

Livello applicazione:  
(RFC), protocollo FTP, protocollo Posta  
Elettronica, servizi di trasporto

Gaia Maselli

Queste slide sono un adattamento delle slide fornite dai libri di testo e pertanto protette da copyright.

- Copyright © 2013 McGraw-Hill Education Italy srl

- All material copyright 1996-2007 J.F Kurose and K.W. Ross, All Rights Reserved

# Gli standard Internet

- ❑ Uno standard Internet è una specifica che è stata rigorosamente esaminata e controllata, utile e accettata da chi utilizza le rete Internet.
- ❑ E' un insieme di regole formalizzate che devono necessariamente essere eseguite
- ❑ Esiste una procedura rigorosa attraverso la quale una specifica diviene uno standard

# Stadi o livelli di maturità

- ❑ **Proposta di standard:** stabile, di interesse per la comunità ma la specifica è ancora immatura
- ❑ **Draft:** 2 implementazioni di successo e specifica matura
- ❑ **Standard:** numero di implementazioni significativo e notevole esperienza dimostrata dagli utenti su questo standard, gli viene assegnato un numero progressivo nella lista degli standard (STD)

# Internet Documents

- ❑ Gli standard di Internet sono documenti pubblici denominati **RFC** (Request For Comments)
- ❑ L'organismo che coordina la stesura degli RFC è l'**IETF** (Internet Engineering Task Force)
- ❑ RFC - Request For Comments
  - RFC3000 in Nov 2000
  - Updated RFCs published with new numbers
  - Not all describe protocols
  - Not all used!
- ❑ [www.ietf.org/rfc.html](http://www.ietf.org/rfc.html)
- ❑ [www.rfc-editor.org](http://www.rfc-editor.org)

# Paradigma CLIENT-SERVER

Nel paradigma client/server la **comunicazione** a livello applicazione avviene **tra due programmi applicativi in esecuzione chiamati processi: un client e un server.**

- Un client è un programma in esecuzione che inizia la comunicazione inviando una richiesta;
- Un server è un altro programma applicativo che attende le richieste dai client.

# *API: Application Programming Interface*

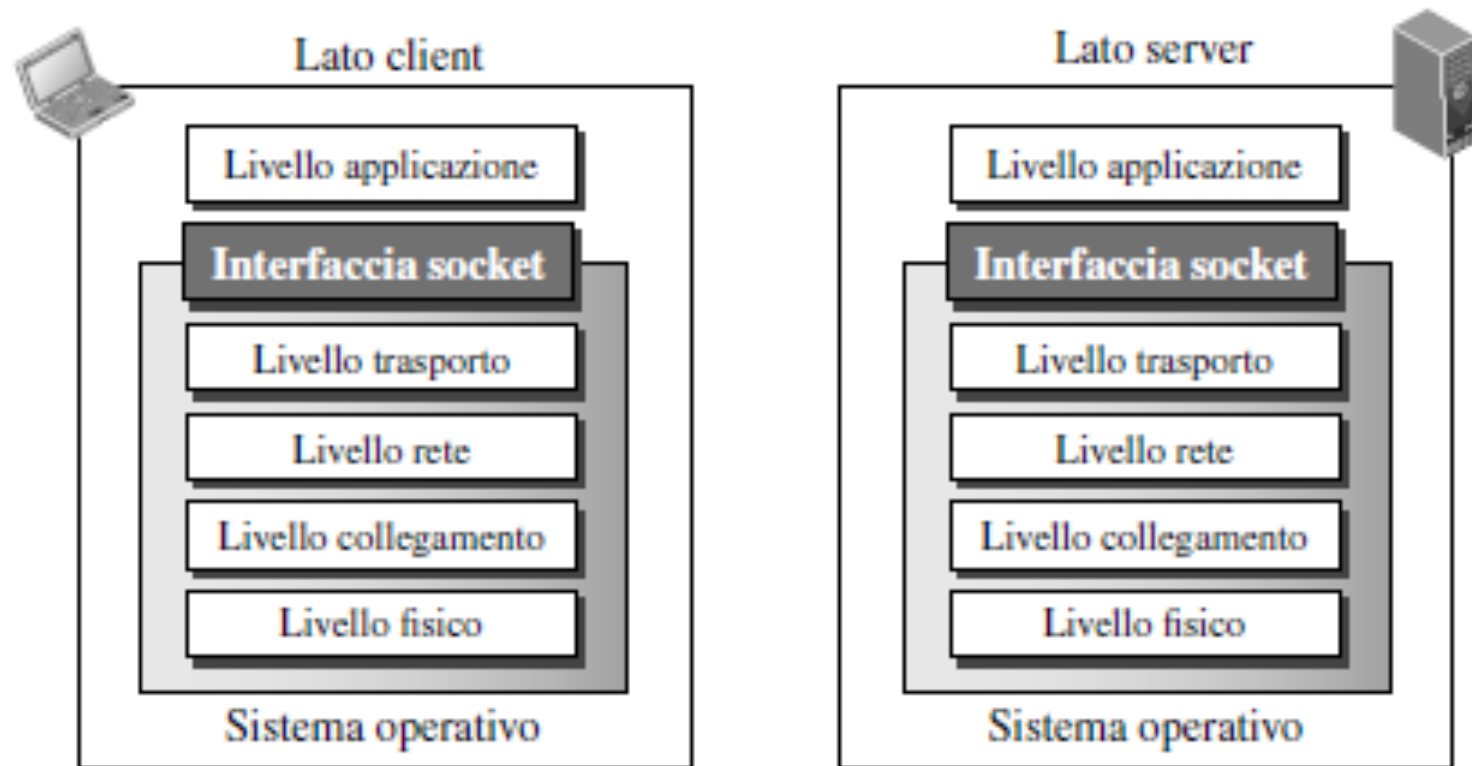
Un linguaggio di programmazione prevede un insieme di istruzioni matematiche, un insieme di istruzioni per la manipolazione delle stringhe, un insieme di istruzioni per la gestione dell'input/output ecc.

Se si vuole sviluppare un programma capace di comunicare con un altro programma, è necessario un nuovo insieme di istruzioni per chiedere ai primi quattro livelli dello stack TCP/IP di aprire la connessione, inviare/ricevere dati e chiudere la connessione.

Un insieme di istruzioni di questo tipo viene chiamato API (Application Programming Interface).

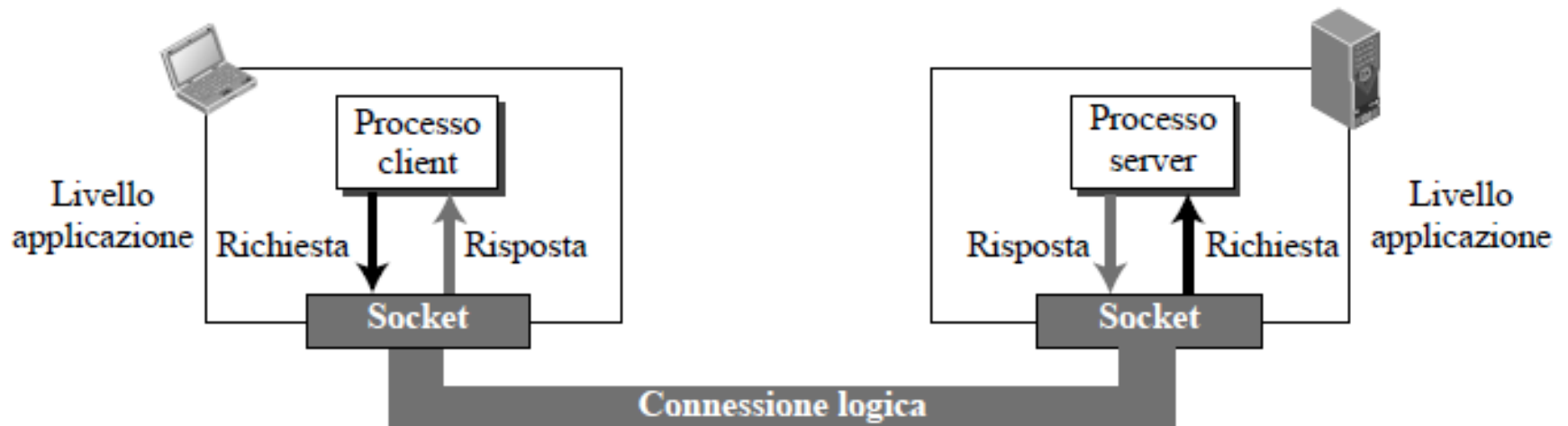
# *API di comunicazione*

## ❑ Socket



# Comunicazione tra processi: socket

- ❑ Appare come un terminale o un file ma non è un'entità fisica
- ❑ Astrazione
- ❑ Struttura dati creata e utilizzata dal programma applicativo
- ❑ Comunicare tra un processo client e un processo server significa comunicare tra due socket create nei due lati di comunicazione





# Comunicazione tra Processi

**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

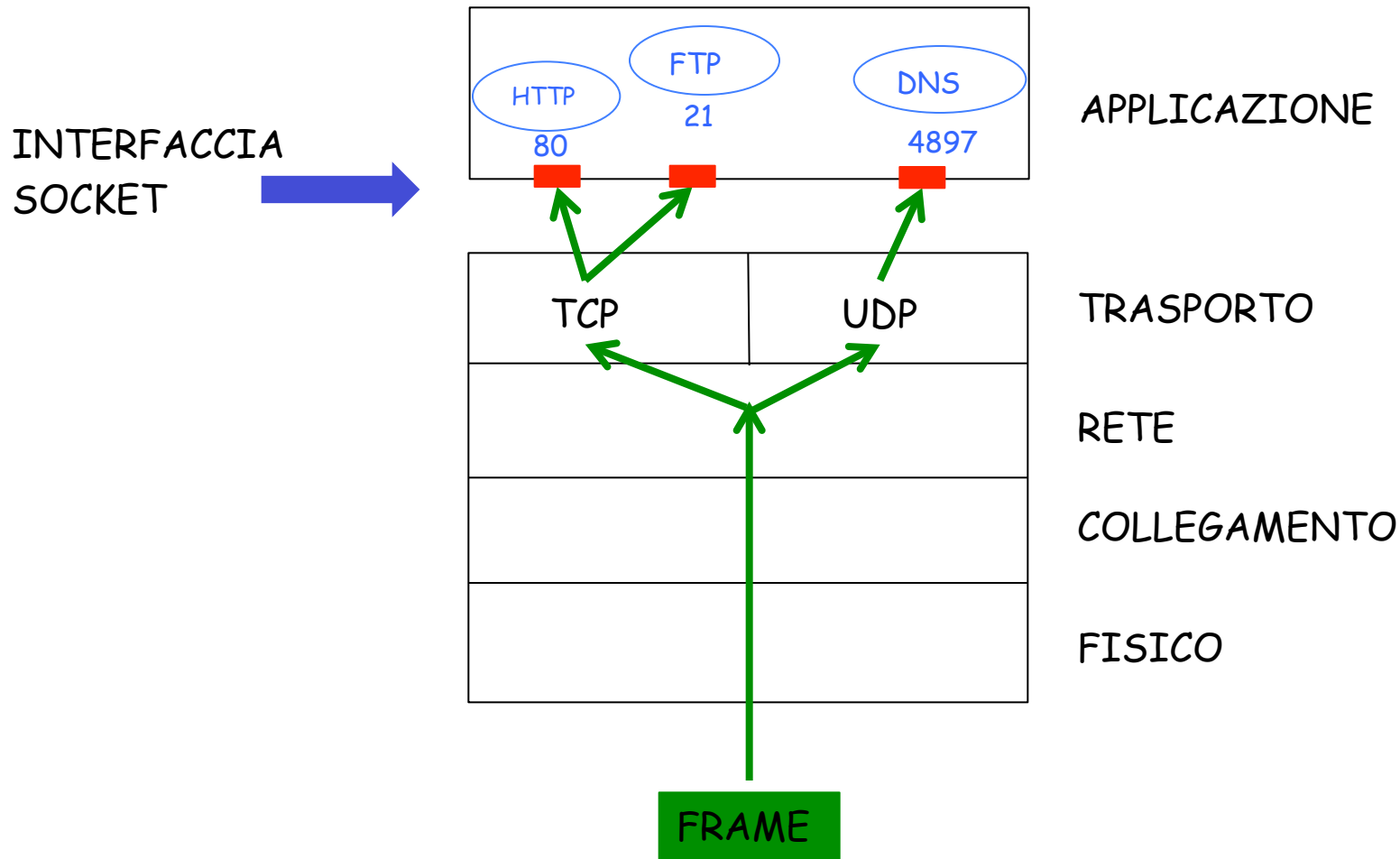
- Nota: le applicazioni con architetture P2P hanno processi client e processi server

# Indirizzamento dei processi

- ❑ 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 ha un indirizzo IP univoco a 32 bit (es. 127.0.0.1)
- ❑ **D:** È sufficiente conoscere l'indirizzo IP dell'host su cui è in esecuzione il processo per identificare il processo stesso?
- ❑ **Risposta:** 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
- ❑ Per inviare un messaggio HTTP al server gaia.cs.umass.edu:
  - ❖ **Indirizzo IP:** 128.119.245.12
  - ❖ **Numero di porta:** 80

**N.B.** Il numero di porta è una delle informazione contenute negli header di livello di trasporto per capire a quale applicazione bisogna riportare il messaggio

# Come viene recapitato un pacchetto all'applicazione

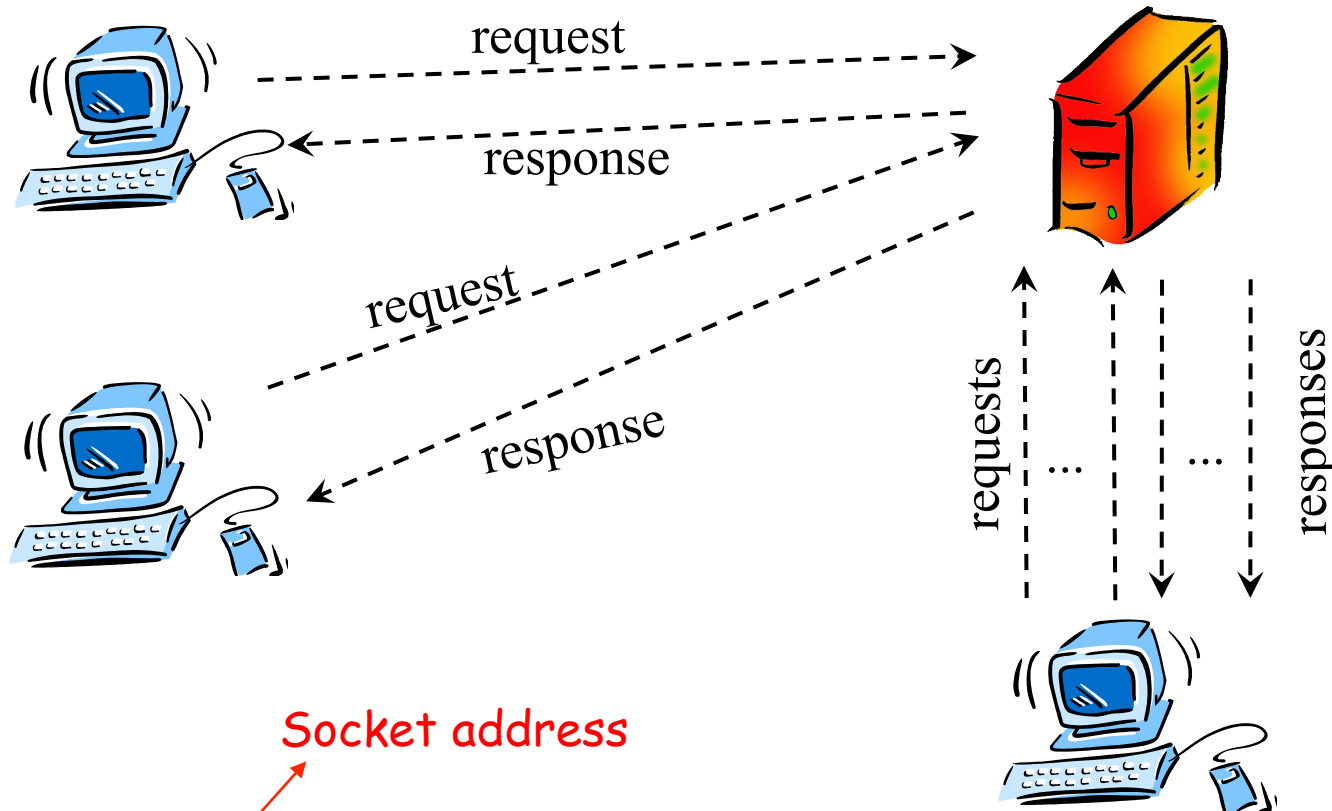


# Port numbers

- ❑ 16 bit address (0-65535)
- ❑ well known port numbers for common servers
  - ❖ FTP 20, TELNET 23, SMTP 25, HTTP 80, POP3 110, ... (full list: RFC 1700)
- ❑ number assignment (by IANA)
  - ❖ 0 not used
  - ❖ 1-255 reserved for well known processes
  - ❖ 256-1023 reserved for other processes
  - ❖ 1024-65535 dedicated to user apps

# Programmi Client e Server

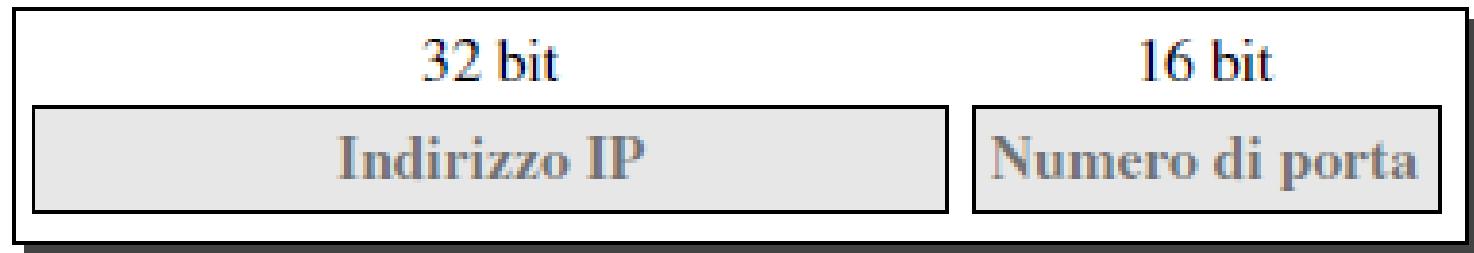
- Normalmente più client possono inviare richieste ad uno stesso server
- Un client può fare più richieste contemporanee



Socket address

- Un flusso di informazioni tra due processi identificato da una quadrupla (indirizzo IP sorgente, porta sorgente, indirizzo IP destinazione, porta destinazione)
- Di solito il client NON USA un well known port #
  - OS assegna un num. di porta disponibile

Un socket address



Socket Address

# Individuare i socket address

- ❑ L'interazione tra client e server è bidirezionale
- ❑ È necessaria quindi una coppia di indirizzi socket: *locale* (mittente) e *remoto* (destinatario)
- ❑ L'indirizzo locale in una direzione è l'indirizzo remoto nell'altra direzione
- ❑ Come vengono definiti/individuati questi indirizzi?

# Individuare i socket lato server

- ❑ Il server ha bisogno di un socket address locale (server) e uno remoto (client) per comunicare
- ❑ Socket address locale: fornito dal sistema operativo
  - ❖ Conosce l'indirizzo IP del computer su cui il server è in esecuzione
  - ❖ Il numero di porta è noto al server perché assegnato dal progettista (numero well known o scelto)
- ❑ Socket address remoto
  - ❖ è il socket address locale del client che si connette
  - ❖ Poiché numerosi client possono connettersi, il server non può conoscere a priori tutti i socket address, ma li trova all'interno del pacchetto di richiesta
- ❑ N.B. il socket address locale di un server non cambia (è fissato e rimane invariato), mentre il socket address remoto varia ad ogni interazione con client diversi (anche con stesso client su connessioni diverse)



# Individuare i socket lato client

- ❑ Il client ha bisogno di un socket address locale (client) e uno remoto (server) per comunicare
- ❑ Socket address locale: fornito dal sistema operativo
  - ❖ Conosce l'indirizzo IP del computer su cui il client è in esecuzione
  - ❖ Il numero di porta è assegnato temporaneamente dal sistema operativo (numero di porta effimero o temporaneo – non utilizzato da altri processi)
- ❑ Socket address remoto
  - ❖ Numero di porta noto in base all'applicazione (http porta 80)
  - ❖ Indirizzo IP fornito dal DNS (Domain Name System)
  - ❖ Oppure porta e indirizzo noti al programmatore quando si vuole verificare il corretto funzionamento di un'applicazione

È possibile individuare degli indirizzi a due livelli anche nella comunicazione telefonica.

Il numero di telefono identifica l'azienda, mentre l'estensione identifica un utente specifico all'interno dell'azienda.

Il numero di telefono che identifica l'azienda corrisponde in questo caso all'indirizzo IP, l'estensione che identifica il particolare utente corrisponde al numero di porta.

# *Utilizzo dei servizi di livello trasporto*

Una coppia di processi fornisce servizi agli utenti di Internet, siano questi persone o applicazioni.

La coppia di processi, tuttavia, deve utilizzare i servizi offerti dal livello trasporto per la comunicazione, poiché non vi è una comunicazione fisica a livello applicazione.

Nel livello trasporto della pila di protocolli TCP/IP sono previsti due protocolli principali:

- ❑ Protocollo UDP
- ❑ Protocollo TCP

# Quale servizio richiede l'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
- ❑ Applicazioni come la posta elettronica non hanno particolari requisiti di temporizzazione

## Throughput

- ❑ 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 si rende disponibile

## Requisiti del servizio di trasporto di alcune applicazioni comuni

| <b>Applicazione</b>           | <b>Tolleranza<br/>alla perdita<br/>di dati</b> | <b>Throughput</b>                                       | <b>Sensibilità<br/>al tempo</b> |
|-------------------------------|--|---|---------------------------------|
| Trasferimento file            | No   | Variabile   | No                              |
| Posta elettronica             | No   | Variabile   | No                              |
| Documenti Web                 | No   | Variabile   | No                              |
| Audio/video<br>in tempo reale | Sì   | Audio: da 5 Kbps a 1 Mbps<br>Video: da 10 Kbps a 5 Mbps | Sì, centinaia di ms             |
| Audio/video<br>memorizzati    | Sì   | Come sopra  | Sì, pochi secondi               |
| Giochi interattivi            | Sì   | Fino a pochi Kbps                                       | Sì, centinaia di ms             |
| Messaggistica<br>istantanea   | No   | Variabile   | Sì e no                         |

# Servizi dei protocolli di trasporto Internet

## Servizio di TCP:

- ❑ ***Orientato alla connessione:*** è richiesto un setup fra i processi client e server
  - ❑ *Somiglia al **sistema telefonico**:* come per eseguire una telefonata, l'utente deve stabilire una connessione, usarla e quindi rilasciarla.
  - ❑ Funziona come un tubo: il trasmettitore vi spinge oggetti (bit) a una estremità e il ricevitore li prende dall'altra. L'ordine è conservato ovvero i bit arrivano nella sequenza con cui sono stati trasmessi.
- ❑ ***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 sovraccaricata
- ❑ ***non offre:*** temporizzazione, garanzie su un'ampiezza di banda minima, sicurezza (alcune possono essere implementate, Es. SSL)

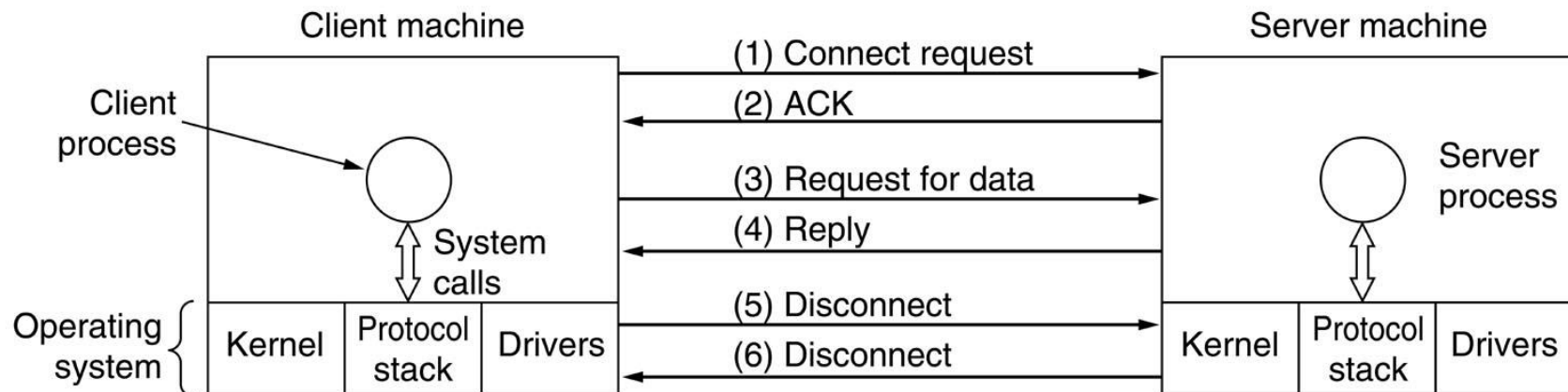
# Primitive di servizio

- Insieme minimo di *primitive* di servizio per implementare un semplice servizio *orientato alla connessione*

| Primitiva  | Significato                                      |
|------------|--|
| LISTEN     | Attesa bloccante di una connessione in arrivo    |
| CONNECT    | Stabilisce una connessione con un pari in attesa |
| RECEIVE    | Attesa bloccante per un messaggio in arrivo      |
| SEND       | Manda un messaggio al pari                       |
| DISCONNECT | Termina una connessione                          |

# Esempio di interazione client-server

- ❑ Interazione client-server in una rete orientata alla connessione
- ❑ Prima di tutto il *server* esegue una LISTEN per indicare che è pronto ad accettare connessioni in arrivo.
- ❑ Il server rimane bloccato fin quando appare una richiesta di connessione, CONNECT, da un client (1)
- ❑ Il server si mette in RECEIVE e invia una conferma di connessione (2)
- ❑ Il client esegue una SEND per trasmettere le sue richieste (3), ed esegue una RECEIVE per mettersi in attesa sulla risposta del server
- ❑ Il server esegue una SEND per inviare la sua risposta (4)
- ❑ Il client riceve la risposta e invia DISCONNECT per richiesta di chiusura della connessione (5)
- ❑ Il server conferma chiusura connessione con DISCONNECT (6)





# Servizi dei protocolli di trasporto Internet

## Servizio di UDP:

- ❑ *Senza connessione*: non è richiesto alcun setup fra i processi client e server
- ❑ trasferimento dati inaffidabile fra i processi d'invio e di ricezione
  - ❑ *Somiglia al **sistema postale***: Ogni messaggio è instradato attraverso il sistema postale in modo indipendente dagli altri
  - ❑ E' possibile che due messaggi mandati alla stessa destinazione arrivino in tempi diversi
- ❑ *non offre*: setup della connessione, affidabilità, controllo di flusso, controllo della congestione, temporizzazione né ampiezza di banda minima e sicurezza

D: Perché esiste UDP?

## Applicazioni Internet: protocollo a livello applicazione e protocollo di trasporto

| <b>Applicazione</b>        | <b>Protocollo a livello applicazione</b> | <b>Protocollo di trasporto sottostante</b> |
|----------------------------|--|--|
| Posta elettronica          | SMTP [RFC 2821]                          | TCP  |
| Accesso a terminali remoti | Telnet [RFC 854]                         | TCP  |
| Web                        | HTTP [RFC 2616]                          | TCP  |
| Trasferimento file         | FTP [RFC 959]                            | TCP  |
| Multimedia in streaming    | HTTP (es. YouTube)<br>RTP [RFC 1889]     | TCP o UDP                                  |
| Telefonia Internet         | SIP, RTP, proprietario<br>(es. Skype)    | Tipicamente UDP                            |

# Livello di applicazione

- ❑ FTP
- ❑ Posta elettronica
  - ❖ SMTP, POP3, IMAP

# Interazione in rete

- ❑ Come possiamo interagire con una macchina remota?
- ❑ Direttamente
  - ❖ l'utente è loggato su un host remoto e interagisce mediante un terminale (telnet)
  - ❖ L'utente non si accorge di differenze tra i due sistemi
  - ❖ L'utente deve conoscere le convenzioni del sistema remoto

# Interazione in rete

- ❑ Come possiamo interagire con una macchina remota?
- ❑ Indirettamente
  - ❖ L'utente non deve loggarsi sull'host remoto
  - ❖ L'utente non deve sapere come usare l'host remoto
  - ❖ Un processo intermedio rende la maggior parte delle differenze nei comandi e nelle convenzioni invisibili all'utente
  - ❖ Il trasferimento di file (FTP) richiede di conoscere solo un insieme standard di comandi di trasferimento per il sistema locale al fine di utilizzare il sistema remoto

# File Transfer Protocol (FTP)

- ❑ Programma di trasferimento file DA/A un host remoto
- ❑ Comando per accedere ed essere autorizzato a scambiare informazioni con l'host remoto

```
ftp NomeHost
```

vengono richiesti nome utente e password

- ❑ Trasferimento di un file da un host remoto

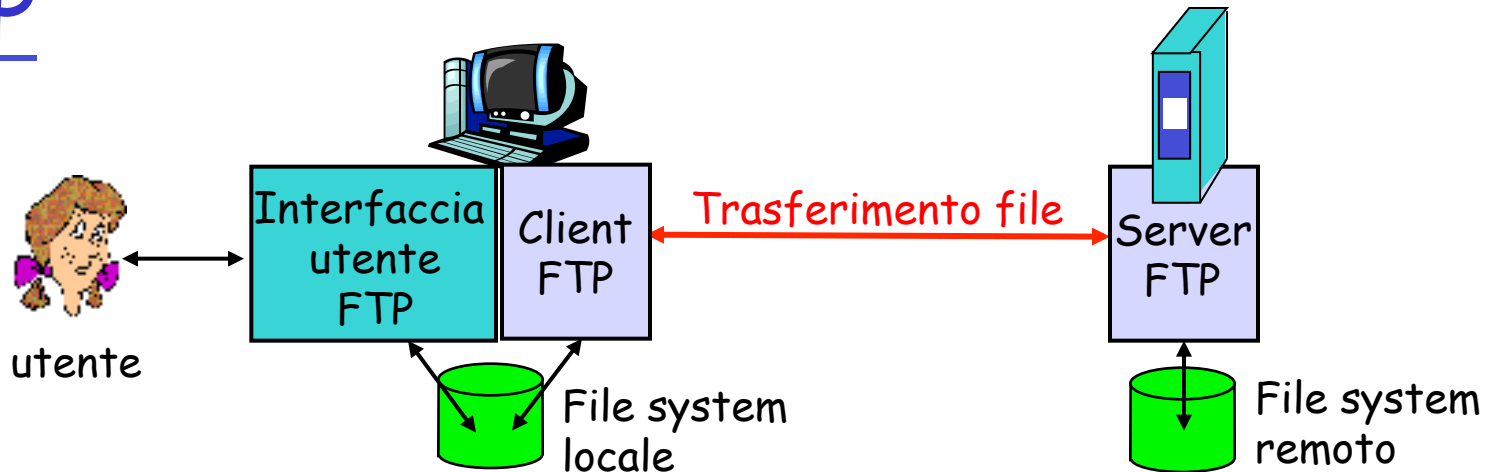
```
ftp> get file1.txt
```

- ❑ Trasferimento di un file a un host remoto

```
ftp> put file2.txt
```

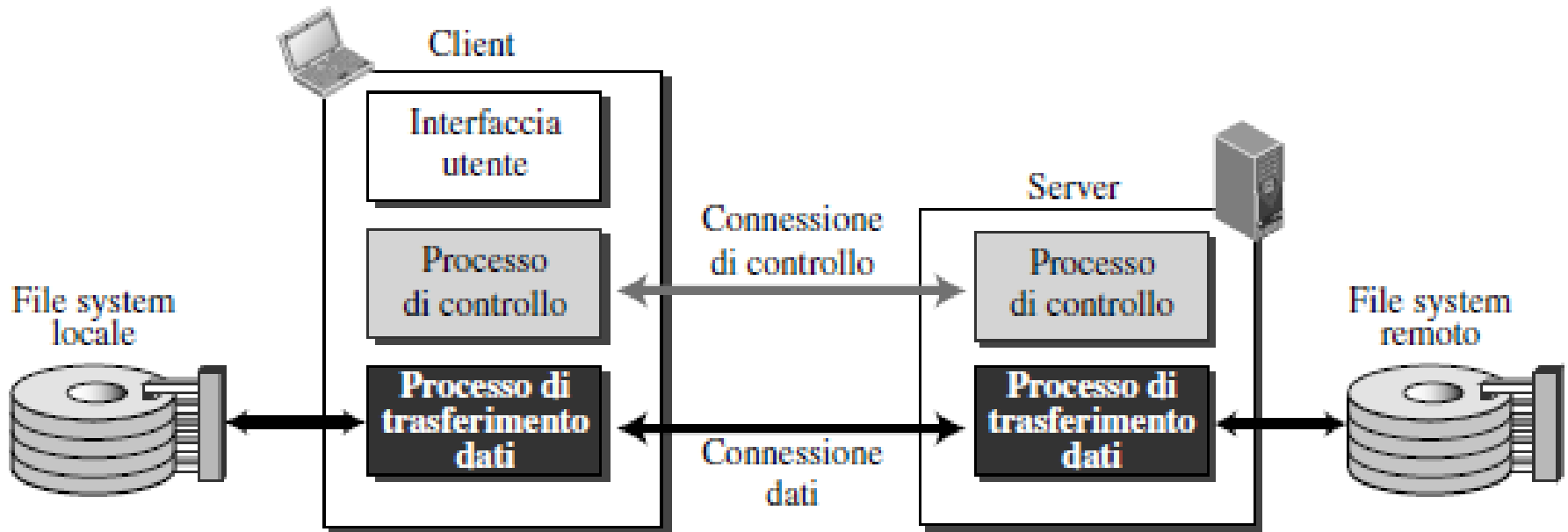
- ❑ Ci sono comandi per cambiare directory in locale e sull'host remoto, cancellare file, etc.

# FTP



- ❑ Modello client/server
  - ❖ *client*: il lato che inizia il trasferimento (a/da un host remoto)
  - ❖ *server*: host remoto
- ❑ Quando l'utente fornisce il nome dell'host remoto (`ftp NomeHost`), il processo client FTP stabilisce una connessione TCP sulla porta 21 con il processo server FTP
- ❑ Stabilita la connessione, il client fornisce nome utente e password che vengono inviate sulla connessione TCP come parte dei comandi
- ❑ Ottenuta l'autorizzazione del server il client può inviare uno o più file memorizzati nel file system locale verso quello remoto (o viceversa)

# FTP client e server



- **Connessione di controllo:** si occupa delle informazioni di controllo del trasferimento e usa regole molto semplici, così che lo scambio di informazioni si riduce allo scambio di una riga di comando (o risposta) per ogni interazione
- **Connessione dati:** si occupa del trasferimento del file

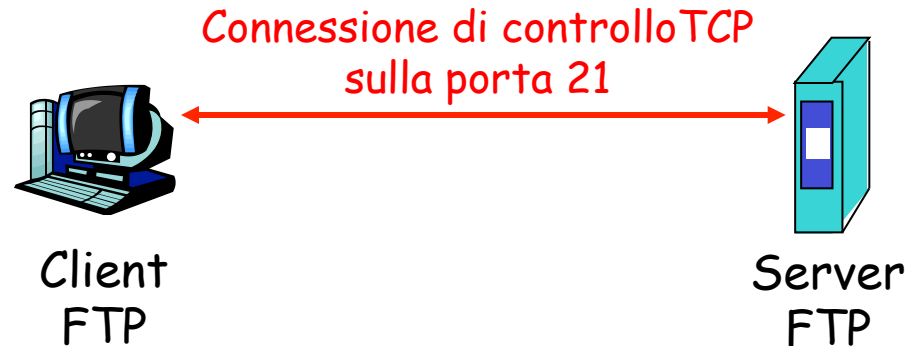


# FTP: connessione di controllo

- Connessione di controllo (porta 21) viene usata per inviare informazioni di controllo
- L'apertura della connessione di controllo viene richiesta dal client al comando

`ftp NomeHost`

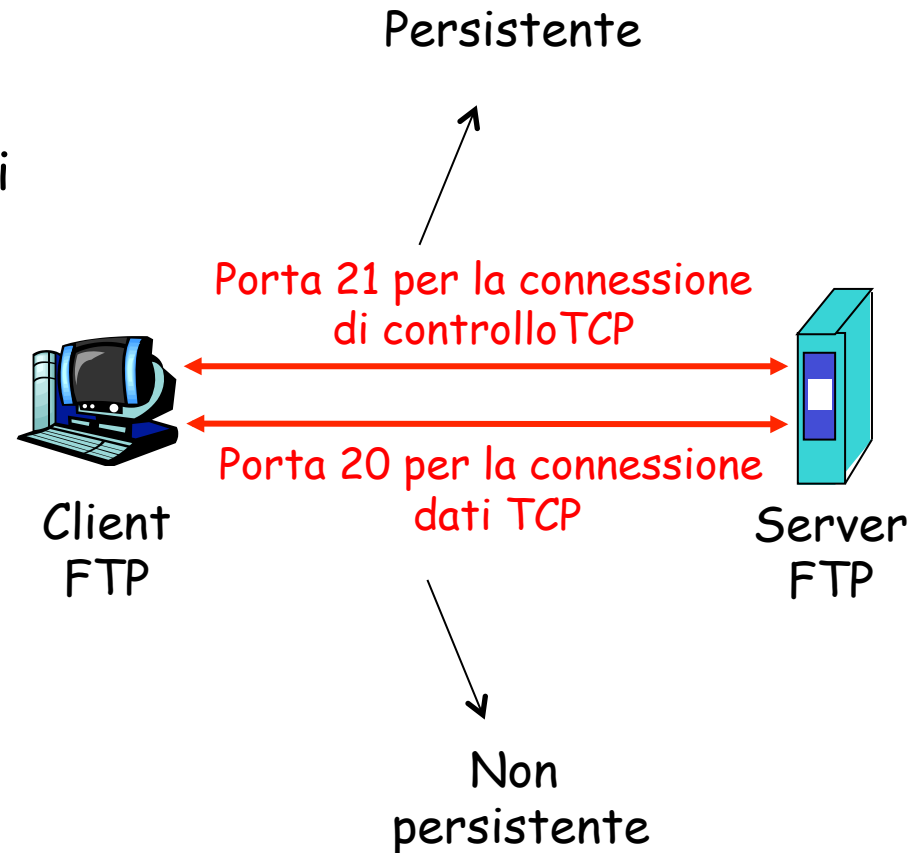
- tutti i comandi eseguiti dall'utente sono trasferiti sulla connessione di controllo
- Esempi di informazioni trasferite sulla connessione di controllo
  - Identificativo utente
  - Password
  - Comandi per cambiare directory
  - Comandi per richiedere invio (put) e ricezione (get) di file



- Connessione di controllo: **"fuori banda"** (*out of band*)
- HTTP utilizza la stessa connessione per messaggi di richiesta e risposta e file, per cui si dice che invia le informazioni di controllo **"in banda"** (*in-band*)
- Il server FTP mantiene lo **"stato"**: directory corrente, autenticazione precedente

# FTP: connessione dati

- ❑ Connessione dati: quando il server riceve un comando per trasferire un file (es. `get`, `put`), apre una connessione dati TCP sulla porta 20 con il client
- ❑ Dopo il trasferimento di un file, il server chiude la connessione
- ❑ La connessione dati viene aperta dal server e utilizzata per il vero e proprio invio del file.
- ❑ Si crea una nuova connessione per ogni file trasferito all'interno della sessione



# Lifetime delle connessioni

- ❑ La connessione di controllo rimane attiva durante l'intera sessione FTP
- ❑ La connessione dati viene aperta e chiusa per ogni trasferimento file
  - ❖ Viene stabilita ogni volta che viene ricevuto un comando di trasferimento sulla connessione di controllo
  - ❖ Viene chiusa al termine del trasferimento
  - ❖ La connessione dati può essere aperta e chiusa più volte se vengono trasferiti più file all'interno della stessa sessione

# Comandi e risposte FTP

- ❑ Esiste una corrispondenza uno a uno tra il comando immesso dall'utente e quello FTP inviato sulla connessione di controllo
- ❑ Ciascun comando è seguito da una risposta spedita dal server al client (codice di ritorno + testo)

## Comandi comuni:

- ❑ Inviati come testo ASCII sulla connessione di controllo
- ❑ **USER *username***
- ❑ **PASS *password***
- ❑ **LIST**  
elenca i file della directory corrente (*dir*), la lista di file viene inviata dal server su una nuova connessione dati
- ❑ **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

# Principali comandi FTP

Codifica standard chiamata NVT ASCII sia per i comandi che per le risposte

| <i>Comando</i> | <i>Argomenti</i>       | <i>Descrizione</i>                              |
|----------------|------------------------|---|
| ABOR           |                        | Interruzione del comando precedente             |
| CDUP           |                        | Sale di un livello nell'albero delle directory  |
| CWD            | Nome della directory   | Cambia la directory corrente                    |
| DELE           | Nome del file          | Cancella il file                                |
| LIST           | Nome della directory   | Elenca il contenuto della directory             |
| MKD            | Nome della directory   | Crea una nuova directory                        |
| PASS           | Password utente        | Password  |
| PASV           |                        | Il server sceglie la porta                      |
| PORT           | Numero di porta        | Il client sceglie la porta                      |
| PWD            |                        | Mostra il nome della directory corrente         |
| QUIT           |                        | Uscita dal sistema                              |
| RETR           | Nome di uno o più file | Trasferisce uno o più file dal server al client |
| RMD            | Nome della directory   | Cancella la directory                           |
| RNTO           | Nome (del nuovo) file  | Cambia il nome del file                         |
| STOR           | Nome di uno o più file | Trasferisce uno o più file dal client al server |
| USER           | Identificativo         | Identificazione dell'utente                     |

# Esempi di risposte FTP

Le risposte sono composte da due parti: un numero di 3 cifre e un testo. La parte numerica costituisce il codice della risposta, quella di testo contiene i parametri necessari o informazioni supplementari

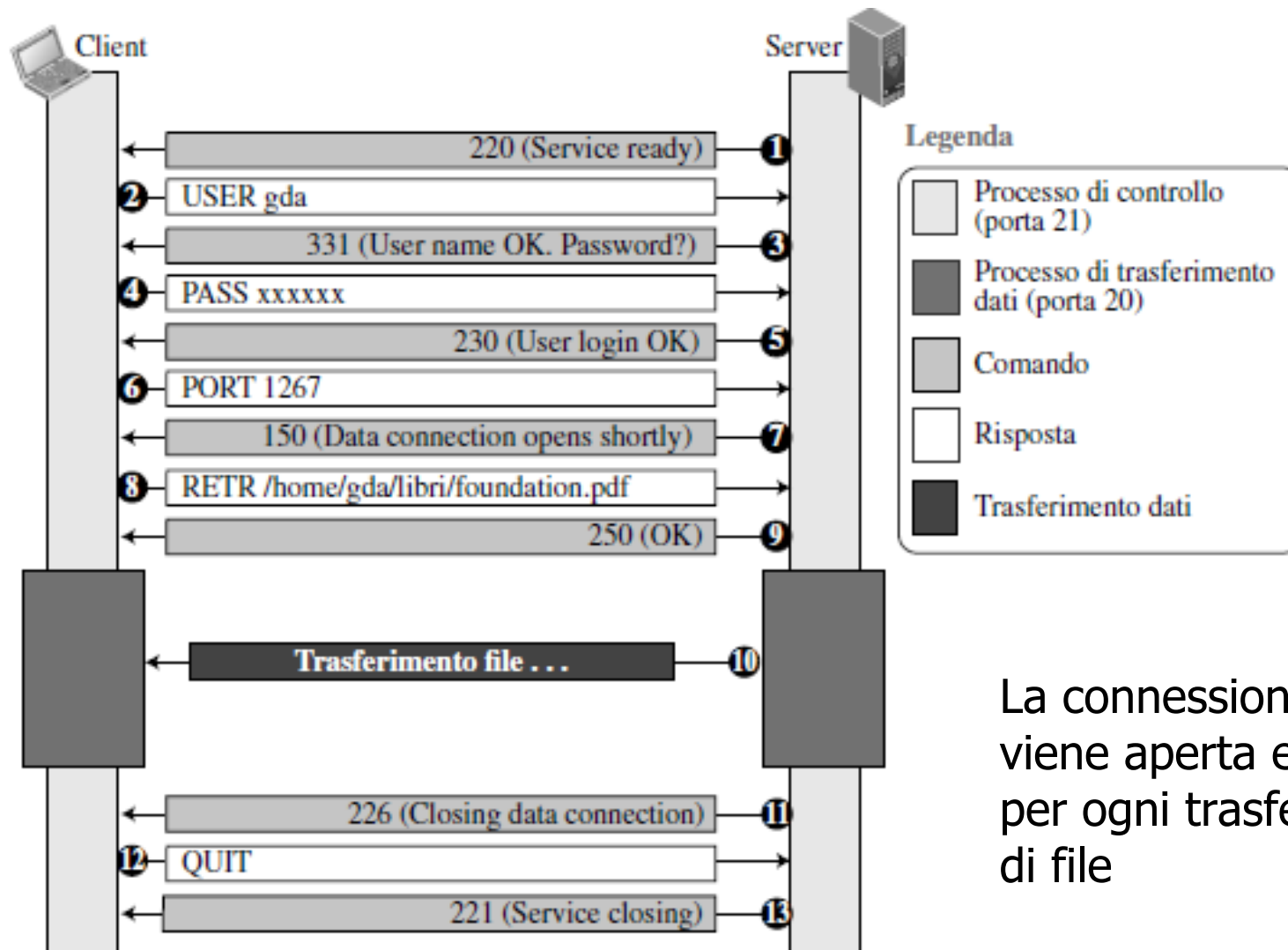
La tabella riporta alcuni codici (non il testo)

| <i>Codice</i> | <i>Descrizione</i>           | <i>Codice</i> | <i>Descrizione</i>                                 |
|---------------|------------------------------|---------------|--|
| 125           | Connessione dati aperta      | 250           | Azione sul file OK                                 |
| 150           | Stato del file OK            | 331           | Nome dell'utente OK; in attesa della password      |
| 200           | Comando OK                   | 425           | Non è possibile aprire la connessione dati         |
| 220           | Servizio pronto              | 450           | Azione sul file non eseguita; file non disponibile |
| 221           | Servizio in chiusura         | 452           | Azione interrotta; spazio insufficiente            |
| 225           | Connessione dati aperta      | 500           | Errore di sintassi; comando non riconosciuto       |
| 226           | Connessione dati in chiusura | 501           | Errore di sintassi nei parametri o negli argomenti |
| 230           | Login dell'utente OK         | 530           | Login dell'utente fallito                          |

# Connessione per lo scambio dei dati

- ❑ Si usa la porta 20 e client e server sono inversi
- 1. Il client, non il server, effettua un'apertura passiva (in attesa di connessioni) su una porta effimera
- 2. Il client invia questo numero di porta al server (per mezzo del comando PORT)
- 3. Il server, ricevuto il numero di porta, effettua un'apertura attiva sulla porta effimera segnalata dal client (dalla porta 20)

# esempio



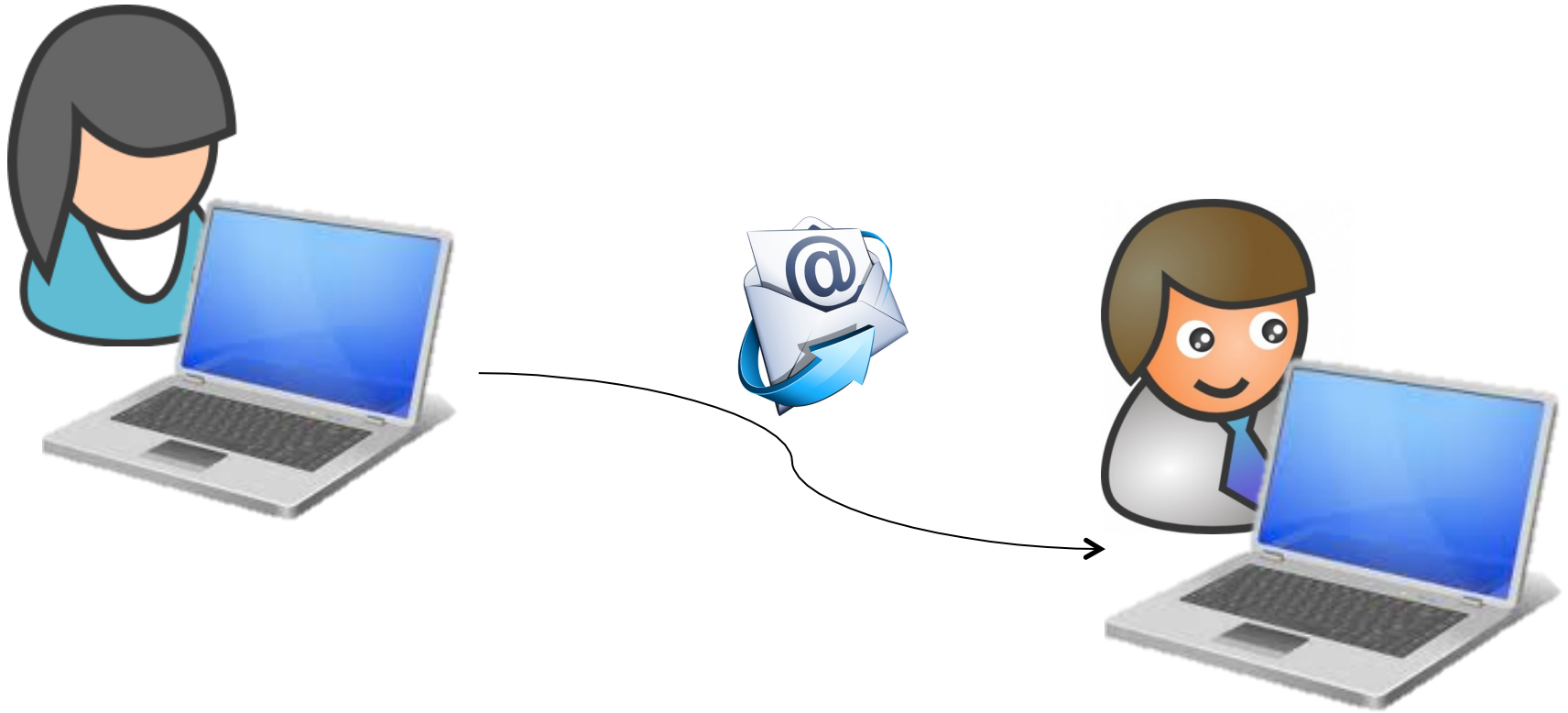
La connessione dati  
viene aperta e chiusa  
per ogni trasferimento  
di file



# Livello applicazione

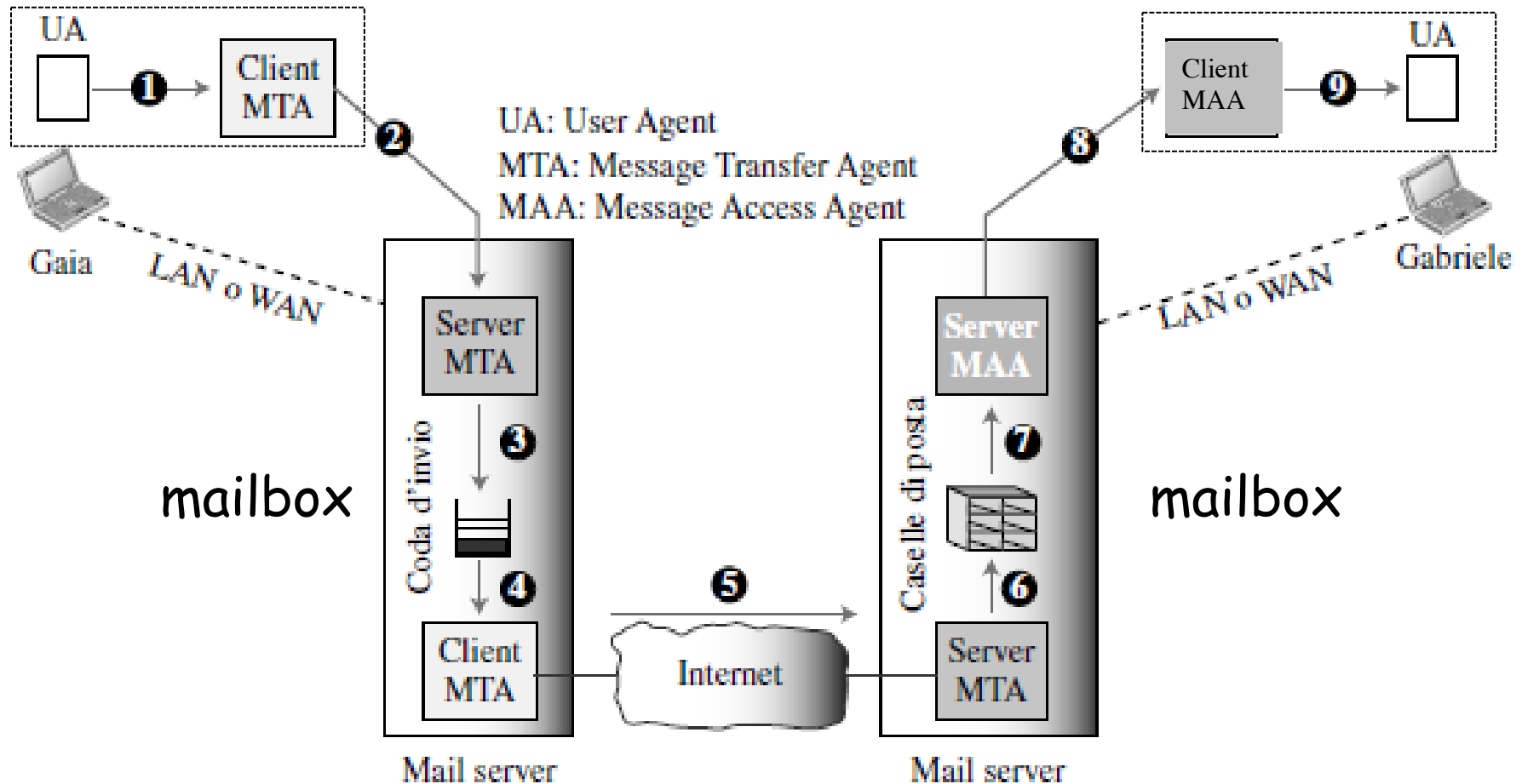
- ❑ FTP
- ❑ Posta elettronica
  - ❖ SMTP, POP3, IMAP

# Caratteristiche dell'applicazione?



- Transazione unidirezionale (non c'è richiesta/risposta come in HTTP e FTP)
- Destinatario: server sempre attivo?

# Posta elettronica: scenario classico



## Tre componenti principali:

1. User agent: usato per scrivere e inviare un messaggio o leggerlo
2. Message Transfer Agent: usato per trasferire il messaggio attraverso Internet
3. Message Access Agent: usato per leggere la mail in arrivo

# User agent

- ❑ Lo user agent viene attivato dall'utente o da un timer: se c'è una nuova email informa l'utente
- ❑ detto anche "mail reader"
- ❑ composizione, editing, lettura dei messaggi di posta elettronica
- ❑ esempi:
  - ❑ Eudora, Outlook, Thunderbird
  - ❑ Mail, Pine, Elm
- ❑ i messaggi in uscita o in arrivo sono memorizzati sul server
- ❑ Il messaggio da inviare viene passato al Mail Transfer Agent
- ❑ Come comunicano i Mail Transfer Agent?

# Message Transfer Agent

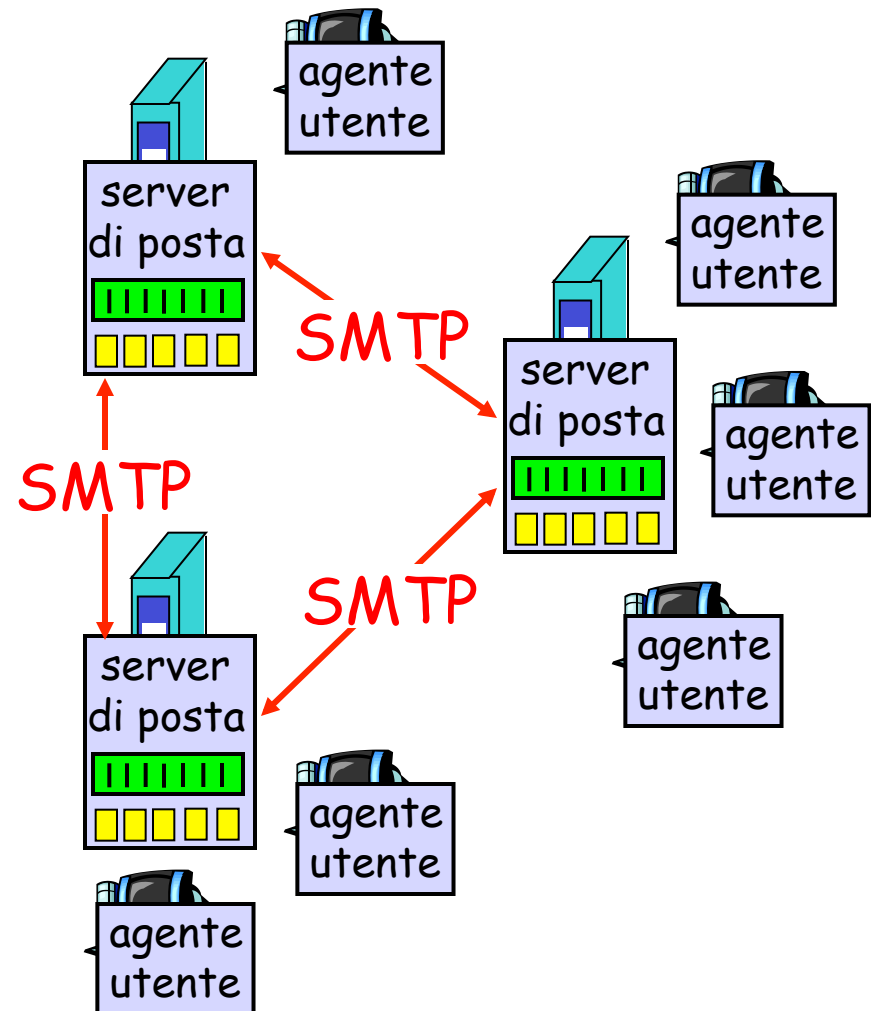
## Come comunicano gli MTA?

### Server di posta

- ❑ **Casella di posta** (*mailbox*)  
contiene i messaggi in arrivo per l'utente
- ❑ **Coda di messaggi** da trasmettere (tentativi ogni x minuti per alcuni giorni)

### Protocollo SMTP (Simple Mail Transfer Protocol)

- ❑ tra server di posta per inviare messaggi di posta elettronica
- ❑ Tra agente utente del mittente e il suo server di posta

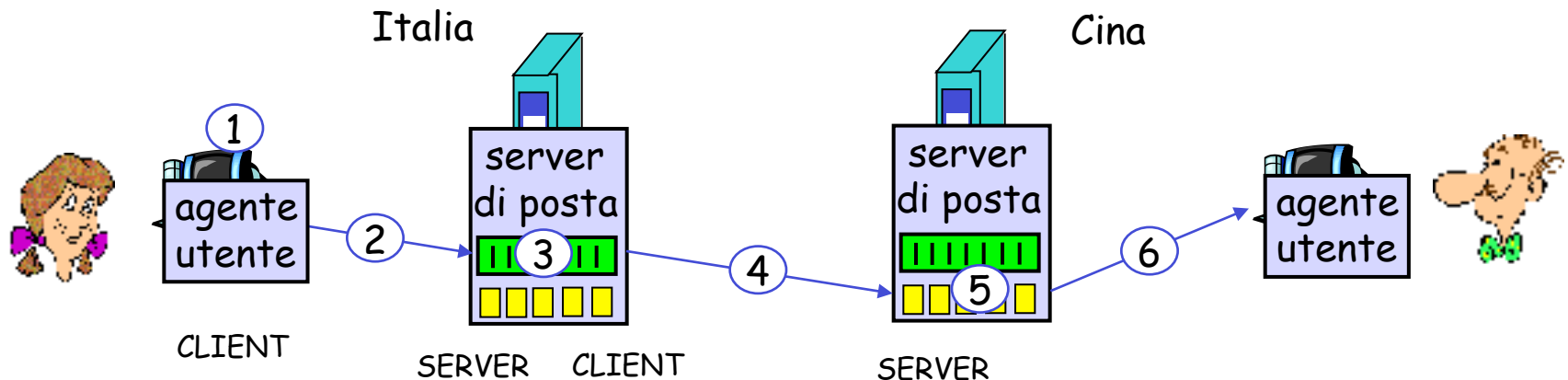


# Posta elettronica: SMTP [RFC 5321]

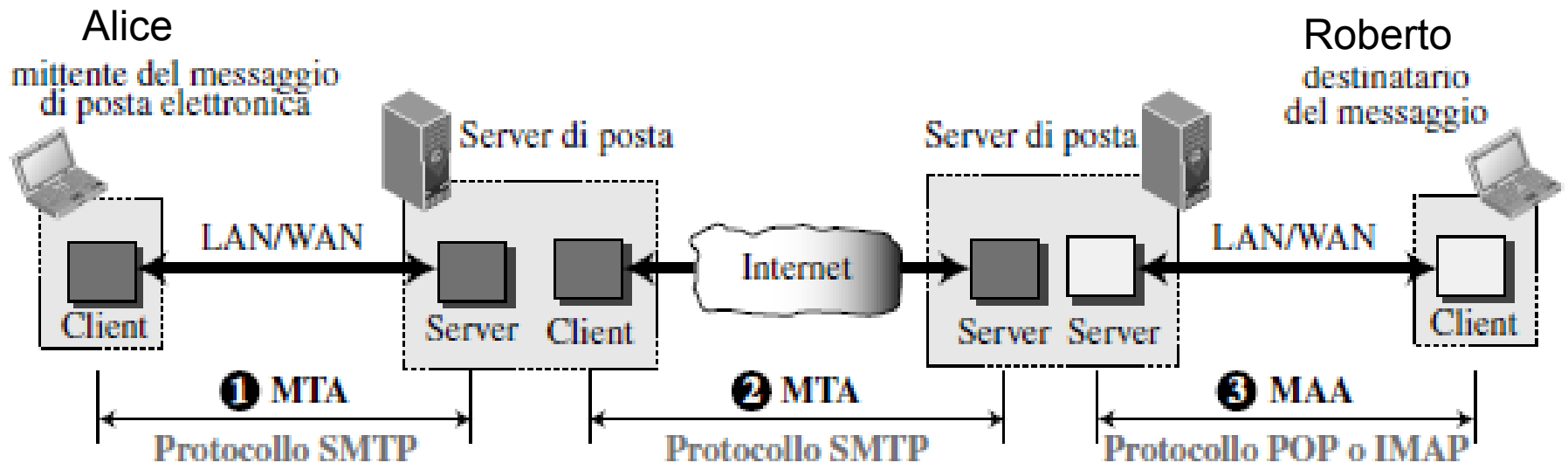
- ❑ usa TCP per trasferire in modo affidabile i messaggi di posta elettronica dal client al server, porta 25
- ❑ trasferimento diretto: dal server trasmittente al server ricevente
- ❑ tre fasi per il trasferimento
  - ❖ Handshaking
  - ❖ trasferimento di messaggi
  - ❖ chiusura
- ❑ interazione comando/risposta
  - ❖ **comandi**: testo ASCII
  - ❖ **risposta**: codice di stato ed espressione
- ❑ i messaggi devono essere nel formato ASCII

# Scenario: Alice invia un messaggio a Roberto

- 1) Alice usa il suo agente utente per comporre il messaggio da inviare "a" `rob@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 Roberto
  - 4) Il client SMTP invia il messaggio di Alice sulla connessione TCP
  - 5) Il server di posta di Roberto riceve il messaggio e lo pone nella casella di posta di Roberto
  - 6) Roberto invoca il suo agente utente per leggere il messaggio
- N.B.: nessun server intermedio



# Protocolli utilizzati





# Scambio di messaggi al livello di protocollo

- ❑ Il client SMTP (che gira sull'host server di posta in invio) fa stabilire una connessione sulla porta 25 verso il server SMTP (che gira sull'host server di posta in ricezione)
- ❑ Se il server è inattivo il client riprova più tardi
- ❑ Se il server è attivo, viene stabilita la connessione
- ❑ Il server e il client effettuano una forma di handshaking (il client indica indirizzo email del mittente e del destinatario)
- ❑ Il client invia il messaggio
- ❑ Il messaggio arriva al server destinatario grazie all'affidabilità di TCP
- ❑ Se ci sono altri messaggi si usa la stessa connessione (**connessione persistente**), altrimenti il client invia richiesta di chiusura connessione

# Esempio di interazione SMTP

Client: crepes.fr

Server: hamburger.edu

La seguente transazione inizia appena si stabilisce la connessione TCP

S: 220 hamburger.edu

C: HELO crepes.fr

S: 250 Hello crepes.fr, pleased to meet you

C: MAIL FROM: <alice@crepes.fr>

Si ripete da qui  
per mail multiple

S: 250 alice@crepes.fr... Sender ok

C: RCPT TO: <rob@hamburger.edu>

S: 250 rob@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: .

Messaggio di posta

S: 250 Message accepted for delivery

C: QUIT

S: 221 hamburger.edu closing connection

## Esempio di interazione SMTP:

- ❑ `telnet servername 25`
- ❑ Riceverete la risposta 220 dal server
- ❑ Digitate i comandi `HELO`, `MAIL FROM`, `RCPT TO`, `DATA`, `QUIT`

Questo vi consente di inviare messaggi di posta elettronica senza usare il client di posta

# SMTP: note

- ❑ SMTP usa connessioni persistenti (ripete i passi da MAIL FROM:)
- ❑ 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 e SMTP vengono utilizzati per **trasferire file** da un host all'altro
- ❑ HTTP: **pull** (gli utenti scaricano i file e inizializzano le connessioni TCP)
- ❑ SMTP: **push** (Il server di posta spedisce il file e inizializza la connessione TCP)
- ❑ HTTP: ciascun oggetto è incapsulato nel suo messaggio di risposta
- ❑ SMTP: più oggetti vengono trasmessi in un unico messaggio

# 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

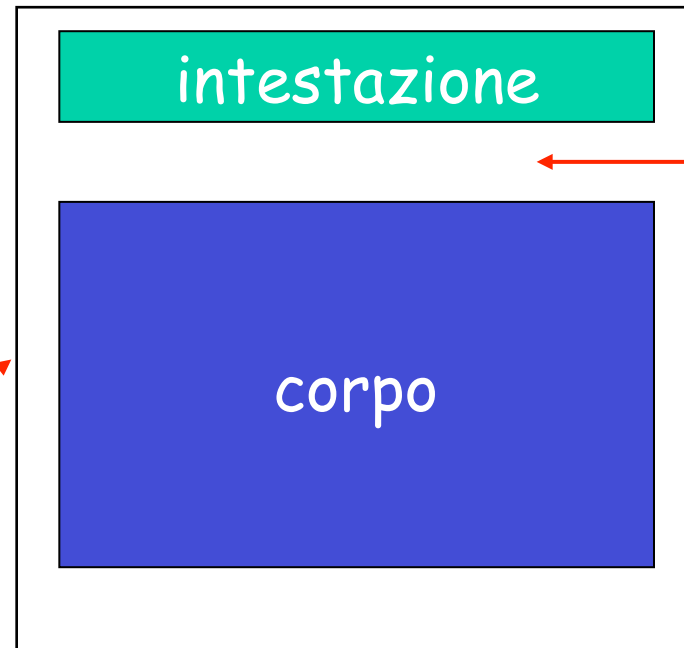
- ❖ To:
- ❖ From:
- ❖ Subject:

*differenti dai comandi SMTP!*

□ corpo

- ❖ il "messaggio", soltanto caratteri ASCII

|         |  |
|---------|--|
| To      | indirizzo di uno o più destinatari.  |
| From    | indirizzo del mittente.  |
| Cc      | indirizzo di uno o più destinatari a cui si invia per conoscenza.            |
| Bcc     | blind Cc: gli altri destinatari non sanno che anche lui riceve il messaggio. |
| Subject | argomento del messaggio.   |
| Sender  | chi materialmente effettua l'invio (ad es. nome della segretaria).           |




riga  
vuota  
(CRLF)

# Esempio formato messaggio

Behrouz Forouzan  
20122 Olive Street  
Bellbury, CA 91000

Firouz Mosharraf  
1400 Los Gatos Street  
San Louis, CA 91005



Behrouz Forouzan  
20122 Olive Street  
Bellbury, CA 91000  
Jan. 10, 2011

Oggetto: Riparazione

Caro Firouz,  
ti scrivo per informarti che,  
dopo l'ultima riparazione,  
ora la rete funziona perfettamente.

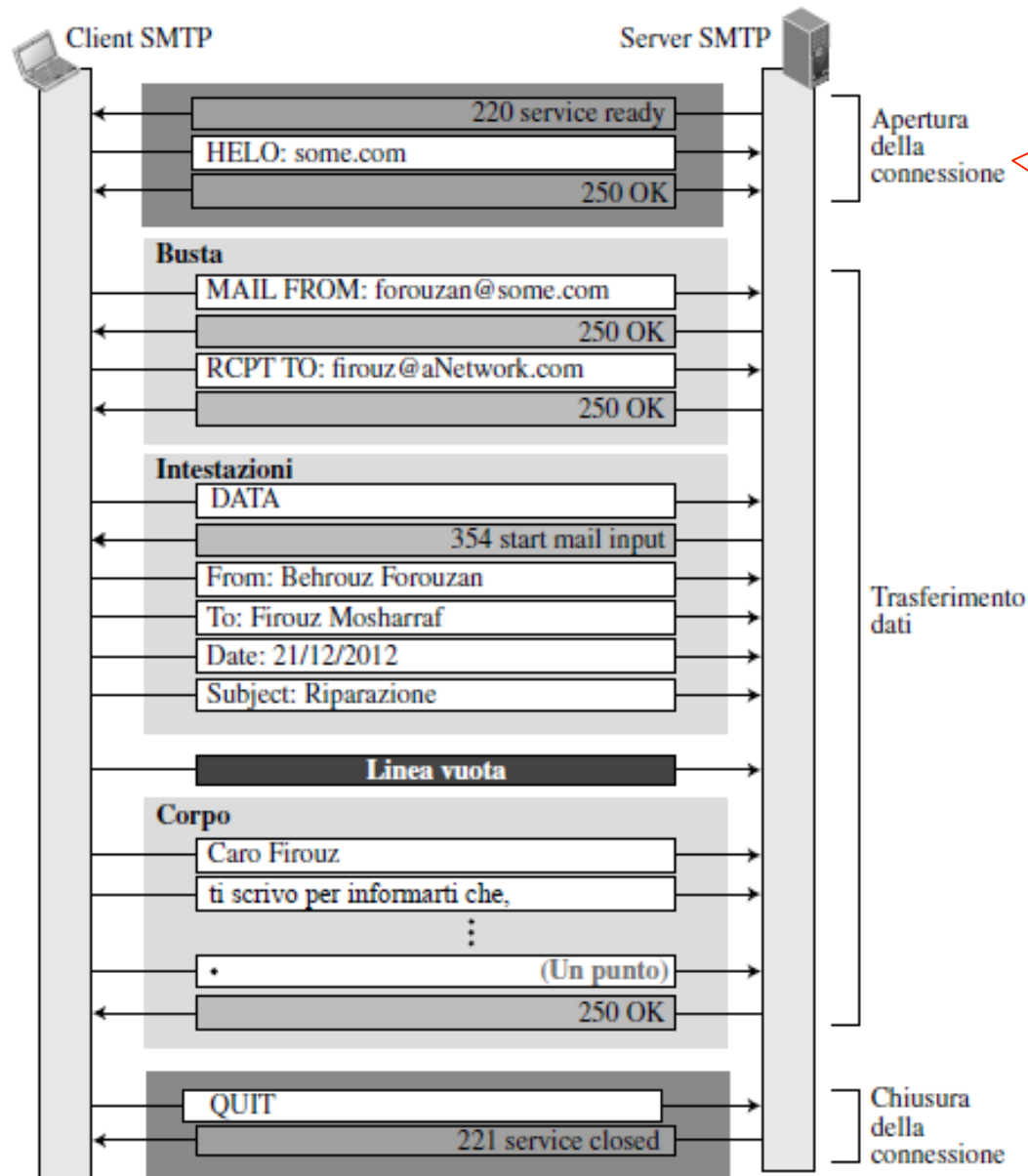
A presto,  
Behrouz.

Posta tradizionale

|              |  |           |
|--------------|--|-----------|
|              | Mail From: forouzan@some.com<br>RCPT To: mosharraf@aNetwork.com  | Busta     |
| Intestazioni | From: Behrouz Forouzan<br>To: Firouz Mosharraf<br>Date: 21/12/2012<br>Subject: Riparazione                         | Messaggio |
|              | Caro Firouz,<br>ti scrivo per informarti che,<br>dopo l'ultima riparazione,<br>ora la rete funziona perfettamente. |           |
| Corpo        | A presto,<br>Behrouz.  |           |

Posta elettronica

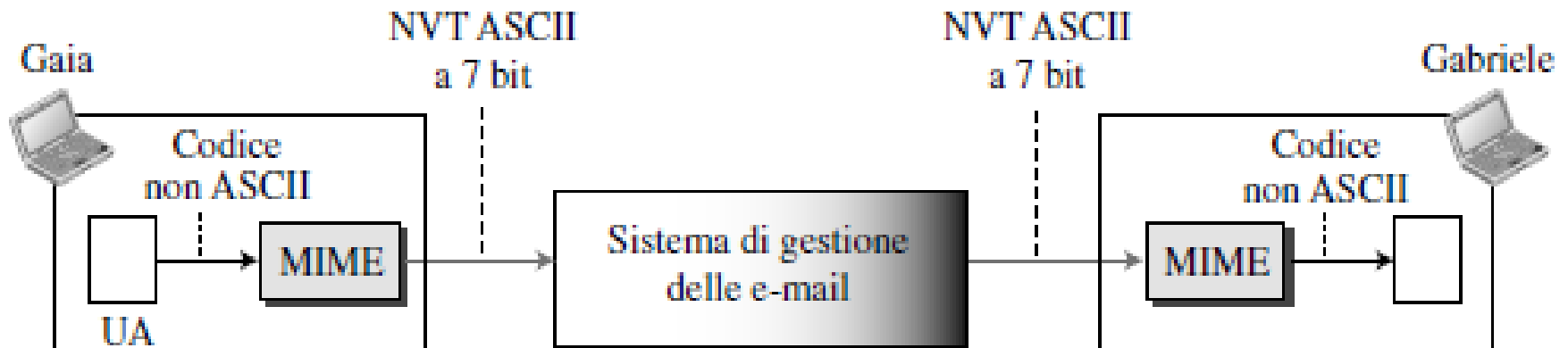
# Esempio: fasi trasferimento



E' successiva  
all'apertura della  
connessione TCP

# Protocollo MIME

- ❑ Come si possono inviare messaggi in formati non ASCII a 7 bit?
- ❑ Alcune lingue non sono supportate dai caratteri ASCII a 7 bit (francese, tedesco, giapponese, etc.)
- ❑ File binari audio e video non possono essere inviati
- ❑ Bisogna convertire i dati!
- ❑ I dati convertiti vengono consegnati al client MTA





# Formato del messaggio: estensioni di messaggi multimediali

- ❑ Per inviare contenuti diversi dal testo ASCII si usano intestazioni aggiuntive
- ❑ MIME (Multipurpose Internet Mail Extension): estensioni di messaggi di posta multimediali, RFC 2045, 2046
- ❑ Alcune righe aggiuntive nell'intestazione dei messaggi dichiarano il tipo di contenuto MIME

Versione MIME

metodo usato per  
codificare i dati in ASCII

Tipo di dati  
multimediali, sottotipo,  
dichiarazione  
dei parametri

Dati codificati  
(corpo del messaggio)

```
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
```

# Formato del messaggio ricevuto

- ❑ Un'altra classe di righe di intestazione viene inserita dal server di ricezione SMTP
- ❑ ES. Il server di ricezione aggiunge **Received:** in cima al messaggio, che specifica il nome del server che ha inviato il messaggio (from), il nome del server che lo ha ricevuto (by), e l'orario di ricezione

```
Received: from crepes.fr by hamburger.edu; 12 Oct 98 15:27:39  
GMT
```

```
From: alice@crepes.fr
```

```
To: bob@hamburger.edu
```

```
Subject: Picture of yummy crepe.
```

```
MIME-Version: 1.0
```

```
Content-Transfer-Encoding: base64
```

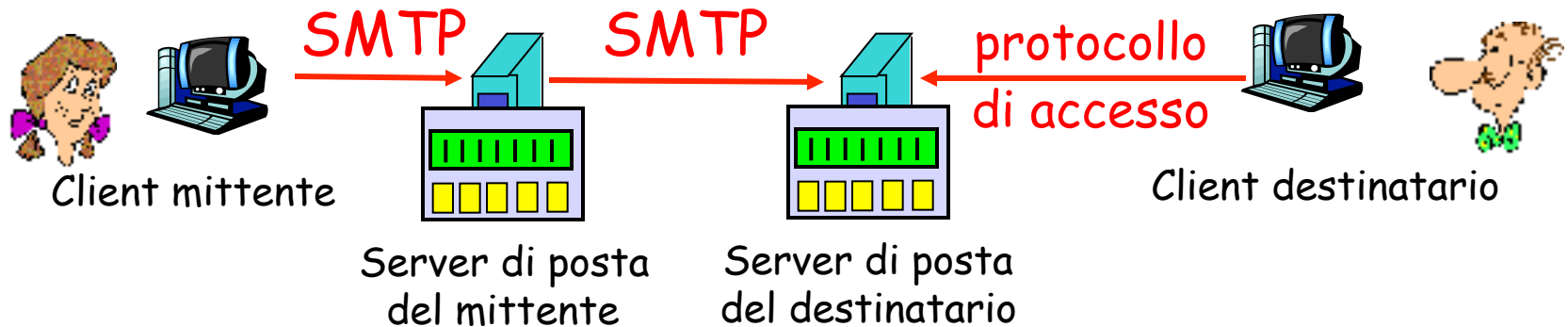
```
Content-Type: image/jpeg
```

```
base64 encoded data .....
```

```
.....
```

```
.....base64 encoded data
```

# Protocolli di accesso alla posta



- ❑ SMTP: consegna/memorizzazione sul server del destinatario
- ❑ SMTP non può essere usato dall'agente utente del destinatario perché è un protocollo push, mentre l'utente deve eseguire un'operazione pull
- ❑ Protocollo di accesso alla posta: ottenere i messaggi dal server
  - ❖ POP3: Post Office Protocol - versione 3 [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: gmail, Hotmail, Yahoo! Mail, ecc.

# Protocollo POP3

- ❑ RFC 1939
- ❑ POP3 permette al client ricevente la posta di aprire una connessione TCP verso il server di posta sulla porta 110
- ❑ Quando la connessione è stabilita si procede in 3 fasi
  1. **Autorizzazione:** L'agente utente invia nome utente e password per essere identificato
  2. **Transazione:** L'agente utente recupera i messaggi
  3. **Aggiornamento:** Dopo che il client ha inviato il `QUIT`, e quindi conclusa la connessione, vengono cancellati i messaggi marcati per la rimozione


# Protocollo POP3: comandi

## Fase di autorizzazione


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

## Fase di transazione, client:

- ❑ `list`: elenca i numeri dei messaggi
- ❑ `retr`: ottiene i messaggi in base al numero
- ❑ `dele`: cancella
- ❑ `quit`



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



```
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 e IMAP

## Ancora su POP3

- ❑ Il precedente esempio usa la modalità "scarica e cancella"
- ❑ Roberto non può rileggere le e-mail se cambia client
- ❑ Modalità "scarica e mantieni": mantiene i messaggi sul server dopo averli scaricati
- ❑ POP3 è un protocollo senza stato tra le varie sessioni
- ❑ POP3 non fornisce all'utente alcuna procedura per creare cartelle remote ed assegnare loro messaggi, l'utente può crearle solo localmente al suo computer

# IMAP: Internet Mail Access Protocol

- ❑ Cosa succede con POP3 se si accede alla mail da computer diversi?
  - ❖ Il server non mantiene le cartelle create localmente al proprio programma di posta
- ❑ 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

# IMAP: Internet Mail Access Protocol

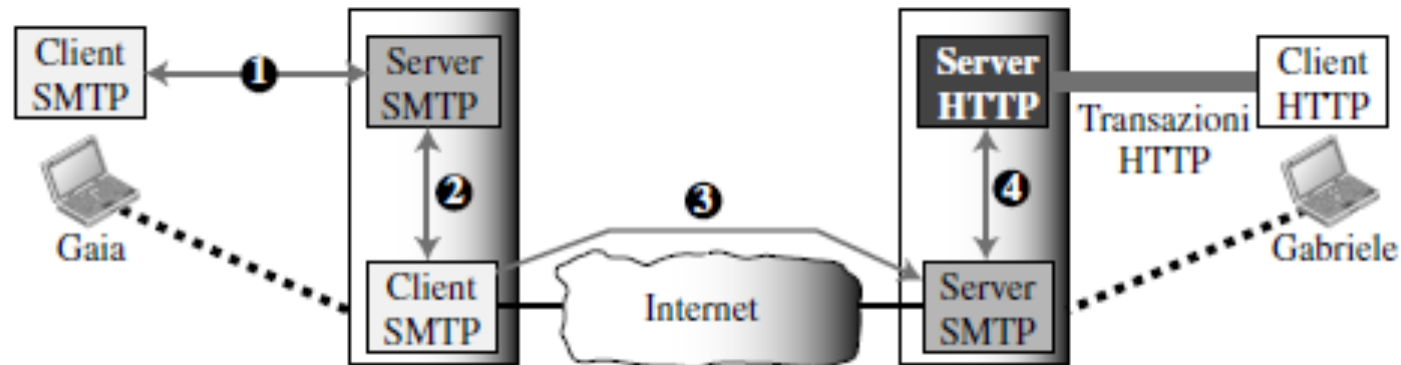
- ❑ RFC 3501
- ❑ Un server IMAP associa a una cartella (INBOX) ogni messaggio arrivato al server
- ❑ Il protocollo IMAP fornisce comandi agli utenti per
  - ❖ Creare cartelle e spostare messaggi da una cartella all'altra
  - ❖ Effettuare ricerche nelle cartelle remote
- ❑ I server IMAP conservano informazioni di stato sull'utente da una sessione all'altra
  - ❖ nomi cartelle
  - ❖ associazioni messaggi-cartelle
- ❑ IMAP presenta anche comandi che permettono agli utenti di ottenere componenti di un messaggio
  - ❖ Intestazione
  - ❖ Parte di un messaggio
  - ❖ Ricerca parole chiave



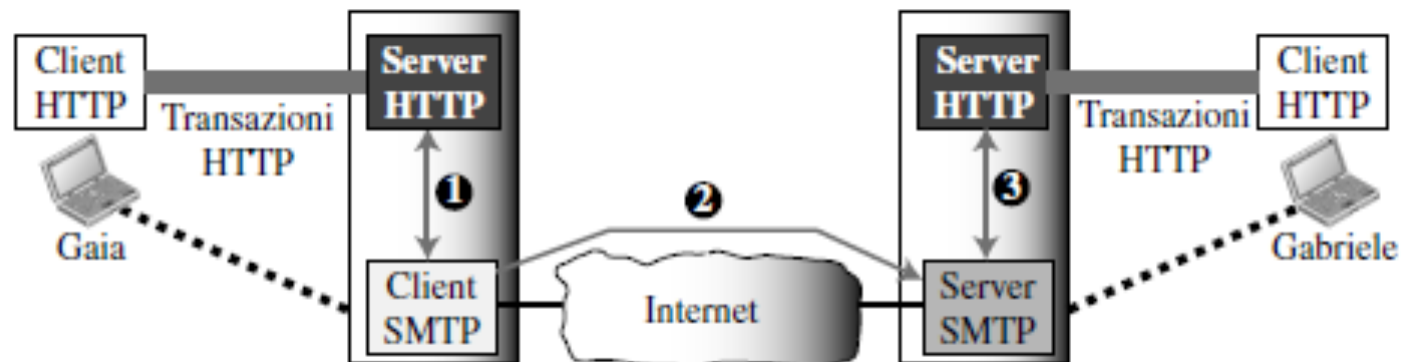
# HTTP

- ❑ Alcuni mail server forniscono accesso alla mail via web (ovvero mediante il protocollo HTTP)
- ❑ Agente utente: web browser
- ❑ L'utente comunica con il suo mailbox mediante http
- ❑ Il ricevente accede al suo mailbox mediante il protocollo HTTP
- ❑ SMTP rimane il protocollo di comunicazione tra mail server

# webmail



### Caso I: solo il destinatario utilizza HTTP



### Case II: il mittente e il destinatario utilizzano HTTP