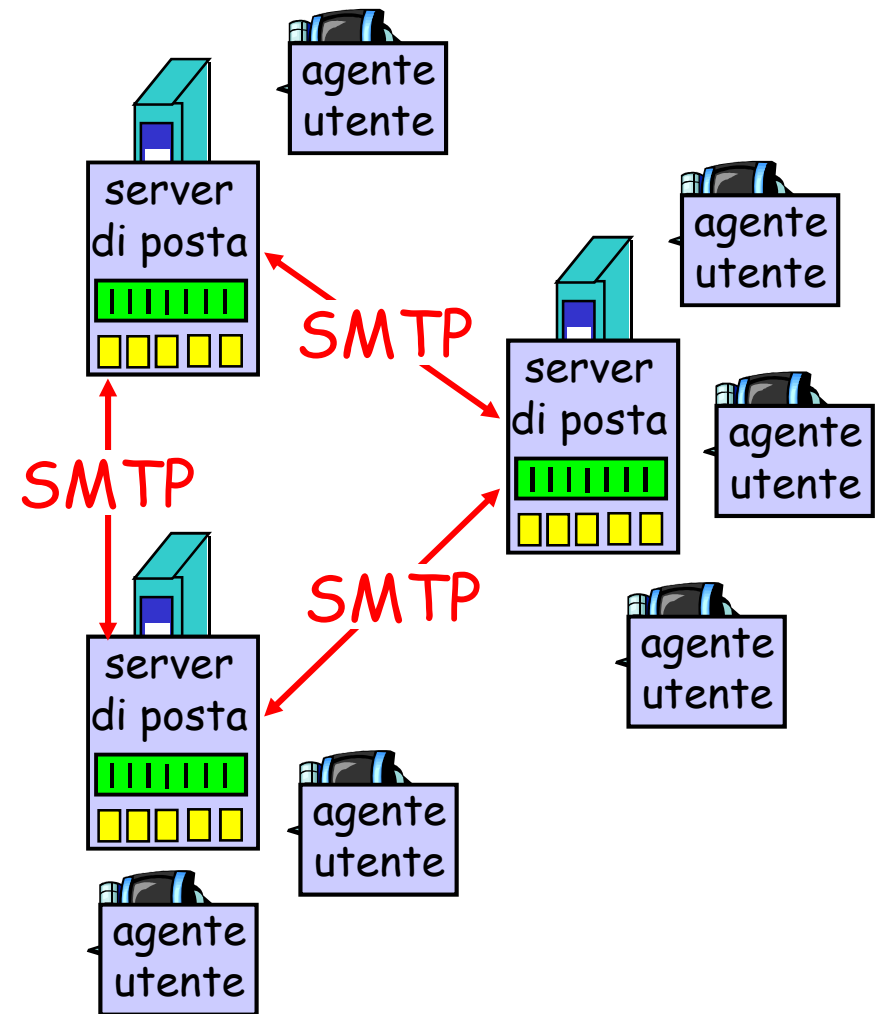
- ❑ detto anche "mail reader"
- ❑ composizione, editing, lettura dei messaggi di posta elettronica
- ❑ esempi: Eudora, Outlook, Netscape Messenger
- ❑ i messaggi in uscita o in arrivo sono memorizzati sul server di posta.



# 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



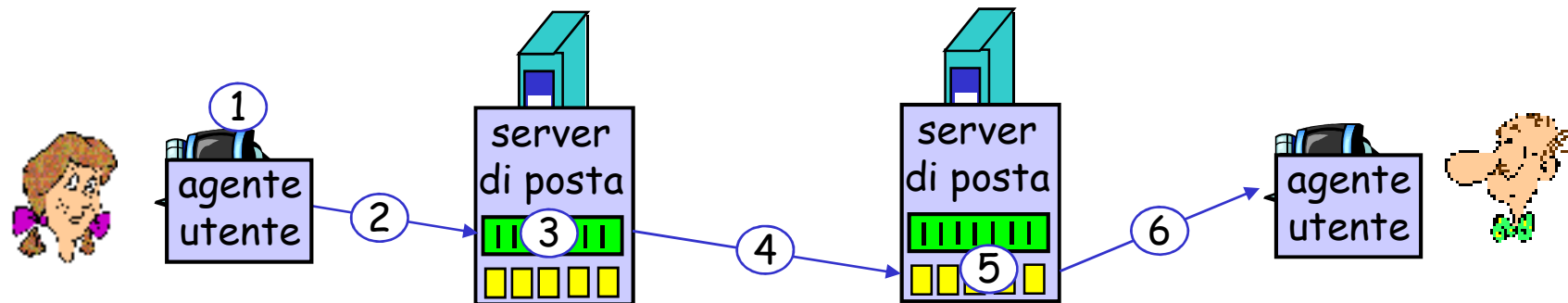


# Posta elettronica: SMTP [RFC 2821]

- ❑ L'SMTP usa TCP per trasferire in modo affidabile i messaggi di posta elettronica dal client al server, utilizzando la porta 25.
- ❑ Si ha un trasferimento diretto dal server trasmittente al server ricevente.
- ❑ Il trasferimento dei messaggi di posta avviene in 3 fasi:
  - ❖ handshaking (saluto)
  - ❖ trasferimento di messaggi
  - ❖ chiusura
- ❑ L'interazione fra i server è del tipo comando/risposta:
  - ❖ **comandi:** testo ASCII
  - ❖ **risposta:** codice di stato ed espressione
- ❑ **I messaggi devono essere nel formato ASCII a 7 bit.**
- ❑ Non è possibile trasmettere in un messaggio di e-mail caratteri non ASCII senza un'opportuna codifica.
- ❑ La soluzione è quella di usare il formato MIME (Multipurpose Internet Mail Extensions) che definisce una struttura del corpo del messaggio e come decodificare i messaggi non ASCII.

# Scenario: Alice invia un messaggio a 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
```

# Formato dei messaggi di posta elettronica

SMTP: protocollo per scambiare messaggi di posta elettronica

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

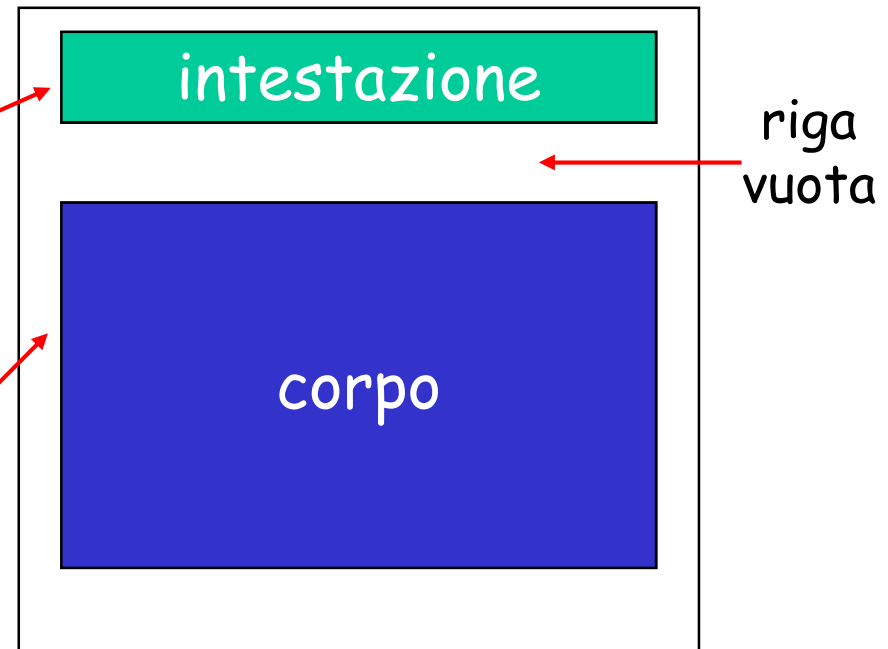
- righe di intestazione, per esempio

- ❖ To:
- ❖ From:
- ❖ Subject:

*differenti dai comandi SMTP!*

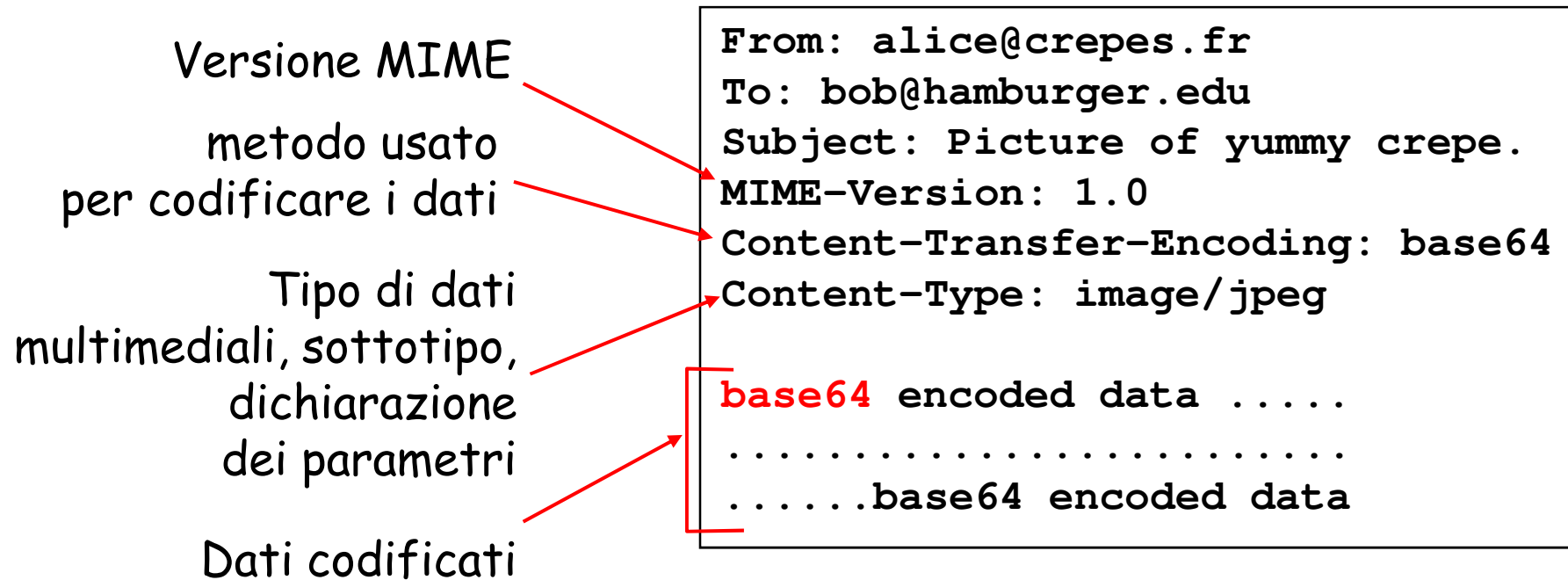
- corpo

- ❖ il "messaggio", soltanto caratteri ASCII

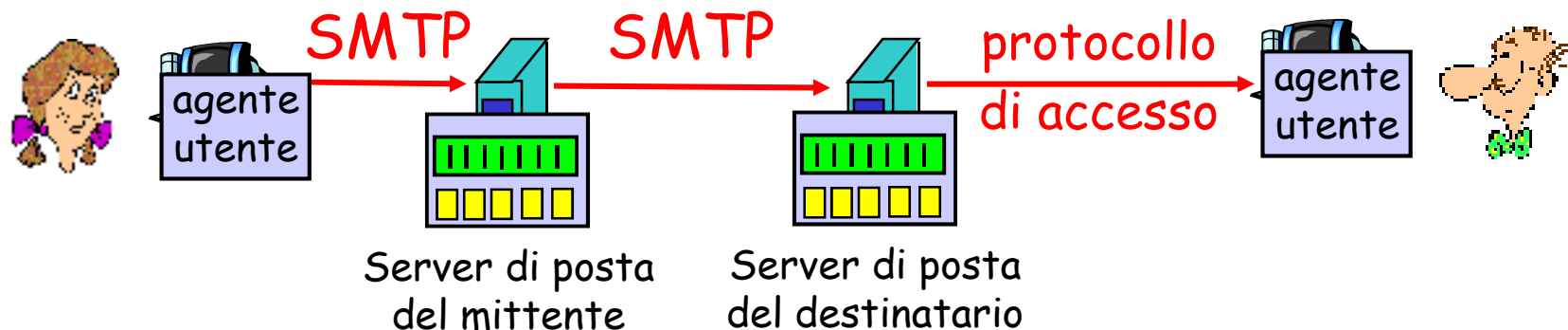


# Formato del messaggio: estensioni di messaggi multimediali

- ❑ MIME [RFC 2045, 2056] aggiunge dei campi di intestazione per definire la struttura del corpo del messaggio.
- ❑ I campi sono gestiti dall'agente utente.



# Protocolli di accesso alla posta



- ❑ SMTP si occupa di consegnare/memorizzare i messaggi sul server del destinatario.
- ❑ Per ottenere i messaggi dal server l'agente utente utilizza un protocollo di accesso alla posta:
  - ❖ 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.

# 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.

# Livello di applicazione

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



# Domain Name System (DNS)



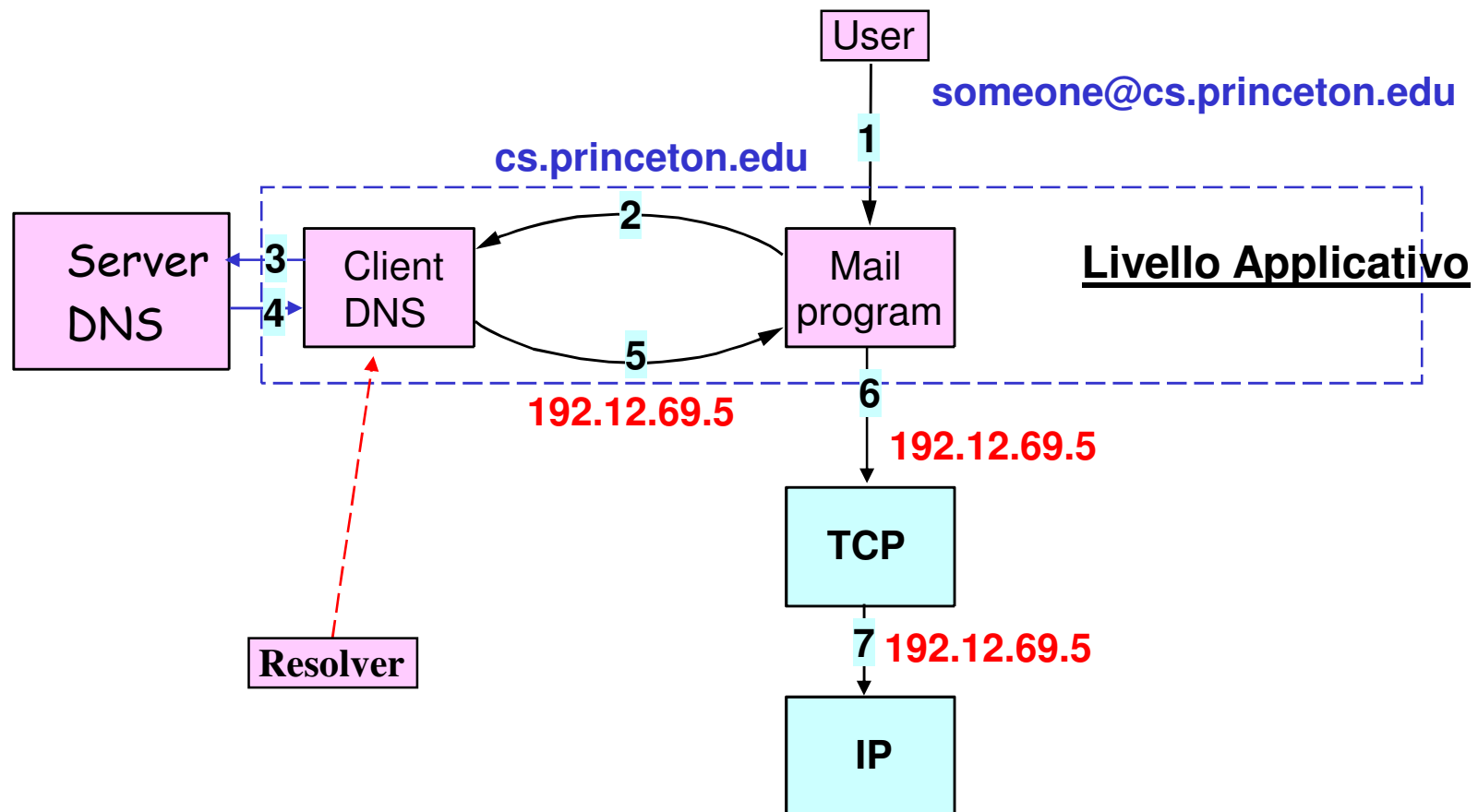
# Domain Name System (DNS)

- ❑ Il DNS [RFC 1034, 1035] 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 anche 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, mentre quella di convertire un indirizzo IP in nome è detto **risoluzione inversa**.
- ❑ Gli esseri umani trovano più facile ricordare nomi testuali (mentre gli host sono raggiungibili utilizzando gli indirizzi IP numerici).

# DNS: caratteristiche principali

- ❑ Database distribuito
- ❑ Basato sul modello client/server
- ❑ Tre componenti principali:
  - ❖ spazio dei nomi e informazioni associate (Resource Record)
  - ❖ 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)

# 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

**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)

# 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?

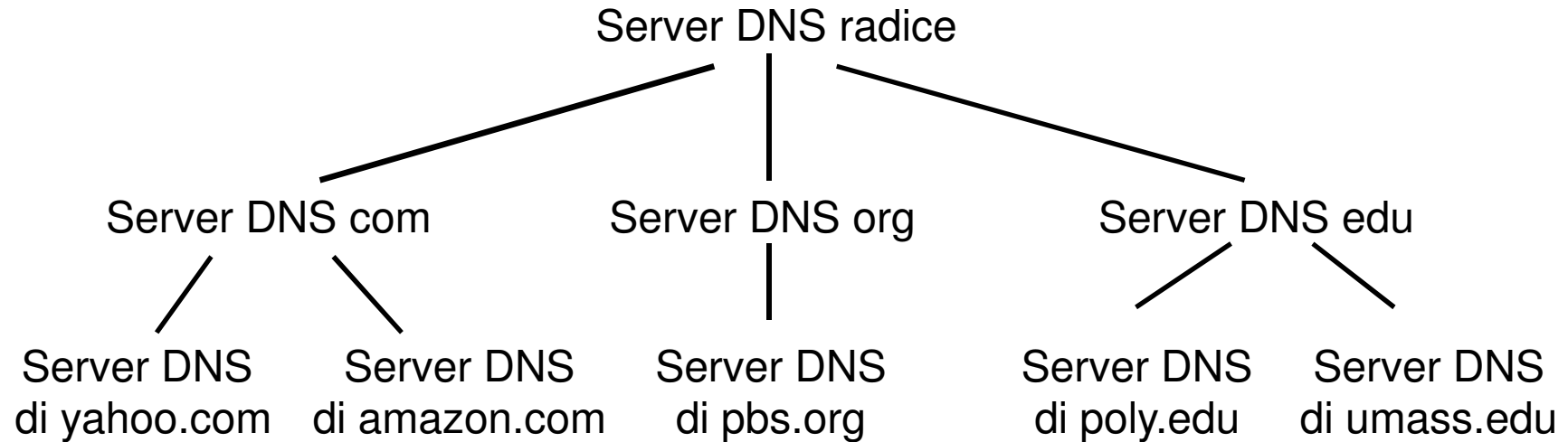
- ❑ singolo punto di guasto
- ❑ volume di traffico
- ❑ database centralizzato distante
- ❑ manutenzione

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 un'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.
- ❑ 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](http://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](http://www.amazon.com)



# DNS: server DNS radice

- ❑ Viene contattato da un server DNS locale che non può tradurre il nome.
- ❑ Il 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

# 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 mantenuti dall'organizzazione o dal service provider

# Server DNS locale

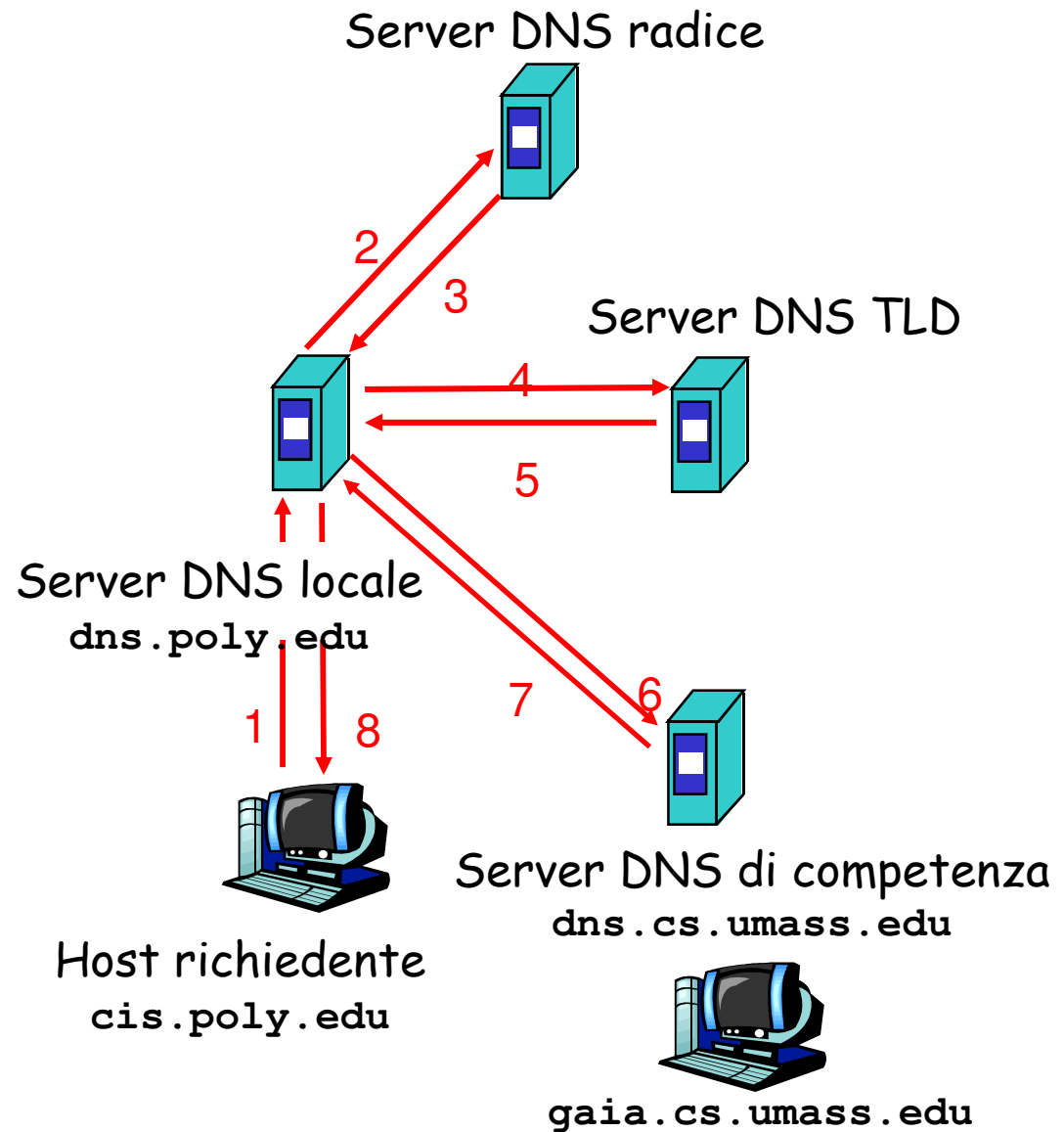
- ❑ Non appartiene strettamente alla gerarchia dei server.
- ❑ Ciascun ISP (università, società, ISP residenziale) ha un server DNS locale.
  - ❖ detto anche "default name server"
- ❑ Quando un host effettua una richiesta DNS, la query viene inviata al suo server DNS locale
  - ❖ il server DNS locale 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:

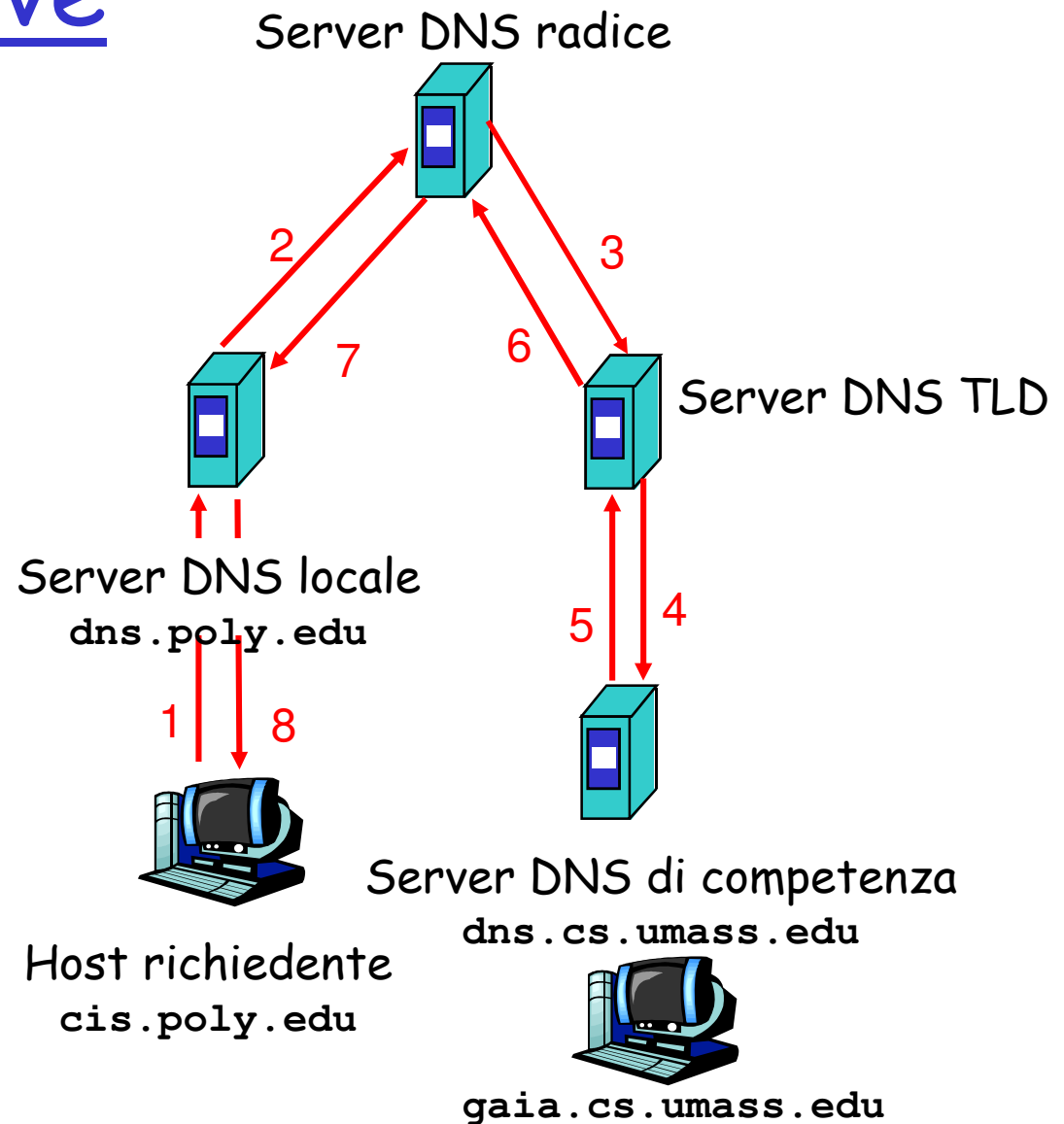
- Il server contattato risponde con il nome del server da contattare
- "Non conosco questo nome, ma chiedi a questo server"



# Query ricorsiva

## Query ricorsiva:

- Affida il compito di tradurre il nome al server DNS contattato
- Compito difficile?



# 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: caching e aggiornamento dei record

- Una volta che un server DNS impara la mappatura, la inserisce nella *cache*
  - ❖ le informazioni nella cache vengono invalidate dopo un certo periodo di tempo
  - ❖ 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



# Types

<i>Type</i>	<i>Mnemonic</i>	<i>Description</i>
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.

# Messaggi DNS

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

Intestazione del messaggio

- **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

Identificazione	Flag	12 byte
Numero di domande	Numero di RR di risposta	
Numero di RR autorevoli	Numero di RR aggiuntivi	
Domande (numero variabile di domande)		
Risposte (numero variabile di record di risorsa)		
Competenza (numero variabile di record di risorsa)		
Informazioni aggiuntive (numero variabile di record di risorsa)		

# Messaggi DNS

