Università di Roma Tor Vergata
Corso di Laurea triennale in Informatica

Sistemi operativi e reti

A.A. 2018-2019

Pietro Frasca

Parte II: Reti di calcolatori

Lezione 6 (30)

Giovedì 21-03-2019

Servizi forniti dai protocolli di trasporto

- Lo strato di trasporto di Internet ha due protocolli (dal 2000 è presente anche **SCTP**):

  UDP *(User Datagram Protocol)*
  TCP *(Transmission Control Protocol).*

- Per realizzare un'applicazione di rete, è necessario decidere dapprima se scegliere di usare UDP o TCP. I due protocolli offrono un modello di servizio molto diverso alle applicazioni.

Servizi TCP

- Il TCP fornisce alle applicazioni un servizio orientato alla connessione e un servizio di trasferimento affidabile dei dati.
- ***Servizio orientato alla connessione****:* Prima che due processi applicativi inizino a comunicare tra loro, il TCP effettua una procedura iniziale detta **handshake** ("**stretta di mano**"), terminata la quale si instaura una connessione TCP fra i **socket** dei due processi.
- La comunicazione è di tipo **full-duplex** che consente ai due processi di inviare contemporaneamente messaggi in entrambe le direzioni. La comunicazione può essere chiusa sia dal client che dal server.
- ***Servizio di trasporto affidabile****: Il TCP* garantisce ai processi che tutti i dati trasmessi arrivano a destinazione senza errori e nello stesso ordine di partenza.

- Il TCP fornisce anche un servizio di **controllo della congestione**, un servizio per regolare il traffico sulla rete. Il **controllo della congestione** del TCP modula la velocità di trasmissione dei dati di un processo (client o server) quando la rete è congestionata.
- Con il TCP la velocità di trasmissione è regolata dal servizio di controllo della congestione, che riduce la velocità media di trasmissione del mittente quando la rete è congestionata. Pertanto, il TCP non garantisce una velocità minima di trasmissione dei dati e quindi non garantisce anche un ritardo minimo.

Servizi UDP

- L’UDP è un protocollo di trasporto semplice.
- L’UDP è un protocollo *senza connessione*, quindi non c’è la fase di handshake prima che i due processi inizino a comunicare.
- L’UDP fornisce un servizio di trasferimento dati non affidabile. Pertanto, i messaggi che arrivano al socket ricevente possono non arrivare in ordine o non arrivare per niente.
- L’UDP non implementa il servizio di controllo della congestione, così un processo può inviare dati nel socket fino alla massima velocità consentita.
- L’UDP è più adeguato per applicazioni soft real-time le quali generalmente possono tollerare qualche perdita di dati, ma richiedono che la velocità di trasmissione non scenda sotto una determinata soglia.
- Come il TCP, l’UDP non dà garanzie sul ritardo.

Applicazioni	Protocollo dello strato di applicazione	Protocollo di trasporto sottostante
Posta elettronica	SMTP (RFC 821)	TCP
Accesso a terminale remoto	Telnet (RFC 854)	TCP
Web	HTTP (RFC 2616)	TCP
Trasferimento di file	FTP (RFC 959)	TCP
File server remoto	NFS (McKusik 1996)	UDP o TCP
Streaming multimediale	Spesso proprietario (per esempio, Real Networks)	UDP o TCP
Telefonia Internet	Spesso proprietario (per esempio, Dialpad)	Tipicamente UDP

Applicazioni di rete

- Descriveremo quattro applicazioni oggi molto diffuse in Internet: il **Web**, **FTP** (**trasferimento di file)**, la **posta elettronica** e il **DNS** ***(Domain Name System,*** **Sistema dei nomi**). Inoltre parleremo, di alcuni tecnologie e protocolli usati nelle applicazioni di **condivisione di file da pari a pari (P2P, Peer to Peer).**

Il Web

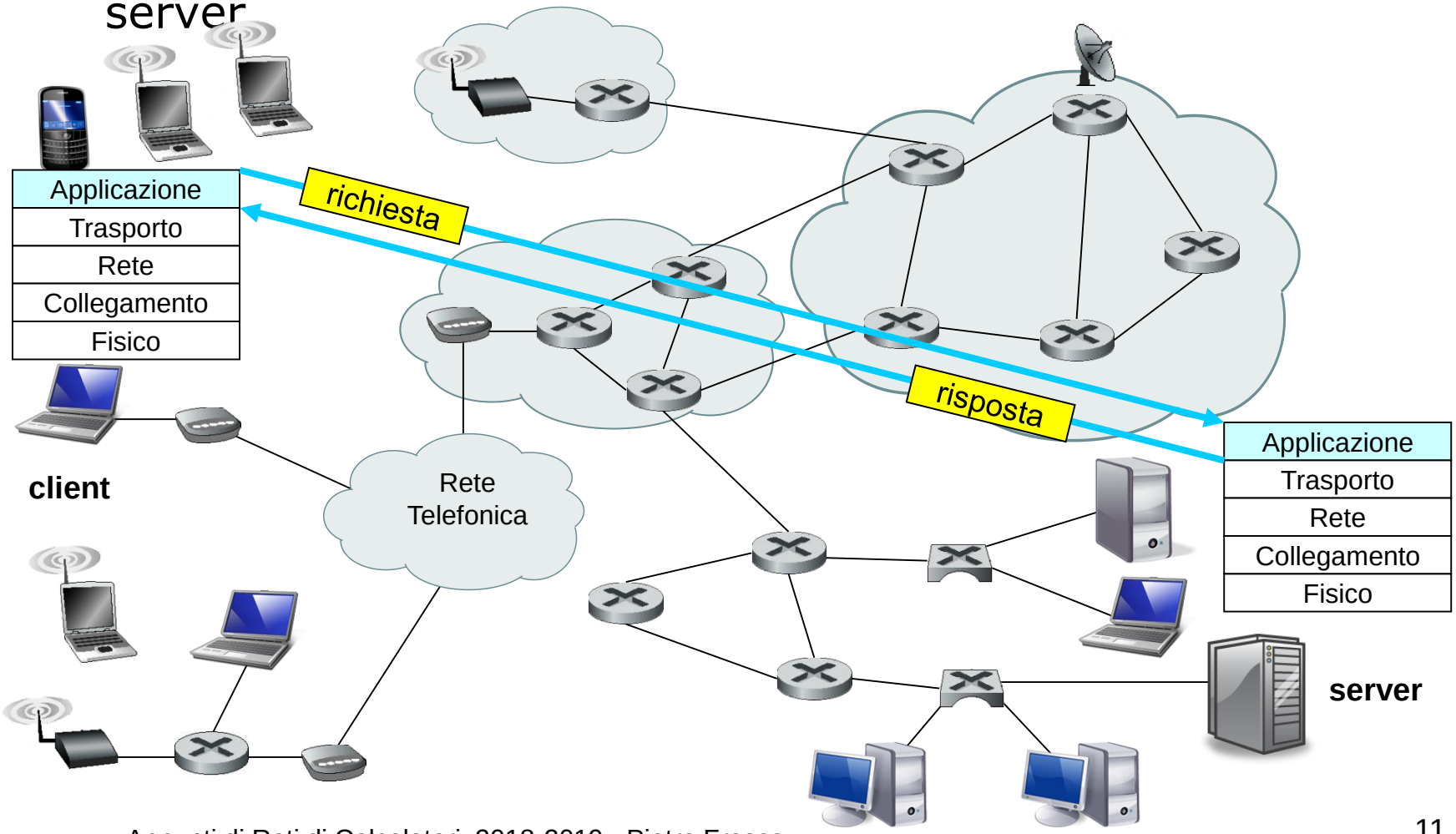
- Fino al 1990 Internet era usata soprattutto usata in ambito universitario per il collegamento a host remoti, per trasferire dati da host locali a host remoti e viceversa, per accedere a notizie e per scambiarsi messaggi di posta elettronica.
- All'inizio degli anni '90, fu realizzata l'applicazione **World Wide Web (Web).** L'enorme crescita delle tecnologie di rete, la diffusione del Web e delle sue applicazioni (web application) hanno cambiato profondamente il modo di interagire delle persone, sia all'interno che all'esterno dell'ambiente lavorativo, cambiando vari comportamenti e modalità di lavorare delle persone.
- Il Web ha permesso la nascita di migliaia di società. Ha portato Internet a diventare praticamente l'unica e sola rete per dati. Prima, soprattutto presso le università e le grandi società, erano molto diffuse le reti SNA di IBM e DECNET di Digital (ora HP).

- Una delle caratteristiche più importanti del Web è che permette la fruizione delle informazioni a ***richiesta***. Gli utenti accedono alle informazioni e ai contenuti multimediali quando lo desiderano. Questo funzionamento è molto diverso dalle trasmissioni radio/televisive, che richiedono agli utenti di sintonizzarsi in determinati orari per ottenere informazioni che sono trasmesse in base ad un palinsesto programmato.
- Oltre a essere a richiesta, il Web ha molte altre interessanti caratteristiche. Consente agli utenti di pubblicare informazioni in modo molto semplice ed economico.
- I **motori di ricerca** aiutano nella ricerca delle informazioni.
- Una **pagina web** (**documento**) è formata da file di vario formato, come file di testo HTML, immagine JPEG, video divx, mp4, etc.) che sono indirizzabili attraverso **URL (Uniform Resource Locator).**

- Il Web utilizza vari standard e tecnologie:
  - Il **protocollo HTTP *(HyperText Transfer Protocol*)**. E' il protocollo dello strato di applicazione Web, che consente il trasferimento di file (hyperText) tra server web e browser.
  - **HTML e CSS** per la formattazione dei documenti;
  - i **browser** (per esempio, Google Chrome, Netscape Navigator, Mozilla Firefox, Safari e Microsoft Internet Explorer) che sono il lato client dell'applicazione web.
  - i **server Web** (per esempio, i server Apache, IIS (Internet Information Services) di Microsoft e Netscape) che sono il lato server dell'applicazione web.
  - **URL (URI)**, (Uniform Resource Locator) identificatore di risorse.
  - **DBMS** (Sistemi di gestione di basi di dati)
  - **Linguaggi di programmazione (lato client)** come Java (applet), JavaScript, VBScript, etc.
  - **Linguaggi di programmazione (lato server)** come java, JavaScript, php, VBScript (asp), etc, per la generazione dinamica delle pagine

Lati client e server di un'applicazione web

- Un'applicazione web ha due "lati", un **lato client** e un **lato server**. Un browser implementa il lato client del protocollo HTTP, e un server Web ne implementa il lato server.

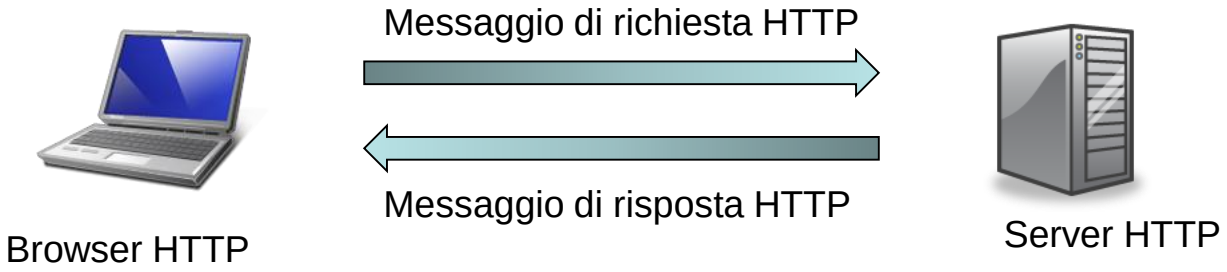


- Per esempio, se una pagina Web contiene testo HTML e cinque immagini JPEG, allora la pagina Web è composta da sei file: il file base HTML più le cinque immagini.
- Il file base HTML contiene gli URL (indirizzi) dei file. Ciascun URL ha tre componenti fondamentali: il **protocollo** utilizzato, il **nome** *(o l'indirizzo IP)* del server che contiene gli oggetti e il nome del **percorso (path)** per raggiungere il file.
- Per esempio, l'URL **http://www.pf.uniroma2.it/reti/lezioni/lezione5.html** ha come hostname **www.pf.uniroma2.it** e **/reti/lezioni/lezione4.html** come nome del path del file html e come protocollo http.
- Un browser visualizza la pagina Web richiesta e consente molte caratteristiche di navigazione e configurazione. I server Web implementano il lato server dell'HTTP e memorizzano i file web, ciascuno indirizzabile da un URL.
- Server Web molto diffusi sono Apache, Microsoft Internet Information Server e Netscape Enterprise Server.

Il protocollo HTTP

- L'**HTTP (Hypertext Transfer Protocol)** è il protocollo dello strato di applicazione del Web. L'HTTP è implementato in due parti: un programma **client** e uno **server** i quali comunicano tra loro, scambiandosi **messaggi di richiesta e di risposta**.
- La prima versione HTTP/1.0 fu realizzata nel 1991. Nel 1997 il protocollo fu aggiornato alla versione HTTP/1.1 che eliminò alcuni problemi e limitazioni della versione precedente.
- Le due versioni sono compatibili ed entrambe usano il **TCP** come protocollo dello strato di trasporto.
- Le due versioni dell'HTTP sono descritte negli [RFC 1945] e [RFC 2616] rispettivamente per le versioni 1.0 e 1.1.
- HTTP/2 è la nuova versione di HTTP. E' stato sviluppato dal Working Group Hypertext Transfer Protocol dell'Internet Engineering Task Force. E' stato proposto come standard nel dicembre 2014.

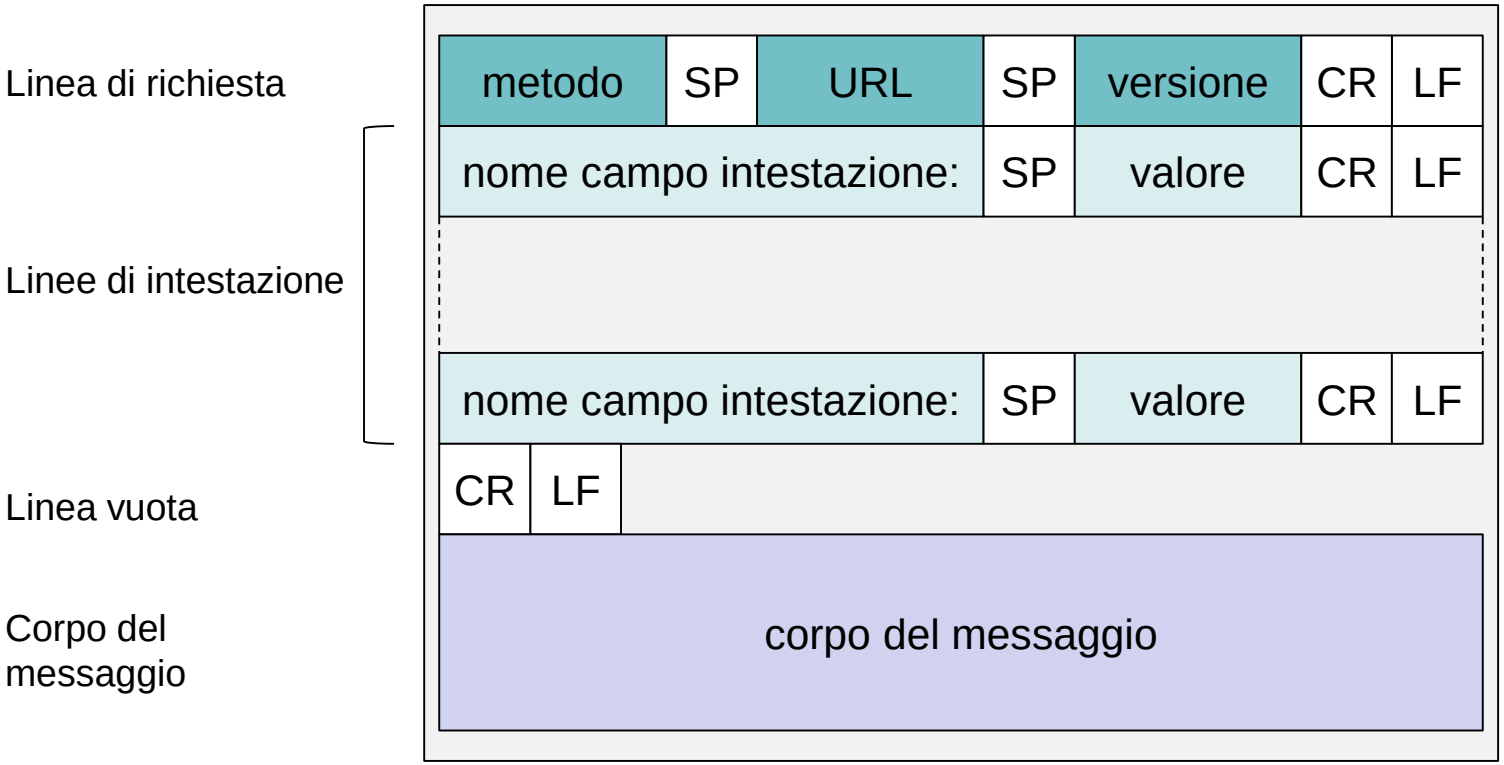
- Il funzionamento, a grandi linee, dell'interazione tra client e server è illustrata in figura.



- Quando un utente richiede una pagina Web (per esempio, cliccando su un hyperlink), il browser crea e invia messaggi di richiesta HTTP per gli oggetti nella pagina al server. Il server riceve la richiesta e risponde con messaggi di risposta HTTP contenenti gli oggetti richiesti.
- E' importante notare che l'HTTP non gestisce alcuna informazione relativa ai client che richiedono file al server. Dato che un server non conserva le informazioni relative al client è detto **protocollo senza stato** *(stateless protocol).*

Formato del messaggio HTTP

- Ci sono due tipi di messaggi HTTP, **messaggi di richiesta** e **messaggi di risposta**.
- Il formato del messaggio di richiesta è illustrato nella figura seguente.



Esempio di messaggio di richiesta HTTP

- Un tipico messaggio di richiesta HTTP:

```
GET /lezioni/lez2.html HTTP/1.1
Host: www.pf.uniroma2.it
Accept-Language: it-IT
User-Agent: Mozilla/4.0
Connection: close
```

- Il messaggio di richiesta è scritto in formato ASCII che è leggibile. Quindi, non è garantita la riservatezza delle informazioni.
- Un messaggio di richiesta può avere un numero di linee variabile, anche una sola linea se la versione HTTP è la 1.0.
- La prima linea di un messaggio di richiesta HTTP è detta **linea di richiesta** *(request line).*
- Le linee successive prendono il nome di **linee di intestazione** *(header line).*
- La linea di richiesta ha tre campi, separati da uno spazio (SP):
  - **il campo metodo**
  - **il campo URL**
  - **il campo versione dell'HTTP**.
- Il campo metodo può assumere vari valori, tra cui **GET**, **POST** e HEAD.

- Il metodo GET è il metodo più usato. Consente al browser di richiede un oggetto, specificandolo nel campo URL.
- Nel nostro messaggio di esempio, la linea di richiesta indica che il browser richiede la pagina `/lezioni/lez2.html` e che viene utilizzata la versione HTTP/1.1.
- Ora guardiamo le linee di intestazione dell'esempio.

  - **Host: www.pf.uniroma2.it** specifica il nome del server sul quale è memorizzato l'oggetto richiesto.
  - **Accept-Language: it-IT**, indica che il browser è configurato per richiedere preferibilmente una versione del file in italiano, se disponibile.
  - **User-Agent: Mozilla/4.0**, specifica il nome del browser utilizzato per la richiesta. Mozilla/4.0 è un browser prodotto da Netscape. Questa linea di intestazione è utile per problemi di compatibilità, in quanto il server potrebbe inviare differenti versioni dello stesso file a differenti tipi di browser per ottenere una corretta visualizzazione.

- **Connection: close**, indica che il browser richiede al server di utilizzare la connessione non persistente, nonostante che il browser che invia questo messaggio di richiesta implementi HTTP/1.1 che per default funziona con le connessioni persistenti (keep-alive).

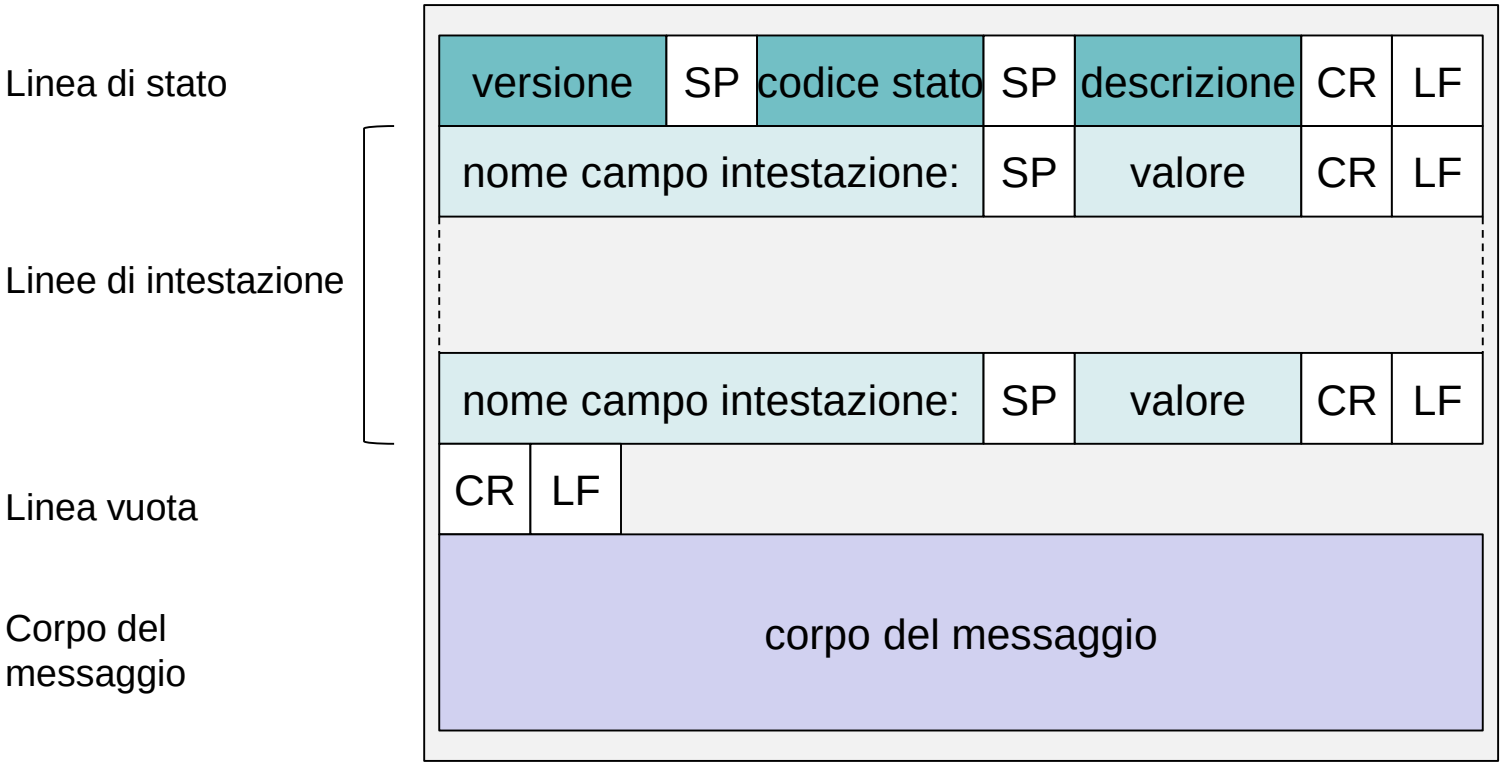
- Dopo le linee dell'intestazione c'è una riga vuota (CR+LF) e quindi il "**corpo del messaggio**" *(entity body).* Il corpo del messaggio è usato con il metodo POST ma non è usato con il metodo GET.
- Generalmente, il metodo POST si usa quando nella pagina web fornita al client sono presenti ***form*** (moduli), costituiti da elementi per l'input di dati. In questo caso il corpo del messaggio contiene ciò che l'utente ha inserito nei campi di input del form.
- Tuttavia, anche se con dei limiti, il metodo GET può essere usato per inviare i valori inseriti nei campi di un form. In questo caso i campi del form sono accodati all'url sottoforma di una stringa detta **QUERY_STRING**.

- Ad esempio, nell'URL:

  http://www.cs.uniroma2.it/persone/rubrica.php?nome=mario&numero=0672545411
- la parte che segue il punto interrogativo prende il nome di **query_string**.
- Il metodo HEAD è utile per il debugging. Quando un server riceve una richiesta che usa il metodo HEAD, invia una risposta ma non invia l'oggetto richiesto.
- La versione HTTP/1.0 permette tre tipi di metodi: GET, POST e HEAD. La versione HTTP/1.1 oltre a questi tre metodi consente altri metodi, tra cui PUT e DELETE. Il metodo PUT consente ad un utente, di caricare un oggetto su una specifica directory di un Web server (funzione di upload) mentre DELETE si usa per cancellare un oggetto da un server Web.

Messaggio di risposta HTTP

- Il formato del messaggio di risposta è illustrato nella figura seguente.



- Il messaggio di risposta è composto da tre parti:
  - una linea iniziale detta **linea di stato**,
  - un numero variabile di **linee di intestazione**
  - **corpo del messaggio**.

- La **linea di stato** è composta da tre campi: **versione**, **codice di stato** e **descrizione** (del codice di stato)
- Il campo **versione** indica la versione del protocollo HTTP, ad esempio 1.1.
- I campi **codice di stato** e **descrizione**, specificano il risultato di una richiesta. I codici di stato più usati e le relative descrizioni sono:
  - **200 OK**: indica che la richiesta è stata soddisfatta e l'oggetto richiesto è stato inviato;
  - **301 Moved Permanently**: indica che l'oggetto richiesto è stato spostato definitivamente; il nuovo URL è specificato nell'intestazione **Location:** del messaggio di risposta; il browser referenzierà automaticamente il nuovo URL;

- **400 Bad Request**: è un codice di errore che indica che la richiesta non è stata correttamente interpretata dal server;
- **404 Not Found**: l'oggetto richiesto non è stato trovato sul server;
- **505 HTTP Version Not Supported**: la versione richiesta del protocollo HTTP non è supportata dal server.

- Il messaggio può avere varie linee di intestazione, come mostrato nell'esempio seguente:

```
HTTP/1.1 200 OK
Date: Mon, 19 Mar 2012 16:00:00 GMT
Last-Modified: Mon, 5 Mar 2012 10:30:24 GMT
Server: Apache/2.0.52 (win32) PHP/5.1.2
Connection: close
Content-Length: 8002
Content-Type: text/html

(dati dati dati dati dati. . .)
```

- **Date:** indica l'ora e la data in cui è stato inviato il messaggio di risposta dal server.
- **Last-Modified**: indica l'ora e la data relativa alla creazione o dell'ultima modifica del file richiesto.
- **Server:** indica il nome del server web che ha inviato il messaggio di risposta (in questo caso Apache per il sistema operativo Windows a 32 bit).
- **Connection:** `close` per avvisare il client che il server chiuderà la connessione TCP al termine della trasmissione del messaggio.
- **Content-length:** indica la dimensione in byte del file da inviare (nell'esempio 8002 byte).
- **Content-type**: indica il tipo di contenuto del file inserito nel corpo del messaggio, che nell'esempio è testo HTML. Il tipo di file è specificato dall'intestazione Content-type: e non dall'estensione del file.

- Il **corpo del messaggio** contiene il file richiesto (rappresentato nell'esempio da *dati dati dati dati dati*...).
- Per vedere un messaggio di risposta HTTP si può usare Telnet collegandosi ad un server Web. Una volta connessi è necessario digitare una linea di messaggio di richiesta per una risorsa memorizzata sul server. Per esempio per un sistema unix o windows:

```
telnet mat.uniroma2.it 80
GET /~frasca/index.htm http/1.1
```

GET condizionato

- Per aumentare la velocità di trasferimento dei documenti e diminuire la quantità di traffico Web, i browser utilizzano due tipi di cache, uno posto in memoria principale e l'altro residente su memoria secondaria.
- Quando un browser ottiene una pagina, la visualizza, ne mantiene il contenuto in memoria ram e salva tutti i file che la compongono nella cache su memoria secondaria, all'interno di una specifica cartella.
- Quando un browser richiede un oggetto, verifica prima se esso si trova nelle cache, prima in memoria ram poi su memoria secondaria. Se è presente lo carica dalla cache.
- Oltre alle suddette **cache,** interne al client, è possibile utilizzare anche un **server cache** esterno detto **server proxy**.
- L'uso delle cache riduce i tempi di risposta per ottenere una pagina web, ma ovviamente crea il problema di aggiornamento della pagina.

- In altre parole, se la pagina originale nel server Web viene modificata la pagina presente nella cache del client non è aggiornata.
- L'HTTP risolve questo problema con un meccanismo detto **GET condizionato**, basato sulla linea di intestazione **If-Modified-Since**.
- Per descrivere il funzionamento del GET condizionato, consideriamo il seguente esempio.

1. un browser richiede un oggetto, non presente nella cache, al server web www.pf.uniroma2.it:

```
GET /img/schema1.gif HTTP/1.1
Host: www.pf.uniroma2.it
…
…
```

2. il server web invia al client un messaggio di risposta con l'oggetto richiesto:

```
HTTP/1.1 200 OK
Date: Tue, 13 Mar 2012 10:25:26 GMT
Last-Modified: Sat, 25 Feb 2012 11:34:56
Server: Apache/2.2.2 (Unix) PHP/5.1.6
Content-Length: 10022
Content-Type: image/gif

(dati dati dati …)
```

Il browser visualizza l'oggetto (nell'esempio un immagine gif) e lo salva anche nella cache su disco, o su altro dispositivo di memoria secondaria. Il browser oltre al file salva anche il suo URL e l'ultima data di modifica del file stesso che recupera dal campo **Last-Modified**.

3. Successivamente, l'utente richiede lo stesso file e supponiamo che questo sia ancora presente nella cache. Dato che il file potrebbe essere stato modificato sul server web, il browser inserisce nel messaggio di richiesta la linea di intestazione **If-Modified-Since:**

```
GET /img/schema1.gif HTTP/1.1
Host: www.pf.uniroma2.it
If-modified-since: Sat, 25 Feb 2012 11:34:56
…
```

Il valore della linea di intestazione

`If-modified-since:`

è uguale al valore della linea di intestazione **`Last-Modified:`** che era stata inviata al server tempo prima.

- Questo messaggio di GET condizionato richiede al server di inviare il file solo se è stato modificato dopo la data specificata nella linea **If-modified-since**.
- Supponiamo che l'oggetto non abbia subito modifiche dalla data `Sat, 25 Feb 2012 11:34:56.` Allora:

4. il server Web invia un messaggio di risposta al client:

```
HTTP/1.1 304 Not Modified
Date: Tue, 20 Mar 2012 14:25:26 GMT
Server: Apache/2.2.2 (Unix) PHP/5.1.6

(corpo del messaggio vuoto)
```

Il server Web invia ancora un messaggio di risposta, ma non in inserisce nel corpo del messaggio il file richiesto.

- Il rinvio dell'oggetto richiesto è inutile, poiché nella cache del client è presente una copia aggiornata, e aumenterebbe il tempo di trasferimento dell'oggetto, soprattutto se questo è di grandi dimensioni.
- Il messaggio di risposta dell'esempio contiene nella linea di stato il codice **304** e la descrizione **`Not Modified`**, che indica al client che il file richiesto non è stato modificato e quindi può utilizzare la copia del file presente nella cache.