Università di Roma Tor Vergata
Corso di Laurea triennale in Informatica

Sistemi operativi e reti

A.A. 2017-18

Pietro Frasca

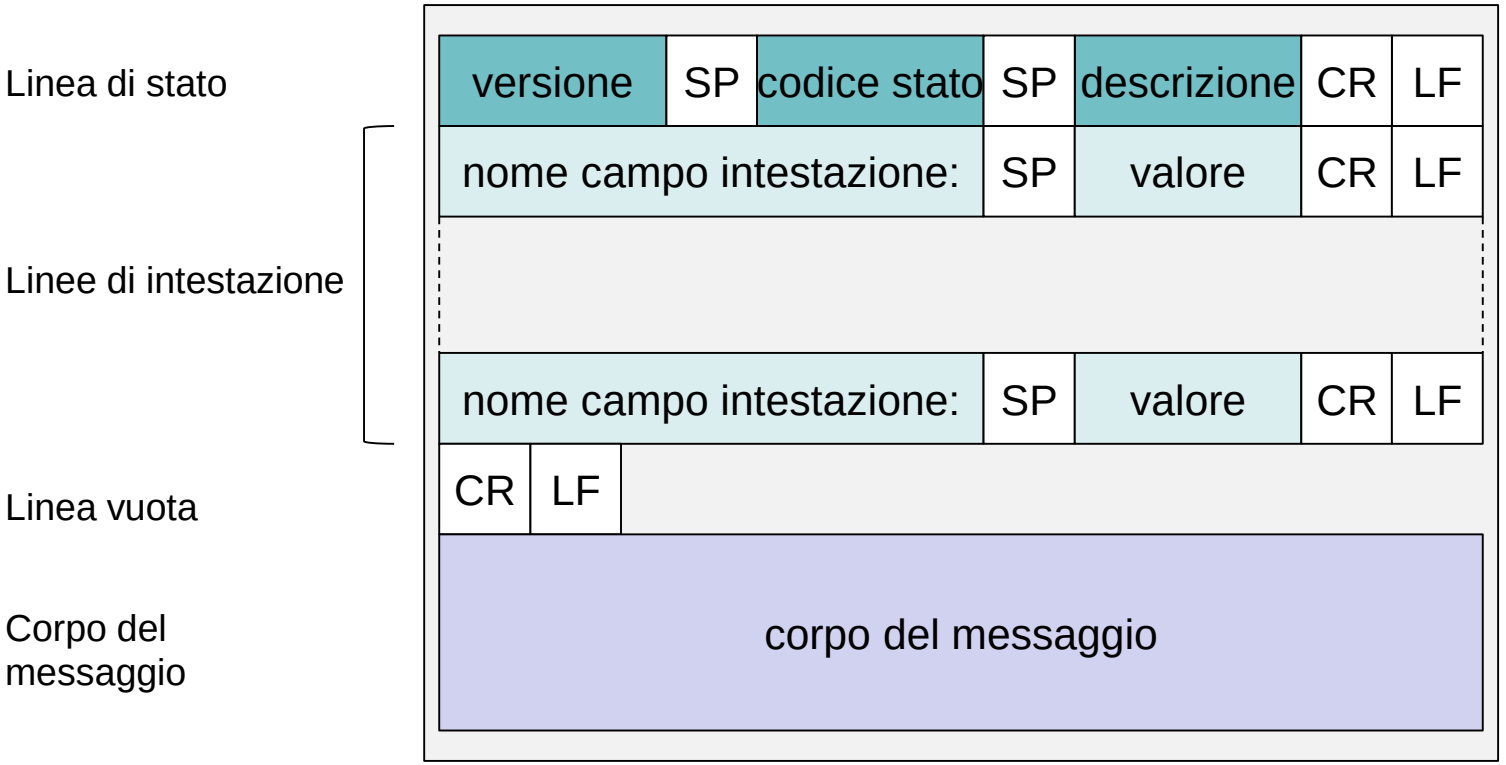
Parte II: Reti di calcolatori

Lezione 7 (31)

Martedì 27-03-2018

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.

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 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;
2. 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.
3. 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 delle attività svolte dagli utenti.
- L'uso dei cookie è molto diffuso nei siti di commercio elettronico.
- Il funzionamento dei cookie si basa su quattro componenti:

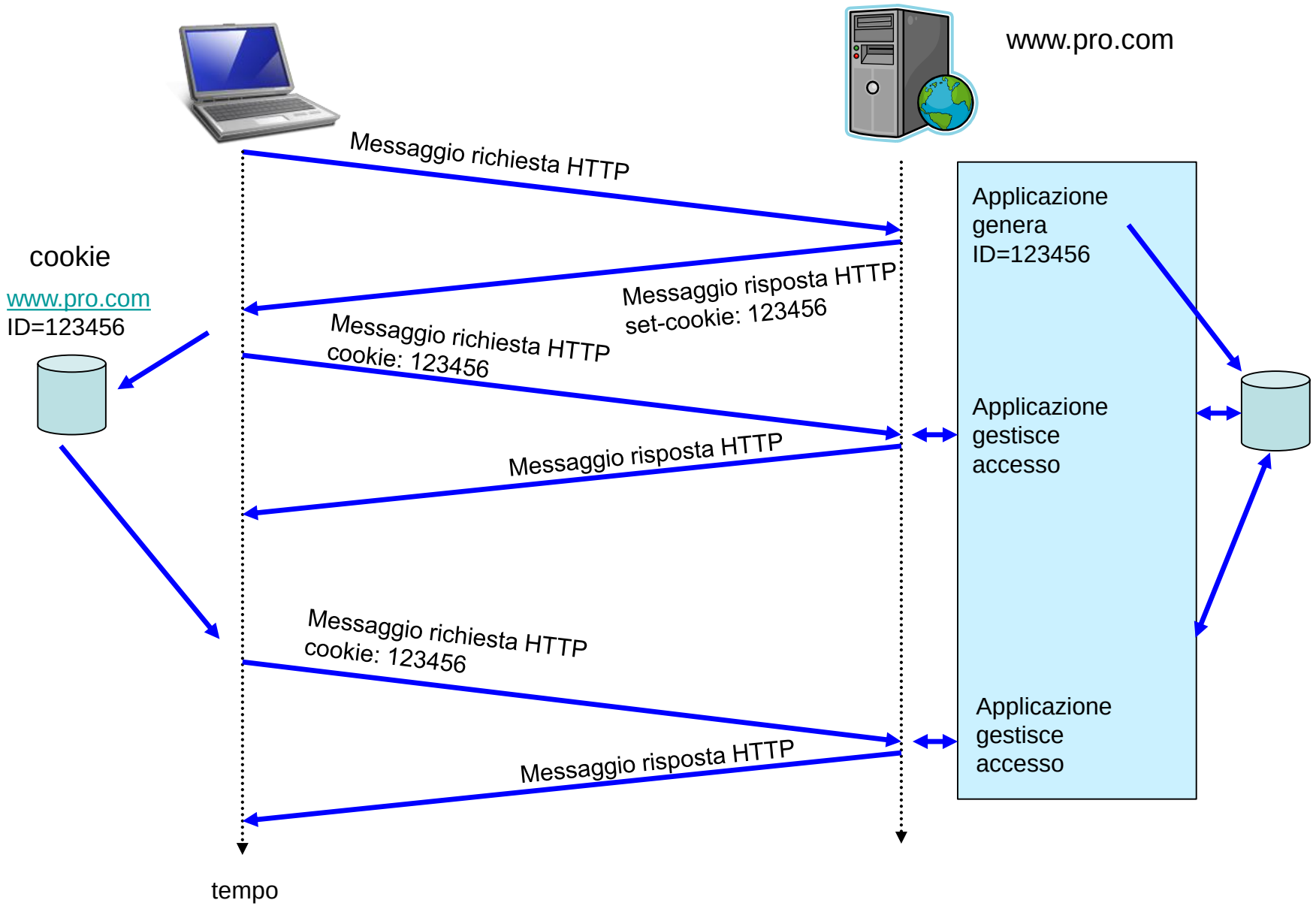
1. una linea di intestazione **Set-cookie**: inserita nel messaggio di risposta HTTP (lato server);
2. una linea di intestazione **Cookie:** presente nel messaggio di richiesta HTTP (lato client);
3. un file cookie gestito dal browser dell'utente;
4. Un'applicazione di gestione dei cookie nel sito Web.

- 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`**, crea e


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

memorizza in un'apposita cartella un file cookie associato al sito. Nel file cookie sono salvate varie informazioni, comprese **il nome del server** e **il numero di identificazione** contenuto nell'intestazione Set-cookie.

4. Quando l'utente, navigando in questo sito, richiede una pagina, il browser consulta il file cookie, relativo a questo sito, estrae il suo numero di identificazione 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.