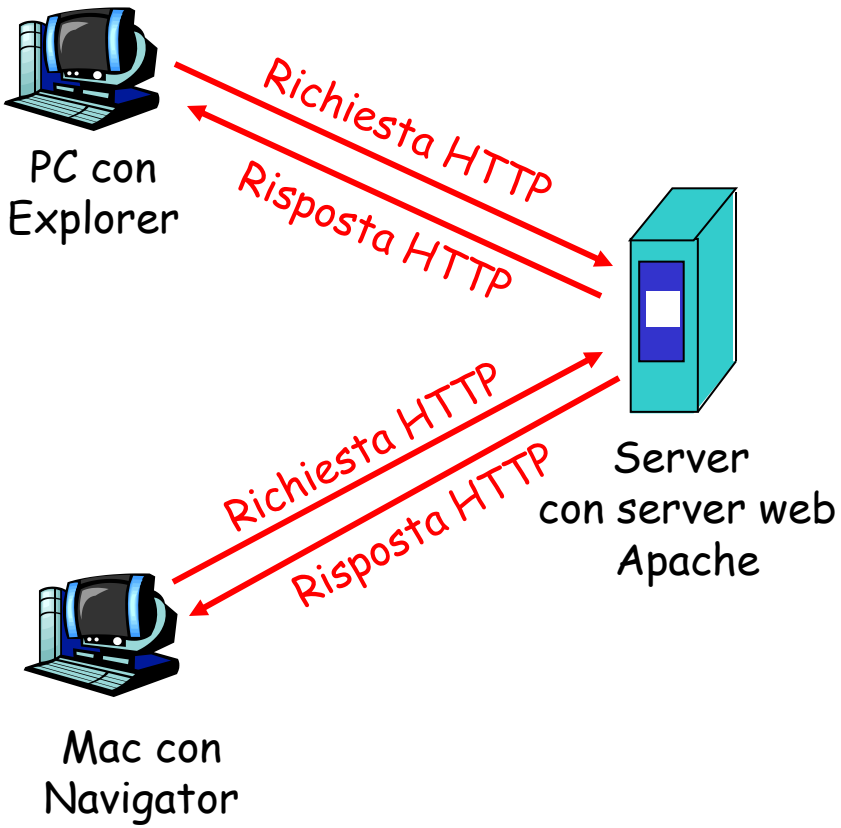
# Livello applicazione: Sessioni HTTP, FTP, Posta Elettronica

Gaia Maselli

# HTTP

## HTTP: hypertext transfer protocol

- Protocollo a livello di applicazione del Web
- Modello client/server
  - *client:* il browser che richiede, riceve, "visualizza" gli oggetti del Web
  - *server:* il server web invia oggetti in risposta a una richiesta
- HTTP definisce in che modo i client web richiedono pagine ai server web e come questi le trasferiscono ai client



# HTTP: meccanismo per mantenere lo stato

- HTTP è un protocollo "senza stato" (stateless): ogni coppia "domanda del client"/"risposta del server" è indipendente dalle altre
- Il server una volta servito il client se ne dimentica (non mantiene informazioni sulle richieste fatte)
- Il meccanismo dei Cookie rappresenta un modo per creare una sessione di richieste e risposte HTTP "con stato" (stateful)
- La sessione rappresenta un contesto più largo rispetto alla richiesta/risposta.
- Questo contesto o sessione può essere utilizzato per creare per esempio "shopping cart", in cui le selezioni dell'utente possono essere aggregate, o un giornale online può presentare contenuti personalizzati in base alle letture precedenti dell'utente

# Sessione

- Ci possono essere diversi tipi di sessione in base al tipo di informazioni scambiate e la natura del sito.
- Caratteristiche generali di una sessione:

1. Ogni sessione ha un inizio e una fine.
2. Ogni sessione ha un tempo di vita relativamente corto.
3. Sia il client che il server possono chiudere la sessione.
4. La sessione è **implicita** nello scambio di informazioni di stato.

Sessione vs. connessione

N.B. Per "sessione" NON si intende connessione persistente, ma una sessione logica creata da richieste e risposte HTTP. Una sessione può essere creata su connnessioni persistenti e non persistenti.

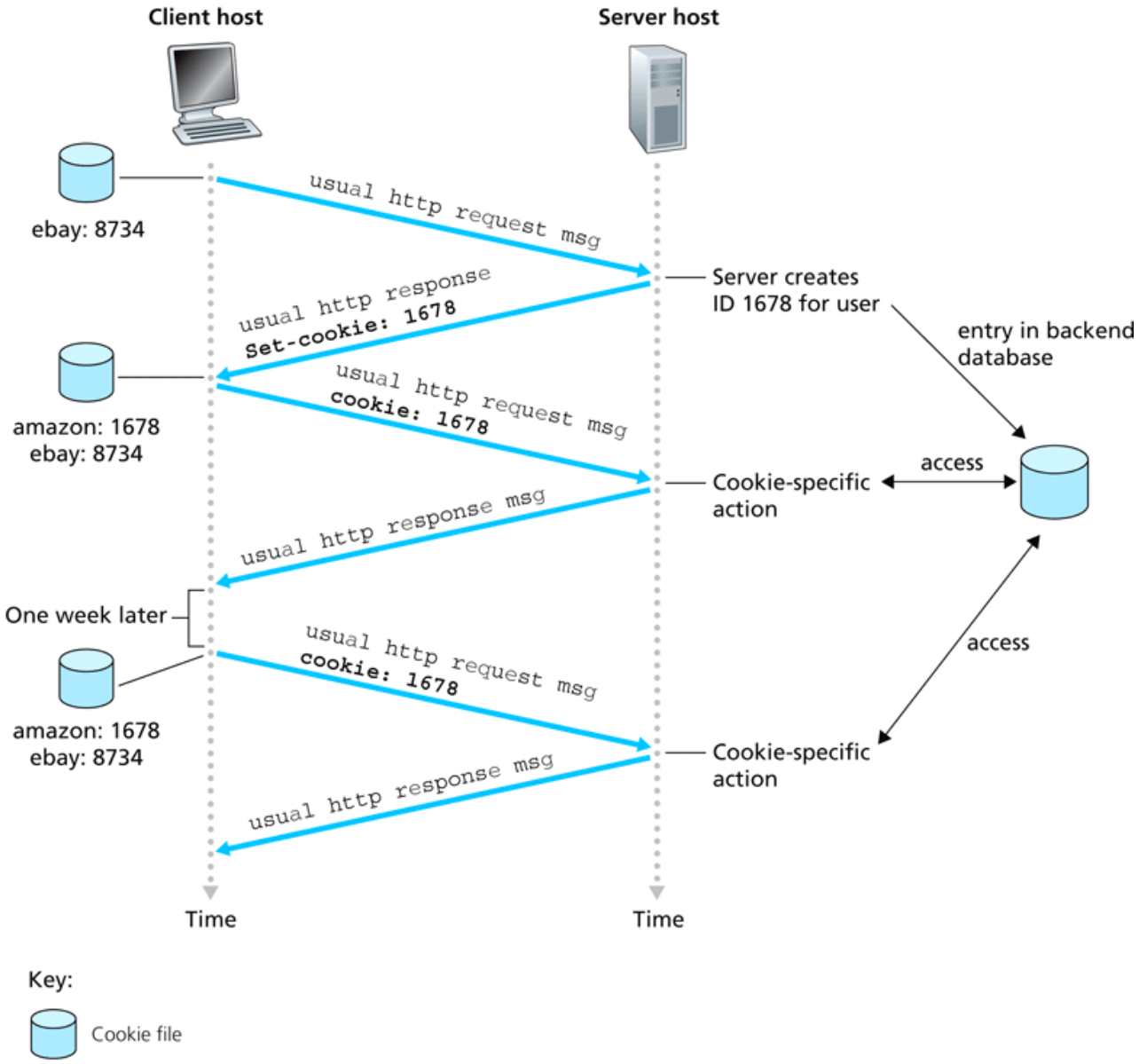
# Cookie

- Il server mantiene tutte le informazioni riguardanti il client su un file e gli assegna un identificatore (cookie) che viene fornito al client
- Il cookie inviato è un **identificatore di sessione (SID)**
- Per evitare che il cookie sia utilizzato da utenti "maligni" l'identificatore è composto da una stringa di numeri
- Il client ogni volta che manda una richiesta al server fornisce il suo identificatore (cookie)
- Esempio

```
== Server -> User Agent ==
Set-Cookie: SID=31d4d96e407aad42
== User Agent -> Server ==
Cookie: SID=31d4d96e407aad42
```

- Il server mediante il cookie fornito dal client accede al relativo file e fornisce risposte personalizzate
- Per informazioni dettagliate su cookie e sessioni guardare RFC 2109 obsoleted by RFC 2965, obsoleted by RFC 6265

# Cookie


Client host
Server host
ebay: 8734
usual http request msg
usual http response
Set-cookie: 1678
Server creates
ID 1678 for user
entry in backend
database
amazon: 1678
ebay: 8734
usual http request msg
cookie: 1678
access
Cookie-specific
action
usual http response msg
One week later
usual http request msg
cookie: 1678
access
amazon: 1678
ebay: 8734
Cookie-specific
action
usual http response msg
Time
Time
Key:
Cookie file

# Durata di un cookie

- Il server chiude una sessione inviando al client una intestazione `Set-Cookie` nel messaggio con

  `Max-Age=0`

- `Max-Age=delta-seconds`

  L'attributo `Max-Age` definisce il tempo di vita in secondi di un coockie. Dopo delta secondi il client dovrebbe rimuovere il cookie. Il valore zero indica che il cookie deve essere rimosso subito.

# Cookie e HTML

- Le intestazioni set-cookie e cookie sono contenute nei messaggi di risposta e richiesta HTTP

- Il codice HTML inviato come corpo di un messaggio di risposta HTTP contiene invece codice di personalizzazione della pagina in base al cookie

# Altra soluzione per mantenere lo stato

Per mantenere lo stato e quindi creare una sessione

- Il client mantiene tutte le informazioni sullo stato della sessione e le inserisce in ogni richiesta inviata al server
  - Metodo POST
  - Inserendole nella URL

Vantaggi

- facile da implementare
- non richiede l'introduzione di particolare funzionalità sul server

Svantaggi

- può generare lo scambio di grandi quantità di dati
- le risorse del server devono essere re-inizializzate ad ogni richiesta

# HTML

- Markup language
  - I comandi per la formattazione sono inseriti in modo esplicito nel testo
  - Esempio: il codice html

    ```
    testo in <b>grassetto</b>
    ```

    produce

    testo in **grassetto**
- HTML definisce come deve essere visualizzata una pagine web (o pagina html) e fornisce al tempo stesso formattazione contenuto
- Tag per la formattazione del testo
- Tag per altri scopi: inclusione immagine, link ipertestuale, ecc.

# Pagina HTML

## Struttura

```
<HTML>
<HEAD>
...
<TITLE>...</TITLE>
...
</HEAD >
<BODY>
...
</BODY>
</HTML>
```

## Tag marcatori

| | |
|---|---|
| *HTML* | Inizio e fine del documento |
| *HEAD* | Questa parte non viene mostrata e contiene metainformazioni sul documento (creatore, data di "scadenza", e se c'è, il titolo) |
| *TITLE* | Il titolo del documento: appare come titolo della finestra che lo contiene |
| *BODY* | Il suo contenuto viene visualizzato nella finestra |

## Alcuni tag di formattazione

| | |
|---|---|
| <B>...</B> | Grassetto (*bold*) |
| <I>...</I> | Corsivo (*italic*) |
| <Hx>...</Hx> | Intestazione (*heading*) di livello x (da 1 a 6) |
| <PRE>...</PRE> | Testo visualizzato esattamente come è scritto (*preformatted*), con spazi multipli, caratteri di fine linea, ecc. |

# Livello di applicazione

- FTP
- Posta elettronica
  - SMTP, POP3, IMAP

# File Transfer Protocol (FTP)

- Programma di trasferimento file da o verso un host remoto
- Comando per accedere ed essere autorizzato a scambiare informazioni con l'host remoto

  `ftp NomeHost`

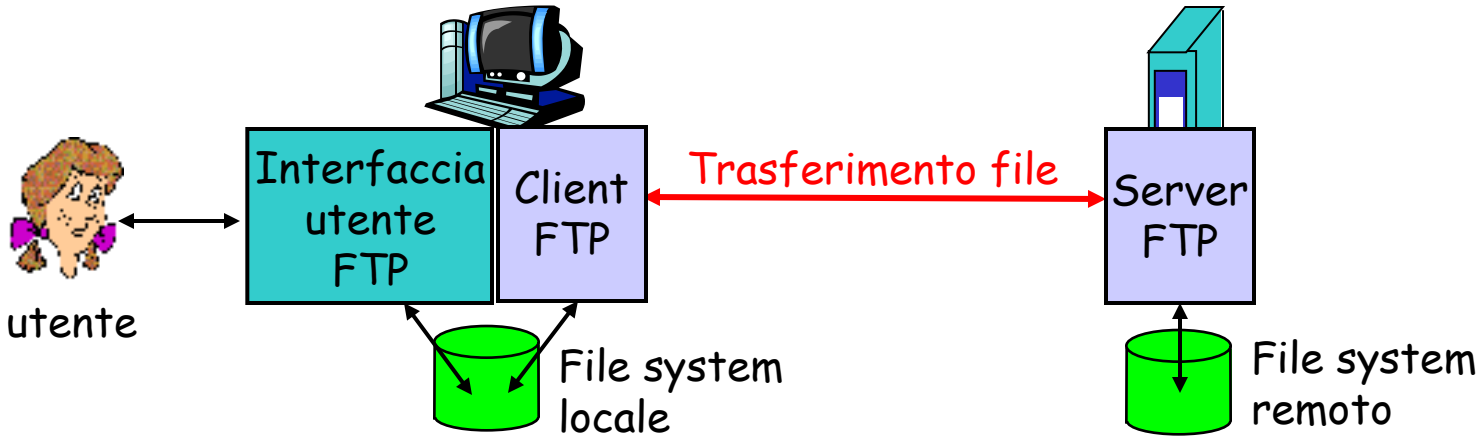
  vengono richiesti nome utente e password
- Trasferimento di un file da un host remoto

  `ftp> get file1.txt`
- Trasferimento di un file verso un host remoto

  `ftp> put file2.txt`
- Ci sono comandi per cambiare directory in locale e sull'host remoto, cancellare file, etc.

- Il protocollo FTP è descritto nella RFC 949

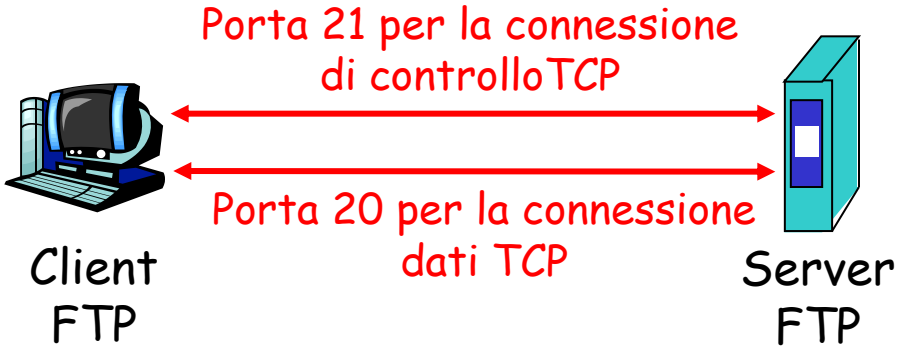
# FTP




- 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, 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: connessione di controllo

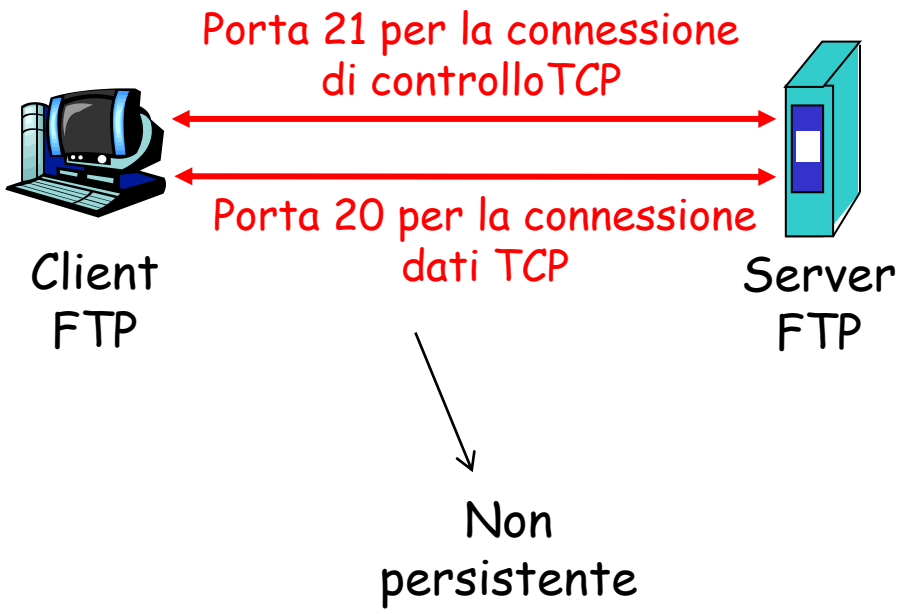
- FTP utilizza due **connessioni** TCP parallele: **controllo** e **dati**
- Connessione di controllo (porta 21) viene usata per inviare informazioni di controllo
- tutti i comandi eseguiti dall'utente sono trasferiti sulla connessione di controllo
- Esempi di informazioni trasferite sulla connessione di controllo
  - Identificativo utente
  - Password
  - Comandi per cambiare directory
  - Comandi per inviare (put) e ricevere (get) 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 informazione 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, 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




- Il server FTP mantiene lo "stato": directory corrente, autenticazione precedente

# 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

## 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`**

# Altri comandi Unix di trasferimento file

- `sftp`: secure file transfer program
- `rcp`: remote file copy
- `scp`: secure copy (remote file copy program)

# Livello di applicazione

- FTP
- Posta elettronica
  - SMTP, POP3, IMAP

# Posta elettronica

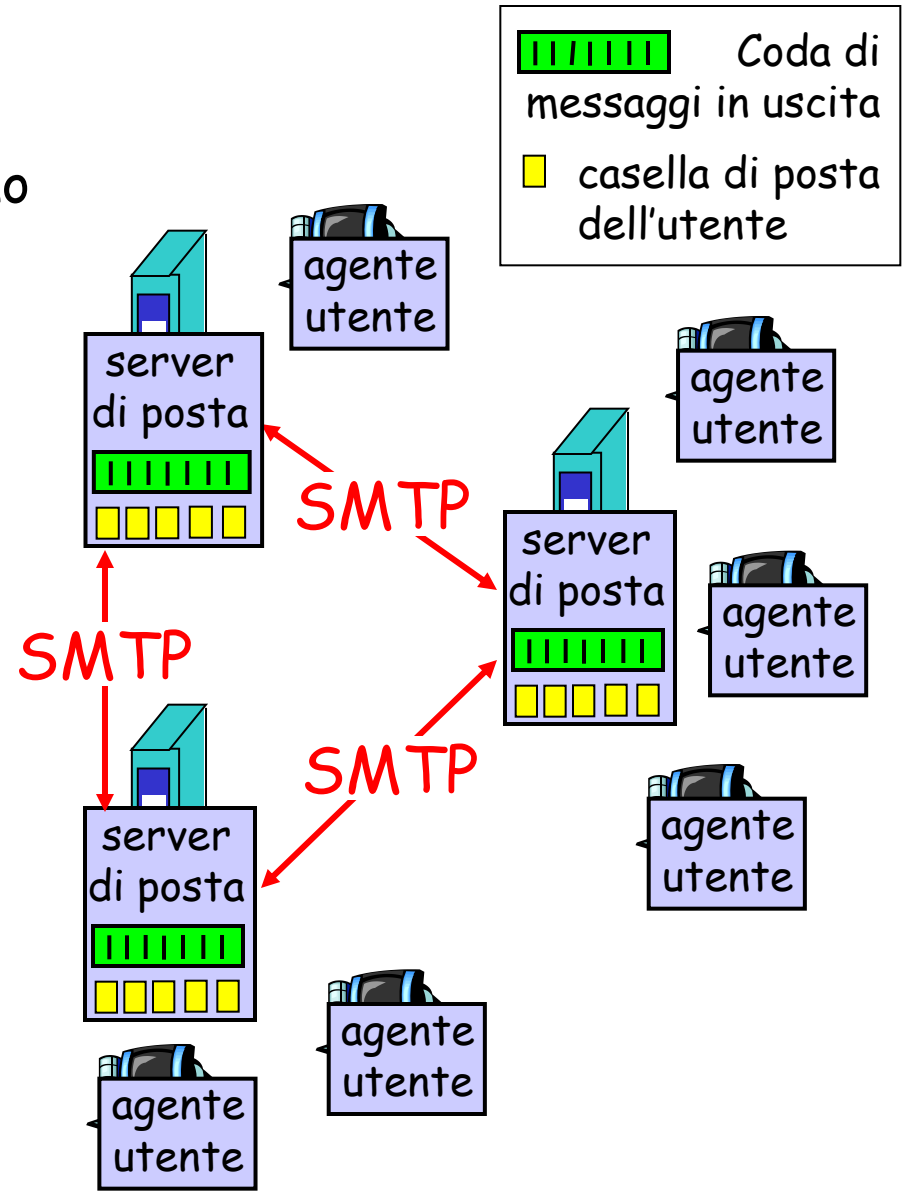
Mezzo di comunicazione asincrono

## Tre componenti principali:

- agente utente
- server di posta
- simple mail transfer protocol: SMTP

## Agente utente

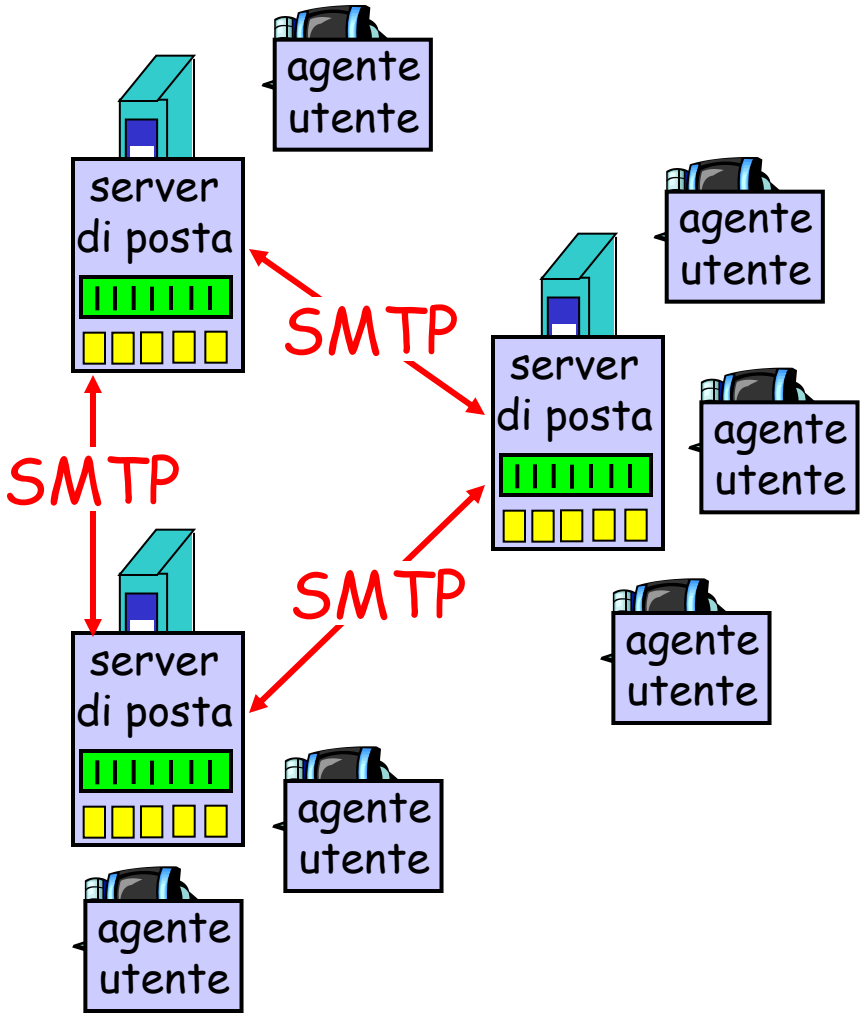
- detto anche "mail reader"
- composizione, editing, lettura dei messaggi di posta elettronica
- esempi: Eudora, Outlook, elm, Mozilla, Thunderbird
- i messaggi in uscita o in arrivo sono memorizzati sul server



# 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 (tentativi ogni x minuti per alcuni giorni)
- Protocollo SMTP tra server di posta per inviare messaggi di posta elettronica
  - client: server di posta trasmittente
  - "server": server di posta ricevente



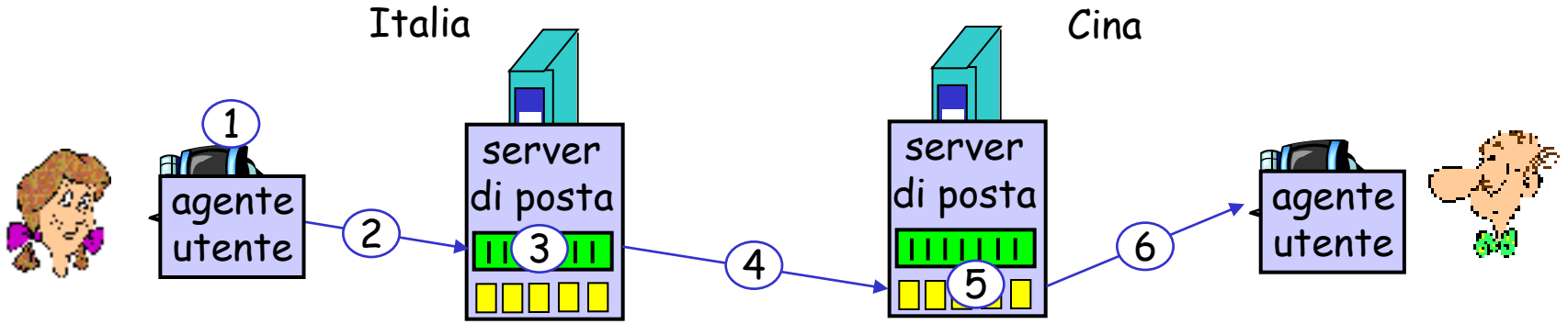
# Posta elettronica: SMTP [RFC 2821]

- 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 fasi 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

# 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



# 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>
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: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

Si ripete da qui per mail multiple

Messaggio di posta

# 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 (lettore)

# 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

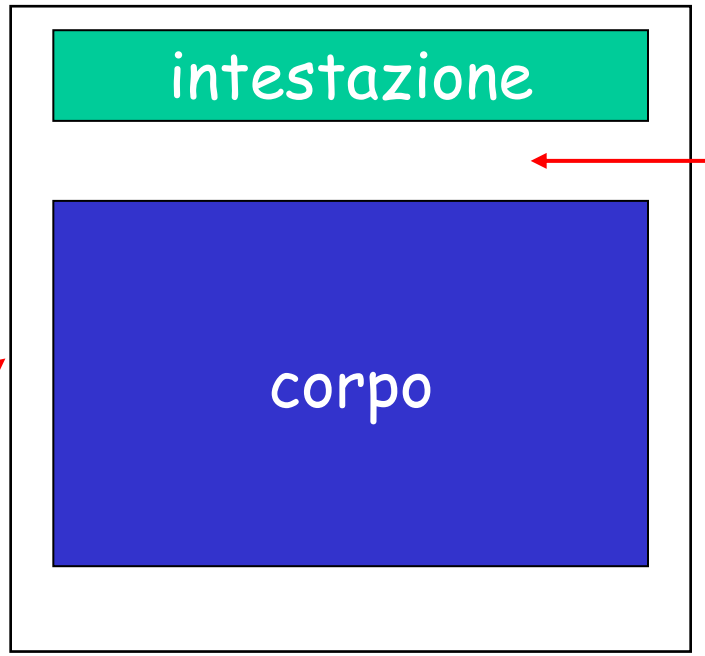
SMTP: protocollo per scambiare messaggi di posta elettronica

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

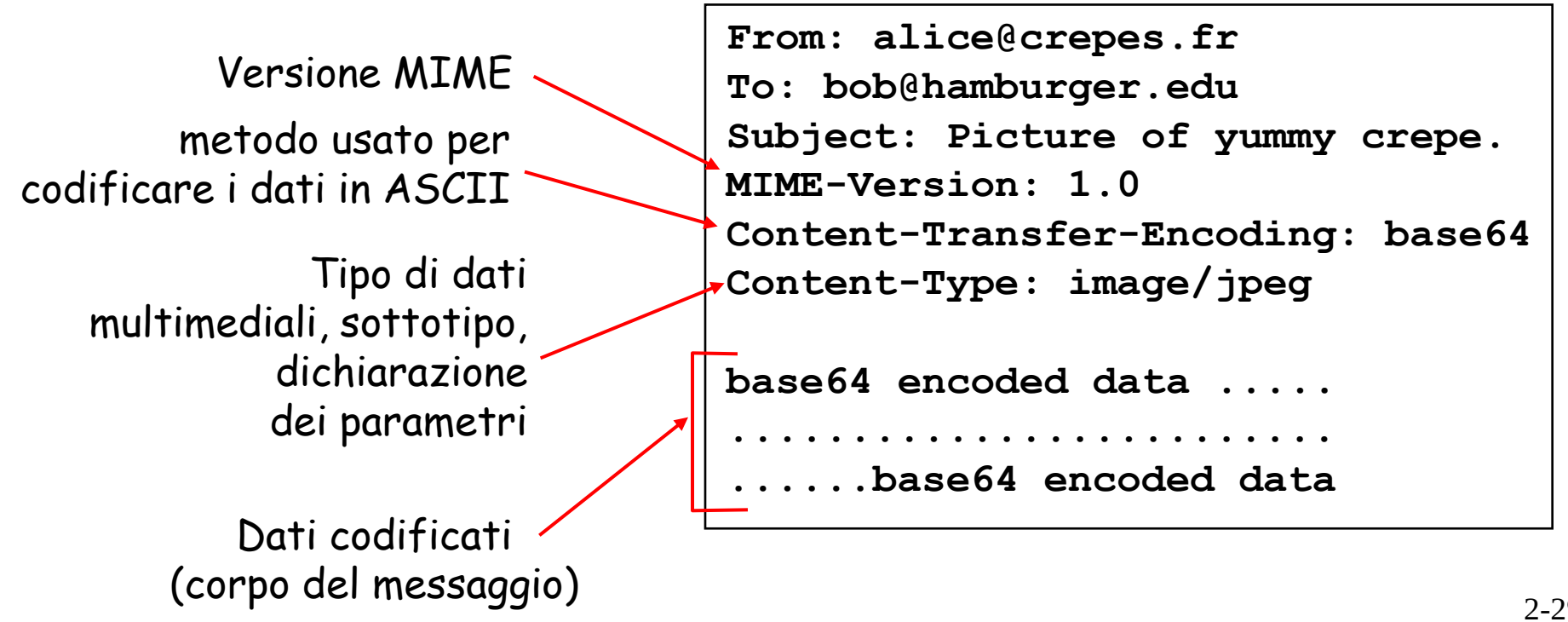
| | |
|---|---|
| To | indirizzo di uno o più destinatari. |
| From | indirizzo del mittente. |
| Cc | indirizzo di uno o più destinatari a cui si invia per conoscenza. |
| Bcc | blind Cc: gli altri destinatari non sanno che anche lui riceve il messaggio. |
| Subject | argomento del messaggio. |
| Sender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




Nota: Come si possono inviare messaggi con formati testuali non ASCII o immagini?

# Formato del messaggio: estensioni di messaggi multimediali

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



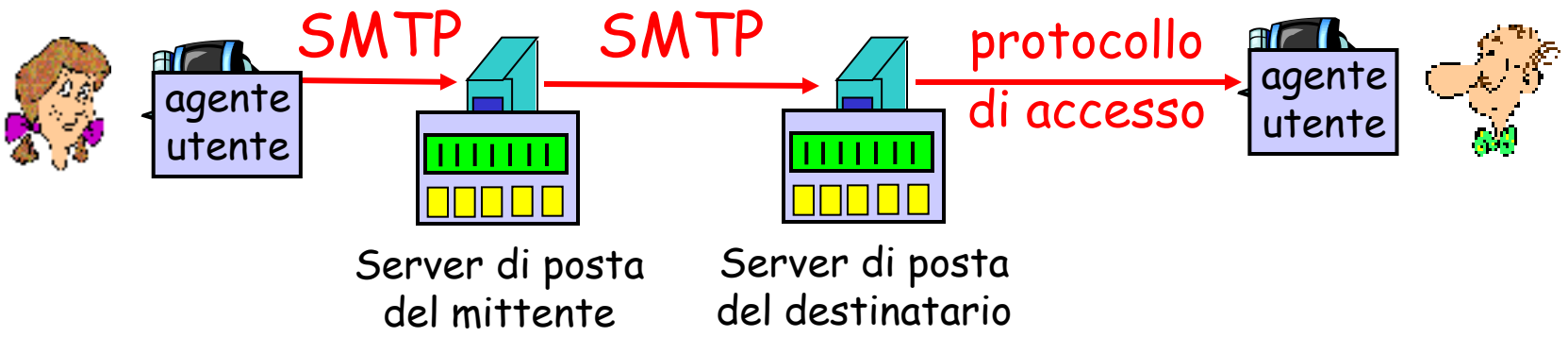
# Formato del messaggio ricevuto

- 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 all'agente utente (client) di aprire una connessione TCP verso il server di posta (il server) 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": copia i messaggi su più client
- POP3 è un protocollo senza stato tra le varie sessioni
- POP3 non fornisce all'utente alcuna procedura per creare cartelle remote ed assegnar loro messaggi

## 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