Università di Roma Tor Vergata
Corso di Laurea triennale in Informatica

Sistemi operativi e reti

A.A. 2016-17

Pietro Frasca

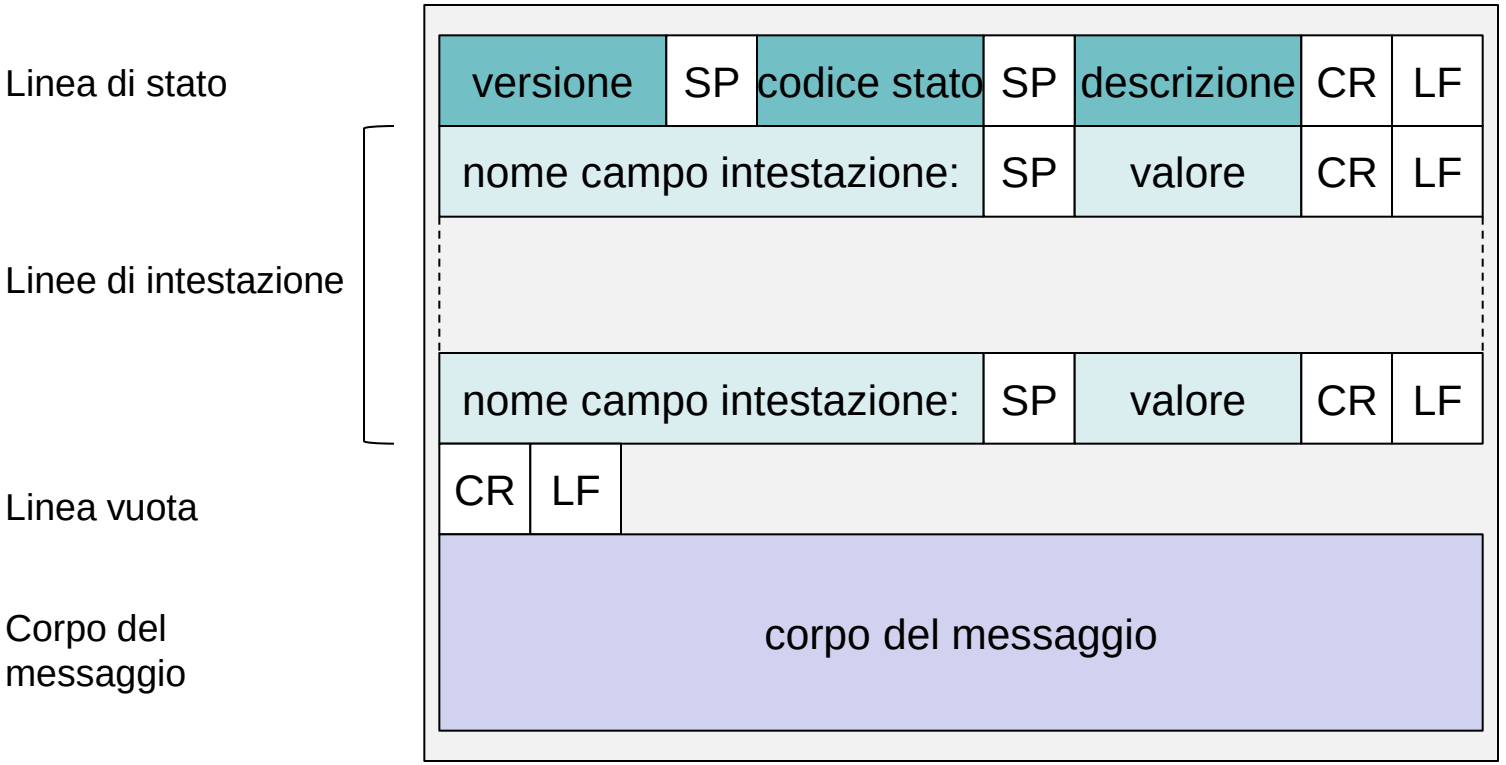
Parte II: Reti di calcolatori

Lezione 6 (30)

Venerdì 24-03-2017

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. Alcuni codici di stato molto 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 livelli di cache: una in memoria ram e una su disco.
- 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 disco, all'interno di una specifica cartella.
- Quando un browser richiede un oggetto, verifica prima se esso si trova nelle cache, prima in memoria poi sul disco. Se è presente lo carica dalla cache.
- Oltre alle suddette **cache** 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 si 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**, che fa uso della linea di intestazione **If-Modified-Since**.
- Per illustrare come funziona il GET condizionato, vediamo un 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
User-Agent: Mozilla/4.0
```

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. 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
user-agent: Mozilla/4.0
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 invia il file richiesto. Il rinvio dell'oggetto richiesto è inutile, dato 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.

Interazione user-server: autorizzazione e cookie

- Il protocollo HTTP è stato progettato senza stato per semplificare lo sviluppo dei server Web che in tal modo possono gestire migliaia di connessioni TCP contemporaneamente.
- Tuttavia in molte applicazioni web è necessario che un sito Web debba identificare gli utenti, e consentire sessioni di lavoro, come ad esempio nelle applicazioni di commercio elettronico.
- L'HTTP fornisce due meccanismi di identificazione degli utenti: **l'autorizzazione** e i **cookie**.

Autorizzazione

Molti siti richiedono agli utenti di digitare uno ***username*** e una ***password*** per poter accedere alle loro pagine. Questa procedura è chiamata **autorizzazione** *(authorization).*

- Ci sono varie modalità di autorizzazione. La più semplice è detta *basic authorization (autorizzazione di base).*
- La richiesta e la risposta, dell'autorizzazione avviene usando speciali intestazioni e codici dell'HTTP.

- La procedura di autorizzazione si svolge nelle seguenti fasi:

  1. Il client richiede una pagina inviando un messaggio di richiesta.
  2. Il server risponde con un messaggio avente:
     A. La linea di stato con codice di stato **401** e descrizione **Authorization Required;**
     B. l'intestazione **`WWW-Authenticate`**: che specifica che il client deve fornire uno username e una password;
  3. Il client (browser) riceve il messaggio di risposta e vedendo la presenza dell'intestazione **`WWW-Authenticate`**: visualizza un finestra di dialogo per consentire all'utente di inserire username e password.
  4. Il client allora rispedisce il messaggio di richiesta, includendo la linea di intestazione **Authorization**: contenente **username** e **password** inseriti dall'utente.

- Il client continua a inviare username e password nei successivi messaggi di richiesta al server. Lo username e la password sono mantenute in variabili del browser (in memoria ram), in modo che l'utente non deve digitarli ogni volta che chiede un nuovo file. In questo modo il sito può identificare l'utente per ciascuna richiesta. Per cancellare lo username e la password è necessario che l'utente chiuda il browser.

Cookie

- I cookie, definiti nella RFC 2965 (obsoleto RFC 2109), sono un meccanismo alternativo che i siti web possono usare per tenere traccia degli utenti.
- L'uso dei cookie è molto diffuso nei siti di commercio elettronico.
- Il funzionamento dei cookie si basa su quattro componenti:

1. una linea di intestazione **Set-cookie**: inserita nel messaggio di risposta HTTP (lato server);
2. una linea di intestazione **Cookie:** presente nel messaggio di richiesta HTTP (lato client);
3. un file cookie gestito dal browser dell'utente;
4. Un'applicazione di gestione dei cookie nel sito Web.

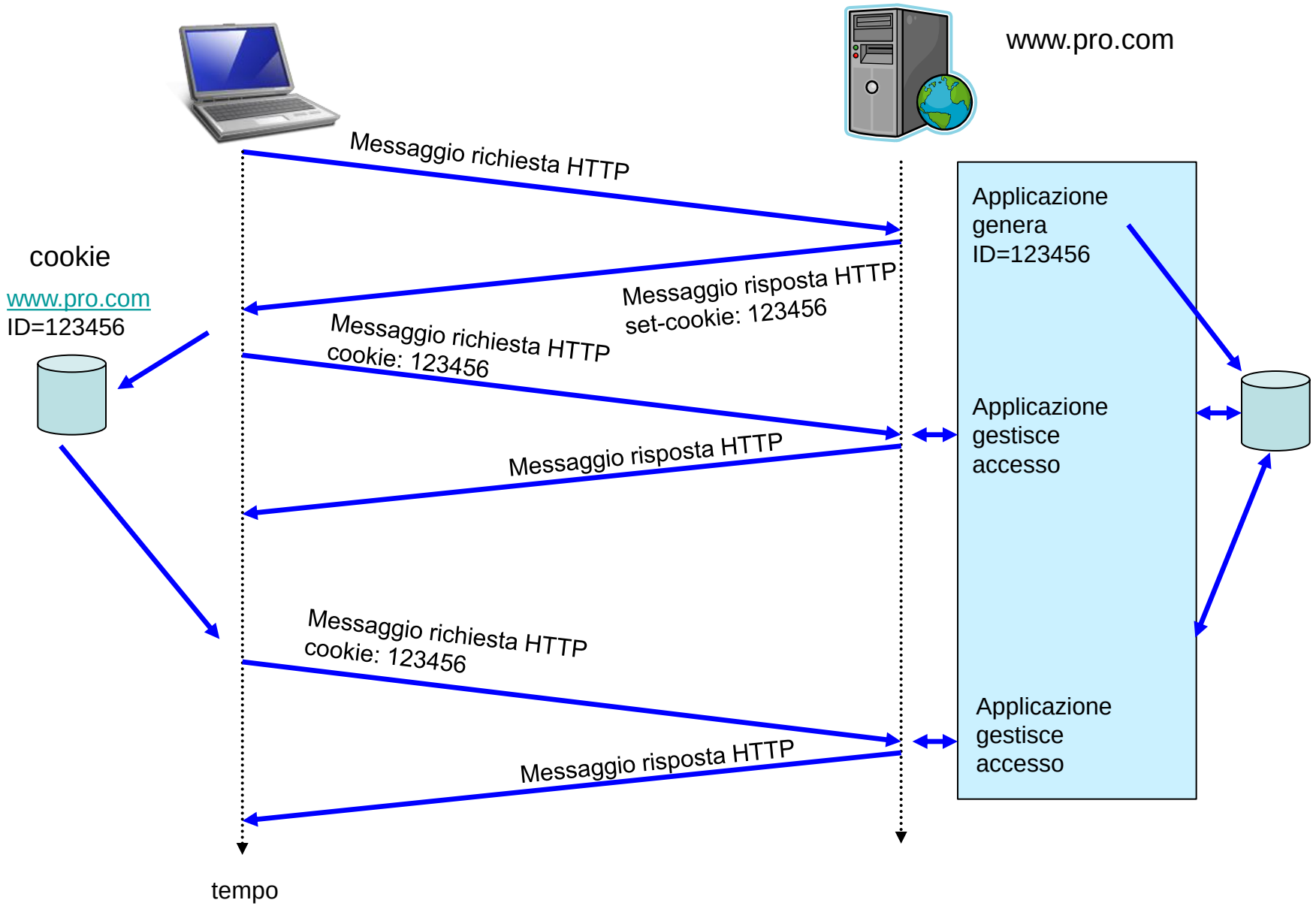
- Esaminiamo un tipico esempio di come sono utilizzati i cookie.

  Supponiamo che un utente, acceda per la prima volta ad una pagina di un sito web di commercio elettronico, ad esempio www.pro.com, e che questo sito usi i cookie.

1. Quando un messaggio di richiesta del browser arriva al server web, l'applicazione web **crea un numero di identificazione unico** e inserisce in una tabella di un database una riga la cui chiave è il numero di identificazione stesso.
2. Il server web risponderà al browser, inserendo nel messaggio di risposta HTTP l'intestazione **Set-cookie**: che contiene il numero di identificazione. Ad esempio, la linea di intestazione potrebbe essere:

   ```
   Set-cookie: 123456
   ```

3. Quando il browser riceve il messaggio di risposta HTTP, vedendo la presenza della linea **`Set-cookie`**, aggiunge una linea al file cookie che esso stesso gestisce. Questa linea contiene **il nome del server** e **il numero di identificazione** contenuto nell'intestazione Set-cookie.


www.pro.com
Messaggio richiesta HTTP
cookie
www.pro.com
ID=123456
Messaggio risposta HTTP
set-cookie: 123456
Messaggio richiesta HTTP
cookie: 123456
Applicazione
genera
ID=123456
Applicazione
gestisce
accesso
Messaggio risposta HTTP
Messaggio richiesta HTTP
cookie: 123456
Applicazione
gestisce
accesso
Messaggio risposta HTTP
tempo

4. Quando l'utente naviga in questo sito, ogni volta che richiede una pagina del sito, il browser consulta il file cookie (relativo a questo sito), estrae il suo numero di identificazione per questo sito e inserisce nella richiesta HTTP la linea d'intestazione **`Cookie:`** con valore pari al numero di identificazione. In questo caso, ogni richiesta HTTP al server web contiene la linea:

   `Cookie: 123456`

   In questo modo, il sito è in grado di registrare le attività che l'utente svolge nel sito web stesso.

- Sebbene il sito web non conosca il nome dell'utente **123456**, sa comunque l'indirizzo IP del suo host, quali pagine ha visitato, in quale ordine, e a che ora.
- Il sito di commercio elettronico può quindi usare i cookie per realizzare un servizio di carrello per gli acquisti gestendo una sessione di lavoro, mantenendo una lista di tutti gli acquisti dell'utente.

- Se tempo dopo, l'utente ritorna a visitare il sito, il suo browser continuerà a inserire la linea di intestazione **Cookie: 123456** nei messaggi di richiesta. Il sito può allora consigliare i prodotti a questo utente in base alle pagine che ha visitato in passato.
- Se l'utente si registra nel sito, fornendo i suoi dati anagrafici, fiscali etc., il sito può associare l'identità di questo utente al suo numero di identificazione.
- In conclusione, i cookie consentono di creare uno **strato di sessione di utente sovrapposto all'HTTP** che è senza stato.
- Sebbene i cookie consentano agli utenti di svolgere operazioni di acquisto on-line, essi rimangono molto sospetti poiché possono essere usati per raccogliere informazioni sul comportamento degli utenti attraverso un grande numero di siti web.

Connessione non persistente e persistente

- Il TCP offre alle applicazioni un servizio di trasferimento di dati affidabile. Il client richiede la connessione al server, dopo la fase di handshake il TCP crea una connessione punto-punto tra le due estremità. Le due parti possono scambiarsi i dati contemporaneamente nelle due direzioni, fino a quando una delle due estremità chiude la connessione.
- L'HTTP usa la connessione TCP in due modalità dette connessione **persistente** e **non persistente**.
- La versione più recente HTTP/1.1 usa le **connessioni persistenti** nella modalità di default. Le connessioni persistenti, consentono di trasferire più file con un'unica connessione TCP.
- La vecchia versione HTTP/1.0 invece usa la connessione non persistente che consente di trasferire un singolo file per ogni connessione TCP.
- Tuttavia, è possibile configurare sia i client che i server HTTP/1.1 per utilizzare le connessioni non persistenti.

Connessione non persistente

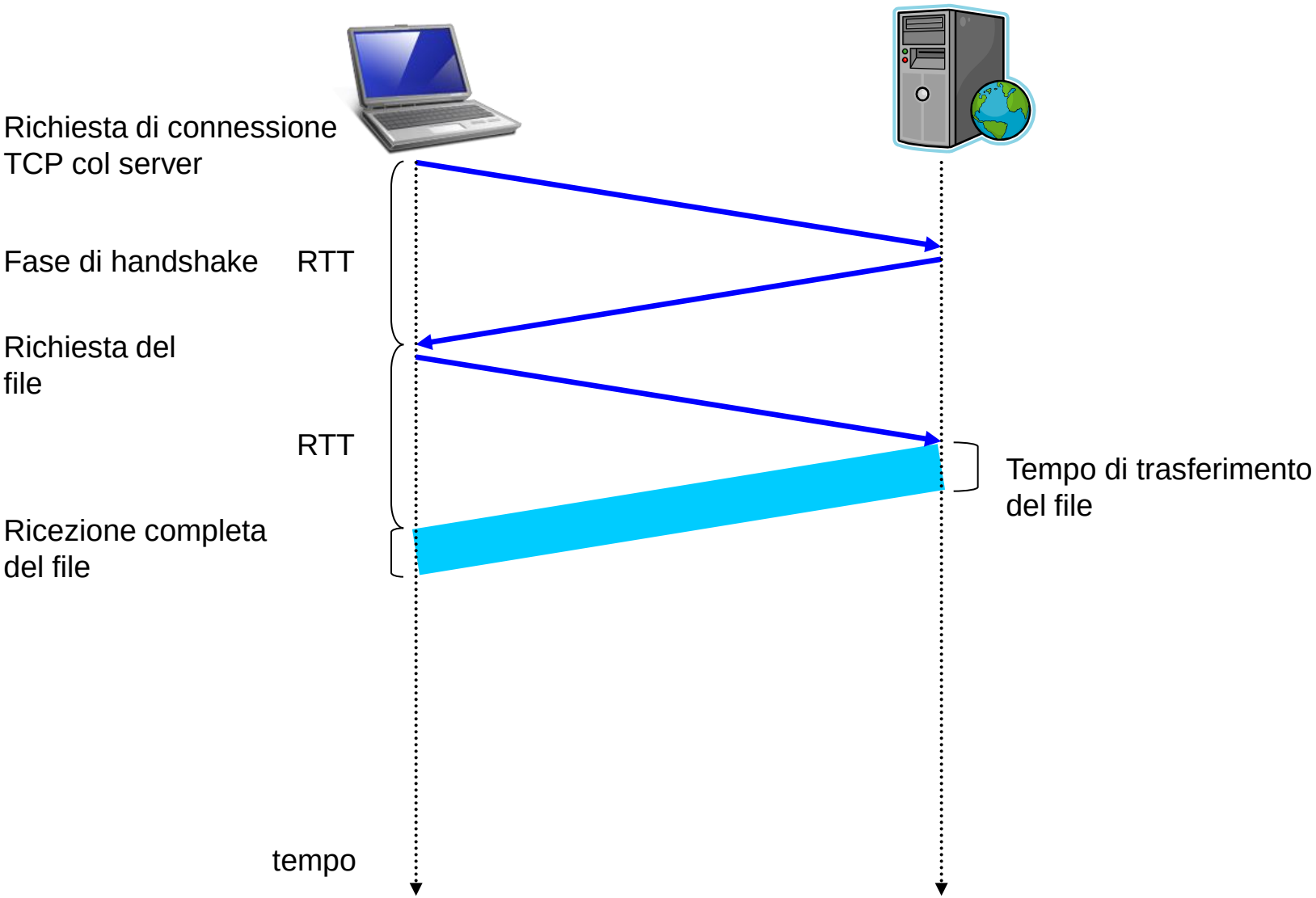
Descriviamo con un esempio come avviene il trasferimento di file con la connessione **non persistente**. Supponiamo che un client richieda una pagina formata da un file base HTLM e da 10 immagini JPEG, per un totale di 11 file. La transazione HTTP avviene nel seguente modo:

1. Il browser instaura una connessione TCP col server (ad esempio **www.pf.uniroma2.it)** alla porta **80**.
2. Il browser invia un messaggio di richiesta HTTP.
3. Il server HTTP riceve il messaggio di richiesta, cerca il file richiesto e se lo trova, lo inserisce in un messaggio di risposta HTTP che invia al client.
4. Il server web **chiude la connessione TCP** instaurata con il client.
5. Il client riceve il messaggio di risposta e riconosce che il file ricevuto è un file HTML, quindi lo interpreta e trova i riferimenti ai 10 oggetti JPEG. La connessione TCP si chiude.
6. La sequenza relativa ai passi 1-4 viene ripetuta per ciascuno dei file JPEG.

Listato del file html dell’esempio:

```
<h1> Esempio di connessione <b>non persistente</b></h1>
<img src=“figura1.jpg"> <img src=“figura2.jpg">
<img src=“figura3.jpg"> <img src=“figura4.jpg">
<img src=“figura5.jpg"> <img src=“figura6.jpg">
<img src=“figura7.jpg"> <img src=“figura8.jpg">
<img src=“figura9.jpg"> <img src=“figura10.jpg">
```

- Notiamo che ciascuna connessione TCP trasporta un messaggio di richiesta e un messaggio di risposta. Quindi, in questo esempio, quando un utente richiede una pagina, si generano **11 connessioni TCP**.
- Il browser (client) può ottenere le 10 immagini JPEG su 10 connessioni TCP in serie, o su più connessioni TCP in parallelo (fino ad un numero massimo prestabilito). Generalmente i browser possono essere configurati con un numero massimo di connessioni parallele, tipicamente 5-10.
- L'uso di connessioni in parallelo diminuisce i tempi di risposta.


Richiesta di connessione TCP col server
Fase di handshake
RTT
Richiesta del file
RTT
Tempo di trasferimento del file
Ricezione completa del file
tempo

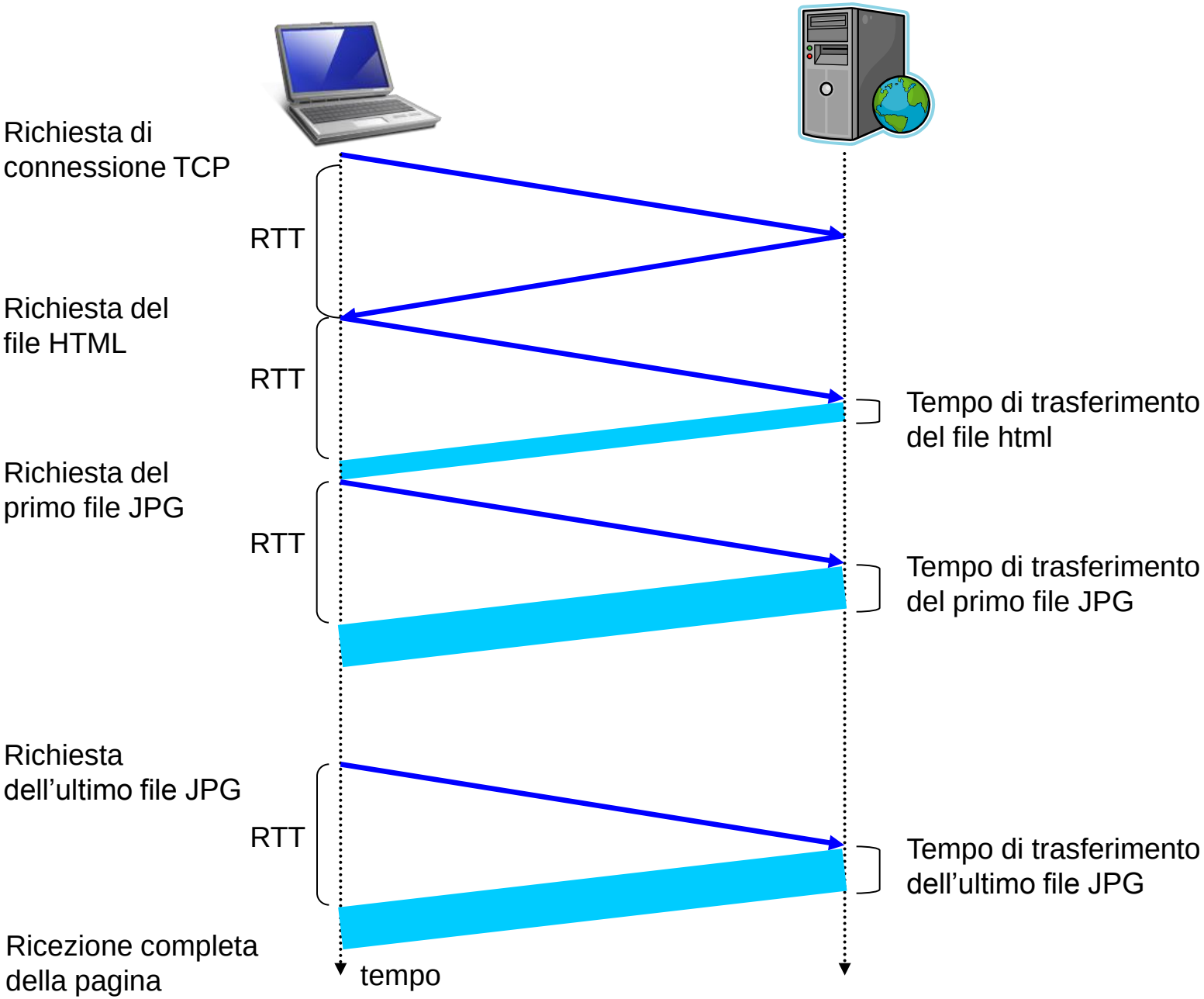
- Il server, quando riceve il messaggio di richiesta, invia il file HTML mediante il TCP. Il tempo che passa dall'inizio della richiesta all'ottenimento della risposta HTTP è di circa due **RTT** più il tempo di trasferimento del file HTML.
- Con **RTT** (**Round Trip Time, Tempo di andata e ritorno)** si indica l'intervallo di tempo che trascorre da quando un piccolo messaggio (come ad esempio il messaggio di richiesta di connessione), parte dal client, arriva al server e ritorna al client.
- Le connessioni non persistenti hanno alcuni svantaggi.
  - per ogni file richiesto è necessario instaurare una nuova connessione TCP che richiede buffer e variabili sia sul client che sul server. Se il server web deve soddisfare richieste di centinaia di client è evidente che risulterà fortemente sovraccarico;
  - prima che inizi il trasferimento di ogni file devono trascorrere due RTT.

Connessione persistente

- La connessione persistente, consente di trasferire più file con un'unica connessione TCP.
- Ad esempio, la pagina Web dell'esempio precedente, costituita da un file html e da 10 immagini jpeg, può essere inviata su un'unica connessione TCP persistente.
- La chiusura della connessione avviene quando il server HTTP rileva che la connessione stessa non è usata da un certo tempo, detto **intervallo di timeout**.
- Esistono due modalità di funzionamento della connessione persistente: **senza parallelismo** e **con parallelismo**.

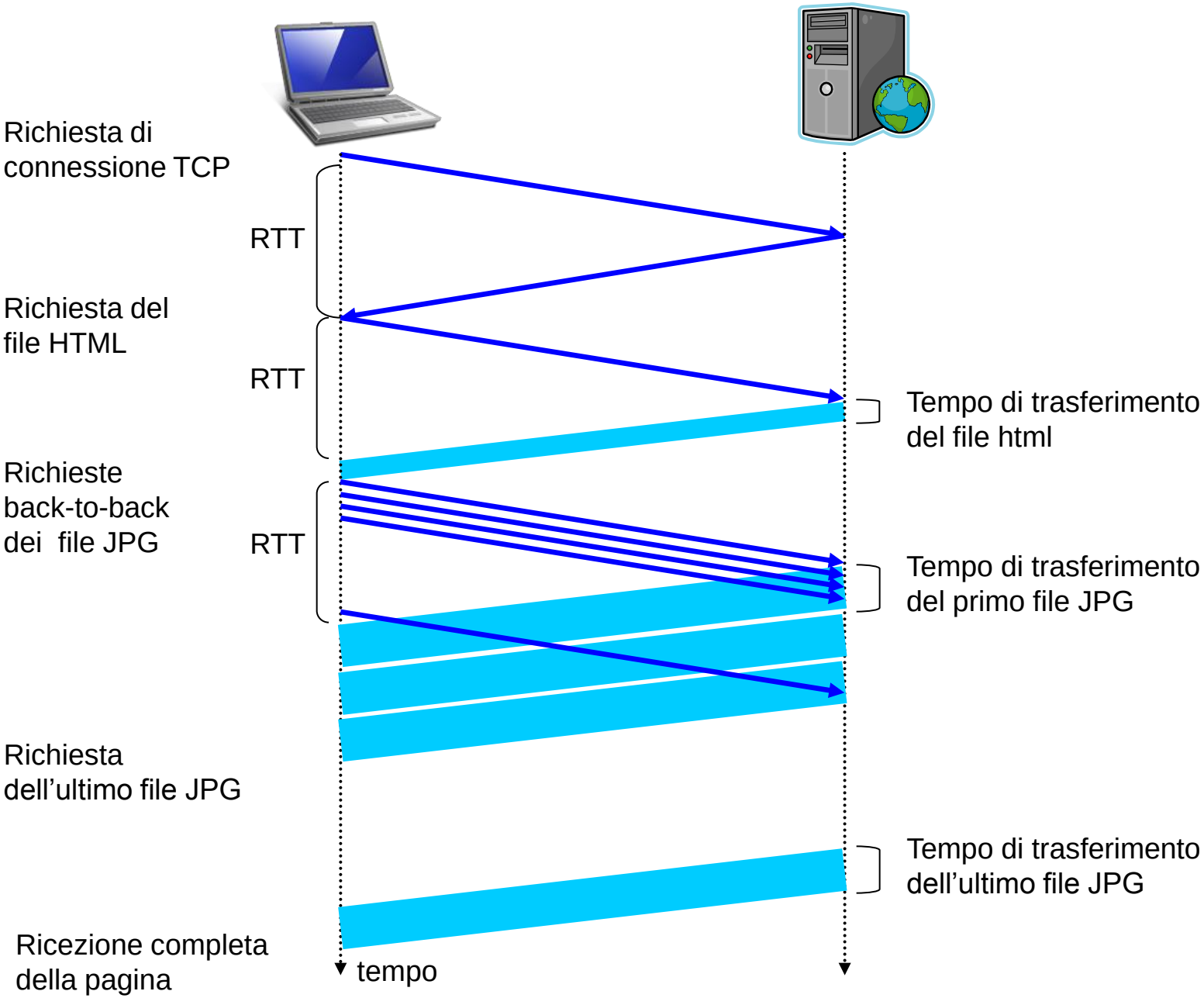
Connessione persistente senza parallelismo

- Nella modalità persistente senza parallelismo, il client può inviare una nuova richiesta solo quando la risposta precedente è stata ricevuta. In questo caso prima che inizi il trasferimento di ciascuna delle 10 immagini, dell'esempio precedente, deve passare **un RTT**.


Richiesta di connessione TCP
RTT
Richiesta del file HTML
RTT
Tempo di trasferimento del file html
Richiesta del primo file JPG
RTT
Tempo di trasferimento del primo file JPG
Richiesta dell'ultimo file JPG
RTT
Tempo di trasferimento dell'ultimo file JPG
Ricezione completa della pagina
tempo

Connessione persistente con parallelismo

- Con la **connessione persistente con parallelismo,** il client può inviare una richiesta appena trova nella pagine HTML base un riferimento di file, senza attendere la risposta alla precedente richiesta.
- Quando il server riceve le richieste, invia i file uno dopo l'altro.
- Il parallelismo, quindi, utilizza **un solo RTT per tutti i file richiesti** (invece di un RTT per ciascun file come avviene quando non si usa il parallelismo).
- Inoltre la connessione TCP con parallelismo resta attiva per tempo più piccolo rispetto a quella senza parallelismo.
- Per default**, l'HTTP/1.1** usa la connessione persistente con parallelismo.


Richiesta di connessione TCP
RTT
Richiesta del file HTML
RTT
Tempo di trasferimento del file html
Richieste back-to-back dei file JPG
RTT
Tempo di trasferimento del primo file JPG
Richiesta dell'ultimo file JPG
Tempo di trasferimento dell'ultimo file JPG
Ricezione completa della pagina
tempo