



POLITECNICO  
MILANO 1863

# Fondamenti di TELECOMUNICAZIONI

Prof. Marco Mezzavilla

# Lezione 8 - Livello Applicativo 1

# INDICE

## 8. LIVELLO APPLICATIVO 1

1. **Applicazioni e architetture**
2. **Web browsing**
3. **Posta elettronica**
4. **Risoluzione di nomi simbolici e caching**
5. **Streaming e peer-to-peer**

Parte I (oggi)

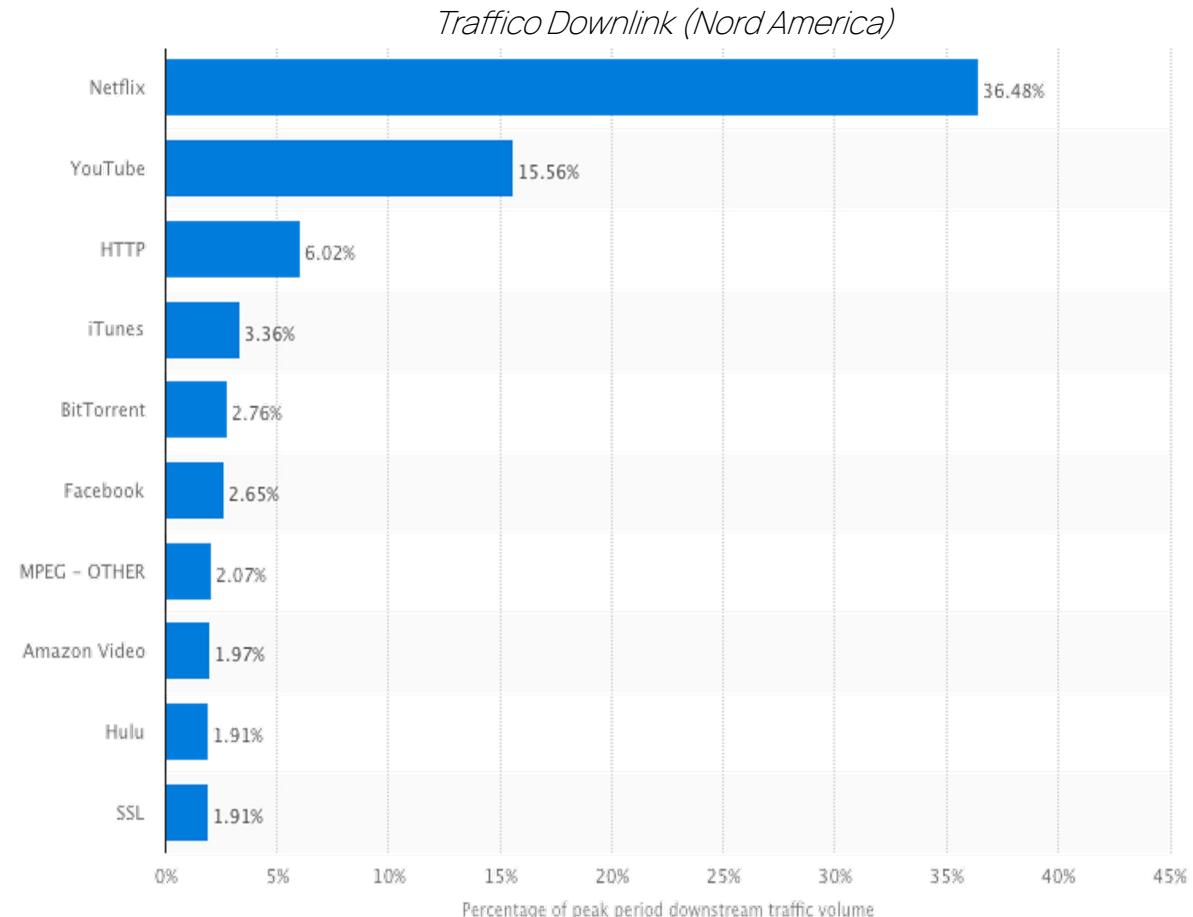
Parte II (domani)

# APPLICAZIONI E ARCHITETTURE

01

# Alcune applicazioni di rete

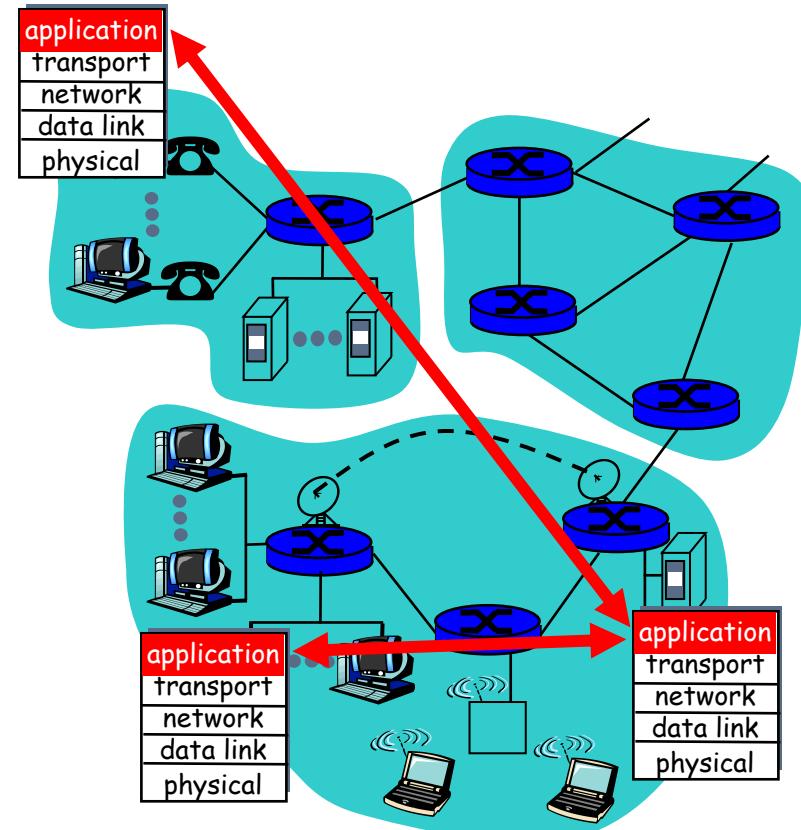
- World Wide Web**
  - Firefox, Safari, Chrome, ..
- Posta elettronica**
  - Mail, Gmail, Outlook, ..
- Social networking**
  - Facebook, Twitter, Instagram, Snapchat, ..
- P2P file sharing**
  - BitTorrent, eMule, ..
- Video streaming**
  - NetFlix, YouTube, Hulu, ..
- Video conference e telefonia**
  - Zoom, Teams, Skype, Hangout, ..
- Instant messaging**
  - Whatsapp, Telegram, ..
- Network games**
- ...



Source: [www.statista.com](http://www.statista.com) (2016)

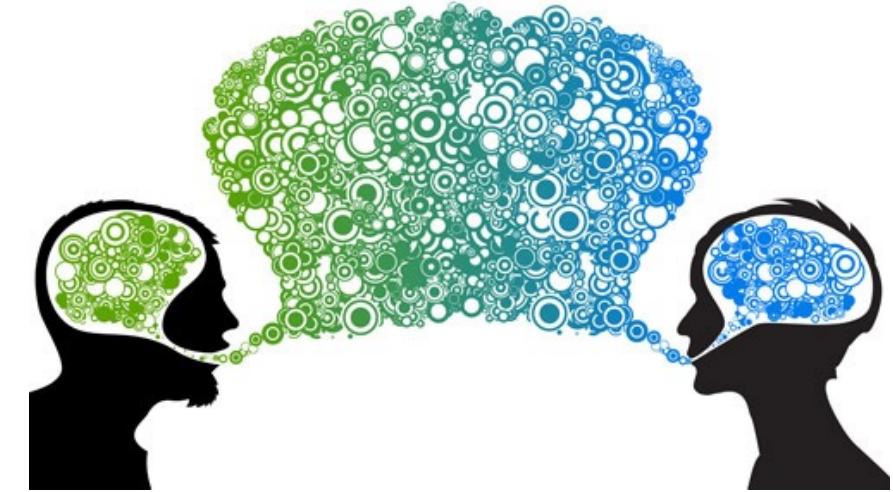
# Alcune applicazioni di rete

- ❑ L'applicazione è un **software** che:
  - ❑ può essere eseguito su diversi terminali e
  - ❑ può comunicare tramite la rete
- ❑ Esempio: il *browser web* (FireFox, Safari, Chrome, ecc..) è un software “in esecuzione” su un dispositivo che comunica con un software in esecuzione su un server web ([www.google.com](http://www.google.com), [www.amazon.com](http://www.amazon.com), ecc..)
- ❑ Inventare una nuova applicazione non richiede di cambiare il funzionamento della rete
- ❑ I nodi della rete non hanno software applicativo
- ❑ Le applicazioni sono solo nei terminali e possono essere facilmente sviluppate e diffuse



# Comunicazione tra processi

- ❑ **Host:** dispositivo d'utente
  - ❑ Laptop, smartphone, desktop
- ❑ **Processo:** programma software in esecuzione su un host
  - ❑ Molti processi possono essere in esecuzione simultaneamente sullo stesso host



# Ingredienti per la comunicazione tra processi remoti

## **Indirizzamento** dei processi

- conoscere il “numero di telefono” dell’interlocutore

## **Protocollo** di scambio dati (decidere la “lingua” con cui si parla”

- Tipi di messaggi scambiati:

- Richieste, risposte

- Sintassi dei messaggi:

- Campi del messaggio e delimitatori

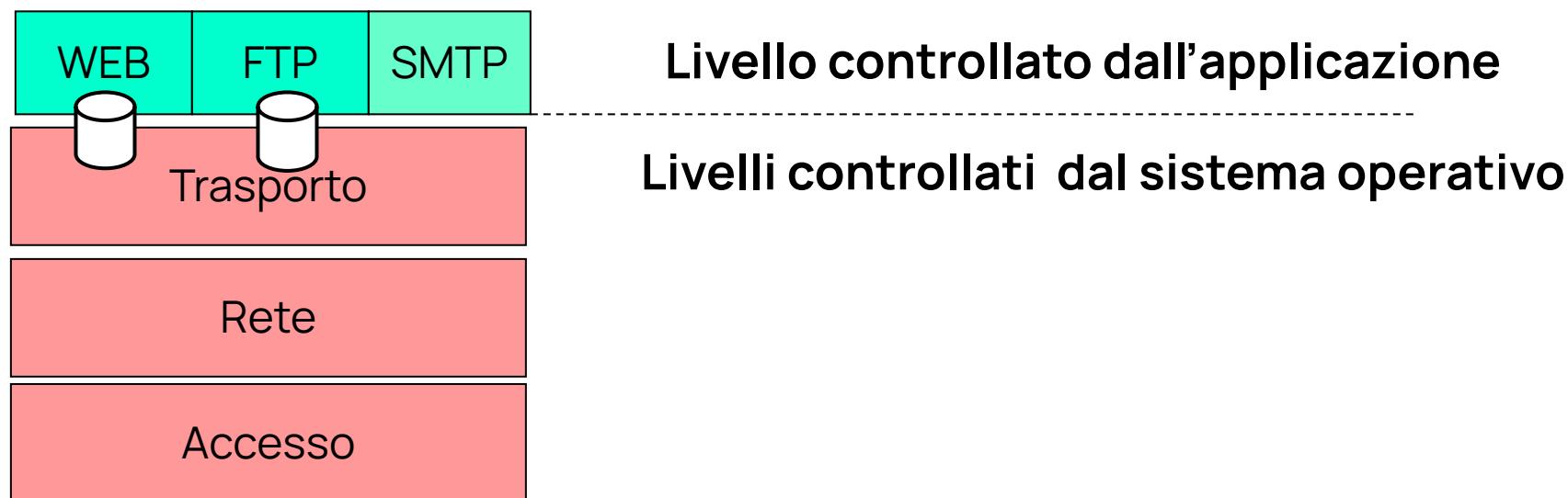
- Semantica dei messaggi:

- Significato dei campi

- Regole su come e quando inviare e ricevere i messaggi

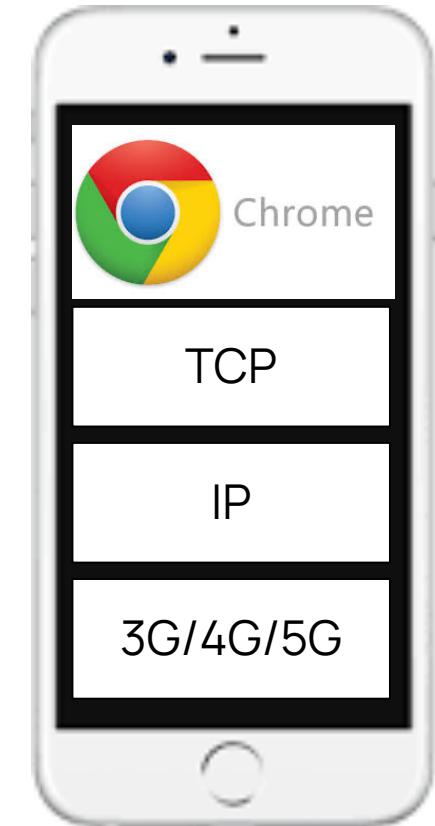
# Indirizzamento di processi applicativi

- Lo scambio di messaggi fra i processi applicativi avviene utilizzando i servizi dei livelli inferiori attraverso i SAP (Service Access Point)
- Ogni processo è associato ad un SAP



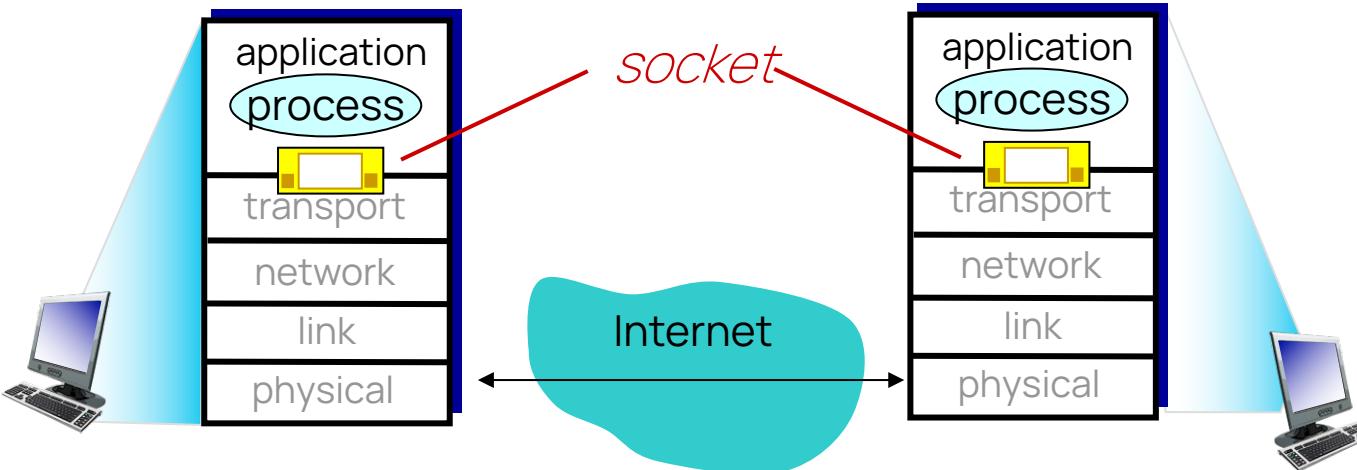
# Indirizzamento di processi applicativi

- ❑ Cosa serve per identificare il mio browser Chrome in esecuzione sul mio host?
  - ❑ Indirizzo del mio host
    - ❑ Indirizzo **IP** univoco di 32 bit (di cui parleremo ampiamente in seguito)
  - ❑ Indirizzo del SAP del processo in esecuzione sul mio host
    - ❑ Numero di **porta**



# Indirizzamento di processi applicativi

- Indirizzo di un processo in esecuzione = indirizzo IP + numero di porta = **SOCKET**
- Esempio:
  - Il server web del Politecnico www.polimi.it è raggiungibile a: 131.175.12.34/80
- Il socket rappresenta la porta di comunicazione
  - Il processo trasmittente mette il messaggio fuori dalla porta
  - La rete raccoglie il messaggio e lo trasporta fino alla porta del destinatario



- **Porte ben note (well-known ports)**: Sono comprese tra **0 e 1023** e sono riservate per servizi di sistema ben noti. Ad esempio:
  - **Porta 80**: HTTP (navigazione web).
  - **Porta 443**: HTTPS (navigazione web sicura).
  - **Porta 25**: SMTP (protocollo per l'invio di email).
- **Porte registrate**: Sono comprese tra **1024 e 49151** e vengono assegnate ad applicazioni utente o servizi meno comuni.
- **Porte dinamiche o private**: Sono comprese tra **49152 e 65535** e vengono solitamente assegnate dinamicamente dai sistemi operativi quando un'applicazione crea un socket.

# Requisiti delle applicazioni

## Perdita di dati

- Alcune applicazioni possono tollerare perdite parziali (e.g., audio)
- Altre applicazioni richiedono a completa affidabilità (e.g., email)

## Ritardo

- Alcune applicazioni richiedono basso ritardo (e.g., telefonia internet, videogiochi interattivi)

## Banda

- Alcune applicazioni richiedono una banda minima di trasferimento (e.g., appl. multimediali)
- Altre applicazioni si adattano alla banda disponibile (e.g., appl. elastiche)



# Requisiti delle applicazioni

Application	Data loss	Bandwidth	Time Sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5kbps-1Mbps video:10kbps-5Mbps	yes, 100 msec
stored audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	few kbps up	yes, 100msec
instant messaging	no loss	elastic	no

# Architetture applicative

## □ Client-server

- I dispositivi coinvolti nella comunicazione implementano o solo il processo client o solo il processo server
- I dispositivi client e server hanno caratteristiche diverse
- I client possono solo eseguire richieste
- I server possono solo rispondere a richieste ricevute

## □ Peer-to-peer (P2P)

- I dispositivi implementano sia il processo client che quello server

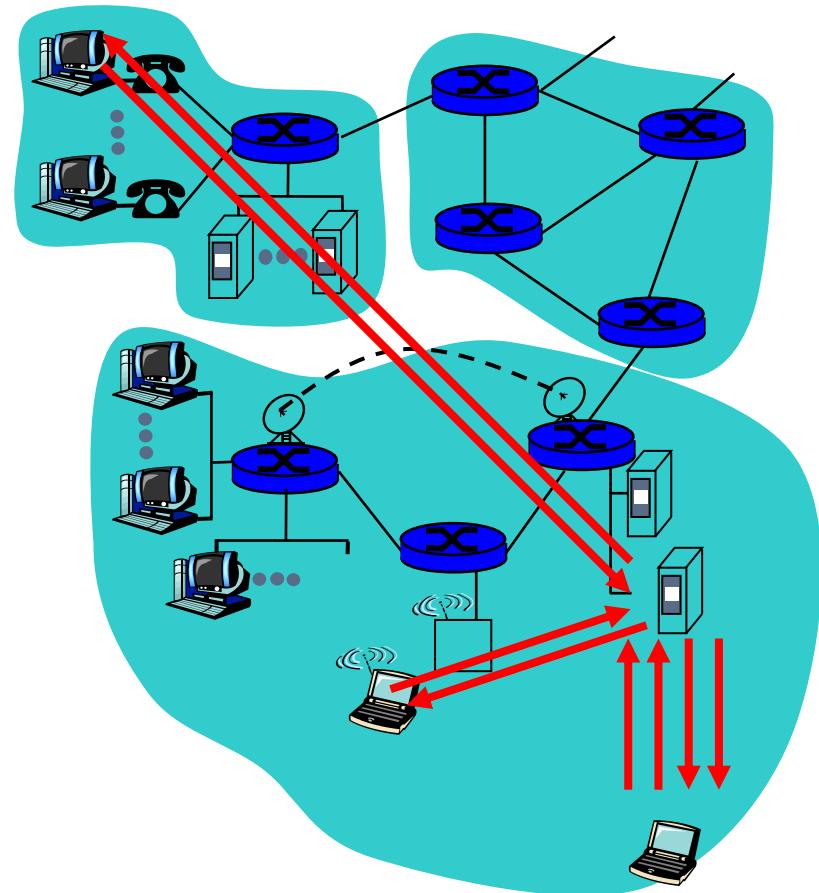
# Architettura client-server

## □ Server:

- Host sempre attivo
- Indirizzo IP permanente
- Possibilità di utilizzo di macchine in cluster
- Possono ricevere richieste da molti client

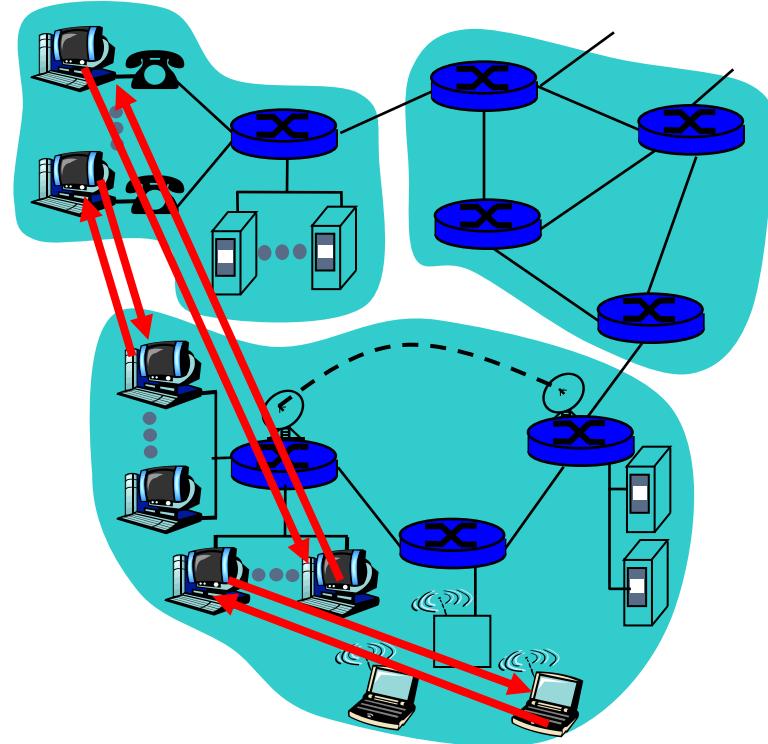
## □ Client:

- Comunicano con il server
- Possono essere connessi in modo discontinuo
- Possono cambiare indirizzo IP
- Non comunicano con altri client
- Possono inviare molte richieste allo stesso server



# Architettura P2P

- Non ci sono server sempre connessi
- Terminali (peers) comunicano direttamente
- I peers sono collegati in modo intermittente e possono cambiare indirizzo IP
- Esempio: BitTorrent



**Fortemente scalabile ma difficile da gestire**

# WEB BROWSING

02

# Cosa contengono i messaggi HTTP – le Pagine Web

- Le pagine web sono fatte di oggetti
- Gli oggetti possono essere file HTML, immagini JPEG, applet Java, file audio file, file video, collegamenti ad altre pagine web..
- Generalmente le pagine web hanno un file HTML (o php) base che “chiama” gli altri oggetti
- Ogni oggetto è indirizzato da una Uniform Resource Locator (URL), es:

`http://www.polimi.it:80/index.html`

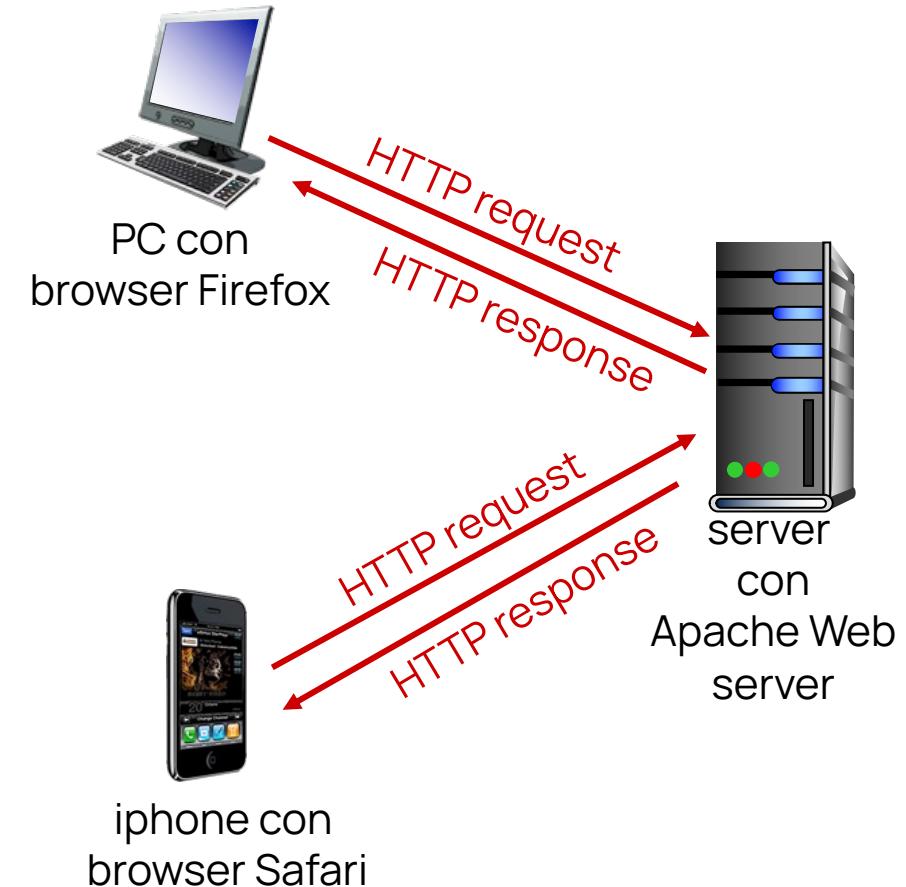
The URL `http://www.polimi.it:80/index.html` is displayed at the top. Below it, four arrows point from descriptive text to specific parts of the URL:

- A solid arrow points from the text "Indica il protocollo applicativo" to the prefix `http://`.
- A solid arrow points from the text "Indica l'indirizzo di rete del server" to the domain name `www.polimi.it`.
- A dashed arrow points from the text "Indica la pagina web richiesta" to the file name `index.html`.
- A dashed arrow points from the text "Indica il numero di porta (opzionale)" to the port number `:80`.

Indica il protocollo applicativo  
Indica l'indirizzo di rete del server  
Indica la pagina web richiesta  
Indica il numero di porta (opzionale)

# La comunicazione HTTP

- Architettura client/server:
  - **client**: browser che effettua richieste HTTP di pagine web (oggetti), le riceve e le mostra all'utente finale
  - **server**: Web server inviano gli oggetti richiesti tramite risposte HTTP
- Nessuna memoria sulle richieste viene mantenuta nei server sulle richieste passate ricevute da un client (protocollo stateless)



# La comunicazione HTTP

- HTTP si appoggia su TCP a livello di trasporto:
  1. Il client HTTP inizia una connessione TCP verso il server (porta 80)
  2. Il server HTTP accetta la connessione TCP da client HTTP
  3. Client e Server HTTP si scambiano informazioni (pagine web e messaggi di controllo)
  4. La connessione TCP tra client e server HTTP viene chiusa

# La comunicazione HTTP

## □ Connessione non persistente

- Una connessione TCP per una sola sessione richiesta-risposta; inviato l'oggetto il server chiude la connessione TCP
- La procedura viene ripetuta per tutti i file collegati al documento HTML base
- Le connessioni TCP per più oggetti possono essere aperte in parallelo per minimizzare il ritardo

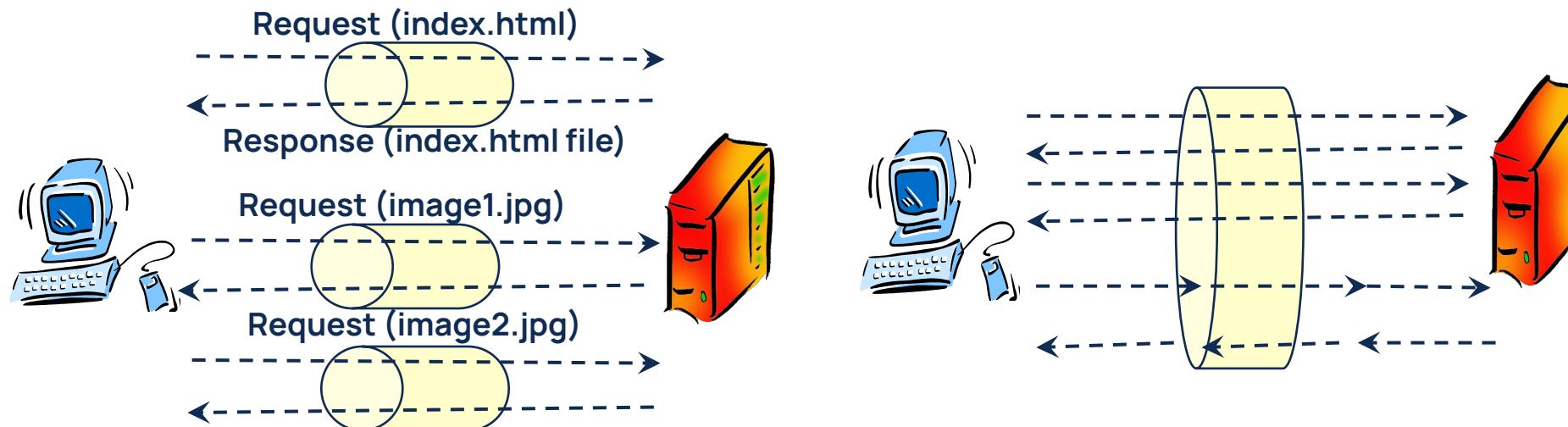
## □ Connessione persistente

- La connessione TCP rimane aperta e può essere usata per trasferire più oggetti della stessa pagina web o più pagine web

- *Without pipelining*: richieste HTTP inviate in serie
- *With pipelining*: richieste HTTP inviate in parallelo

*prevalente al giorno d'oggi*

*agli albori di internet, quando i sistemi erano più semplici.*



# Esempio di connessione non persistente

- L'utente digita nel browser la URL: **www.polimi.it/home/index.html**

(Esempio: l'HTML contiene testo e riferimenti a 10 immagini jpeg)

```
<!DOCTYPE html>
<html lang="it">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Galleria di Immagini</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      text-align: center;
      margin: 0;
      padding: 0;
    }
    .gallery {
      display: grid;
      grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
      gap: 10px;
      padding: 20px;
    }
    .gallery img {
      width: 100%;
      height: auto;
      border: 2px solid #ddd;
      border-radius: 5px;
    }
  </style>
</head>
<body>
```

```
<h1>Galleria di Immagini JPEG</h1>
<div class="gallery">
  
  
  
  
  
  
  
  
  
  
</div>
</body>
</html>
```

```
<a href="image1.jpg" target="_blank">
  
</a>
```

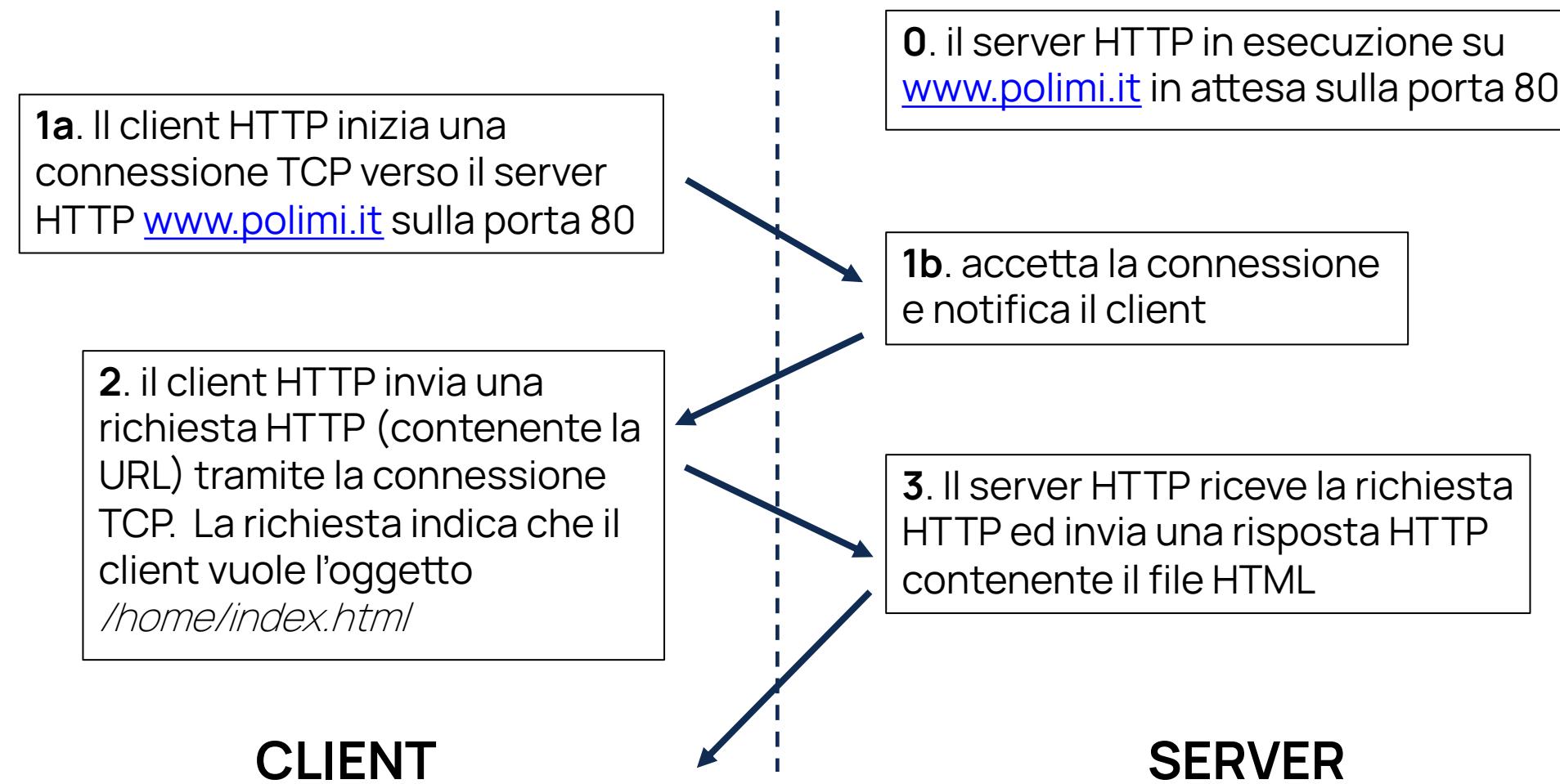
→ Immagini aperte automaticamente

→ Immagini aperte con un clic

# Esempio di connessione non persistente

- L'utente digita nel browser la URL: **www.polimi.it/home/index.html**

(Esempio: l'HTML contiene testo e riferimenti a 10 immagini jpeg)



# Esempio di connessione non persistente

- L'utente digita nel browser la URL: **www.polimi.it/home/index.html**  
*(Esempio: l'HTML contiene testo e riferimenti a 10 immagini jpeg)*

**4.** Il client HTTP riceve il messaggio di risposta contenente il file html e scopre dei collegamenti a 10 oggetti JPEG

**5.** Il server HTTP chiude la connessione TCP.

*I passi da 1 a 5 sono ripetuti per ognuna delle 10 immagini JPEG indicate dal file HTML*

*tempo*

**CLIENT**

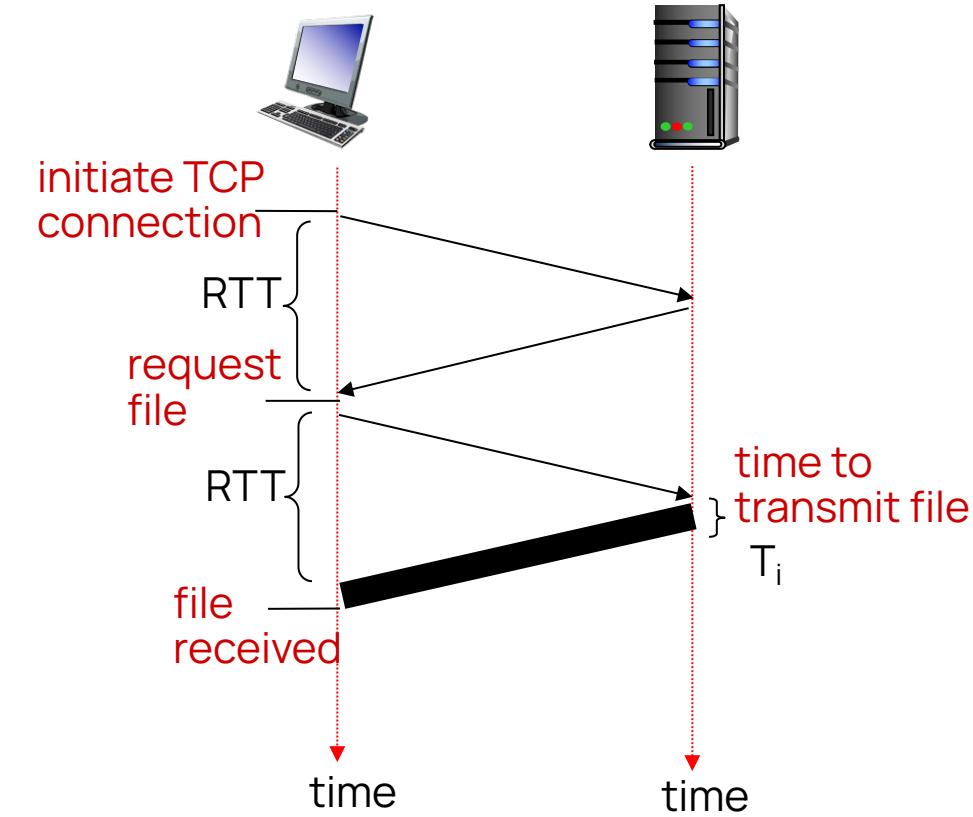
**SERVER**

# Stima del tempo di trasferimento in HTTP

- ❑ Round trip Time (RTT): tempo per trasferire un messaggio “piccolo” dal client al server e ritorno
- ❑ Tempo di risposta di HTTP:
  - ❑ un RTT per iniziare la connessione TCP
  - ❑ un RTT per inviare i primi byte della richiesta HTTP e ricevere i primi byte di risposta
  - ❑ tempo di trasmissione dell’oggetto (file HTML, immagine, ecc..) – i.e.,  $T_i$
- ❑ Supponendo che la pagina web sia composta da 11 oggetti (un file HTML e 10 immagini JPEG), il tempo di download dell’intera pagina web è:

$$T_{nonpers} = \sum_{i=0}^{10} (2RTT + T_i)$$

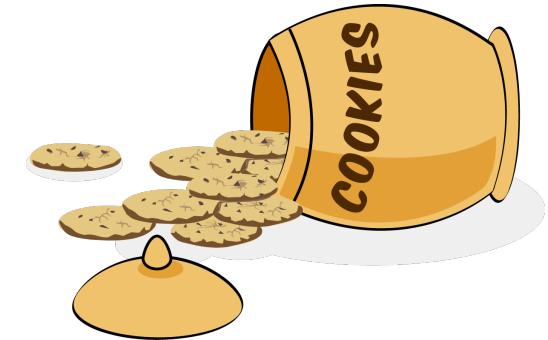
$$T_{pers} = RTT + \sum_{i=0}^{10} (RTT + T_i)$$



# Mantenere uno 'stato' in HTTP: i cookies

## Ingredienti dei cookies:

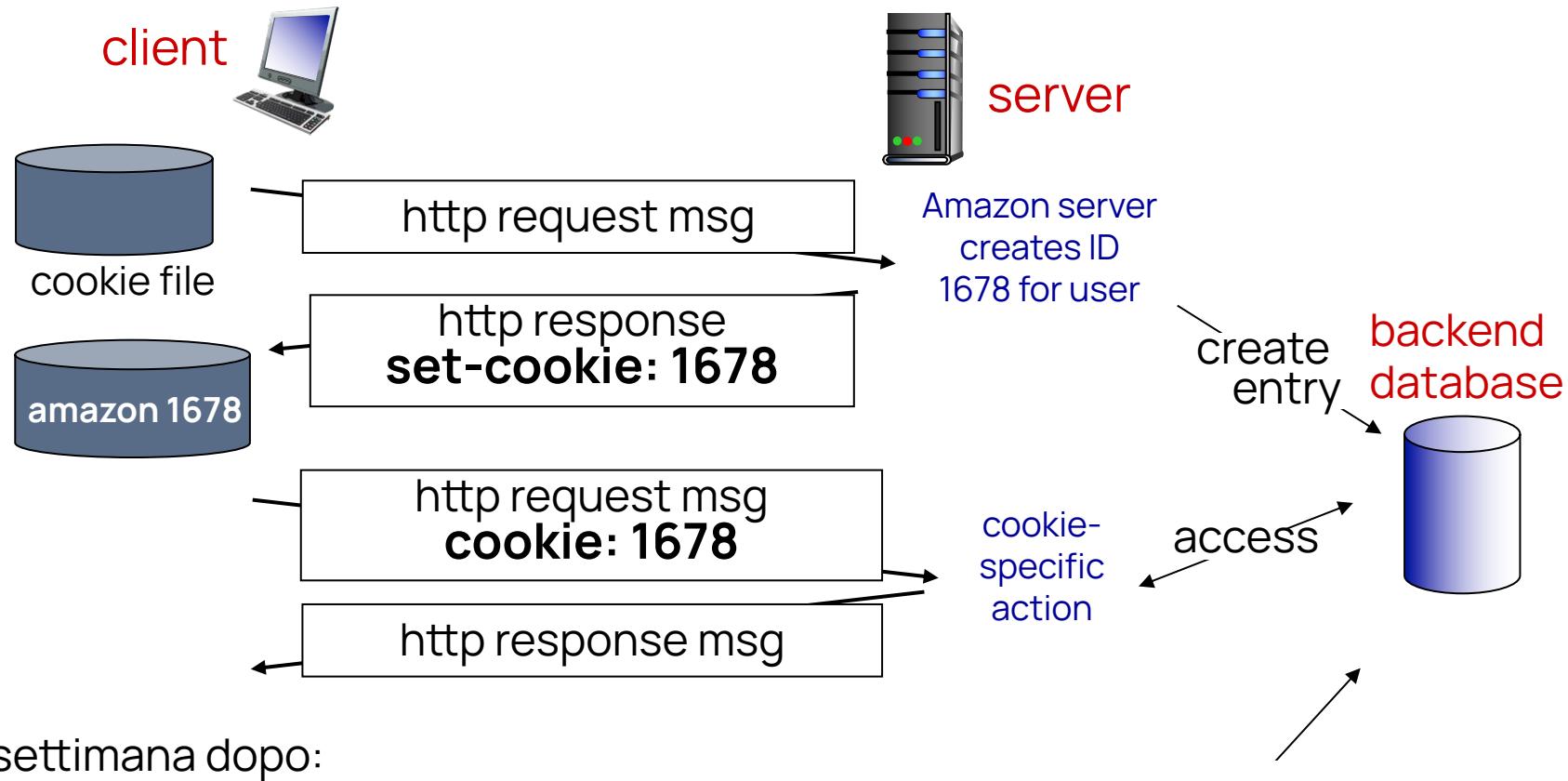
1. Un header "cookie" nelle risposte HTTP
2. Un header "cookie" nelle successive richieste HTTP
3. Una lista di cookie mantenuta sull'host (dal browser)
4. Un database di cookie mantenuto dal sito web



## Esempio:

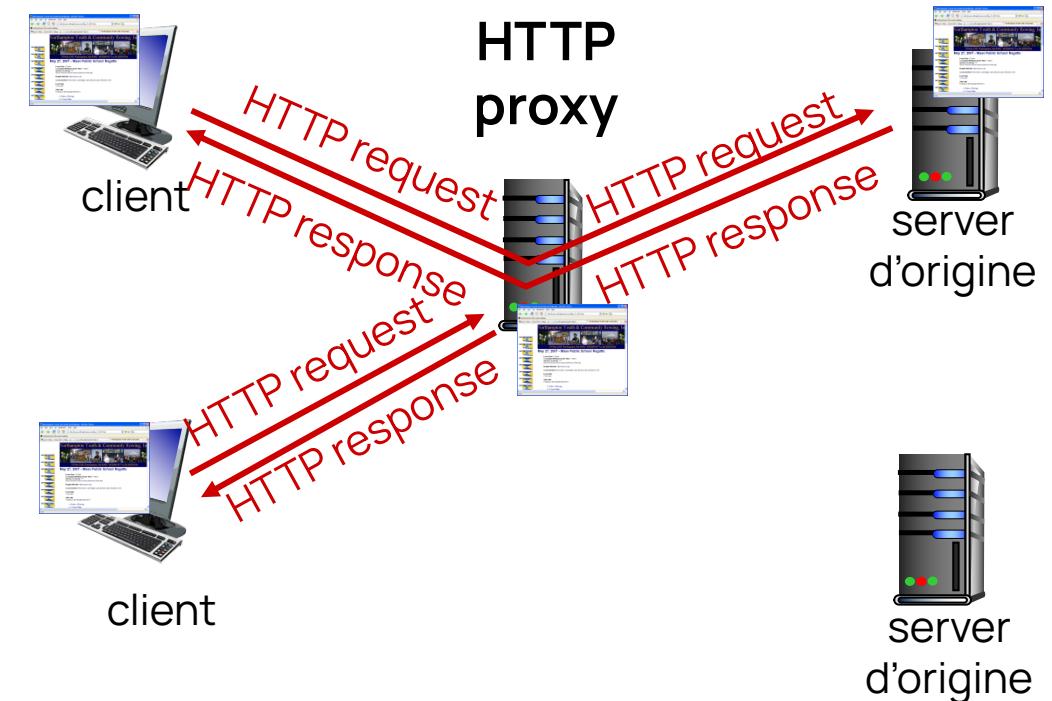
- Matteo visita un sito di e-commerce per la prima volta
- Quando il sito riceve la prima richiesta HTTP, il sito genera:
  - un ID unico
  - una entry nel data base locale che corrisponde all'ID creato
- Quando Matteo torna nel sito, invierà una richiesta HTTP con l'header cookie associato in precedenza che era stato salvato nel suo host (browser nello smartphone, laptop, etc.)
- Il web server lo riconoscerà

# Esempio di uso dei cookies



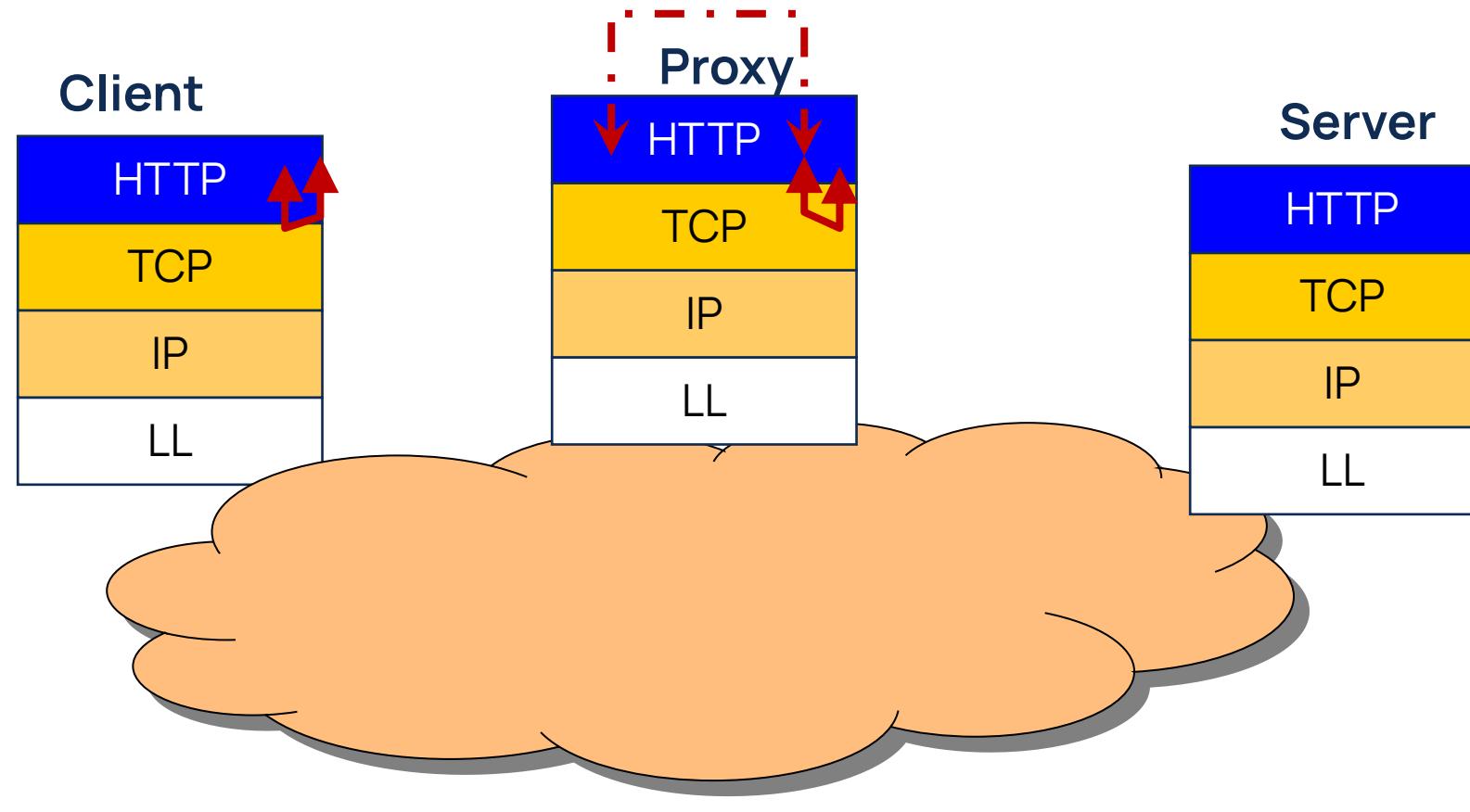
# I proxy HTTP: Cache di rete

- ❑ **Obiettivo:** rispondere alle richieste HTTP senza coinvolgere il server HTTP
  - ❑ Il client HTTP invia tutte le richieste HTTP ad un proxy HTTP:
    - ❑ se l'oggetto richiesto è disponibile nella cache (memoria temporanea) del proxy server, il proxy server risponde con l'oggetto
    - ❑ altrimenti il proxy recupera l'oggetto dal server d'origine e lo restituisce al client
- ❑ La cache può anche essere attiva lato **client**, nel browser locale (smartphone, laptop, etc.)



# La comunicazione HTTP

- I proxy sono degli application gateway, ovvero degli instradatori di messaggi di livello applicativo
- Devono essere sia client (verso i server d'origine) che server (verso i client)
- Il server vede arrivare tutte le richieste dal proxy (mascheramento degli utenti del proxy)

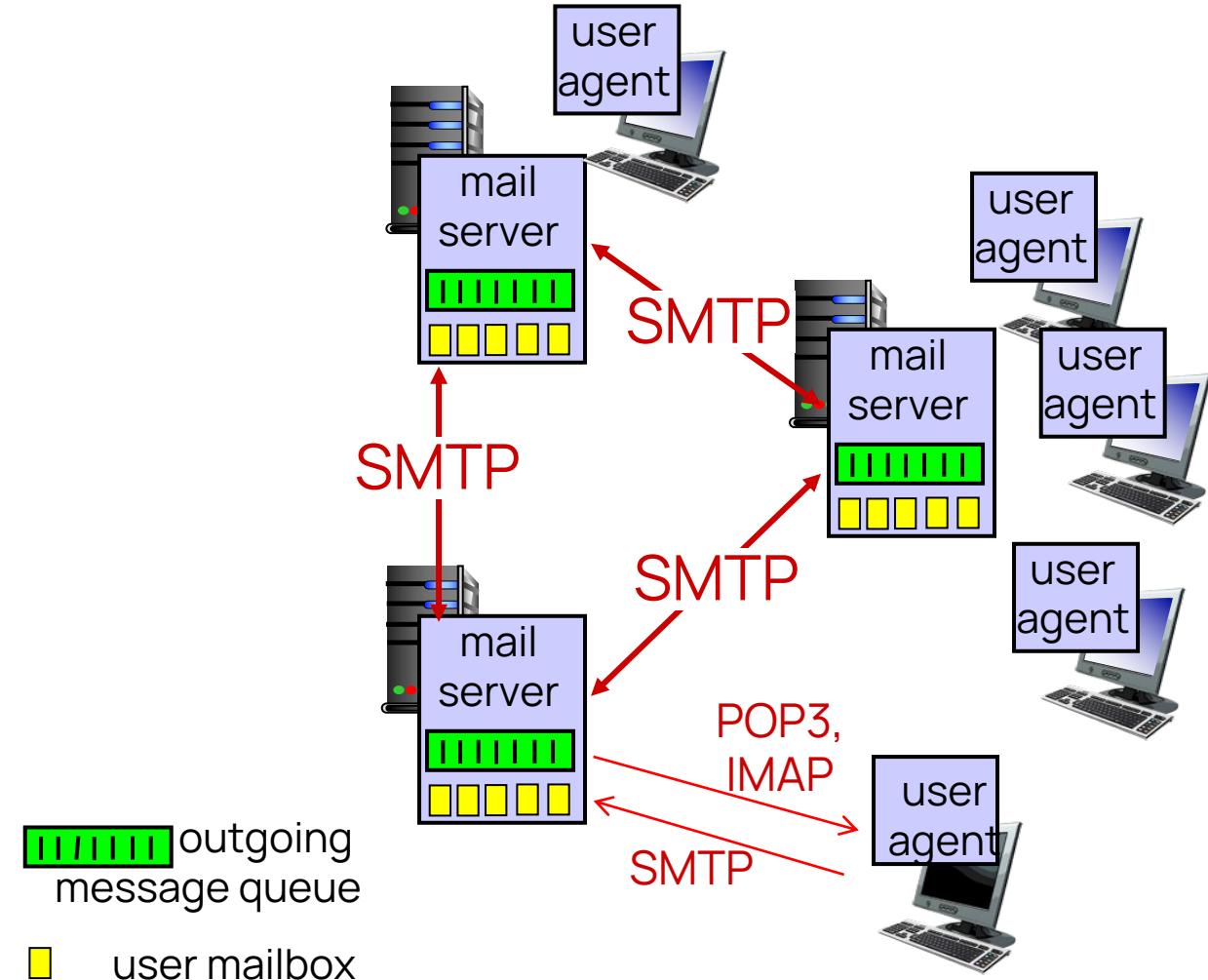


# POSTA ELETTRONICA

03

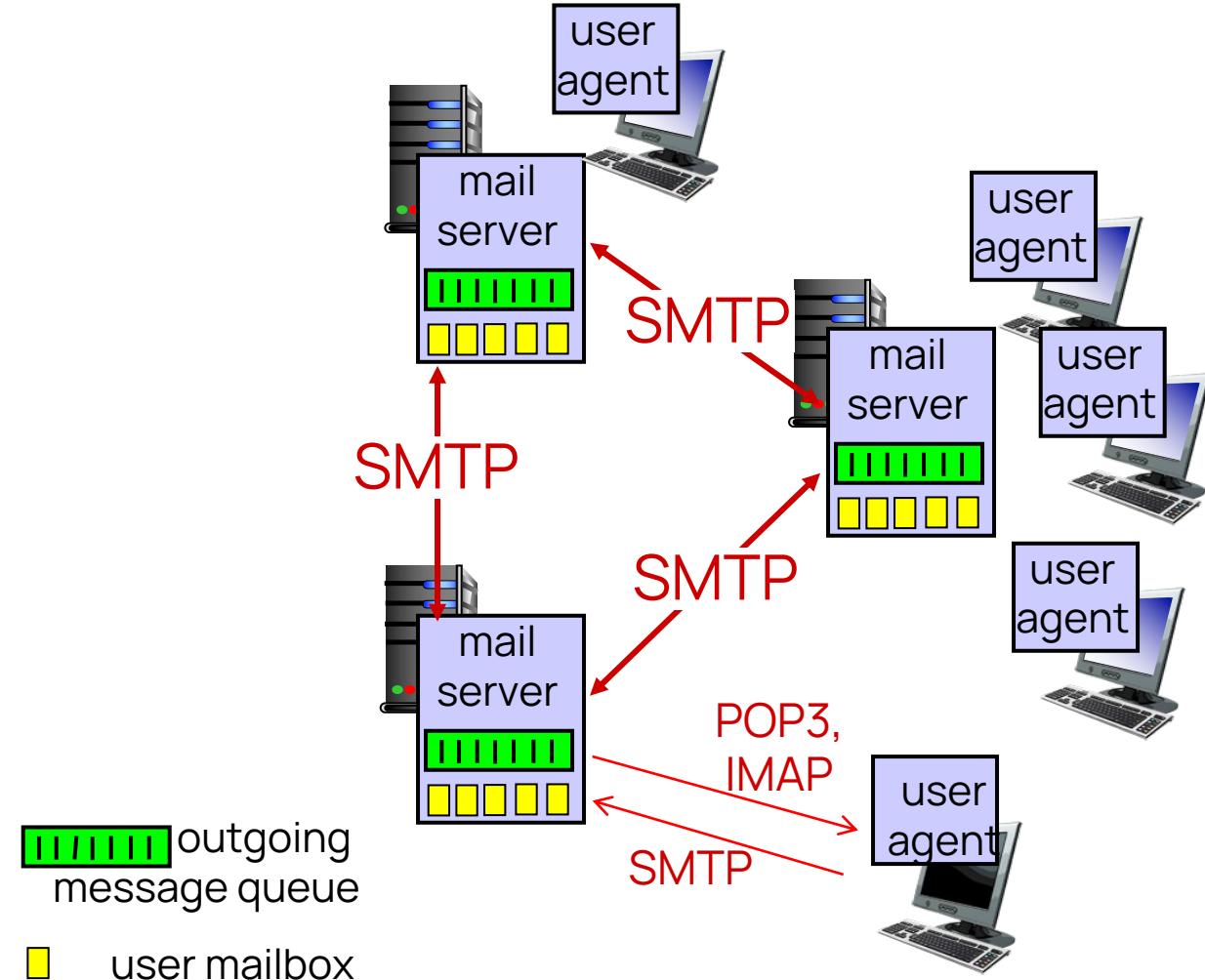
# Il servizio di e-mail

- Client d'utente aka User Agent
  - GMail, Outlook, Thunderbird, etc.
- Mail Server
- Simple Mail Transfer Protocol (**SMTP**): per trasferire email dal client d'utente fino al mail server del destinatario
- Protocolli di accesso ai mail server: per “scaricare” email dal proprio mail server (**POP3, IMAP**)



# I Mail Server

- Per ogni client controllato i mail server **contengono**:
  - una coda di email in ingresso (mailbox)
  - una coda di email in uscita
- I mail server **ricevono**:
  - le mail in uscita da tutti i client d'utente che "controllano"
  - tutte le mail destinate ai client d'utente controllati da altri mail server
- I mail server "**parlano**"
  - SMTP con altri mail server e con i client d'utente in **uplink**
  - POP3/IMAP con i client d'utente in **downlink**

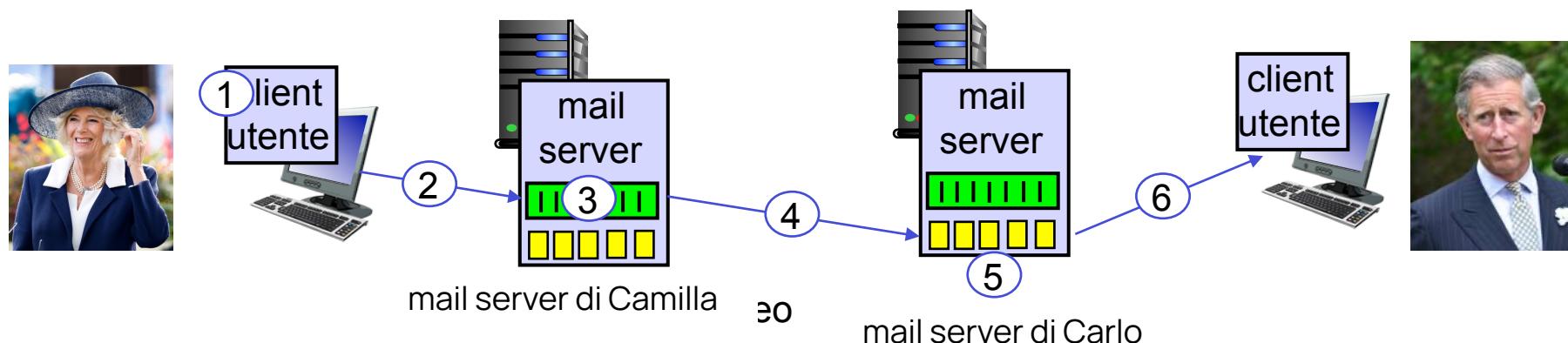


# SMTP: Simple Mail Transfer Protocol

- ❑ E' un protocollo applicativo client-server
- ❑ Quando un mail server riceve un messaggio da un client d'utente
  - ❑ mette il messaggio in una coda
  - ❑ apre una connessione TCP con la **porta 25** del mail server del destinatario
  - ❑ trasferisce il messaggio
  - ❑ chiude la connessione TCP
- ❑ L'interazione tra client SMTP e server SMTP è di tipo comando/risposta
- ❑ Comandi e risposte sono testuali

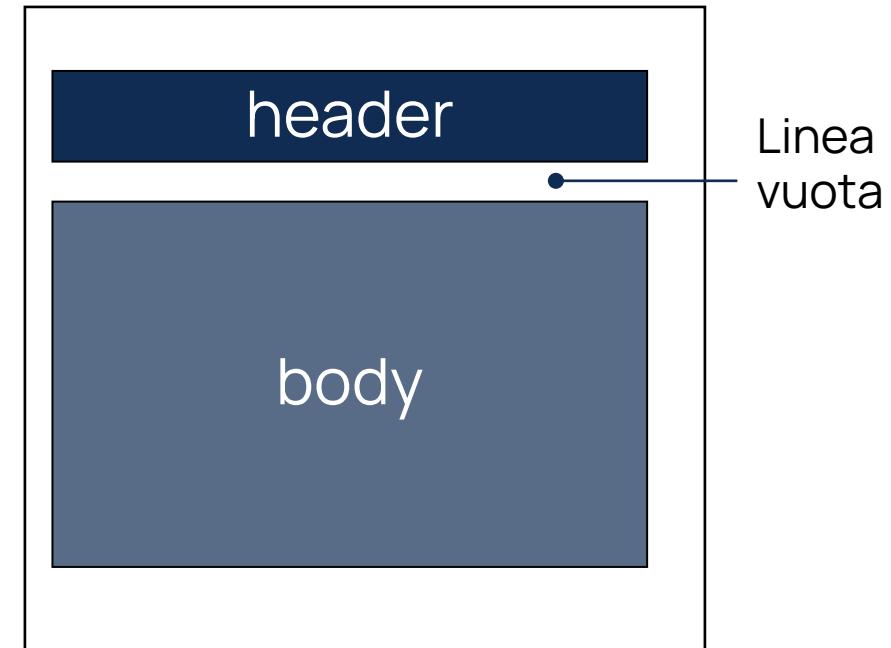
# Esempio di trasferimento SMTP

1. Camilla compone una email destinata a Carlo carlo@miomailserver.com
2. Il client d'utente di Camilla invia la mail al proprio mail server
3. Il mail server di Camilla si comporta come client SMTP ed apre una connessione TCP (porta 25) con il mail server di Carlo
4. Il client SMTP (mail server di Camilla) invia la email sulla connessione TCP
5. Il mail server di Carlo memorizza la mail nella mailbox di Carlo
6. Carlo (in modo asincrono) usa il proprio client d'utente per leggere la mail



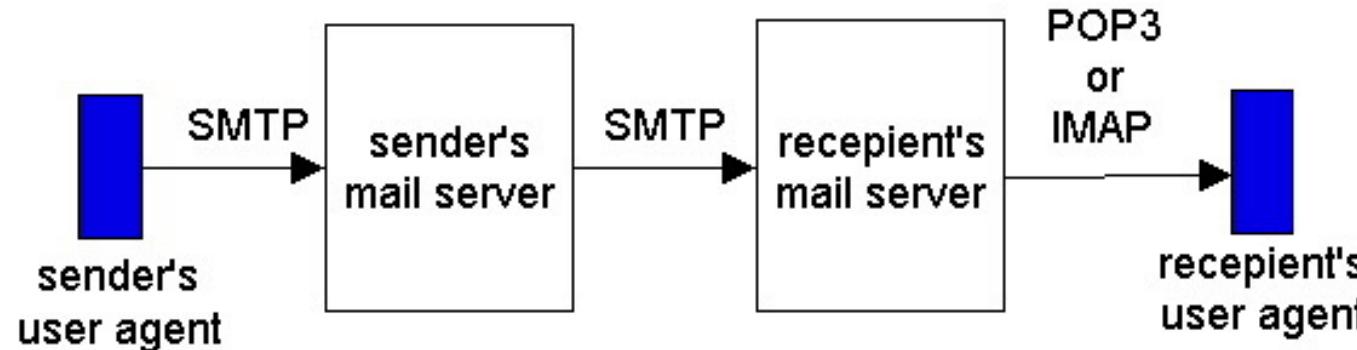
# Il formato delle email (RFC 822)

- ❑ Il formato dei messaggi inviati (tutto ciò che segue il comando SMTP DATA) è specificato
- ❑ **Header:**
  - ❑ To:
  - ❑ From:
  - ❑ Subject:
- ❑ **Body:** il contenuto dell'email



# Protocolli di accesso al mail-box

- ❑ Diversi protocolli sono stati sviluppati per il colloquio tra user agent e server in fase di lettura dei messaggi presenti nel mailbox
  - ❑ **POP3** (Post Office Protocol versione 3, RFC 1939) : download dei messaggi
  - ❑ **IMAP** (Internet Mail Access Protocol, RFC 1730) : download dei messaggi, gestione della mailbox sul mail server



# Fondamenti di TELECOMUNICAZIONI

Prof. Marco Mezzavilla  
[marco.mezzavilla@polimi.it](mailto:marco.mezzavilla@polimi.it)