Livello Applicazione

obiettivo:

- Aspetti implementativi e di uso dei protocolli di livello applicazione
  - paradigma client server
  - Modelli di servizio
- Vediamo esempi di come funzionano le applicazioni su rete

Casi pratici

- esempi
  - http
  - smtp

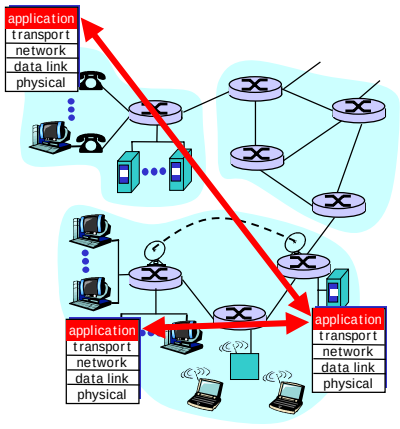
Applicazioni e protocolli del livello applicazione

Applicazione: processi distribuiti

- In esecuzione su host distribuiti in rete
- Scambio messaggi tra applicazioni
- es., email, file transfer, World Wide Web (WWW)

Protocolli di livello applicazione

- Un pezzo di applicazione
- Definiscono il tipo, ordine e struttura dei messaggi da scambiare
- Servizi utente forniti dai livelli inferiori



Applicazioni di rete:

- un processo è un programma in esecuzion su un host in rete
- Sullo stesso host due applicazioni comunicano mediante interprocess communication a cura del sistema operativo
- Processi in esecuzione su host diversi usano invece protocolli di rete di livello applicazione

- uno user agent è un'interfaccia (programma) che consente all'utente di usare i servizi dell'applicazione di rete
  - Es. Web: browser
  - E-mail: mail reader
  - streaming audio/video: media player

paradigma client-server

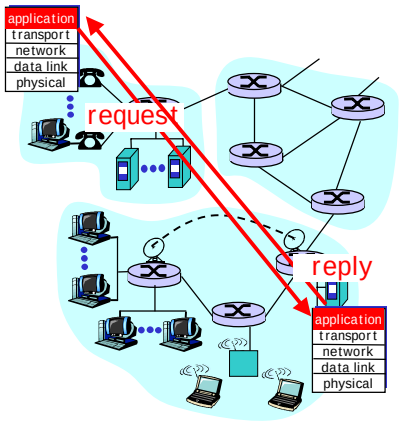
Le applicazioni di rete hanno spesso due agenti: *client* and *server*

Client:

- Contatta il server per primo
- E chiede un servizio al server
- Es. Browser web o mailer per invio e-mail

Server:

- Fornisce i servizi richiesti al client
- es Web server spedisce le pagine web richieste, mail server consegna le mail spedite



Dialogo tra protocolli di livello applicazione

API: application programming interface

- Definisce l'interfaccia tra livello applicazione e livello trasporto
- socket: sono le "API di Internet"
  - Due processi comunicano inviando dati sui socket, e leggendo dati dai socket

D: come fa un processo a determinare con quale altro processo esso chiede di comunicare?

- Indirizzo IP dell'host sul quale si esegue il processo
- "numero di porta" – istruisce il livello trasporto dell'host ricevente su quale sia il processo in esecuzione destinatario dei dati

Servizi di trasporto per le applicazioni

Perdita di dati

- Alcune applicazioni (es., audio) tollerano perdita di dati
- Altre applicazioni (es., file transfer, telnet) richiedono trasferimento affidabile al 100%

Banda trasmissiva

- Alcune applicazioni (es., multimedia) hanno necessità di un minimo di banda garantita
- altre applicazioni ("elastiche") usano la banda disponibile (anche se poca)

Tempo (ritardo)

- Alcune applicazioni (es., telefonia IP su Internet, giochi interattivi in rete) necessitano di bassi ritardi di comunicazione per funzionare bene

Tabella riassuntiva delle caratteristiche dei servizi di livello trasporto maggiormente usati per le applicazioni

Applicazione	Perdita dati	Banda	limiti di tempo
Trasferimento dati	non ammessa	elastica	no
e-mail	non ammessa	elastica	no
World Wide Web	tollerata	elastica	no
real-time audio/video	tollerata	audio: 5Kb-1Mb video:10Kb-5Mb	si, 100's msec
stored audio/video	tollerata	come sopra	si, pochi sec.
Giochi interattivi	tollerata	pochi Kbps	si, 100's msec
Applicazioni finanziarie	non ammessa	elastica	dipende...

Tabella riassuntiva dei servizi di livello trasporto più usati su Internet

Servizio TCP:

- *connection-oriented:* richiede collegamento instaurato tra client e server
- *Trasporto affidabile* senza perdita e con ordinamento
- *Controllo di flusso:* sender non satura il receiver
- *Controllo di congestione:* sender non satura la rete
- *Cosa manca?:* garanzia di ritardo e banda minima

Servizio UDP:

- *trasporto non affidabile:* può perdere e disordinare i dati
- Non fornisce: connessione permanente, garanzia di consegna, controllo di flusso e di congestione, garanzia di ritardo e banda minima

Tabella riassuntiva dei servizi di livello trasporto maggiormente usati per le applicazioni

Applicazione	Protocollo di livello Applicazione usato	Protocollo di livello Trasporto usato
e-mail	smtp [RFC 821]	TCP
Accesso terminale remoto	telnet [RFC 854]	TCP
World Wide Web	http [RFC 2068]	TCP
Trasferimento file	ftp [RFC 959]	TCP
streaming multimediale	proprietario	TCP o UDP
File server remoto	NSF	TCP o UDP
Telefonia IP su Internet	proprietario	Di solito UDP

World Wide Web

- Pagina Web:
  - Fatta da “oggetti”
  - Indirizzata da URL
- Di solito consiste di
  - Pagina HTML (testo)
  - Oggetti allegati
- URL (Uniform Resource Locator) ha tre componenti:
  - Protocollo usato
  - nome dell'host (server)
  - posizione della pagina nel file system del server

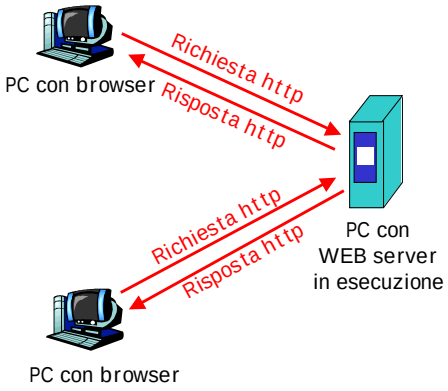
- Browser: applicazione che agisce da client Web:
  - MS Internet Explorer
  - Netscape Communicator
- Web server:
  - Es. Apache (di dominio pubblico)
  - MS IIS: Internet Information Server

http://www.cs.unibo.it/home/faculty/bononi/index.html

Il WWW e il protocollo HTTP

http: hypertext transfer protocol

- È il protocollo di livello applicazione usato per il sevizio WWW
- Modello client/server
- *client:* il browser richiede, riceve e "mostra" gli oggetti (pagine web)
- *server:* Web server invia oggetti (file) che gli sono stati richiesti
- http1.0: RFC 1945
- http1.1: RFC 2068



Il protocollo HTTP

http: usa il servizio trasporto di TCP

- Il client (browser) crea una connessione TCP verso il server (socket su porta 80)
- Il server accetta la connessione TCP
- Client e server si scambiano "messaggi del protocollo HTTP" per gestire lo scambio di pagine web
- Al termine si chiude la connessione TCP

http è "senza stato"

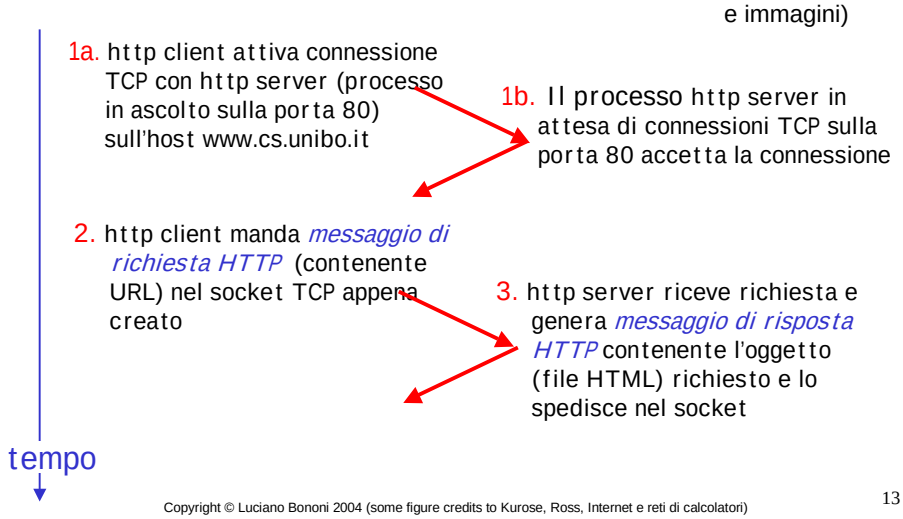
- Il server non ricorda a che punto è arrivato lo scambio con il client

Xche?

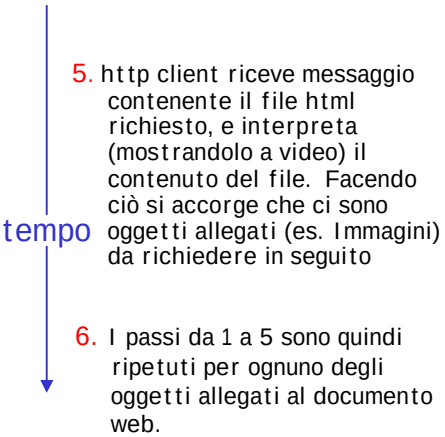
Protocolli che mantengono lo stato sono complessi

Esempio di dialogo HTTP

Se l'utente scrive la richiesta della URL
www.cs.unibo.it/home/faculty/bononi/index.html (es. Pagina HTML che contiene testo e immagini)



Esempio di dialogo HTTP (cont.)



Connessioni HTTP persistenti e non persistenti

Non-persistenti

- Versione HTTP/1.0
- Il server chiude connessione TCP dopo ogni richiesta
- 2 RTTs necessari al client per ottenere ogni oggetto

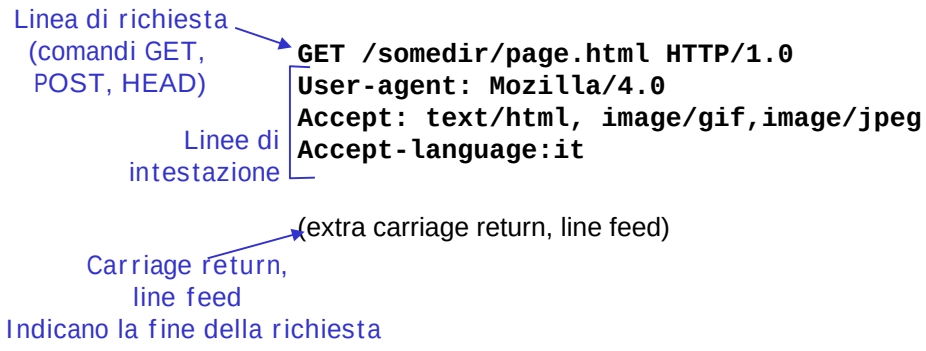
...ma i browser HTTP 1.0 possono aprire più connessioni TCP in parallelo

Persistenti

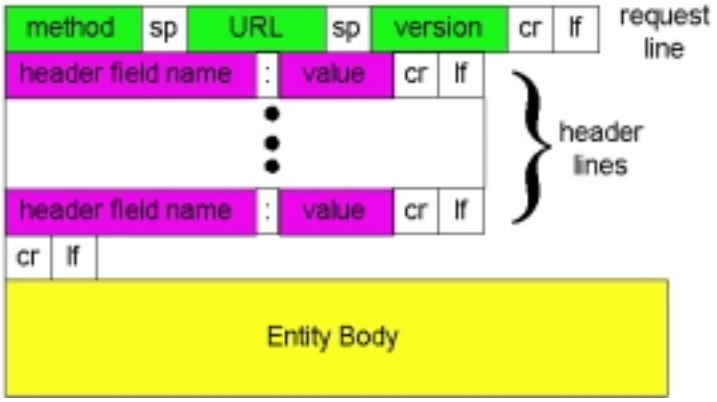
- versione HTTP/1.1
- Sulla stessa connessione TCP il server risponde a più richieste in sequenza
- Client invia tutte le richieste di seguito interpretando il file HTML
- Solo 1 RTT necessario per ottenere gli oggetti della pagina

Formato del messaggio di richiesta HTTP

- Due tipi di messaggi HTTP: *richieste e risposte*
- Richiesta http:
  - Scritta in testo ASCII (si può leggere)



Formato generale del messaggio di richiesta HTTP



Formato del messaggio di risposta HTTP

Linea di stato (codice di stato della richiesta)

Linee di intestazione

dati, oggetti richiesti e inviati dal server

```
HTTP/1.0 200 OK
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 …...
Content-Length: 6821
Content-Type: text/html

dati ...
```

Codici di stato delle risposte HTTP

Sono interpretati dal browser e mostrati all'utente

Alcuni esempi (che avrete visto a volte):

200 OK

- Richiesta soddisfatta. I dati seguono l'intestazione.

301 Moved Permanently

- Oggetto richiesto è stato spostato permanentemente su una nuova destinazione specificata nel campo (Location:)

400 Bad Request

- Messaggio di richiesta incomprensibile

404 Not Found

- Documento richiesto non trovato sul server

505 HTTP Version Not Supported

Esempio pratico (esercitazione minima)

1. Dialoghiamo con il Web server spacciandoci per browser:

```
telnet www.cs.unibo.it 80
```

Apre connessione TCP su porta 80 (su server web del dipartimento). Tutto ciò che scriviamo ora viene inviato dall'applicazione telnet alla porta 80 del server web

2. Scriviamo ora una richiesta GET del protocollo HTTP:

```
GET /~bononi/index.html HTTP/1.0
      (inviare due return)
```

Così inviamo una richiesta minima GET in formato http al server per ottenere il file index.html della mia homepage

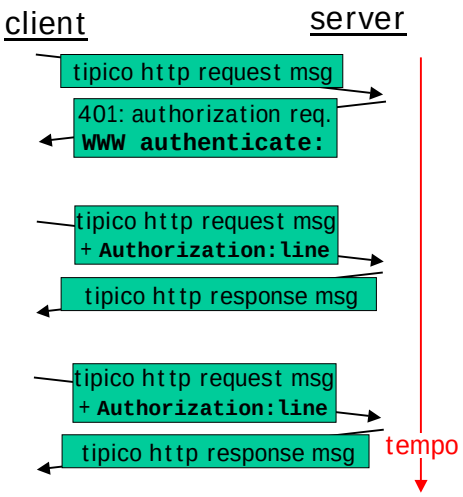
3. Di seguito vedremo la risposta del server, con allegato il file richiesto.
   (poi chiude automaticamente la connessione)

Autenticazione dell'utente

Authenticazione: controllo dell'accesso ai documenti sul server

- stateless: il client deve allegare l'autorizzazione ad ogni richiesta
- Autorizzazione: di solito consiste nella coppia <nome, password>
  - **authorization:** linea di intestazione della richiesta
  - Se è richiesta autorizzazione il server manda linea

    WWW authenticate:



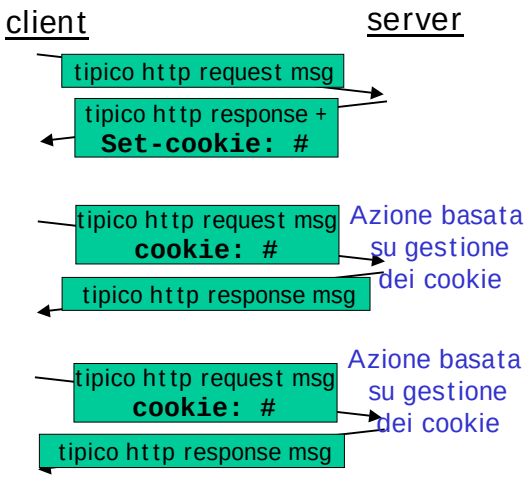
Il browser può memorizzare nome e password per evitare all'utente di reimmetterlo (meglio di no)

Interazione user/server mediante cookies

- server manda "cookie" al client nella risposta

  Set-cookie: 1678453
- client presenta il cookie nelle richieste successive

  cookie: 1678453
- Il server confronta il cookie con le info memorizzate
  - Autenticazione del client
  - Ricorda lo stato della transazione, le scelte e preferenze del client

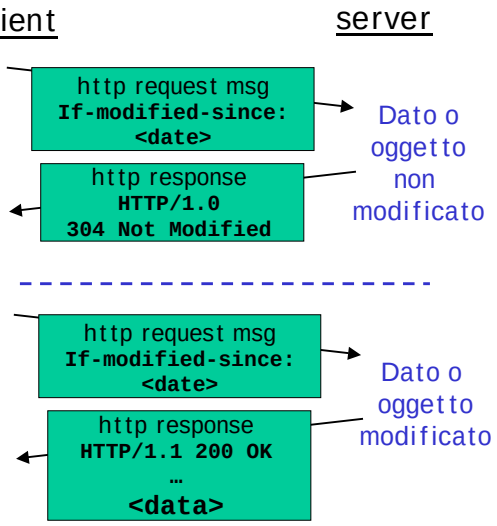


La richiesta GET condizionale

- Xchè?: il server evita di spedire inutili copie dei dati se il client ha in cache dati una copia ancora valida
- client: specifica la data della sua copia in cache nella richiesta HTTP

  `If-modified-since: <date>`
- server: risponde con il dato aggiornato solo se la copia non era più valida, altrimenti

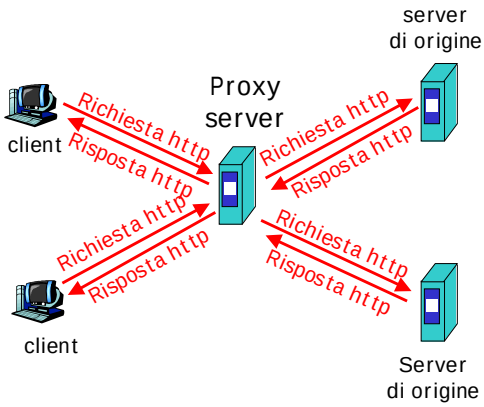
  `HTTP/1.0 304 Not Modified`



Web Caches (e proxy server)

Obiettivo: soddisfare le richieste dei client senza impegnare i server remoti

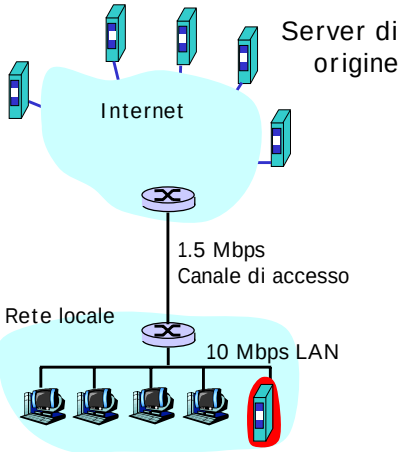
- Utente istruisce il browser sull'accesso mediante web proxy (e web cache)
- client spedisce tutte le richieste HTTP alla web cache
  - Se c'è viene restituito subito il dato
  - Se non c'è la richiesta va al server originale



Perchè serve il Web Caching?

Assunzione: la cache è vicina al client (es. sulla sua rete locale)

- Riduce tempo di risposta
- Riduce traffico di rete verso server remoti



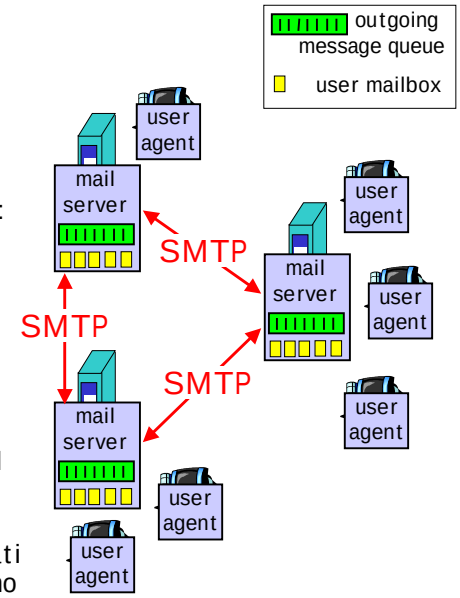
Posta Elettronica

Tre componenti principali:

- user agents
- mail servers
- simple mail transfer protocol: smtp

User Agent

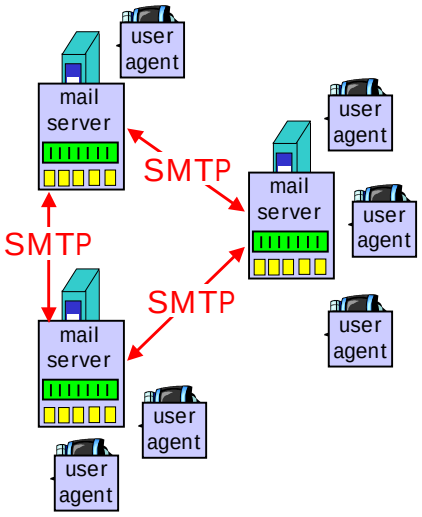
- È il programma per leggere, comporre le mail
- e.g., Eudora, Outlook, elm, Netscape Messenger
- I messaggi in uscita spediti al mail server locale (sempre connesso alla rete)
- Messaggi in arrivo memorizzati sul server locale (mailbox) fino alla lettura



I server di posta elettronica

Mail Servers

- mailbox : cartella contenente i messaggi ricevuti per un utente e non ancora letti
- Coda dei messaggi : coda dei messaggi da spedire a destinazione
- Protocollo smtp usato dai server per inoltrare a destinazione i messaggi
  - I server lavorano sia in fase di spedizione che in fase di inoltro e memorizzazione dei messaggi, in attesa di lettura da parte del client



protocollo SMTP [RFC 821]

- usa connessione TCP per trasmissione affidabile dei messaggi da server a server, sulla porta 25
- Trasferimento diretto tra server di invio e server destinazione
- Tre fasi per il trasferimento
  - handshaking (saluti e collegamento)
  - Trasferimento messaggi
  - Chiusura connessione
- Interazione con comandi/risposte
  - comandi: testo ASCII
  - risposta: codice di stato e frasi di testo
- Messaggi devono essere codificati in ASCII a 7-bit

Esempio di interazione smtp

```
S: 220 cs.unibo.it
C: HELO zuma.cs.unibo.it
S: 250  Hello zuma.cs.unibo.it, pleased to meet you
C: MAIL FROM: <pippo@cs.unibo.it>
S: 250 pippo@cs.unibo.it... Sender ok
C: RCPT TO: <pluto@cs.unibo.it>
S: 250 pluto@cs.unibo.it ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: testo della e-mail..... Bla bla bla bla
C: ............ Bla bla bla bla bla.
C: . /* questo . termina il testo dell'e-mail */
S: 250 Message accepted for delivery
C: QUIT
S: 221 cs.unibo.it closing connection
```


Esercitazione svolta a lezione:

Invio di e-mail senza usare mail client, ma direttamente con dialogo TCP col server.

(note su errori e debolezze del protocollo SMTP)

- **telnet mail.cs.unibo.it 25**
- attendi 220 reply dal server
- immettere HELO, MAIL FROM, RCPT TO, DATA, QUIT

smtp: riassunto

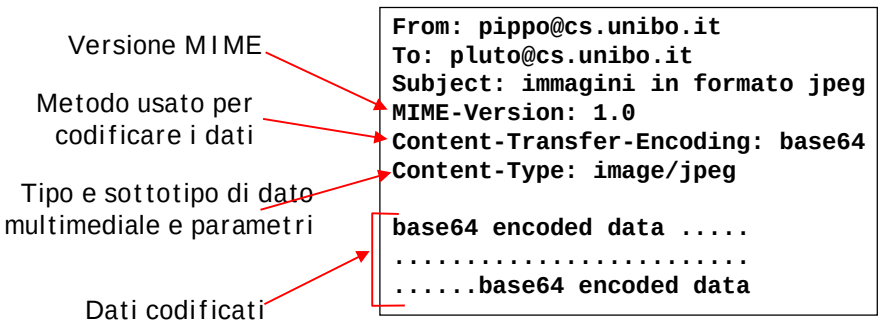
- Smtp usa connessioni persistenti TCP
- smtp richiede che tutto il messaggio sia in 7-bit ascii

Confronto smtp vs. http

- http: pull
- email: push
- Entrambi basati su comandi, intestazioni e risposte ASCII
- http: ogni oggetto è incluso nel suo messaggio di risposta
- smtp: molti oggetti possono essere inclusi in un messaggio singolo (mail attachments) mediante le MIME extensions

Formato messaggi: multimedia extensions

- MIME: multimedia mail extension, RFC 2045, 2056
- Linee addizionali nell'intestazione del messaggio di e-mail dichiarano il tipo di contenuto MIME



Tipi, sottotipi di dati in formato MIME

`Content-Type: type/subtype; parameters`

Text

- Es. subtypes: **`plain, html`**

Image

- Es. subtypes: **`jpeg, gif`**

Audio

- Es. subtypes: **`basic`** (8-bit mu-law encoded), **`32kadpcm`** **(32 kbps coding)**

Video

- Es. subtypes: **`mpeg, quicktime`**

Application

- Altri dati che richiedono di essere decodificati prima di essere "visibili"
- Es. subtypes: **`msword, octet-stream`**

Es. Di codifica Multipart Type

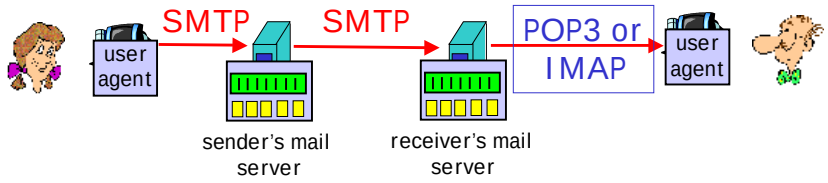
```
From: pippo@cs.unibo.it
To: pluto@cs.unibo.it
Subject: immagine in formato jpeg
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=98766789

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

Caro pluto,
guarda che bella immagine.
--98766789
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 dati codificati .....
.........................
......base64 dati codificati
--98766789--
```

testo

delimitatore

Codifica dell'immagine

Protocolli di accesso all'e-mail



- SMTP: è il protocollo usato per la consegna e la memorizzazione dei messaggi dal server di invio al server destinazione finale
- I protocolli Mail Access sono usati per recuperare le mail memorizzate nel server destinazione finale, e sono ad esempio:
  - POP: Post Office Protocol [RFC 1939]
    - autorizzazione (agent <-->server) e scaricamento mailbox
  - IMAP: Internet Mail Access Protocol [RFC 1730]
    - Ha molte caratteristiche ed è complesso rispetto a POP
    - Permette di gestire e manipolare i messaggi sul server
  - A volte si usa anche HTTP per ricevere i messaggi basandosi sul servizio webmail, es. Hotmail , Yahoo! WebMail, etc.

Es. Protocollo POP3

autorizzazione

- client invia:
  - **user:** dichiara suo nome
  - **pass:** password
- server risponde
  - **+OK (se autorizzato)**
  - **-ERR (altrimenti)**

transazione, il client richiede:

- **list:** elenca numero messaggi
- **retr:** recupera messaggio
- **dele:** cancella messaggio
- **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
```