Livello applicativo

Obiettivi generali:

- Aspetti concettuali/ implementativi dei protocolli applicativi
  - Paradigma client server
  - Modelli dei servizi

Obiettivi specifici:

- Protocolli specifici:
  - http
  - ftp
  - smtp
  - pop
  - dns
  - Programmazione di applicazioni
    - Uso dei socket

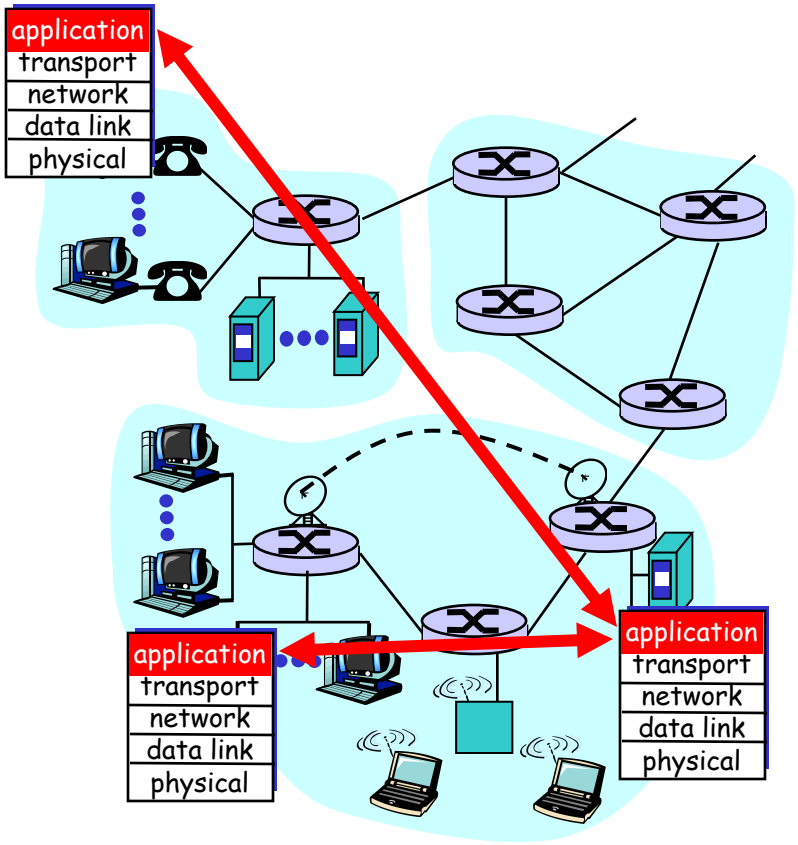
Applicazioni e protocolli applicativi

Applicazione: processi ditribuiti in comunicazione

- In esecuzione su host remoti
- Si scambiano messaggi per eseguire l'applicazione
- Es., posta, FTP, WWW

Protocolli applicativi

- Costituiscono una parte di ogni applicazione
- Definiscono il formato dei messaggi scambiati e il loro significato (azioni)
- Usano i servizi degli strati inferiori



Applicazioni: terminologia essenziale

- Un processo è un programma in esecuzione su un host
- Sullo stesso host i processi comunicano mediante meccanismi definiti dal SO.
- Processi in esecuzione su host diversi comunicano mediante meccanismi definiti dal protocollo dello strato di applicazione (application layer protocol)

- Un agente utente (user agent) è un'interfaccia tra l'utente e l'applicazione di rete.
  - Browser Web
  - E-mail: lettore di posta
  - streaming audio/video: lettore di file audio/video

Paradigma client-server

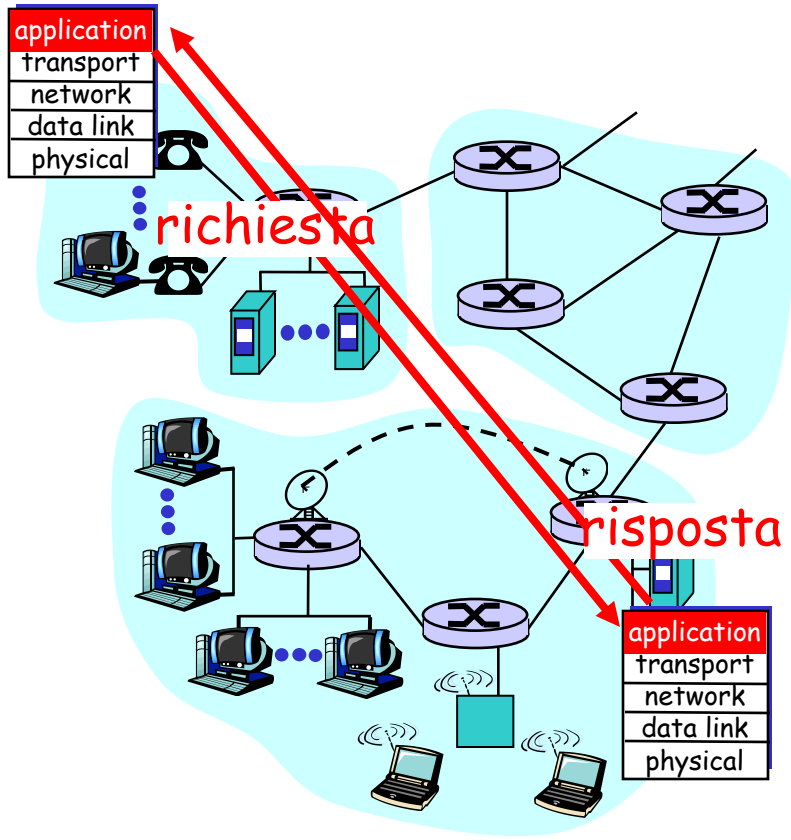
App. di rete tipica consiste di due parti: *client* e *server*

Client:

- Inizia il dialogo col server ("speaks first")
- Di solito richiede un servizio
- Nel caso del Web, il client è integrato nel browser

Server:

- Fornisce il servizio al client, su richiesta
- Es., un Web server invia una pagina Web richiesta, un mail server accede alla casella di posta elettronica



Protocolli di livello applicativo: servizi dagli strati inferiori e identificazione

API: Application Programming Interface

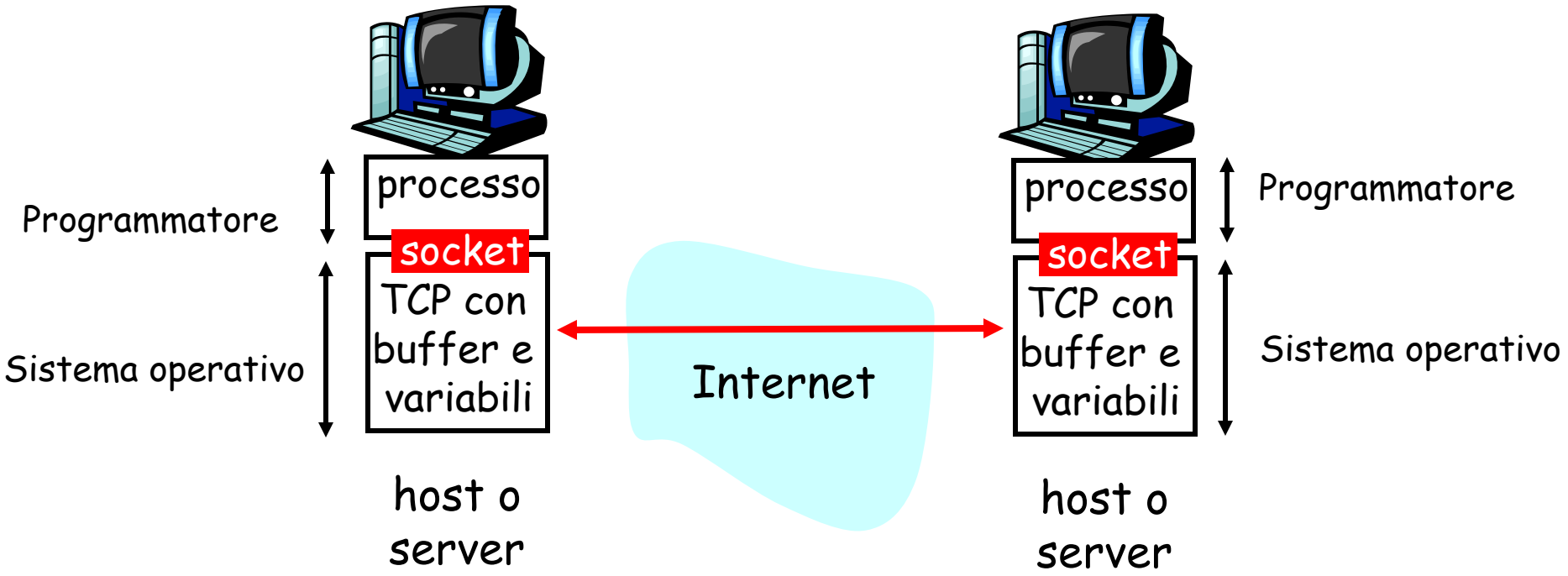
- Definisce l'interfaccia tra applicazione e strato di trasporto
- Socket: API Internet
  - Due processi (applicazione nel modello client server) comunicano inviando/leggendo dati nel/dal socket

D: come può un processo "identificare" quello con cui intende comunicare?

- Indirizzo IP dell'host su cui l'altro processo è in esecuzione
- Numero di porta (port number) - permette all'host ricevente di identificare il processo locale destinatario del messaggio

... di più in seguito.

Socket: funzionamento di base



Requisiti delle applicazioni

Perdita (Data loss)

- Alcune app.ni (es., audio) sono tolleranti (fino a un certo punto)
- Altre (es., FTP, telnet) richiedono affidabilità totale

Ritardo

- Alcune applicazioni (es., telefonia Internet, giochi interattivi in rete) richiedono una banda minima per funzionare con qualità sufficiente

Banda

- Alcune app.ni (soprattutto multimediali) richiedono una banda minima
- Altre (dette "elastiche") usano la banda a disposizione

Nota: alcuni requisiti sono determinati da esigenze percettive umane (es. ritardo nella telefonia Internet)

Transport service requirements of common apps

Application	Data loss	Bandwidth	Time Sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	loss-tolerant	elastic	no
real-time audio/video	loss-tolerant	audio: 5Kb-1Mb video:10Kb-5Mb	yes, 100's msec
stored audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	few Kbps up	yes, 100's msec
financial apps	no loss	elastic	yes and no

Servizi offerti dai protocolli di trasporto Internet

Servizio TCP :

- *Orientato alla connessione:* richiesto "setup" tra client e server
- *Trasporto affidabile (reliable transfer)* tra processi mittente e ricevente
- *Controllo di flusso) flow control:* il mittente non sommerge il ricevente
- *Controllo della congestione (congestion control):* si limita il mittente quando la rete è sovraccarica
- *Non offre:* garanzie di banda e ritardo minimi

Servizio UDP :

- Trasporto non affidabile tra processi mittente e ricevente
- Non offre: connessione, affidabilità, controllo di flusso, controllo di congestione, garanzie di ritardo e banda

D: perché esiste UDP? Può essere conveniente per le applicazioni (si vedrà più avanti)

Applicazioni Internet: loro protocolli e protocollo di transporto usato

Applicazione	Protocollo applicativo	Protocollo di trasporto usato
e-mail	smtp [RFC 821]	TCP
remote terminal access	telnet [RFC 854]	TCP
Web	http [RFC 2068]	TCP
file transfer	ftp [RFC 959]	TCP
streaming multimedia	proprietario (es. RealNetworks)	TCP o UDP
remote file server	NFS	TCP o UDP
Internet telephony	proprietary (e.g., Vocaltec)	tipicamente UDP

WWW: terminologia essenziale

- Pagina Web:
  - È costituita da "oggetti" (di solito: pagina HTML iniziale+oggetti indirizzati)
  - È indirizzata da una URL
- URL (Uniform Resource Locator)
  - Identifica un oggetto nella rete e specifica il modo per accedere ad esso
  - Ha due componenti: nome dell'host e percorso nell'host:

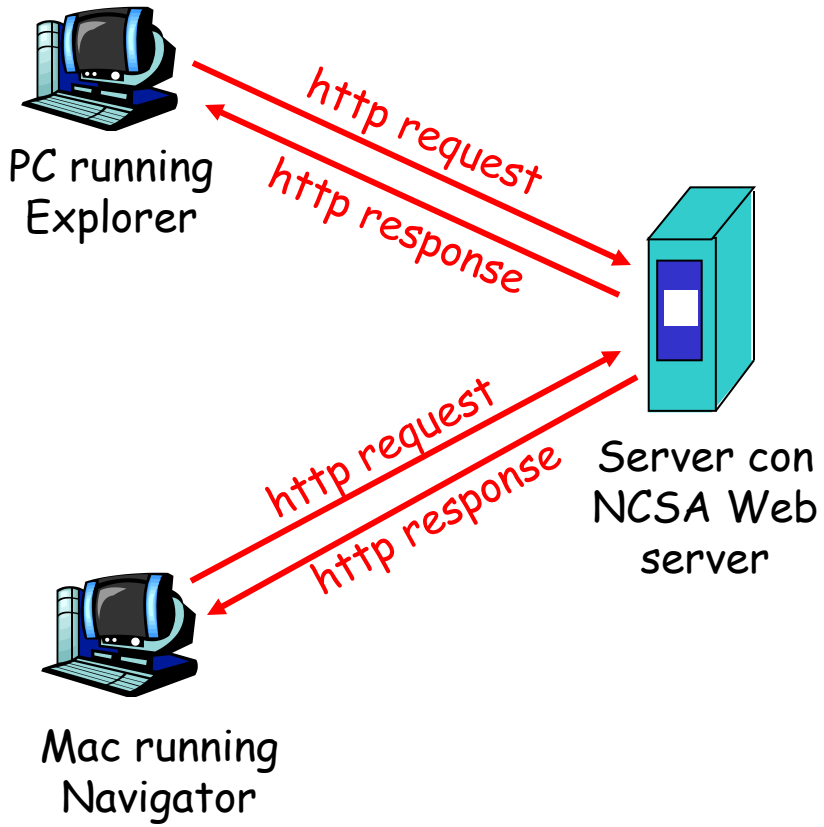
www.someSchool.edu/someDept/pic.gif

- Uno user agent per il Web è detto browser:
  - MS Internet Explorer
  - Google Chrome
  - Mac OS Safari
- un server per il Web è detto Web server:
  - Apache (pubblico dominio)
  - MS Internet Information Server

Il Web: protocollo http

http: hypertext transfer protocol

- Protocollo di livello applicativo per il Web
- Usa il modello client/server
  - *client:* browser che richiede, riceve e "mostra" oggetti Web
  - *server:* Web server che invia oggetti in risposta alle richieste
- http1.0: RFC 1945
- http1.1: RFC 2068



Il protocollo http (cont.)

http: usa TCP:

- Il client inizia una connessione TCP (crea un socket) verso il server sulla porta 80
- Il server accetta la connesione TCP dal client
- Vengono scambiati messaggi http (messaggi del protocollo di livello applicativo) tra il browser (client http) e il Web server (server http)
- La connessione TCP è chiusa

http è "stateless"

- Il server non mantiene informazione sulle richieste precedenti del client

I protocolli che mantengono informazione di stato sono complessi (es. TCP) !

Esempio http

L' utente accede alla URL
www.someSchool.edu/someDepartment/home.index

(contiene testo e i riferimenti a 10 immagini jpeg)

1a. Il client http inizia una connessione TCP verso il server (processo) http sull' host www.someSchool.edu. La porta 80 è quella standard (default) per i server http.

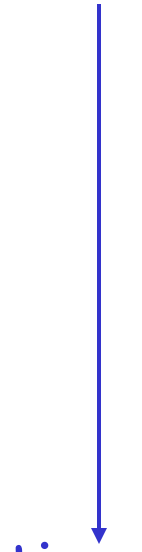
1b. Il server http presso l' host www.someSchool.edu è "in ascolto" sulla porta 80. "Accetta" la richiesta di connessione e ne dà conferma al client

2. Il client http invia un messaggio di richiesta http (*request message*) contenente la URL

3. Il server http riceve il messaggio di richiesta, costruisce un messaggio di rispota (*response message*) contenente l'oggetto richiesto (someDepartment/home.index), inoltra il messaggio nel socket

Tempo

Esempio http (cont.)



4. Il server http chiude la connessione TCP.

5. Il client http riceve il messaggio di risposta contenente il file html, visualizza la pagina html. Analizzando il file html, il browser trova i riferimenti a 10 oggetti jpeg

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

Connessioni persistenti e non-persistenti

Non-persistente

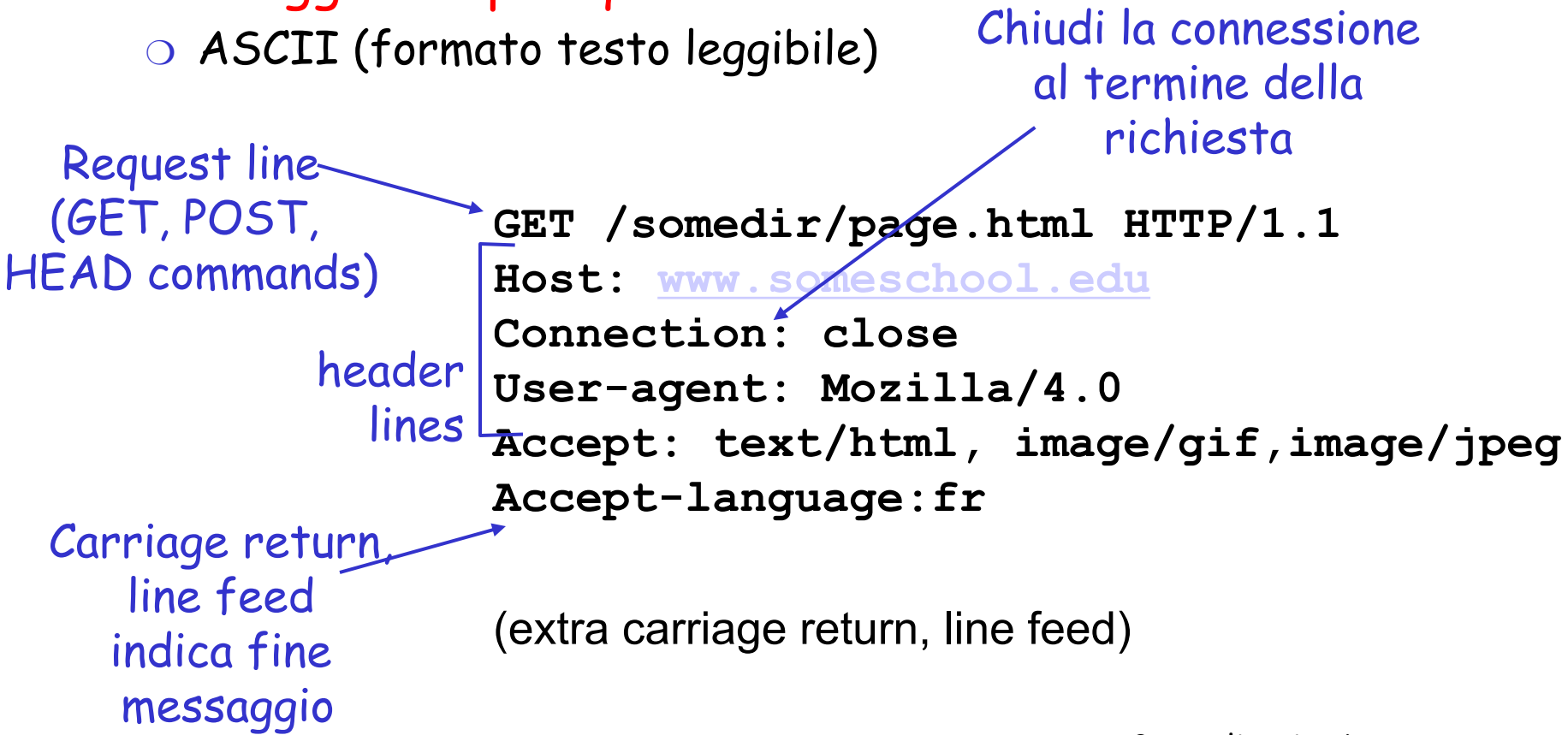
- HTTP/1.0
- Il server analizza la richiesta, risponde e chiude la connessione TCP
- 2 RTTs per ricevere ciascun oggetto
- Ogni oggetto subisce lo "slow start" TCP
- E' possibile parallelizzare le richieste agli oggetti di una pagina

Persistente

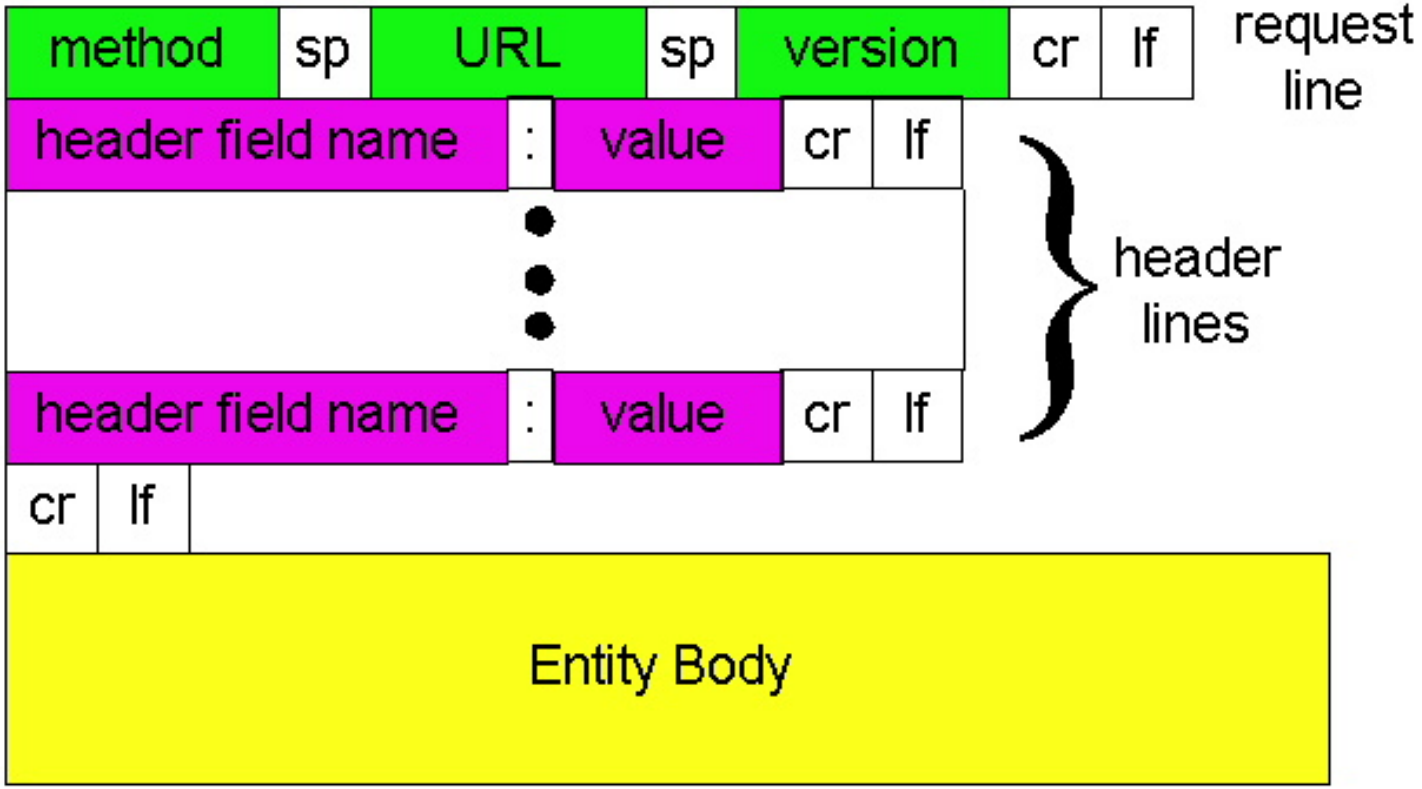
- default per HTTP/1.1
- Sulla stessa connessione TCP : il server analizza una richiesta, risponde, analizza la richiesta successiva,..
- Il client invia richieste per tutti gli oggetti appena riceve la pagina HTML iniziale.
- Si hanno meno RTTs e slow start.
- Connessione incanalata:
  - non si attende la risposta alla richiesta prededente prima di inviare la successiva.
  - Richiesta successiva a ridosso della precedente cosi' come le risposte del server.
  - Solo 2 RTT per ottenere un insieme di oggetti

Formato dei messagi http

- Due tipi di messaggi http: *request, response*
- Messaggio http request:
  - ASCII (formato testo leggibile)



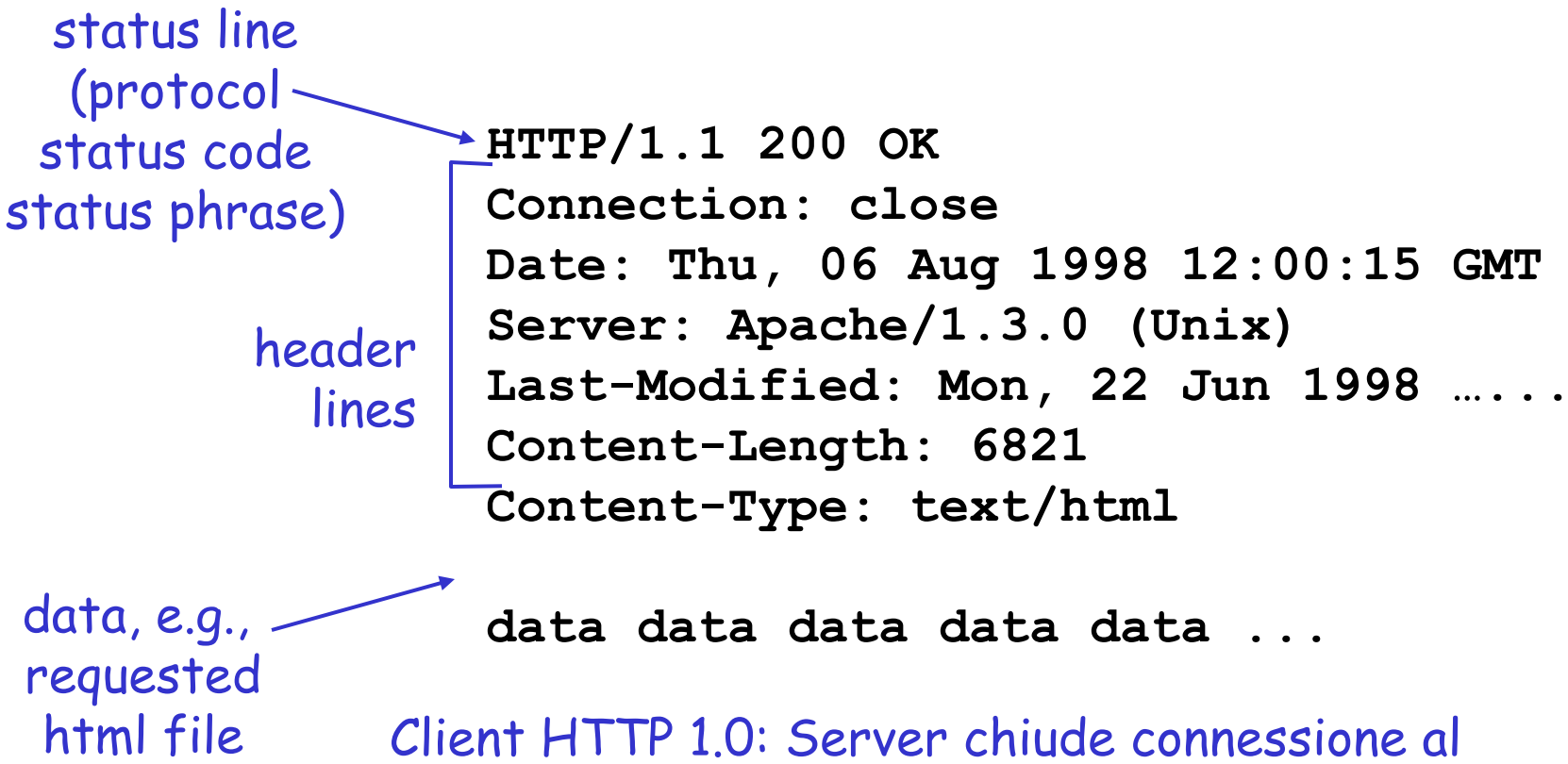
Message http request : formato generale



Message http request : formato generale

- Metodo post:
  - Usato quando l'utente compila una form. Il contenuto dei campi della forms sono disposti nell'Entitiy Body
  - Il comando richiede una pagina Web il cui contenuto dipende dalle informazioni nel campo body
  - Ex: Query inviata ad un motore di ricerca
- Metodo Head:
  - Simile al metodo get ma viene restituito solo l'Head della pagina Web
  - Spesso usato in fase di debugging

Formato del messaggio http response



Client HTTP 1.0: Server chiude connessione al termine della richiesta

Client HTTP 1.1: mantiene aperta la connessione oppure chiude se Connection: close

Risposta: codici di stato

Prima riga del messaggio di risposta server->client.

Alcuni esempi:

200 OK

- Successo, oggetto richiesto più avanti nel messaggio

301 Moved Permanently

- L'oggetto richiesto è stato spostato. Il nuovo indirizzo è specificato più avanti (Location:)

400 Bad Request

- Richiesta incomprensibile al server

404 Not Found

- Il documento non è stato trovato sul server

505 HTTP Version Not Supported

Prova (client)

1. Telnet verso un Web server:

```
telnet www.dis.uniroma1 80
```

Apre connessione TCP verso la porta 80 (default) prsso www.dis.uniroma1.it. Tutto quanto viene digitato è inviato alla porta 80 di www.dis.uniroma1.it

2. Si digita una richiesta http GET:

```
GET /~leon/index.html HTTP/1.0
```

Digitando ciò (carriage return due volte), si invia una richiesta GET al server http

3. Si osservi la risposta!

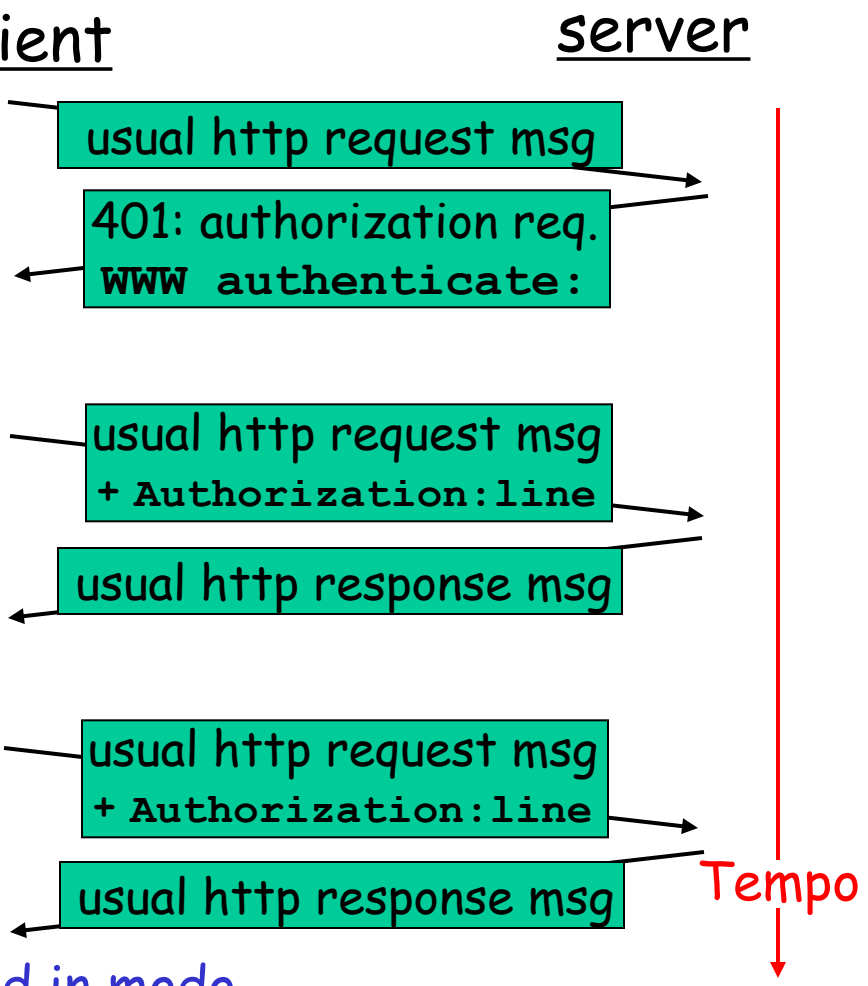
Autenticazione

Obiettivo: controllare l' accesso ai documenti sul server

- stateless: il client deve autenticare ogni richiesta
- autenticazione: tipicamente log e password
  - `authorization:` riga nell'header del messaggio di richiesta
  - Senza autenticazione il server rifiuta la connessione

    `WWW authenticate:`

    Nell' header



Il browser memorizza nome & password in modo che l'utente non debba digitarli ogni volta.

Cookie

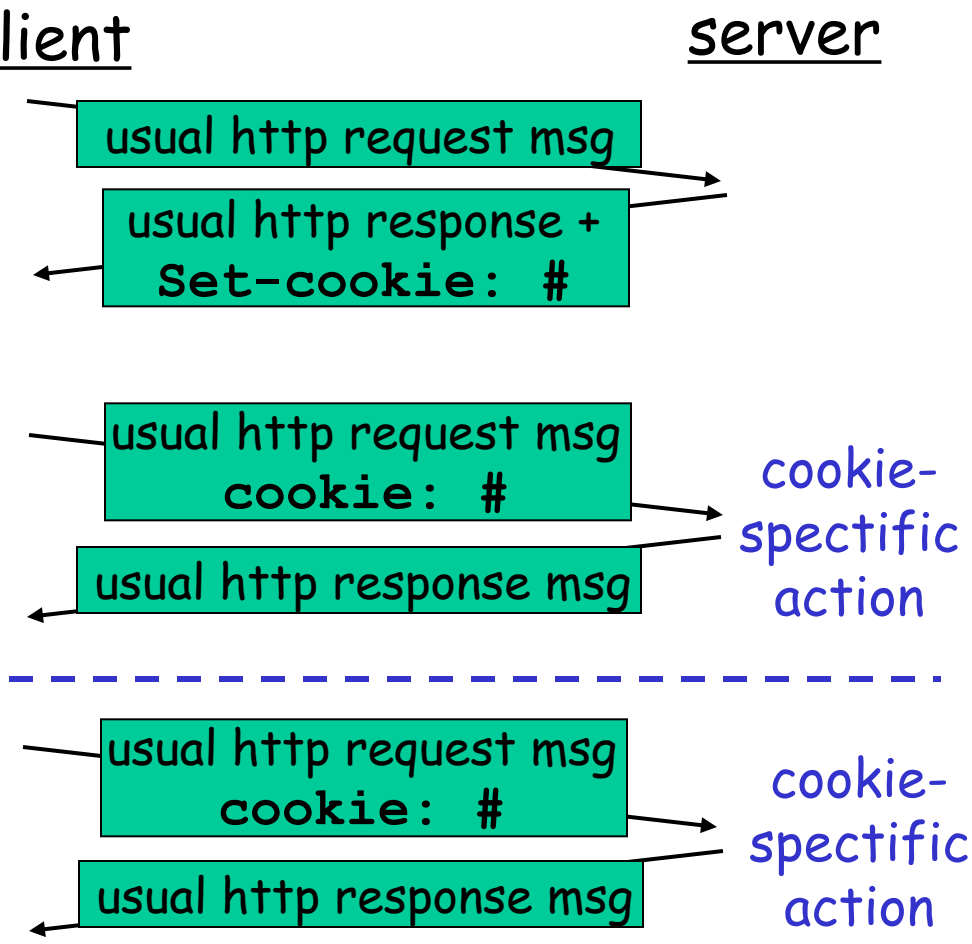
- Il server invia un "cookie" al client con la risposta

  `Set-cookie: 1678453`

- Il client presenta il cookie in accessi successivi

  `cookie: 1678453`

- Il server controlla il cookie presentato
  - Autenticazione
  - Traccia delle preferenze dell'utente

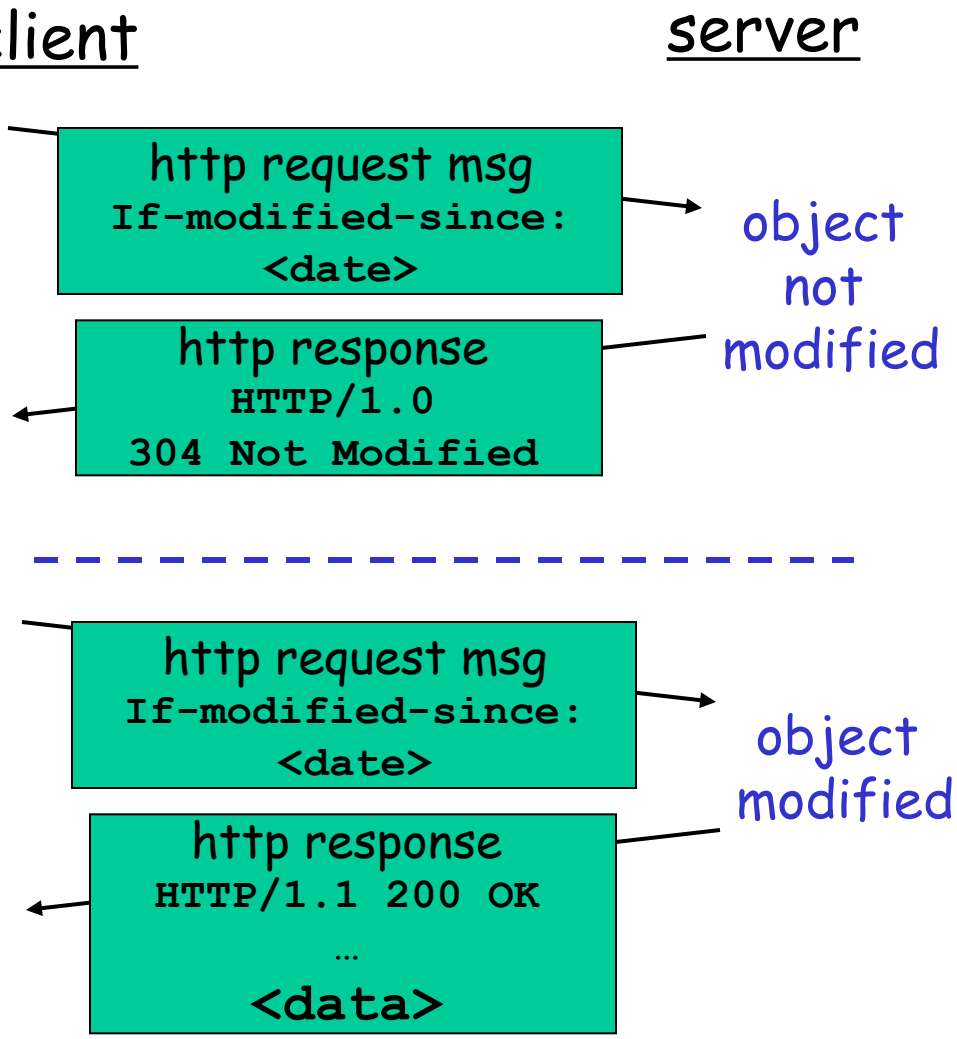


GET condizionale (conditional GET)

- Obiettivo: non inviare oggetti che il client ha già in cache
- client: data dell'oggetto memorizzato in cache

  `If-modified-since: <date>`
- server: la risposta è vuota se l'oggetto in cache è aggiornato:

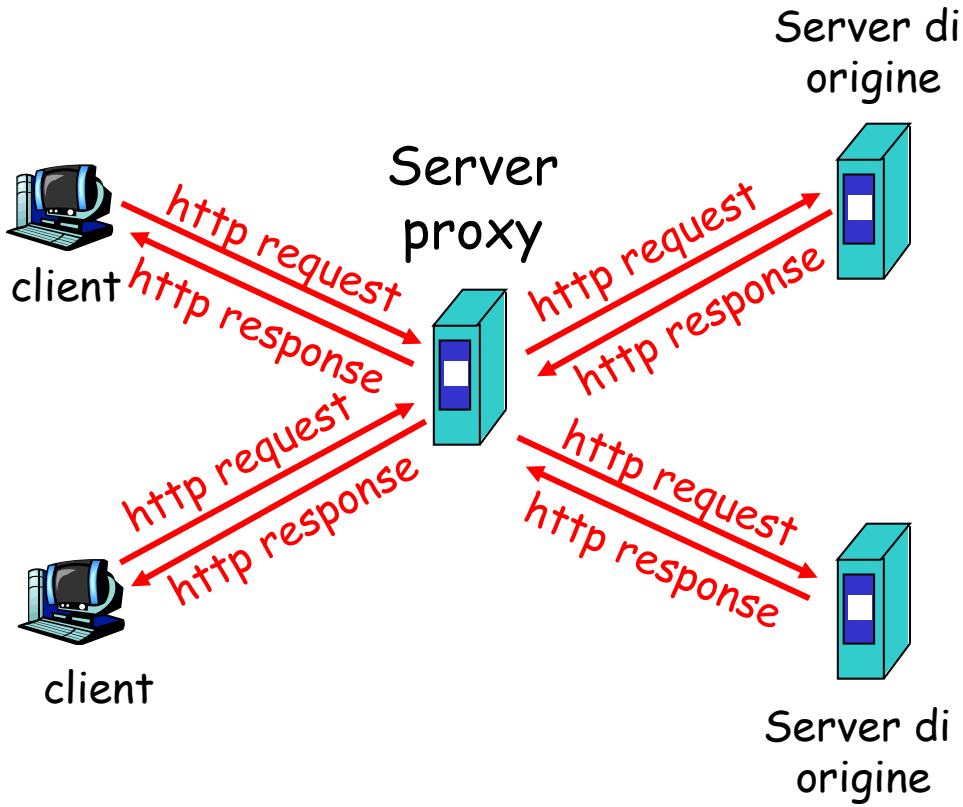
  `HTTP/1.0 304 Not Modified`



Web Cache (proxy server)

Obiettivo: rispondere alle richieste evitando di accedere al server remoto

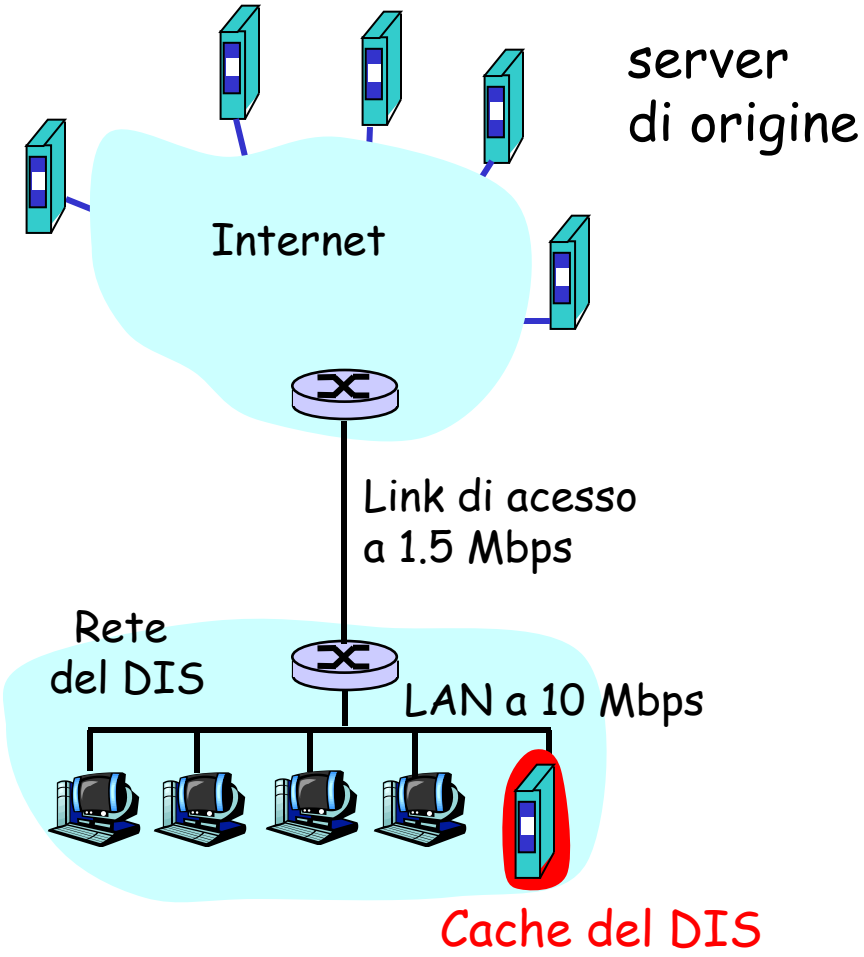
- L'utente configura il browser: accesso attraverso web cache
- Il client invia tutte le richieste al proxy
- La cache restituisce l'oggetto se presente
  - Altrimenti l'oggetto è richiesto prima al server e poi è restituito al client



Perché il Web Caching?

Assunzione: la cache è "vicina" al client (es., stessa rete locale)

- Tempo di risposta minore: la cache è "più vicina" al client
- Diminuisce il traffico verso server lontani
  - Il link di uscita della rete di un ISP istituzionale/locale è spesso un collo di bottiglia



Perché il Web Caching?

- Richieste di 100Kb/sec, 15 richieste per secondo. Intensità di traffico I =1
- Occorre aggiornare la velocità di connessione esterna, i.e. da 1.5 a 10 Mb/sec
- Alternativa: adottare un proxy cache interno alla LAN
- Se riesce a servire il 40% delle richieste, hit rate = 0,4, I = 0,6 (ritardo sale esponenzialmente quando I->1)

Caching cooperativo

- Insieme di proxy cache in cooperazione distribuiti su Internet.
- Possono essere distribuiti in modo gerarchico. Le cache di livello superiore mantengono una copia della pagina mentre questa viene inviata ad una cache di livello inferiore.
- Permette a server con bassa connettività di servire grandi quantità di richieste
- Internet Caching Protocol: comunicazione tra cache

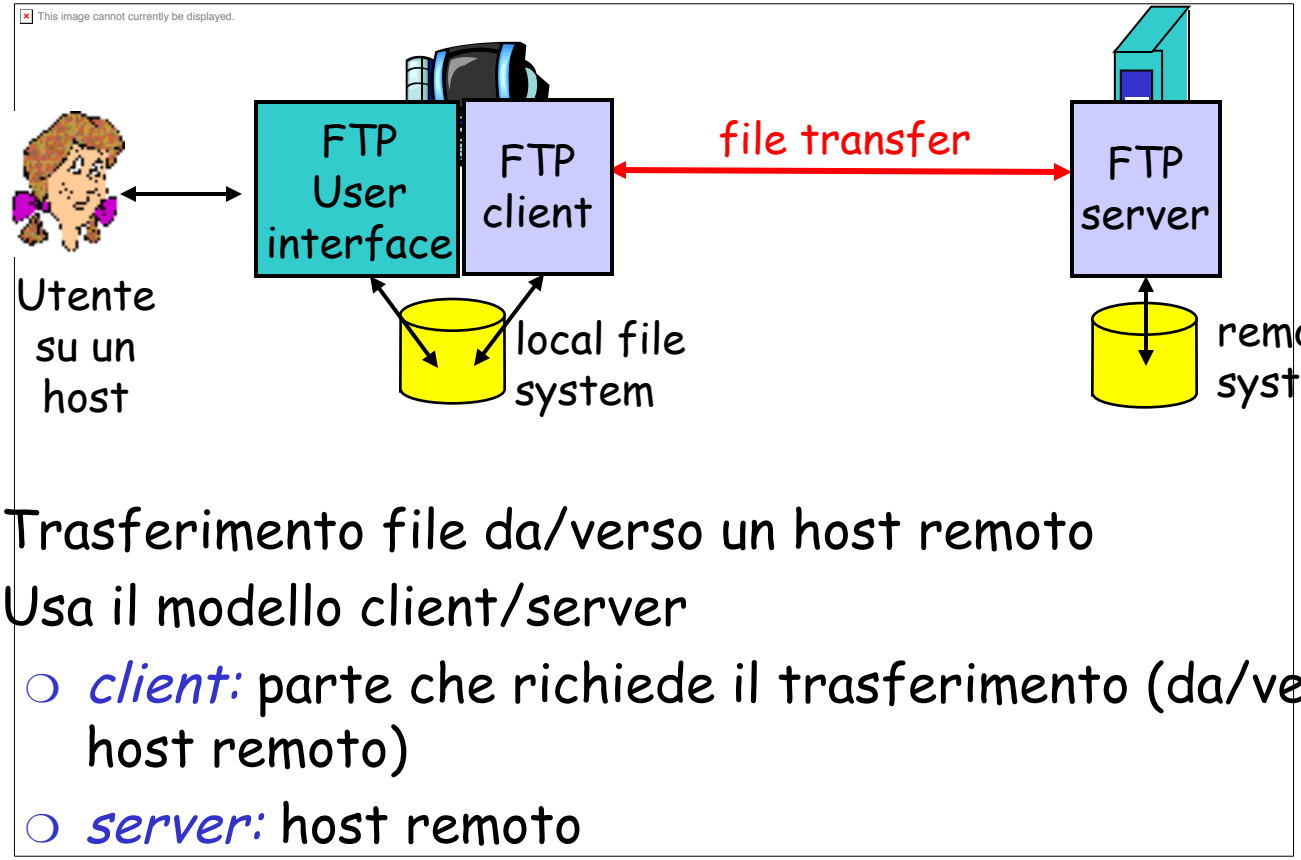
Cluster di Cache

- Permettono di alleviare il carico su una singola cache
- Occorre saper quale cache contiene una specifica pagina
- Si ricorre all'instradamento attraverso tabelle hash (Cache Array Routing Protocol)
- L'elaborazione hash assegna ogni pagina ad una singola cache

Esercizi

- Una pagina Web contiene 5 oggetti.
- Quanti RTT sono necessari per una rchiesta http nel caso di
  a. Connessioni non persistenti
  b. Connessioni persistenti non incanalate
  c. Connessioni persistenti incanalate

ftp: File transfer protocol



- Trasferimento file da/verso un host remoto
- Usa il modello client/server
  - *client:* parte che richiede il trasferimento (da/verso l' host remoto)
  - *server:* host remoto
- ftp: RFC 959
- ftp server: porta 21

ftp: connessioni controllo e dati separate

- Il client contatta il server sulla porta 21, specificando TCP come protocollo di trasporto
- due connessioni TCP parallele:
  - controllo: scambio di messaggi di controllo tra client e server.
    "controllo fuori banda"
  - dati: trasferimento dati da/verso il server
- Entrambi le connessioni aperte dal client
- Il server ftp mantiene info di "stato": directory corrente, autenticazione
- Una nuova connessione per ogni file trasferito

ftp comandi, risposte

Esempi di comandi:

- Inviati come testo ASCII mediante il canale di controllo
- **USER *username***
- **PASS *password***
- **LIST** richiede la lista dei file nella directory corrente (ls)
- **RETR <file>** richiede (get) un file
- **STOR <file>** scarica (put) un file sull' host remoto

Esempi di codici

- Codice di stato e frase (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**

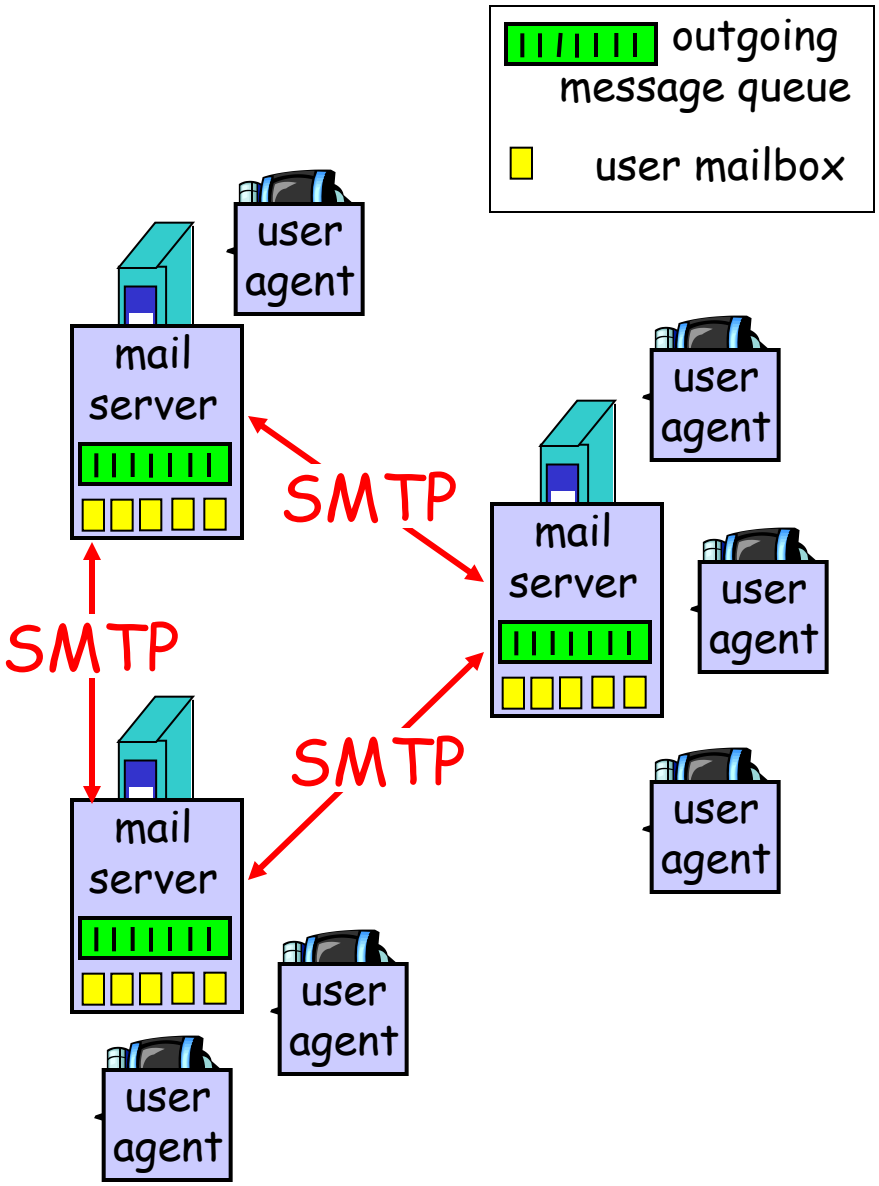
Posta elettronica

Tre componenti principali :

- User agent
- Server di posta
- Simple Mail Transfer Protocol: SMTP

User Agent

- "Lettore di posta"
- Composizione e lettura di messaggi di posta
- Es., Eudora, Outlook, elm, Netscape Messenger
- I messaggi in ingresso/uscita memorizzati sul server

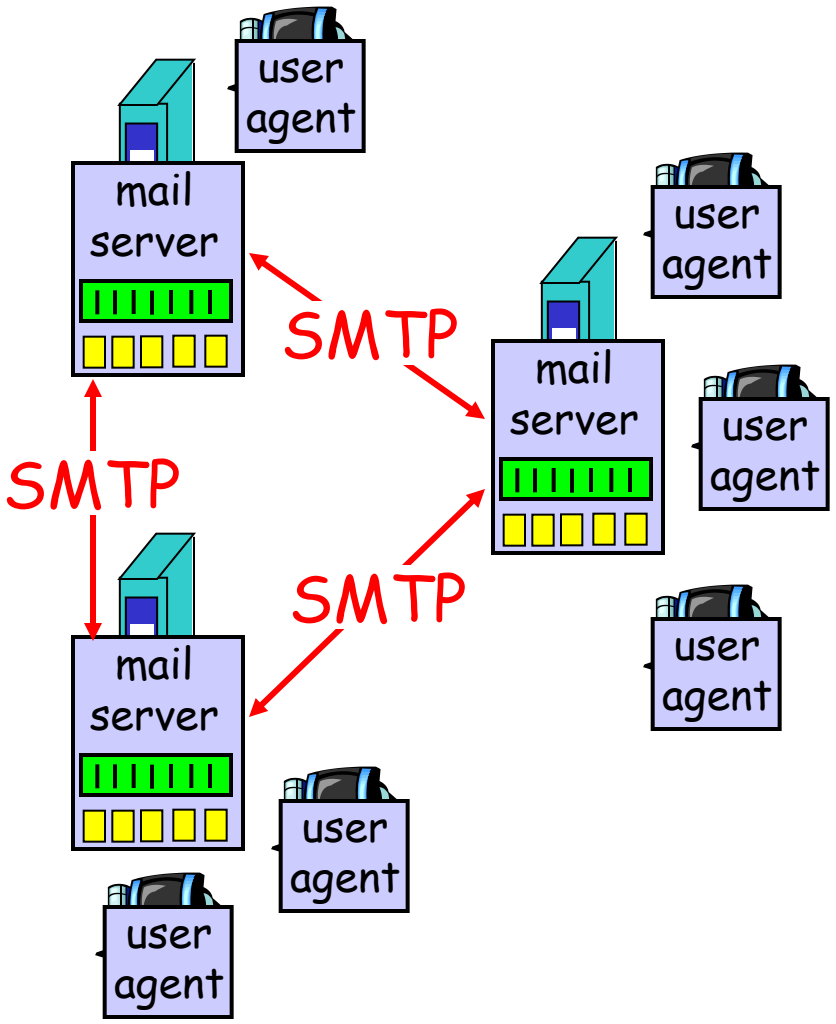


Posta elettronica: mail server

Mail Server

- Mailbox contenente messaggi (non ancora letti) per l'utente
- Coda di messaggi in uscita (non ancora spediti)
- Protocol smtp tra i mail server per il recapito dei messaggi
  - client: il server che invia il messaggio
  - "server": server che riceve il messaggio

Nota: solo una distinzione di ruoli



Recapito di un messaggio di posta elettronica

- Alice compone un messaggio e lo inoltra al suo Mail Server
- Mail Server dispone il messaggio nella coda di messaggi in uscita
- Mail Server di Alice apre una connessione smtp con il Mail Server di Bob ed inoltra il messaggio
- Se il contatto fallisce, l'invio è ripetuto ogni trenta minuti
- Se l'invio fallisce per diversi giorni, mail di notifica inviato ad Alice
- Mail Server di Bob riceve il messaggio dal Mail Server di Alice e lo salva nella Mailbox di Bob
- Bob accede la propria Mailbox specificando Username e Password
- Messaggi possono essere trasferiti dalla Mailbox all'host da cui Bob ha acceduto la Mailbox e/o lasciati sul server
- Bob legge il messaggio di Alice

Posta elettronica: smtp [RFC 821] (1982!)

- Usa tcp per il trasferimento affidabile dei messagi da client a server, porta 25
- Trasferimento diretto: da server a server, non si usano server intermedi di posta
- Tre fasi
  - Handshaking (saluto)
  - Trasferimento di uno o più messaggi (connessione permanente)
  - Chiusura
- Interazione mediante comandi/risposte
  - Comando: testo ASCII
  - Risposta: codice di stato e frase
- Attenzione: I messaggi devono essere comunque riportati in formato ASCII a 7 bit, anche dati multimediali

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
```

Prova

- **`telnet servername 25`**
- Attendi 220 risposta dal server
- Inserisci HELO, MAIL FROM, RCPT TO, DATA, QUIT
- Comandi permettono di inviare email senza usare un User Agent

SMTP: conclusioni

- Connessioni TCP persistenti
- Richiede che il messaggio (header & corpo) sia in formato ascii 7-bit
- Alcune sequenze di caratteri non consentite (es., `CRLF.CRLF`). Conseguenza: il messaggio deve essere codificato
- Il server smtp usa `CRLF.CRLF` per determinare la fine del messaggio

Confronto con http

- http: pull
- email: push
- Entrambi usano un'interazione mediante comandi/risposta in testo ASCII e codici di stato
- http: ogni oggetto incapsulato nel messaggio di risposta
- smtp: un messaggio con più oggetti è inviato mediante un messaggio in più parti

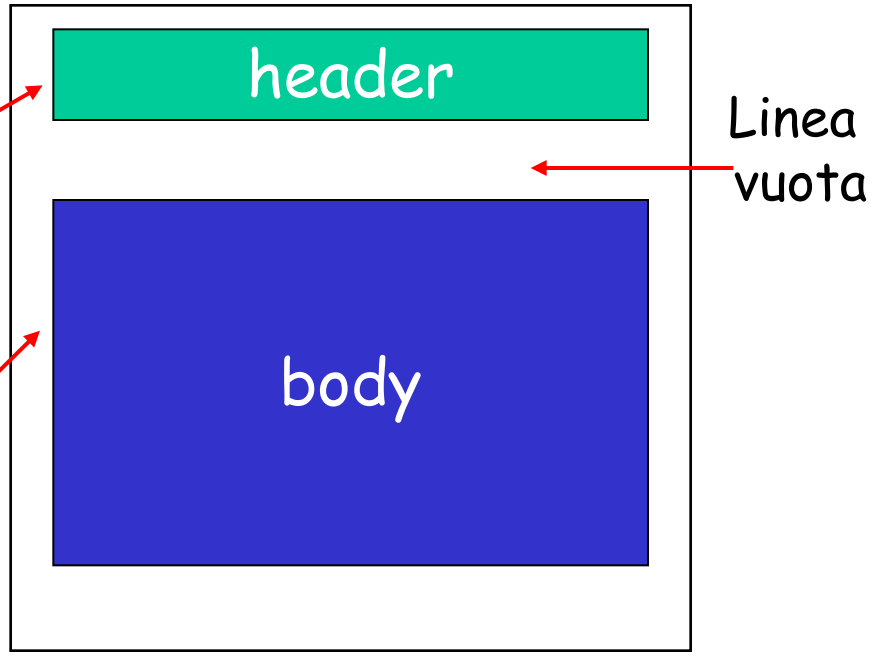
Formato dei messaggi

smtp: protocollo per lo scambio di messaggi di posta

RFC 822: standard per il formato dei messaggi inviati:

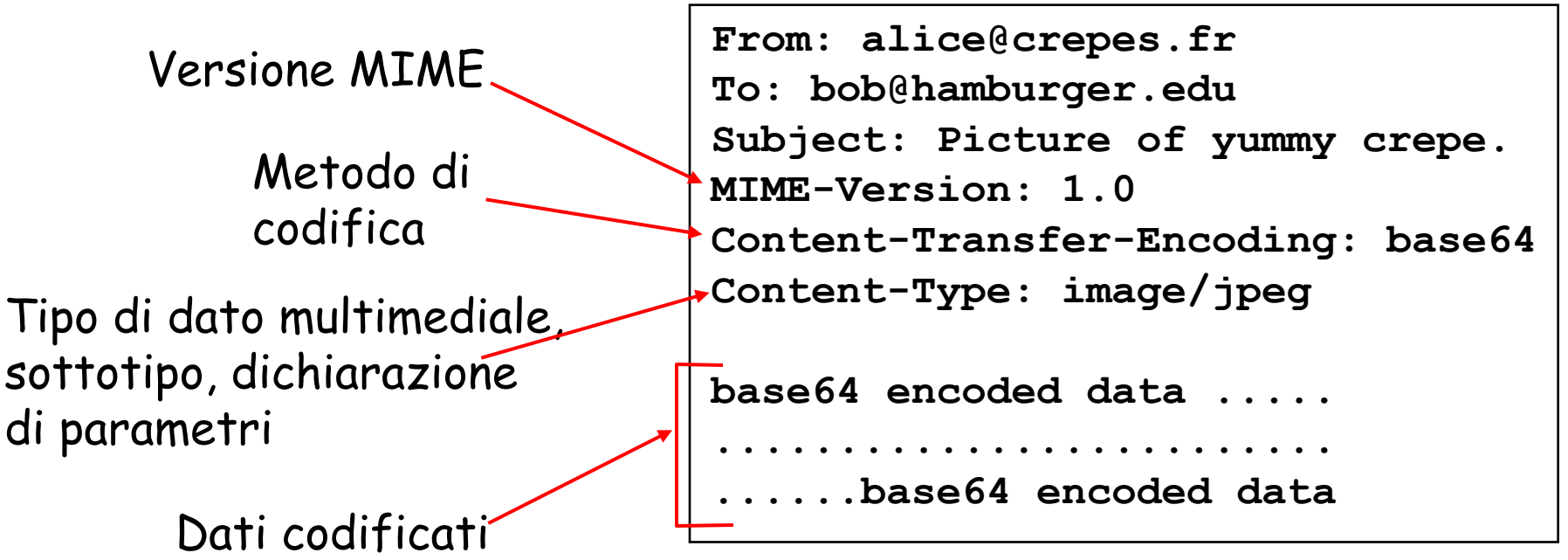
- header, es.,
  - To:
  - From:
  - Subject:

  Diversi dai comandi smtp!
- body
  - Il "messaggio" vero e proprio, solo caratteri ASCII



Formato: estensioni multimediali

- MIME: multipupose internet mail extension, RFC 2045, 2056. Dati Multimediali e di specifiche applicazioni
- Righe addizionali dell' header specificano il tipo del contenuto MIME



Tipi MIME

`Content-Type: type/subtype; parameters`

Text

- Esempi di sottotipi: **`plain, html`**

Image

- Esempi di sottotipi : **`jpeg, gif`**

Audio

- Esempi di sottotipi : **`basic`** (8-bit mu-law encoded), **`32kadpcm`** **(32 kbps coding)**

Video

- Esempi di sottotipi : **`mpeg, quicktime`**

Application

- Dati che devono essere processati da un' applicazione prima di essere "visibili"
- Esempi di sottotipi : **`msword, octet-stream`**

Tipo Multipart

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=98766789

--98766789
Content-Transfer-Encoding: quoted-printable
Content-Type: text/plain

Dear Bob,
Please find a picture of a crepe.
--98766789
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data .....
.........................
......base64 encoded data
--98766789--
```

- E-mail contenenti più oggetti.
- Boundary character: delimitano i messaggi.
- Content-Transfer-Encoding e Content-Type per ogni oggetto

Messaggio ricevuto

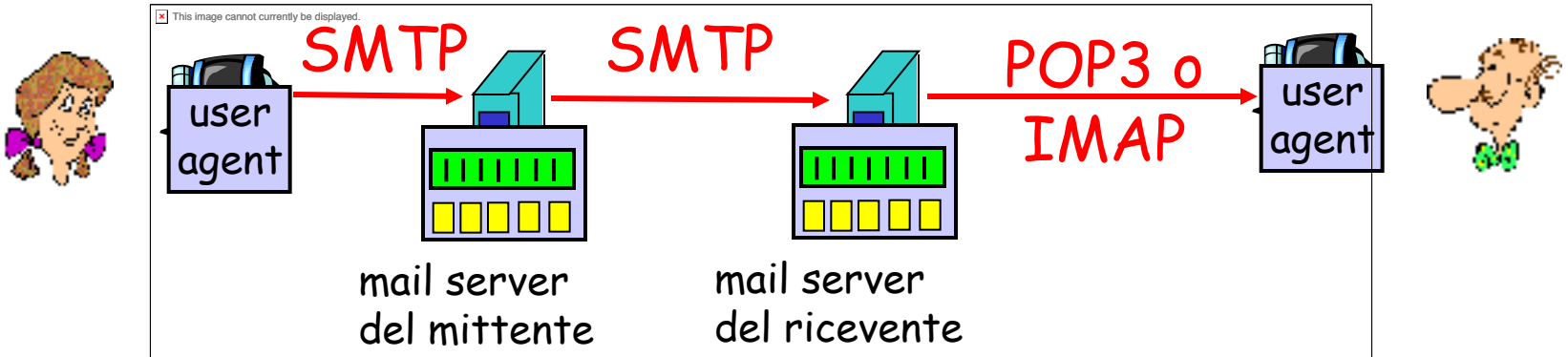
```
Received: from crepes.fr by hamburger.edu; 6 Oct
   2003
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg
```

- Received indica i Mail Server che hanno recapitato il messaggio
- Più linee "Received" se il messaggio è stato inoltrato da più server SMTP lungo il percorso da mittente a destinatario

Protocolli di accesso alla posta

- Soluzione tradizionale: utente legge direttamente la posta sul Mail Server
- L'host su cui è disposto il Mail Server deve essere sempre attivo
- Agenti di posta permettono di trasferire la posta dal Mail Server all'host locale al ricevente
- Possibile visualizzare file multimediali e di specifiche applicazioni
- Occorre un protocollo "Pull" per accedere alla Mailbox collocata sul Mail Server

Protocolli di accesso alla posta



- SMTP: consegna al/memorizzazione nel server di posta del ricevente
- Protocollo di accesso: recupero della posta dal server locale
  - POP: Post Office Protocol [RFC 1939]
    - Autenticazione (agent <-->server) e scaricamento
  - IMAP: Internet Mail Access Protocol [RFC 1730]
    - Più possibilità (più complesso)
    - ~~Manipolazione dei messaggi memorizzati sul server~~
  - HTTP: Hotmail , Yahoo! Mail, ecc.

Protocollo POP3

Fase di autorizzazione

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

Fase di transazione, client:

- `list`: lista numeri e dim. msg
- `retr`: scarica messaggio in base al numero
- `dele`: cancella
- `quit`

```
S: +OK POP3 server ready
C: user alice
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
```

Protocollo POP3

- Scarica ed elimina:
  1. User Agent elimina la posta dalla Mailbox dopo averla scaricata
  2. Un utente disperde la posta sui diversi host da cui accede la Mailbox
  3. User Agent permette di creare cartelle, spostare messaggi, effettuare ricerce nei messaggi

- Scarica e conserva
  1. User Agent conserva la posta sulla Mailbox
  2. Utente può leggere i messaggi da macchine diverse
  3. POP3 stateless, non permette di strutturare i messaggi in directory

Protocollo IMAP

- Permette di gestire cartelle di posta remote come se fossero locali
- IMAP deve mantenere una gerarchia di caretelle per ogni utente
- Permette allo User Agent di scaricare solo parti del messaggio:
  - Intestazione
  - Solo intestazione file MIME Multipart
  - Messaggi di dimensione piccola per utenti a banda limitata
- Stati:
  - Non-authenticated: utente deve fornire username e password per la connessione
  - Authenticated State: utente deve specificare una cartella prima di eseguire comandi che influiscono sul messaggio
  - Selected State: utente può dare comandi che influiscono sul messaggio, e.g. elimina, salva, sposta
  - Logout State: sessione terminata