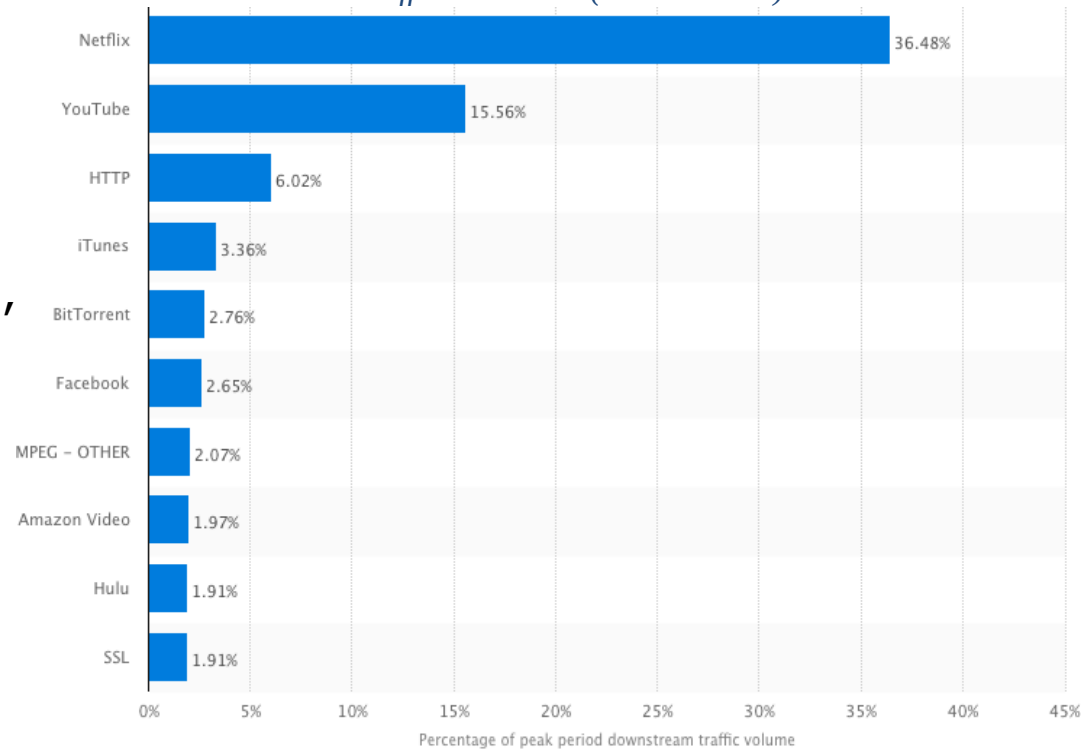
Il Livello Applicativo

Processi e socket, Web, Mail, DNS, peer-to-peer

Alcune applicazioni di rete

- **World Wide Web**
  - HTTP
- **Posta elettronica**:
  - SMTP, Gmail
- **Social networking**:
  - Facebook, Twitter, Instagram , Snapchat, ecc.. (social networking)
- **P2P file sharing**: BitTorrent, eMule, ecc..
- **Video streaming**:
  - NetFlix, YouTube, Hulu
- **Telefonia**:
  - Skype, Hangout, ecc..
- **Network games**
- **Video conference**
- **Massive parallel computing**
- **Instant messaging**
- **Remote login:**
  - TELNET
- …

Traffico Downlink (NordAmerica)



Source: www.statista.com (2016)

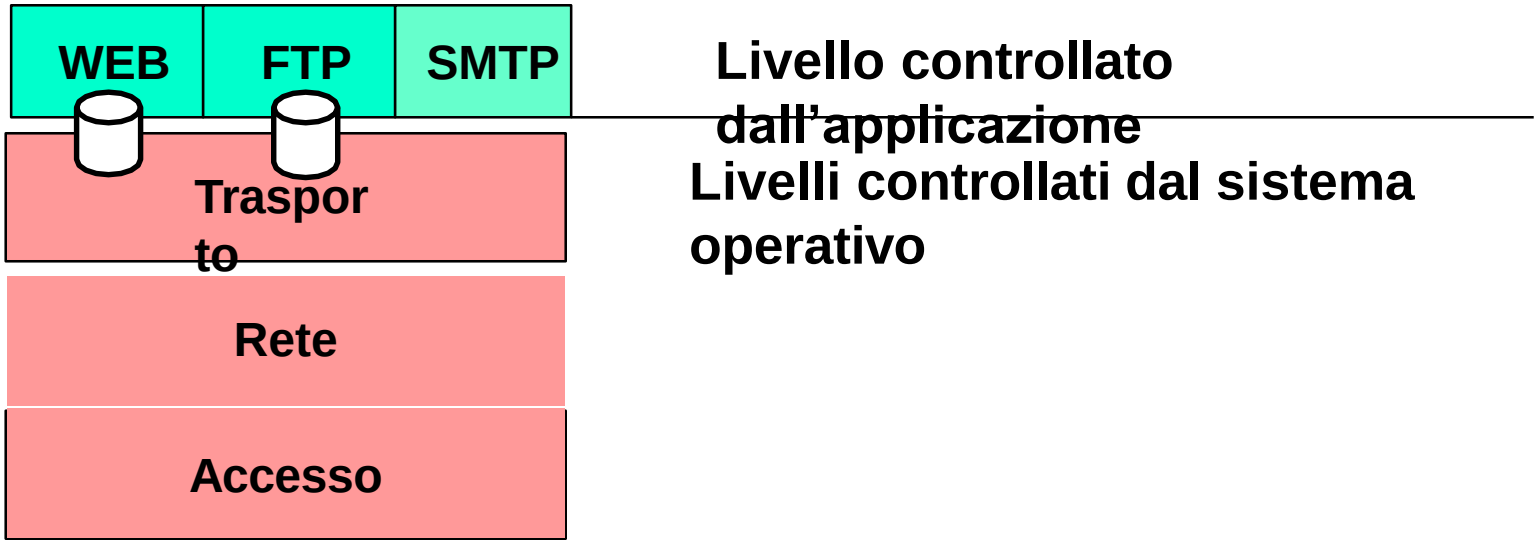
Ingredienti per la comunicazione tra processi remoti

- 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

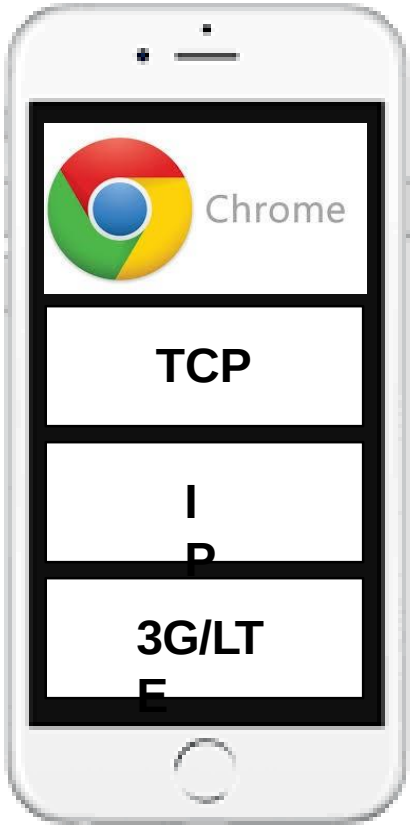
Indirizzamento di processi applicativi

- 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



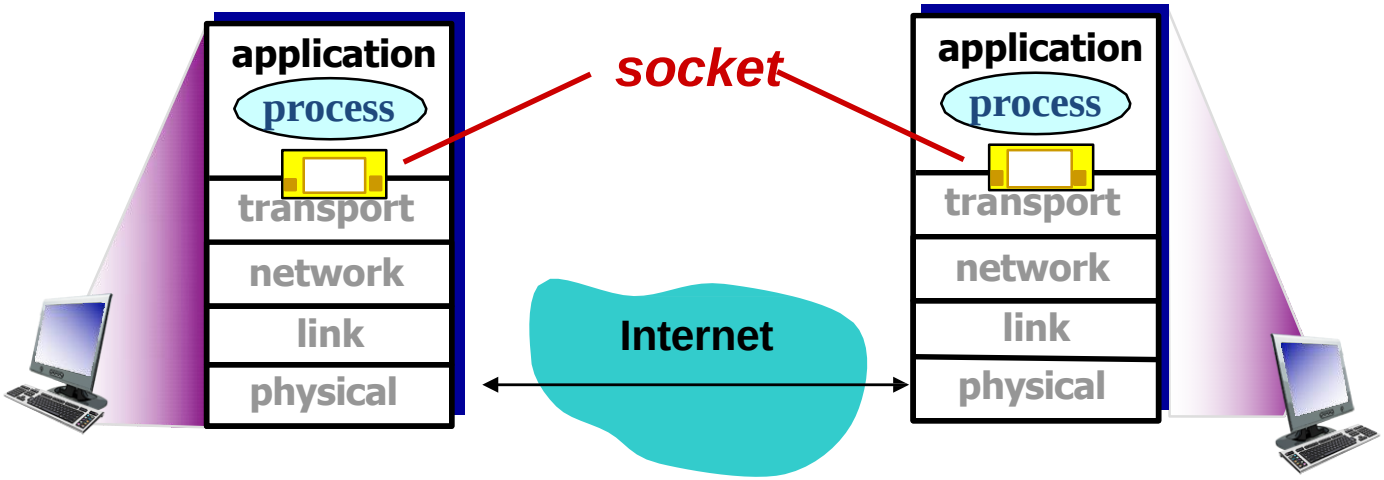
Indirizzamento di processi applicativi

- 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
- Le ***socket*** sono delle porte di comunicazione
  - Il processo trasmittente mette il messaggio fuori dalla porta
  - La rete raccoglie il messaggio e lo trasporta fino alla porta del destinatario
- Attività di laboratorio su ***socket programming***



Requisiti delle applicazioni

- Affidabilità
  - Alcune applicazioni possono tollerare perdite parziali (ad es. audio)
  - Altre applicazioni richiedono a completa affidabilità (ad es. file transfer, telnet)
- Ritardo
  - Alcune applicazioni richiedono basso ritardo (ad es. *Internet telephony, interactive games*)
- Banda
  - Alcune applicazioni richiedono un minimo di velocità di trasferimento (ad es. appl. multimediali)
  - Altre applicazioni si adattano alla velocità disponibile (“appl. elastiche”)

Quale servizio di trasporto scegliere

Servizio TCP:

- *connection-oriented:* instaurazione connessione prima delle scambio dati
- *Trasporto affidabile* senza perdita di dati
- *Controllo di flusso:* il trasmettitore regola la velocità in base al ricevitore
- *Controllo di congestione:* per impedire di sovraccaricare la rete
- *Non fornisce:* garanzie di ritardo e di banda

Servizio UDP:

- Trasferimento non affidabile
- senza connessione
- senza controllo sul traffico
- senza garanzie
- Trasferimento "veloce"

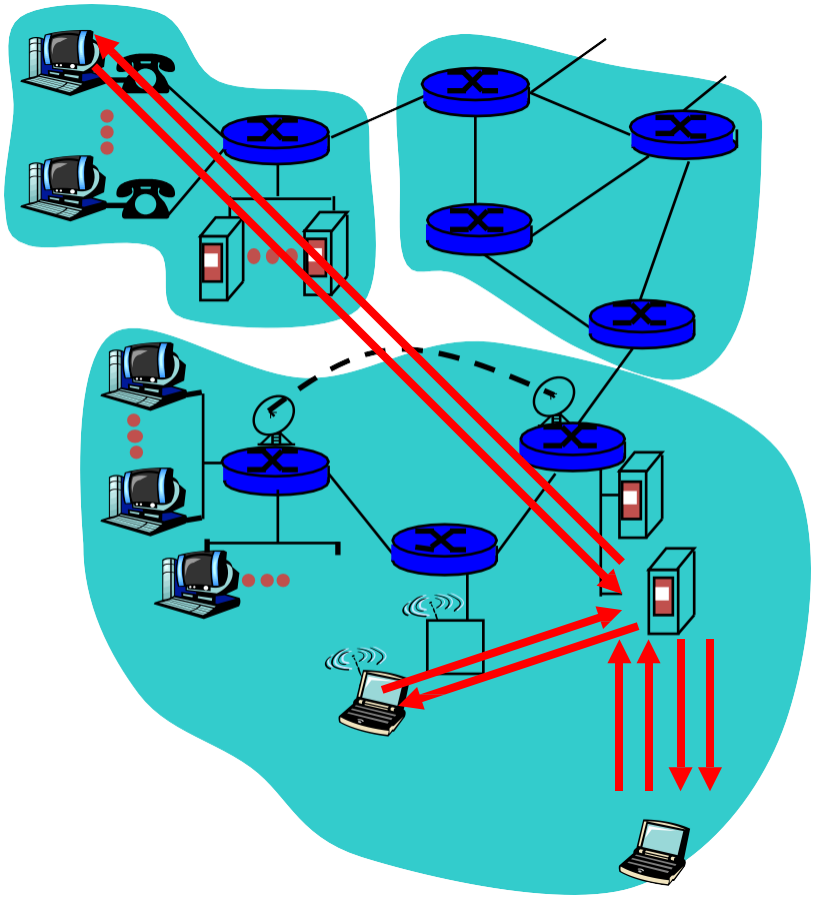
Architetture applicative

- ***Client-server***
  - I dispositivi coinvolti nella comunicazione implementano o solo il processo *client* o solo il processo *server*
  - I dispositivi *client* *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 tutti sia il processo *client* che quello *server*

- ***Ibrida***

Architettura *client-server*



Server:

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

Client:

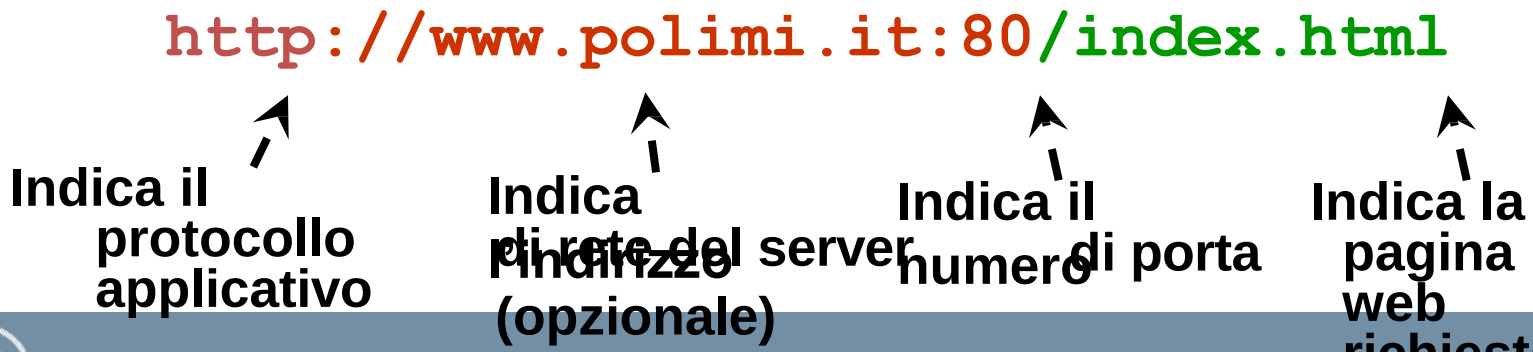
- Comunicano con il *server*
- Possono essere connessi in modo discontinuo
- Possono cambiare indirizzo IP
- Non comunicano con altri client
- Possono inviare molte richieste allo stesso *server*

Il servizio di Web Browsing

Hyper Text Transfer Protocol (HTTP)
RFC 1945, 2616

Cosa contengono i messaggi HTTP - le Pagine Web

- Breve ripasso:
  - le *pagine web* sono fatte di *oggetti*
  - gli *oggetti* possono essere file HTML file, 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:



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

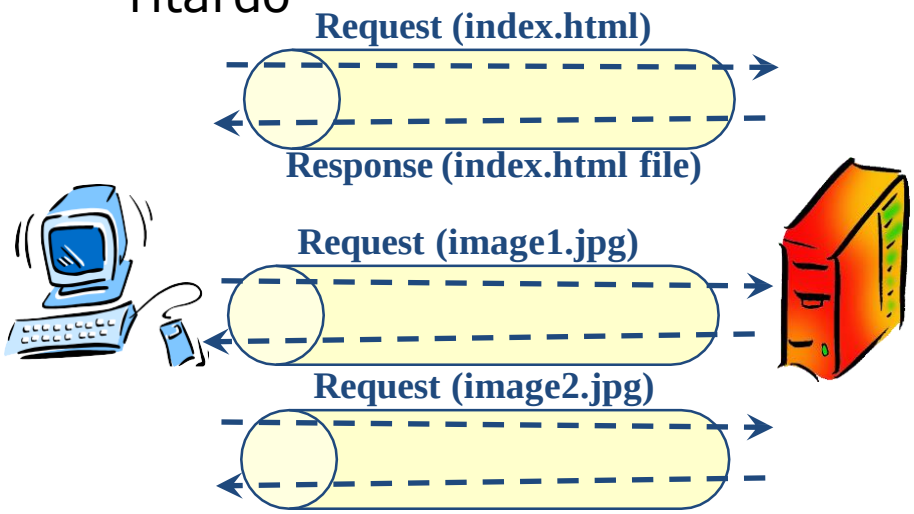
- HTTP si appoggia su TCP a livello di trasporto:
  1. Il client HTTP inizia una connessione TCP verso il server (porta 80)
  2. Il server HTTP accetta connessioni TCP da client HTTP
  3. ***Client* e *Server* HTTP si scambiano informazioni (pagine web e messaggi di controllo)**
  4. La connessione TCP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

Vedremo 1, 2 e 4 nei laboratori su socket programming

Modalità di connessione tra *client* e *server* HTTP

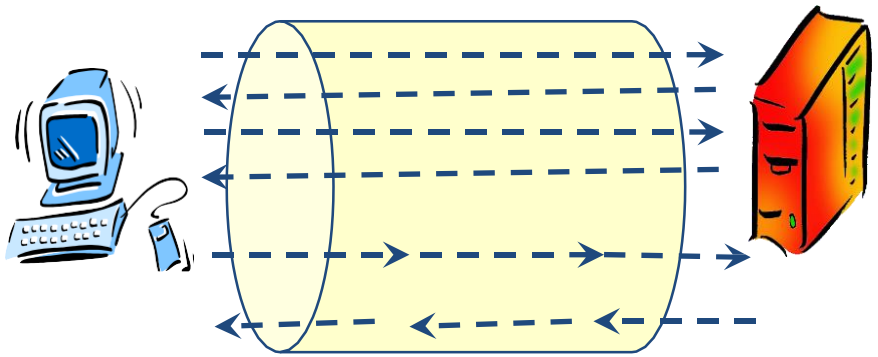
Connessione non persistente

- 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 (default mode HTTP v1.1)



Esempio di connessione non persistente

L'utente digita nel browser la URL:

`www.polimi.it/home/index.html`

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

1a. l client HTTP inizia una connessione TCP verso il server HTTP www.polimi.it sulla porta 80

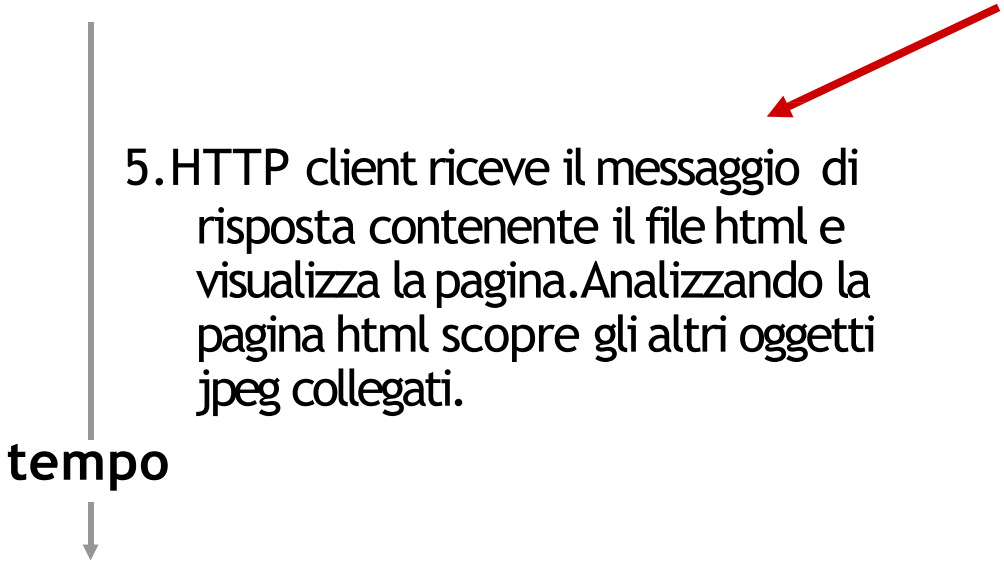
1b. il server HTTP in esecuzione su www.polimi.it in attesa sulla porta 80 accetta la connessione e notifica il client

2 il client HTTP invia una richiesta HTTP (contenente la URL) tramite la connessione TCP. La richiesta indica che il client vuole l'oggetto /home/index.html

3 Il server HTTP riceve la richiesta HTTP ed invia una risposta HTTP contenente il file HTML

tempo

Esempio di connessione non persistente



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

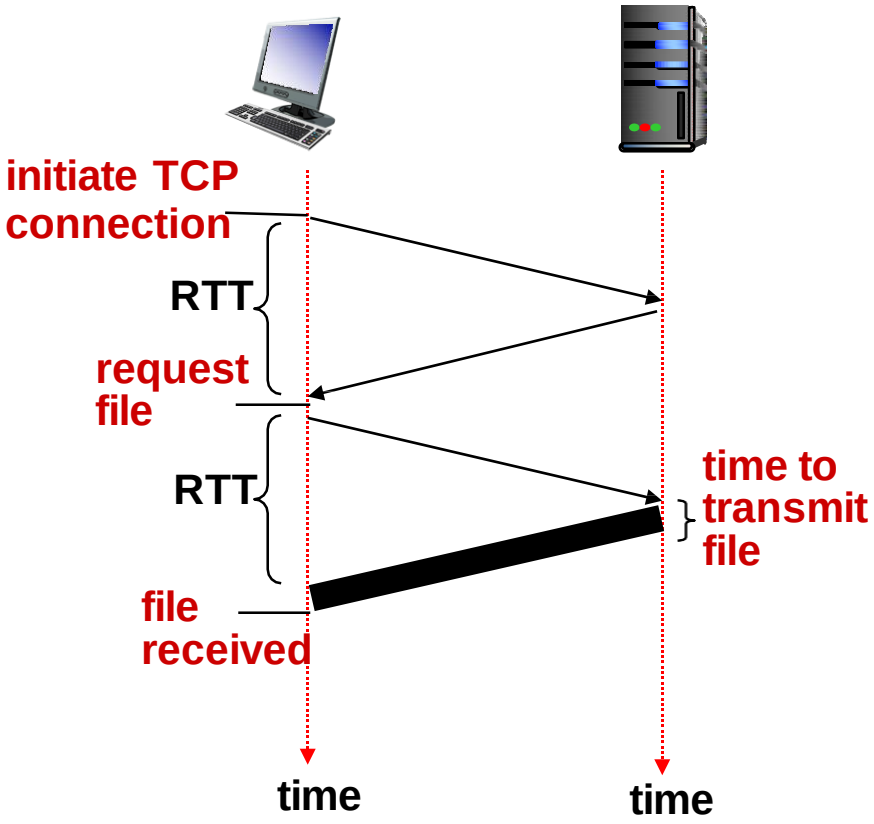
Stima del tempo di trasferimento in HTTP

- *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..)

- Supponendo che la pagina web sia composta da 10 oggetti (un file HTML e 9 immagini JPEG), il tempo di download dell'intera pagina web è:

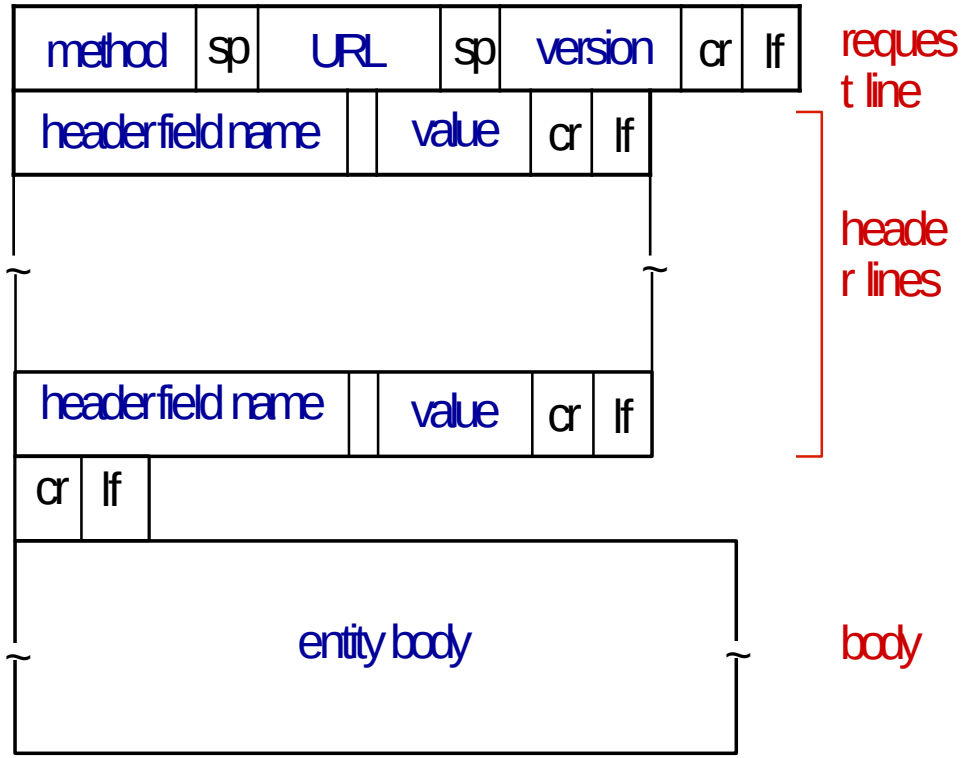
$$T_{nonpers} = \sum_{i=0}^{10} (2RTT + T_i)$$

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



Le richieste HTTP

- I messaggi HTTP sono codificati in ASCII (*human-readable*)




```
GET /index.html HTTP/1.1\r\n
Host: www-net.cs.umass.edu\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```

Esempi di Metodi HTTP

GET	E' usato quando il client vuole scaricare un documento dal server. Il documento richiesto è specificato nell'URL. Il server normalmente risponde con il documento richiesto nel corpo del messaggio di risposta.
HEAD	E' usato quando il client non vuole scaricare il documento ma solo alcune informazioni sul documento (come ad esempio la data dell'ultima modifica). Nella risposta il server non inserisce il documento ma solo degli header informativi.
POST	E' usato per fornire degli input al server da utilizzare per un particolare oggetto (di solito un applicativo) identificato nell'URL.
PUT	E' utilizzato per memorizzare un documento nel server. Il documento viene fornito nel corpo del messaggio e la posizione di memorizzazione nell'URL.
DELETE	cancella il documento specificato nella URL

Esempi di *Header* HTTP

- Gli *header* servono per scambiare informazione di servizio aggiuntiva
- E' possibile inserire più linee di *header* per messaggio

Header name : **Header value**

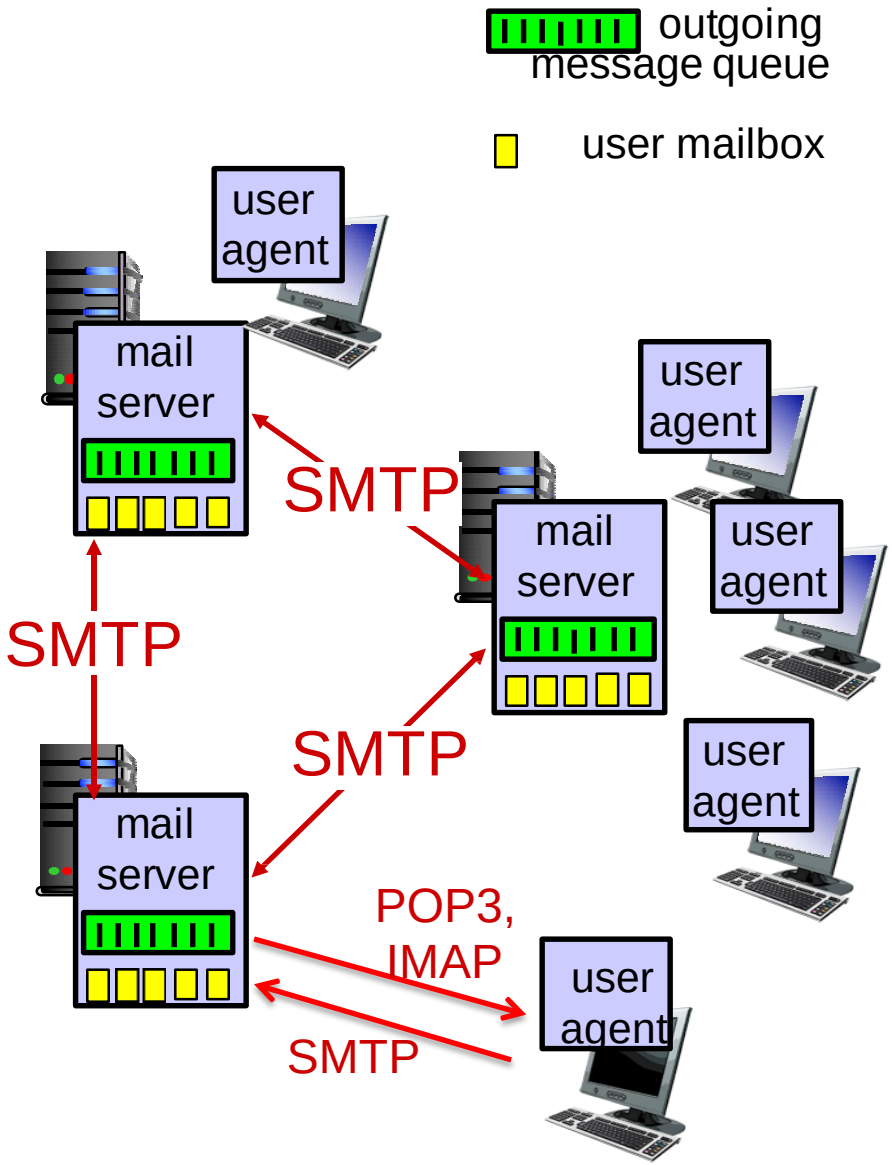
`Cache-control`	Informazione sulla cache
`Accept`	Formati accettati
`Accept-language`	Linguaggio accettato
`Authorization`	Mostra i permessi del client
`If-modified-since`	Invia il doc. solo se modificato
`User-agent`	Tipo di user agent

Il servizio di posta elettronica

Simple Mail Transfer Protocol (SMTP) RFC 5321
Post Office Protocol (POP3) RFC 1939
Internet Mail Control Protocol (IMAP) RFC 3501

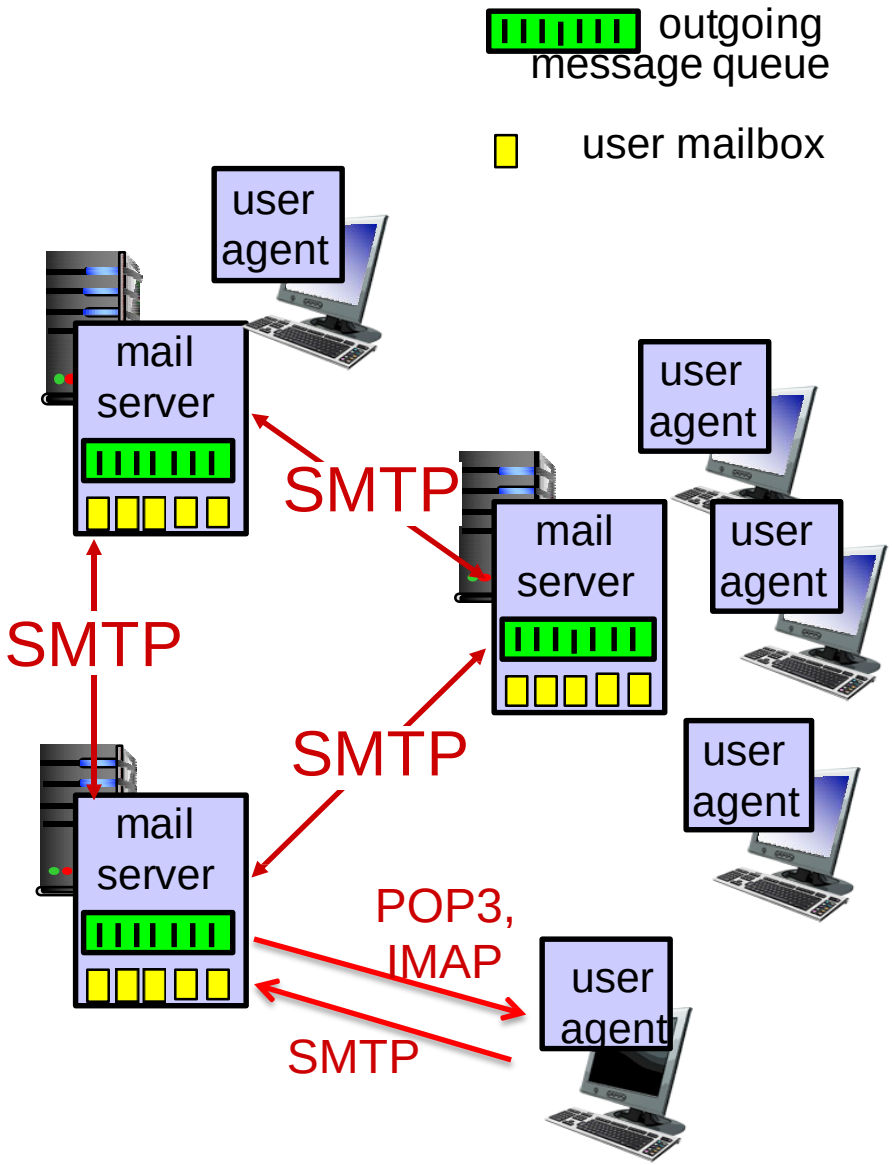
Il servizio di E-mail

- **Client** d'utente aka **User Agent** (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*

- I mail server contengono per ogni client controllato:
  - 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"
  - Ricevono da altri mail server tutte le mail destinate ai client d'utente controllati
- 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

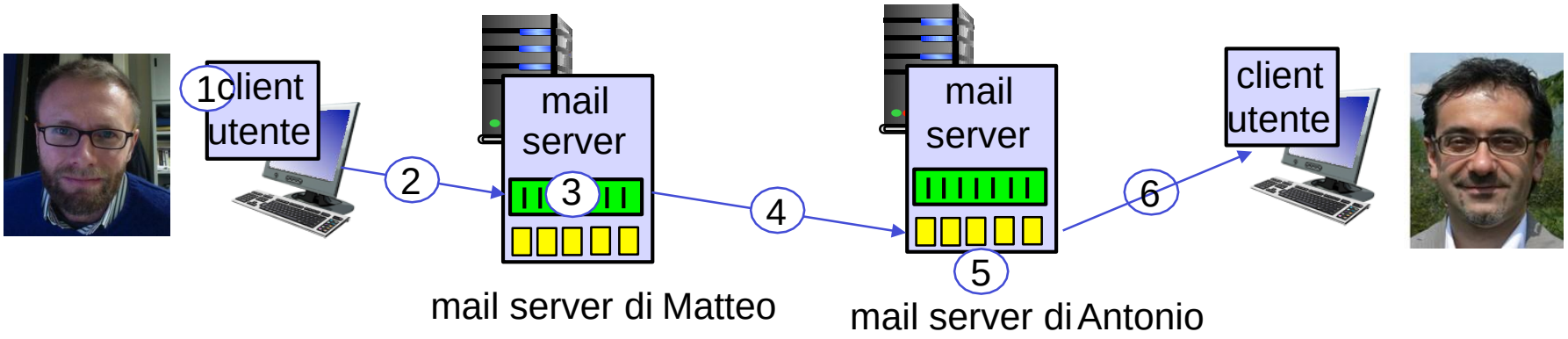


SMTP

- 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 e di tipo comando/risposta
- Comandi e risposte sono testuali
- Richiede che anche il corpo dei messaggi sia ASCII
  - i documenti binari devono essere convertiti in ASCII 7-bit

Esempio di trasferimento SMTP

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



Colloquio tra client e server SMTP

apertura

```
S: 220 antoniomailserver.com
C: HELO matteomailserver.com
S: 250 Hello matteomailserver.com, pleased to meet you
C: MAIL FROM: <matteo@matteomailserver.com>
```

invio

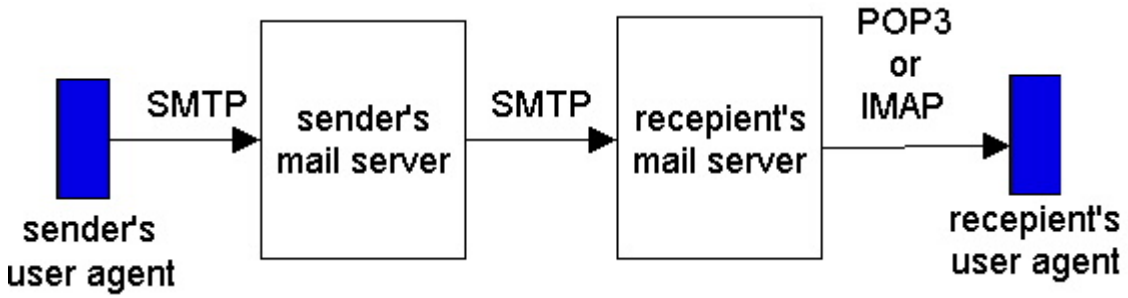
```
S: 250 matteo@matteomailserver.com... Sender ok
C: RCPT TO: <antonio@antoniomailserver.com>
S: 250 antonio@antoniomailserver.com ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Oggi corri al Giuriati?
C: .
S: 250 Message accepted for delivery
```

chiusura

```
C: QUIT
S: 221 antoniomailserver.com closing connection
```

Protocolli di accesso al mailbox

- 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*
  - HTTP



Risoluzione di nomi simbolici

DNS

Domain Name System (DNS)

- Gli indirizzi IP (32 bit) sono poco adatti ad essere usati dagli applicativi

- E' più comodo utilizzare indirizzi simbolici:

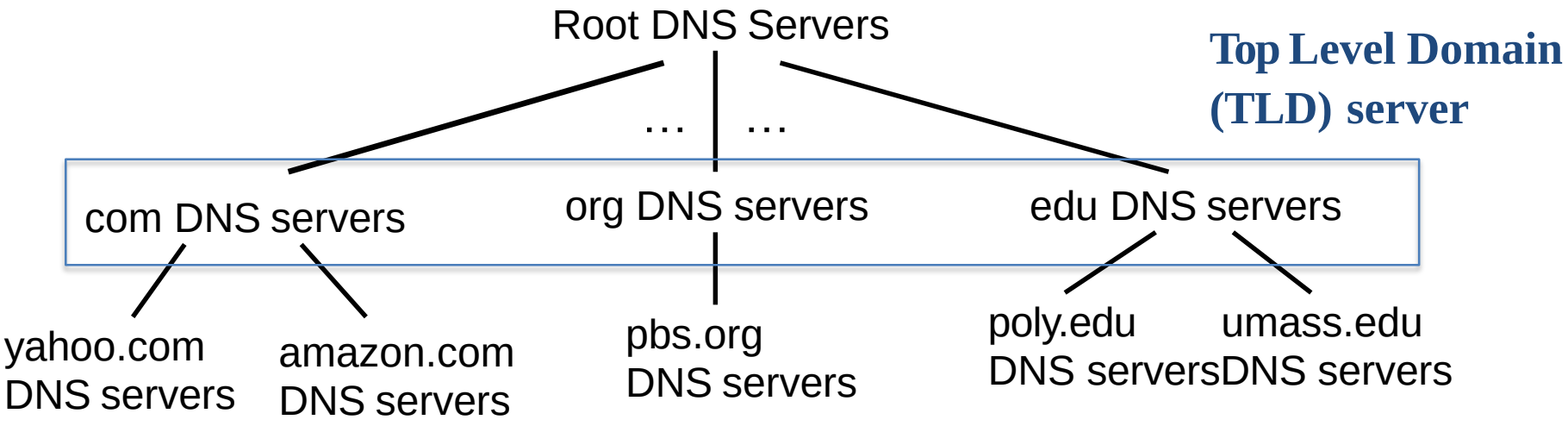
  – E' meglio www.google.com o 74.125.206.99?

- Occorre una mappatura fra indirizzi IP (usati dalle macchine di rete) ed i nomi simbolici (usati dagli esseri umani)

Domain Name System (DNS)

- Ingredienti
  - Database distribuito costituito da molti *name servers* con organizzazione gerarchica
  - Protocollo applicativo basato su UDP tra *name server* e *host* per risolvere nomi simbolici (tradurre nomi simbolici in indirizzi IP)
- Servizi aggiuntivi
  - *host aliasing*
  - *Mail server aliasing*
  - *Load distribution*

Database distribuito e gerarchico



- Ogni livello nella gerarchia ha diversa “profondità” di informazione
- Esempio: un utente vuole l’IP di www.google.com
  - *Root name server sanno come “trovare” i name server che gestiscono i domini .com*
  - *I name server .com sanno come trovare il name server che gestisce il dominio google.com*
  - *I name server google.com sanno risolvere il nome simbolico www.google.com*

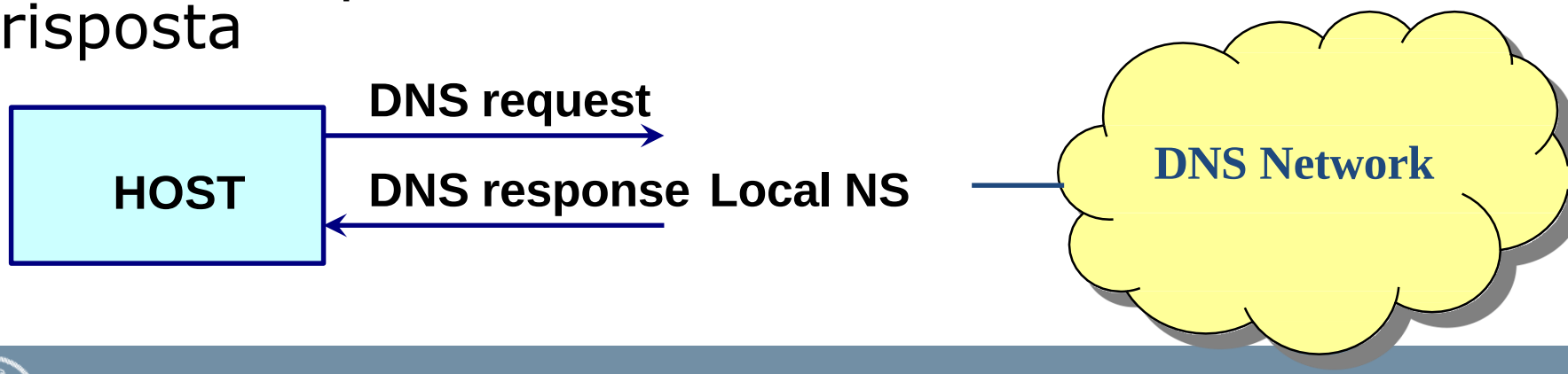
Altri tipi di NS

- *Local Name Servers*
  - Ogni ISP (residenziale, università, compagnia) ha un NS locale
  - Direttamente collegati agli *host*
  - Tutte le volte che un *host* deve risolvere un indirizzo simbolico contatta il Local *Name Server*
  - Il *Local Name Server* (eventualmente) contatta i *Root Name Server* nella gerarchia

- *Authoritative Name Servers*
  - NS "responsabile" di un particolare *hostname*

Come ottenere un mappaggio

- Ogni *host* ha configurato l'indirizzo del LNS
- Le applicazioni che richiedono un mappaggio (browser, ftp, etc.) usano le funzioni del DNS
- Una richiesta viene inviata al server DNS usando UDP come trasporto
- Il server reperisce l'informazione e restituisce la risposta



Formato dei messaggi DNS

- identification: identificativo coppia richiesta/risposta
- flag: richiesta/risposta, authoritative/non auth., iterative/recursive
- number of: relativo al numero di campi nelle sez. successive
- questions: nome richiesto e tipo (di solito A o MX)
- answers: resource records completi forniti in risposta
- authority: contiene altri record forniti da altri server
- additional infor.: informazione addizionale, ad es. il record con l'IP ADDR. per il MX fornito in answers