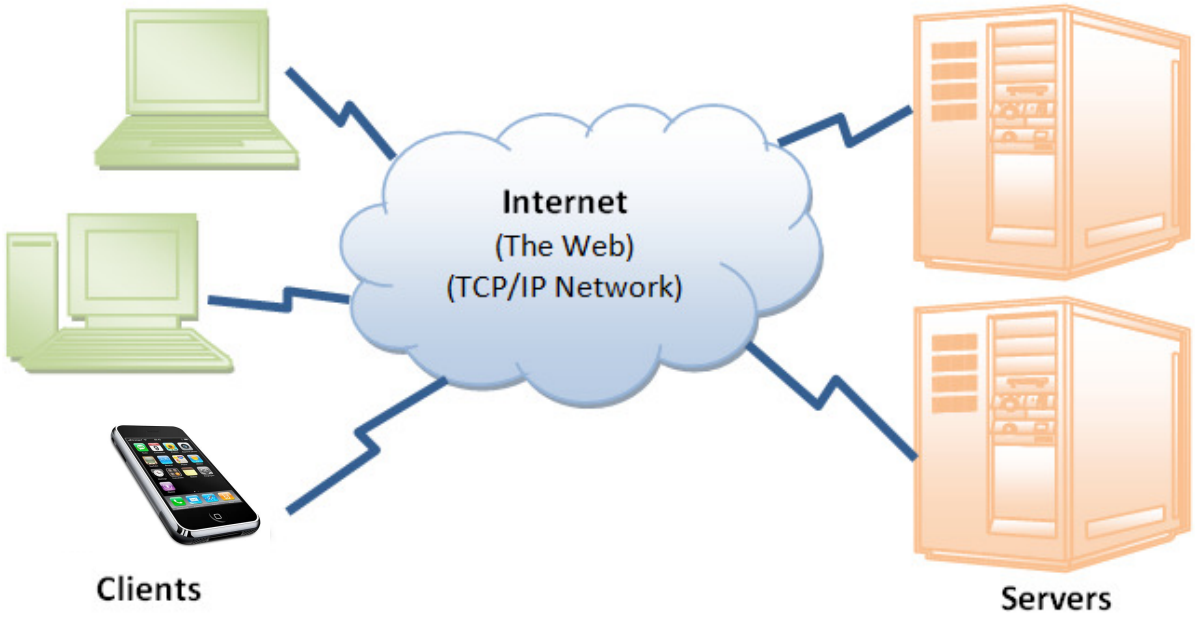
Università degli Studi dell'Insubria
Dipartimento di Scienze Teoriche e Applicate

Programmazione Concorrente e Distribuita
Il protocollo HTTP

Luigi Lavazza
Dipartimento di Scienze Teoriche e Applicate
luigi.lavazza@uninsubria.it

Applicazioni e protocolli



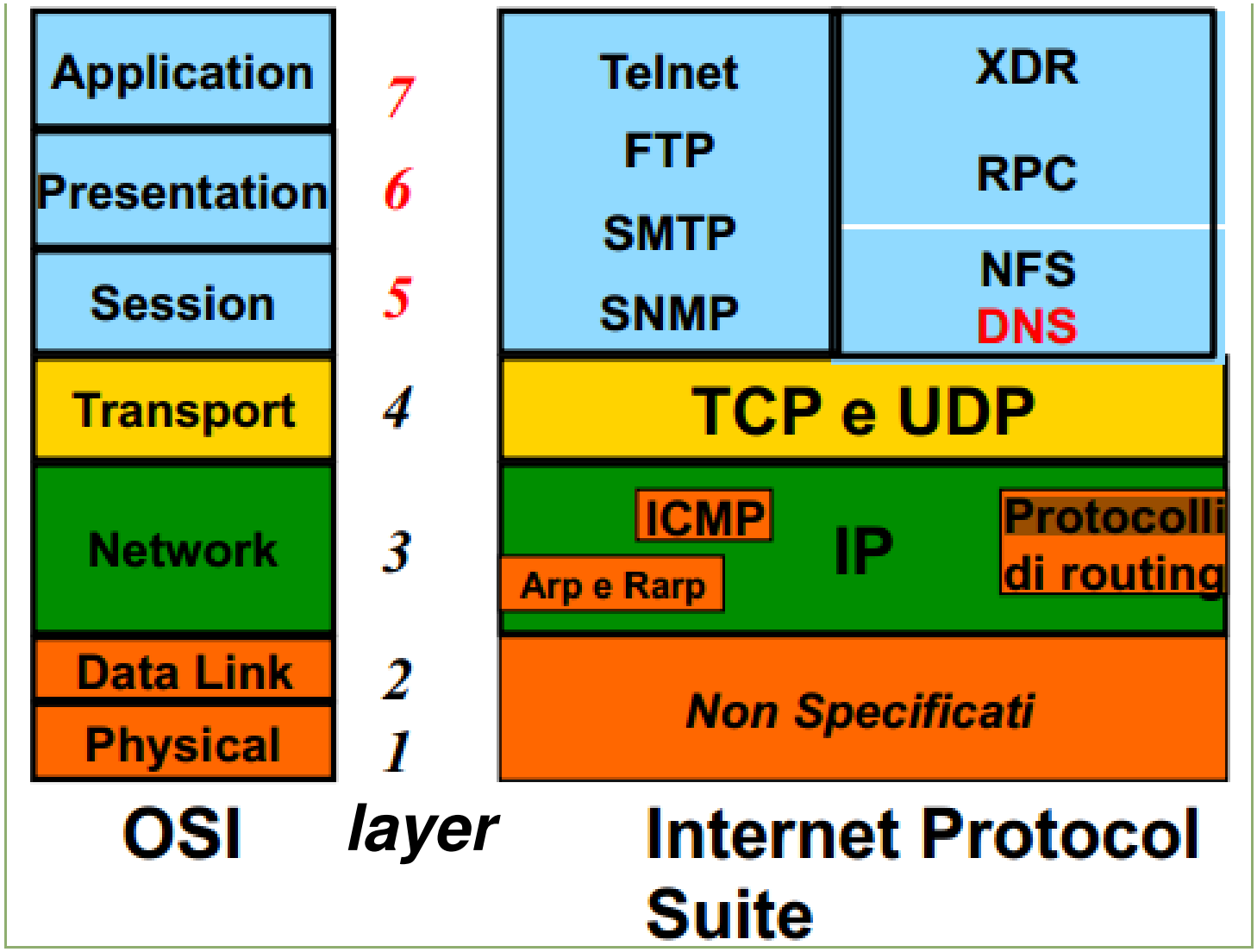
- Molte applicazioni sono in esecuzione contemporaneamente su Internet, come ad esempio le applicazioni per la navigazione web, e-mail, trasferimento di file, lo streaming audio e video, e così via.
- Per far si che client e server comunichino correttamente, queste applicazioni devono accordarsi su uno specifico protocollo a livello di applicazione
  - come HTTP, FTP, SMTP, e così via.

Protocollo

- Definizione di protocollo
  - Un protocollo è un insieme di regole che permettono di trovare uno standard di comunicazione tra diversi computer attraverso la rete.
- Definizione di rete
  - Per rete si intende un insieme di due o più computer connessi tra di loro ed in grado di condividere informazioni.

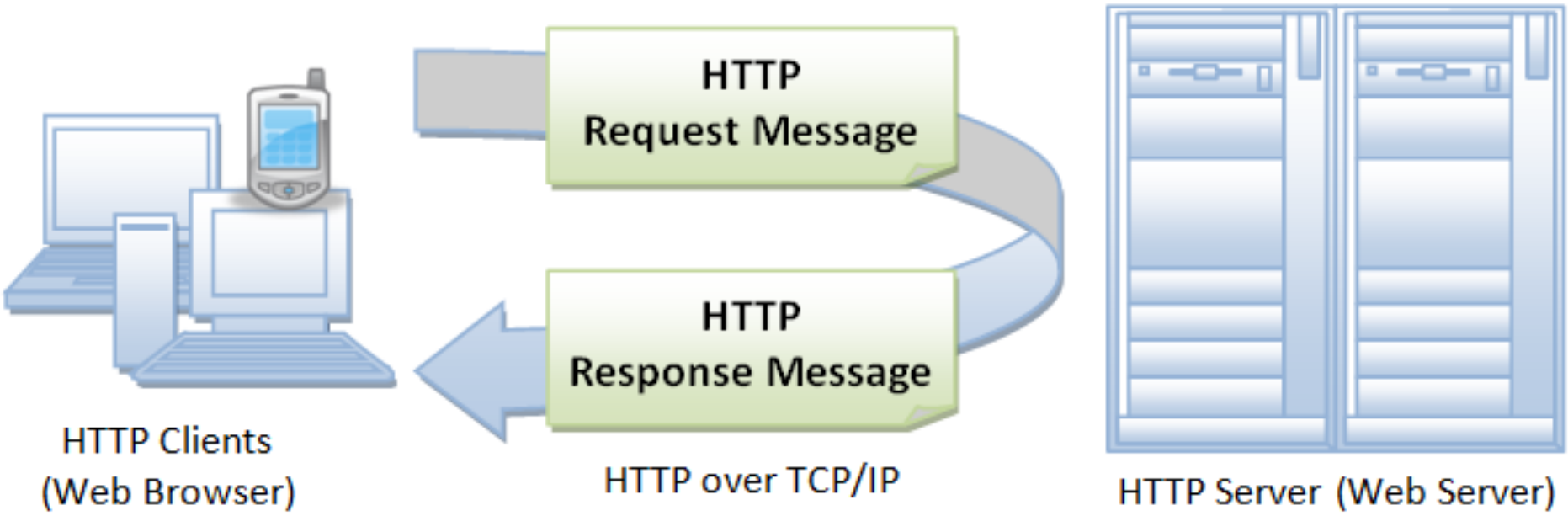
- Un protocollo descrive il formato che il messaggio deve avere.
- Ogni protocollo viene riferito ad una particolare attività, come ad esempio spedire messaggi attraverso la rete, stabilire connessioni remote, oppure trasferire file.

Esempi di protocolli



Il protocollo HTTP

- HTTP (Hypertext Transfer Protocol) è il protocollo di applicazione più popolare in Internet
  - È il protocollo del WEB

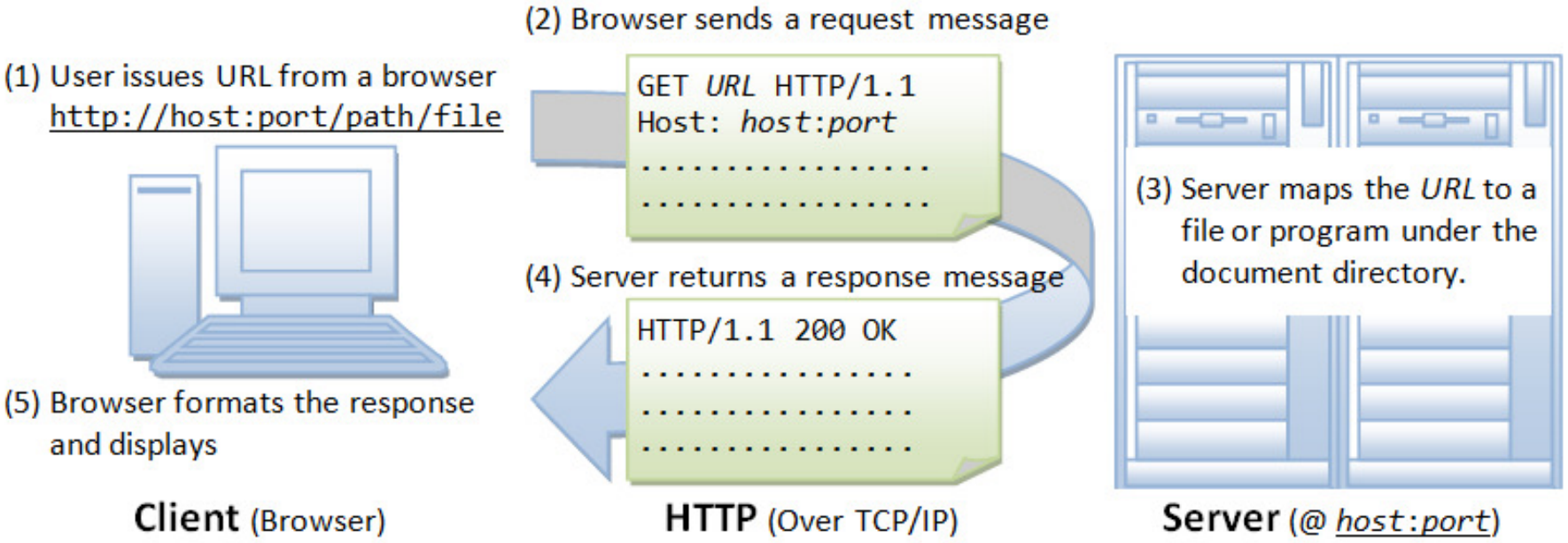


HTTP

- HTTP è il protocollo per la trasmissione di informazioni attraverso il WEB
- Il World Wide Web (WWW) è formato da numerosi server e client che colloquiano attraverso protocolli standard: TCP/IP a livello trasporto e HTTP a livello applicativo
- I dati comunicati sono in formato HTML (HyperText Mark-up Language).
- HTTP è stato creato apposta per il trasferimento di dati HTML
- Tutti i client e server Web comunicano usando il protocollo HTTP
- L'HTTP è un protocollo "stateless" (senza memoria)
  - permette di accedere all'informazione di interesse da un client o server ad un altro attraverso i "links" ipertestuali.

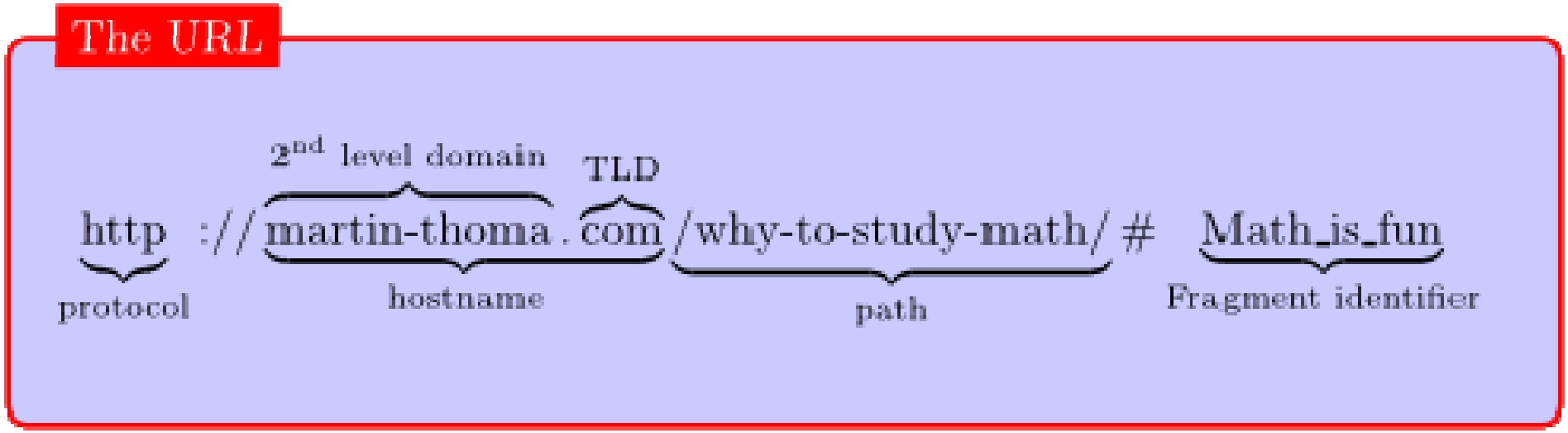
Il World Wide Web o Web

- Il World Wide Web è uno dei principali servizi di Internet che permette di navigare e usufruire di un insieme vastissimo di contenuti.
- Una pagina web è formata da oggetti.
- Gli oggetti possono essere file HTML, immagini (JPEG, GIF, PNG), applet Java, file audio,. . .
- Ogni oggetto è identificato da un URL (Uniform Resource Locator).



Uniform Resource Locator (URL)

- Internet è sistema in cui ogni cliente utilizza le risorse rese disponibili dai vari server.
  - Più precisamente, è una rete, in cui ogni singola connessione può essere vista come una relazione C/S.
- In Internet è necessario avere un metodo per far riferimento alle varie risorse disponibili.
- URL: sequenza di caratteri che identifica univocamente l'indirizzo di una risorsa in Internet



Esempi di URL

- Alcuni esempi di URL (che usano cinque diversi protocolli)
  - http://www.nowhere123.com/docs/index.html
  - ftp://www.ftp.org/docs/test.txt
  - mailto:user@test101.com
  - news:soc.culture.Singapore
  - telnet://www.nowhere123.com/

Struttura di un URL

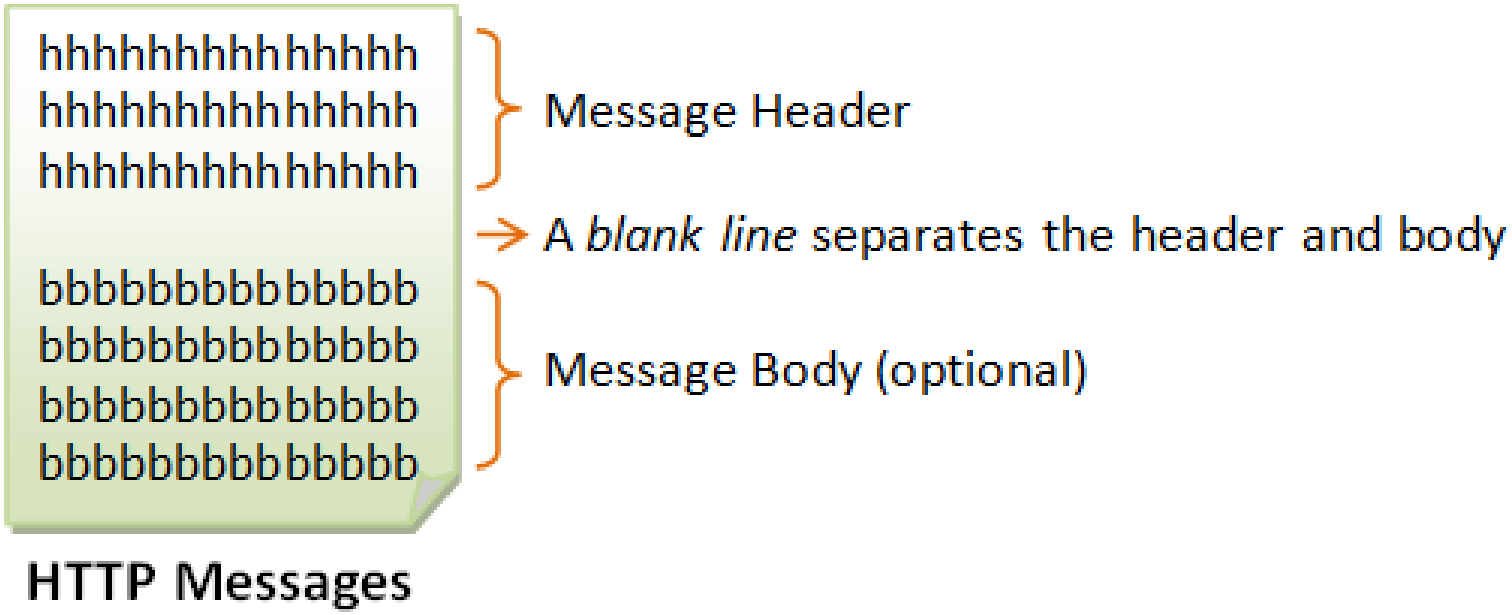
- Ogni URL si compone di varie parti:

protocollo://nomehost<:porta></percorso>
<? querystring>< #fragmentidentifier>

  - protocollo: HTTP, HTTPS, FTP, MMS ecc.
  - nomehost: Rappresenta l'indirizzo fisico del server su cui risiede la risorsa
  - porta (opzionale): la porta del processo server
    - Se non indicata si usa la porta «standard» del servizio. Ad es. 80 per http
  - percorso (opzionale): la risorsa richiesta, completa di path
    - Se non specificato si usa il default **`index.html`**
  - querystring (opzionale): ad esempio **`?par1=val&par2=val`**
  - fragment identifier (opzionale): identificatore di un elemento nella pagina

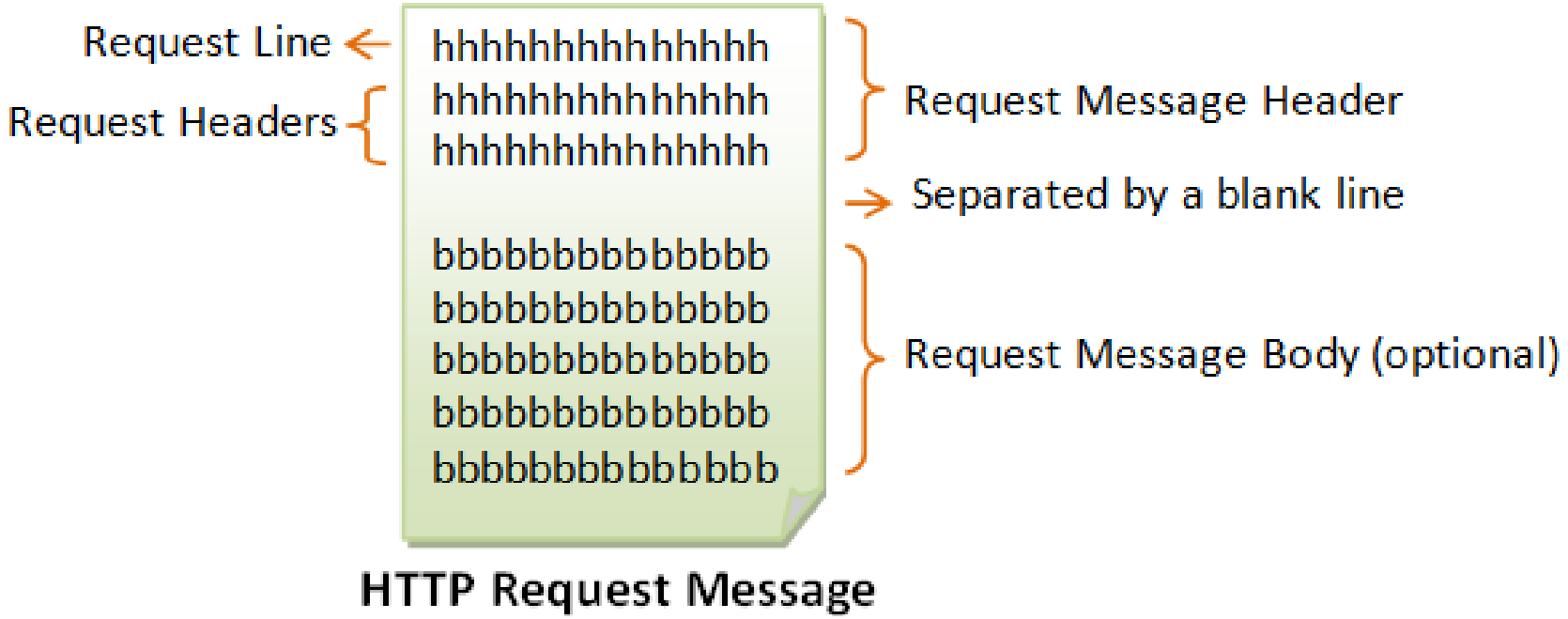
HTTP Request and Response Messages

- Client e Server HTTP comunicano con l'invio di messaggi di testo. Il client invia un messaggio di richiesta al server.
- Il server, a sua volta, restituisce un messaggio di risposta.
- Un messaggio HTTP consiste di un'intestazione ed un corpo facoltativo, separati da una linea vuota, come illustrato di seguito:



HTTP Request Message

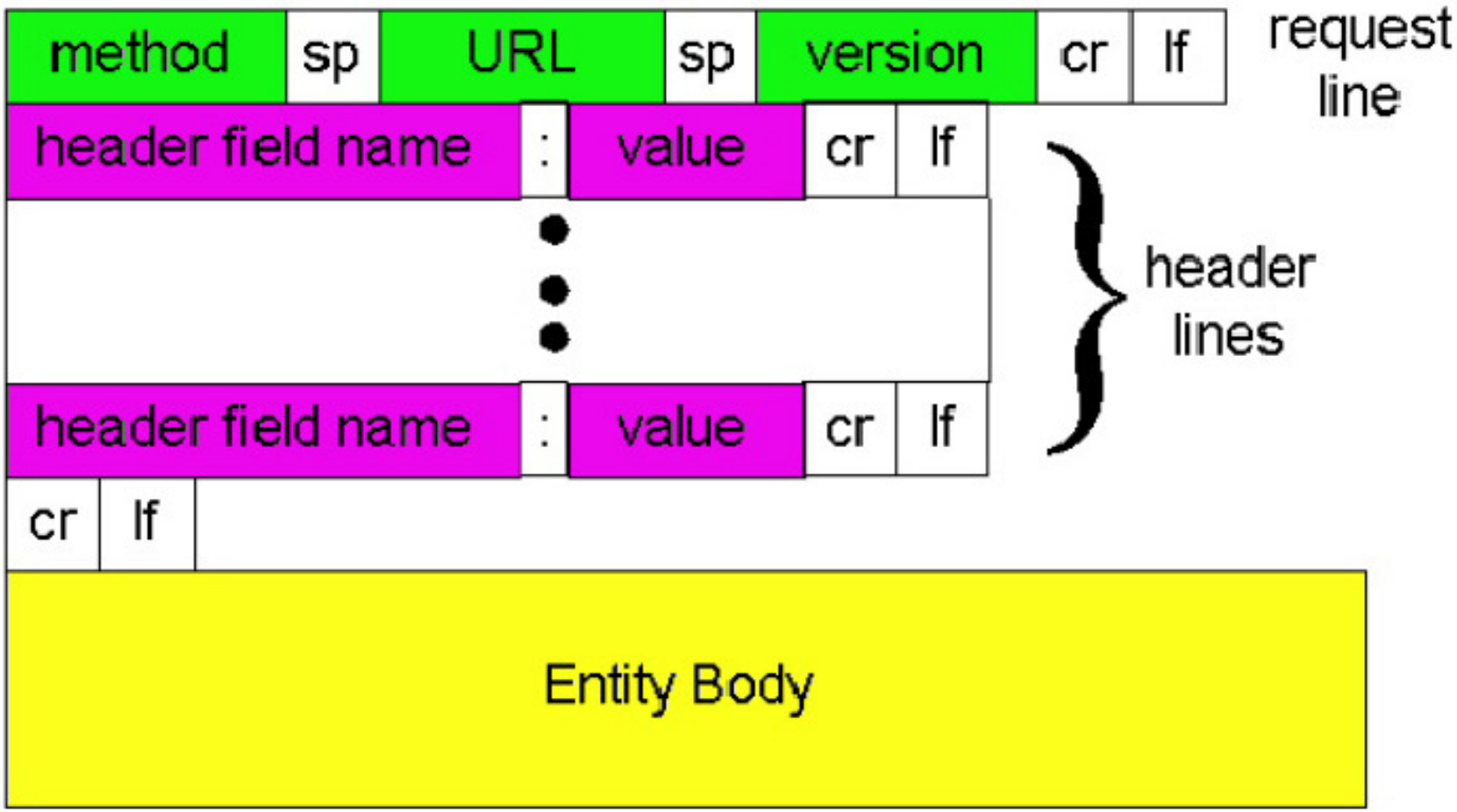
- Il formato di un messaggio di richiesta HTTP è come segue:



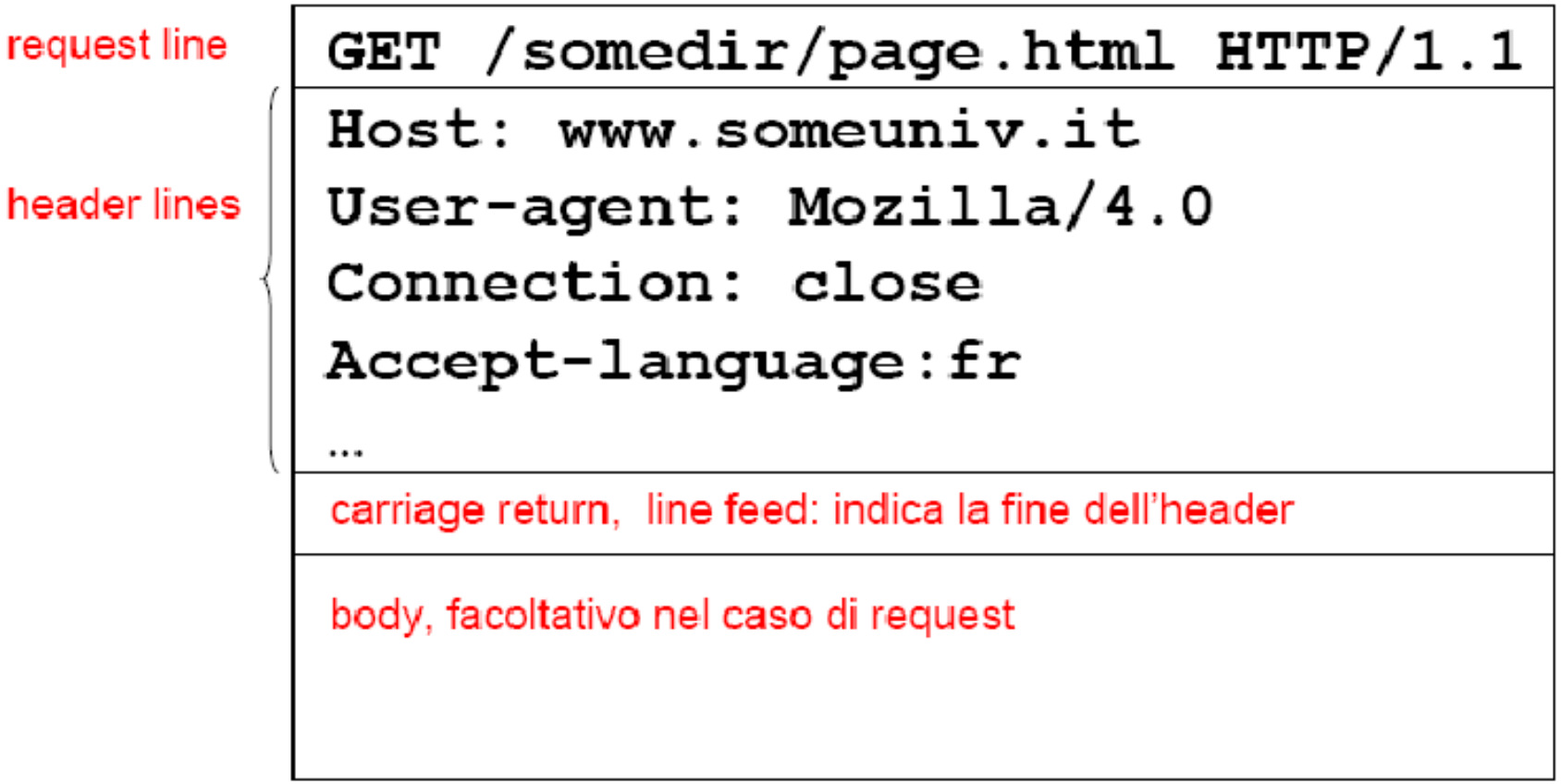
Fasi di comunicazione

- Fasi per l'acquisizione del documento da parte del client:
- Connessione : Il client crea una connessione TCP-IP con il server (il protocollo assume per default la porta 80).
- Richiesta : Il client invia la richiesta mediante una riga di caratteri ASCII terminata da una coppia di caratteri CR-LF (Carriage Return, Line Feed).
- Risposta : La risposta inviata dal server è un messaggio in linguaggio HTML nel quale è contenuto il documento richiesto (o un messaggio d'errore).
- Chiusura connessione : Il server subito dopo aver spedito il documento si sconnette. Comunque anche il client può interrompere la connessione in ogni momento, in questo caso il server non registrerà nessuna condizione d'errore.

Fasi di comunicazione: Messaggio di Richiesta



Fasi di comunicazione: Esempio di Messaggi di Richiesta



Request-line: Metodi

- I metodi indicano l'operazione che deve essere eseguita sulla risorsa
- Nel campo Metodo della Request-Line appare uno solo dei seguenti metodi :
  - "OPTIONS", "GET" ,"HEAD", "POST", "PUT", "DELETE" ,"TRACE", "CONNECT".
- Il metodo utilizzato viene riconosciuto dal server se è implementato su di esso, altrimenti il server restituisce un errore.
- Il metodo "GET" è sempre supportato da qualsiasi server mentre gli altri sono opzionali.
- Alcuni esempi:

```
GET /test.html HTTP/1.1
HEAD /query.html HTTP/1.0
POST /index.html HTTP/1.1
```

Come GET, ma il server non deve includere un message body nella risposta.

Fornisce al server dati da pubblicare.

HTTP Request Methods

- Il protocollo HTTP definisce un insieme di metodi di richiesta.
  - GET: Un client può utilizzare la richiesta GET per ottenere una risorsa web dal server.
  - HEAD: Un client può utilizzare la richiesta HEAD per ottenere l'intestazione della risposta che una richiesta GET avrebbe ottenuto. Dal momento che l'intestazione contiene la data dell'ultima modifica dei dati, questo può essere utilizzato per confrontare la data della copia cache locale.
  - POST: utilizzato per inviare i dati al server web.
  - PUT: chiede al server di memorizzare i dati.
  - DELETE: chiedere al server di cancellare i dati.
  - TRACE: chiedere al server di restituire il messaggio ricevuto, corredato di informazioni diagnostiche
  - OPZIONI: chiedere al server di restituire l'elenco dei metodi di richiesta che supporta.
  - CONNECT: Usato per dire a un proxy di effettuare una connessione ad un altro host. E' spesso usato per connessioni SSL attraverso il proxy.
  - Altri metodi di estensione.

Request-line: URL o “Request-URI”

- In questo campo appare solitamente un absolutePath, usato per inoltrare la richiesta ad un server.
- Indica il path della risorsa sul server originario
  - se l’absolute path è vuoto gli viene assegnato per default lo ”/” per indicare la root del server.
- In generale
  - **`Request-URI = "*"|absoluteURI|abspath|auhority`**
  - L’asterisco ”*” indica che la richiesta non deve essere applicata ad una particolare risorsa, ma al server stesso.
    - Ad esempio: **`OPTIONS * HTTP/1.1`**
  - L’absoluteURI è usato quando la richiesta è stata fatta ad un proxy

Request-line: Versione HTTP

- Nella comunicazione Client/Server è necessario che sia chiaro la versione del protocollo usato per convenire sul formato del messaggio.
- Nella prima linea del messaggio nel campo HTTP-Version field abbiamo :

```
"HTTP" "/" 1*digit "." 1*digit
```

- Es.:

```
HTTP/1.1
```

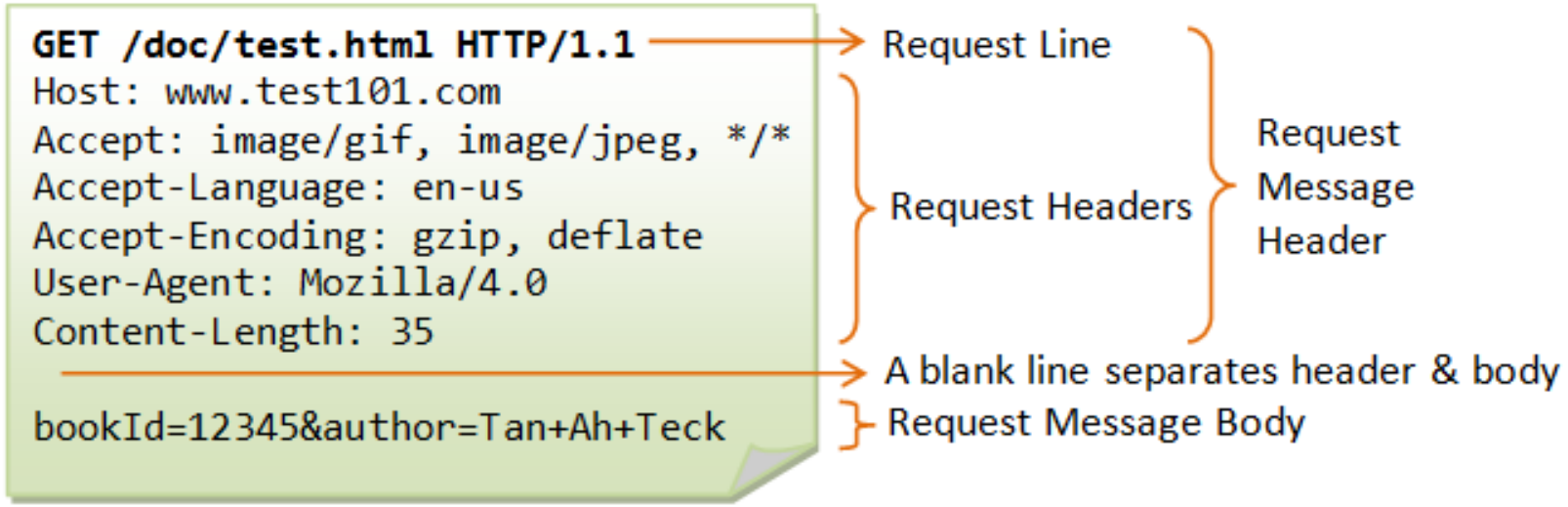
Request Headers

- Request Headers sono delle coppie nome:valore
- possono essere specificati più valori, separati da virgole

```
Request-header-name: request-header-value1, request-
header-value2, ...
```

- Esempi di request headers sono:

```
Host: www.xyz.com
Connection: Keep-Alive
Accept: image/gif, imag/jpeg, */*
Accept-Language: us-en, fr, cn
```



Fasi di comunicazione: esempio Java

```
public class HttpClient {
  void exec(String urlStr) {
    URI uri=null;
    Socket socket=null;
    try { uri = new URI(urlStr);
    } catch (URISyntaxException e1) {
        System.err.println("invalid URL");
        System.exit(0);
    }
    try {
      String host = uri.getHost();
      String path = uri.getRawPath();
      socket = new Socket(host, 80); // Connessione
      BufferedReader in=new BufferedReader(
         new InputStreamReader(socket.getInputStream()));
      PrintWriter out=new PrintWriter(socket.getOutputStream());
```

Fasi di comunicazione: esempio Java

```
    // Richiesta oggetto
    out.print(  "GET " + path + " HTTP/1.1\r\n" +
        "Host: " + host + "\r\n" +
        "Connection: close\r\n\r\n");
    out.flush( );
    // Risposta
    String messageString = "";
    String line;
    System.out.println("*** MESSAGE From " + host + ":\n");
    while ((line = in.readLine()) != null) {
      System.out.println(line);
      if(line.equals("")){
        System.out.println("<fine header>");
        break;
      }
    }
  } catch (IOException e) {
    System.err.println("I/O problems: terminating");
    System.exit(0);
  }
```

Fasi di comunicazione: esempio Java

```
    finally {
      System.out.println("*** closing...");
      try { socket.close(); } catch (IOException e) { }
    }
  }
  public static void main(String[] args) {
    new HttpClient().exec(args[0]);
  }
}
```

Risultato con argomento
http://artelab.dicom.uninsubria.it/

*** MESSAGE From artelab.dicom.uninsubria.it:

HTTP/1.1 200 OK
Date: Sat, 24 Apr 2021 16:38:04 GMT
Server: Apache
Last-Modified: Fri, 21 Jun 2019 07:51:52 GMT
ETag: "23a94-11c0-58bd0bc4a5600"
Accept-Ranges: bytes
Content-Length: 4544
Vary: Accept-Encoding
Connection: close
Content-Type: text/html

<fine header>
*** closing...

Risultato con argomento
http://www.dicom.uninsubria.it/pippo/

*** MESSAGE From www.dicom.uninsubria.it:

HTTP/1.1 404 Not Found
Date: Sat, 24 Apr 2021 16:27:49 GMT
Server: Apache
Content-Length: 196
Connection: close
Content-Type: text/html; charset=iso-8859-1

<fine header>
*** closing...

Risultato con argomento
http://informatica.dista.uninsubria.it/?q=node/14

*** MESSAGE From informatica.dista.uninsubria.it:

HTTP/1.1 303 See Other
Date: Sat, 24 Apr 2021 16:42:04 GMT
Server: Apache
Location: https://www.uninsubria.it/siti-tematici-o-federati/siti-dei-dipartimenti/dipartimento-di-scienze-teoriche-e-applicate-dista/i
Content-Length: 332
Connection: close
Content-Type: text/html; charset=iso-8859-1

<fine header>
*** closing...

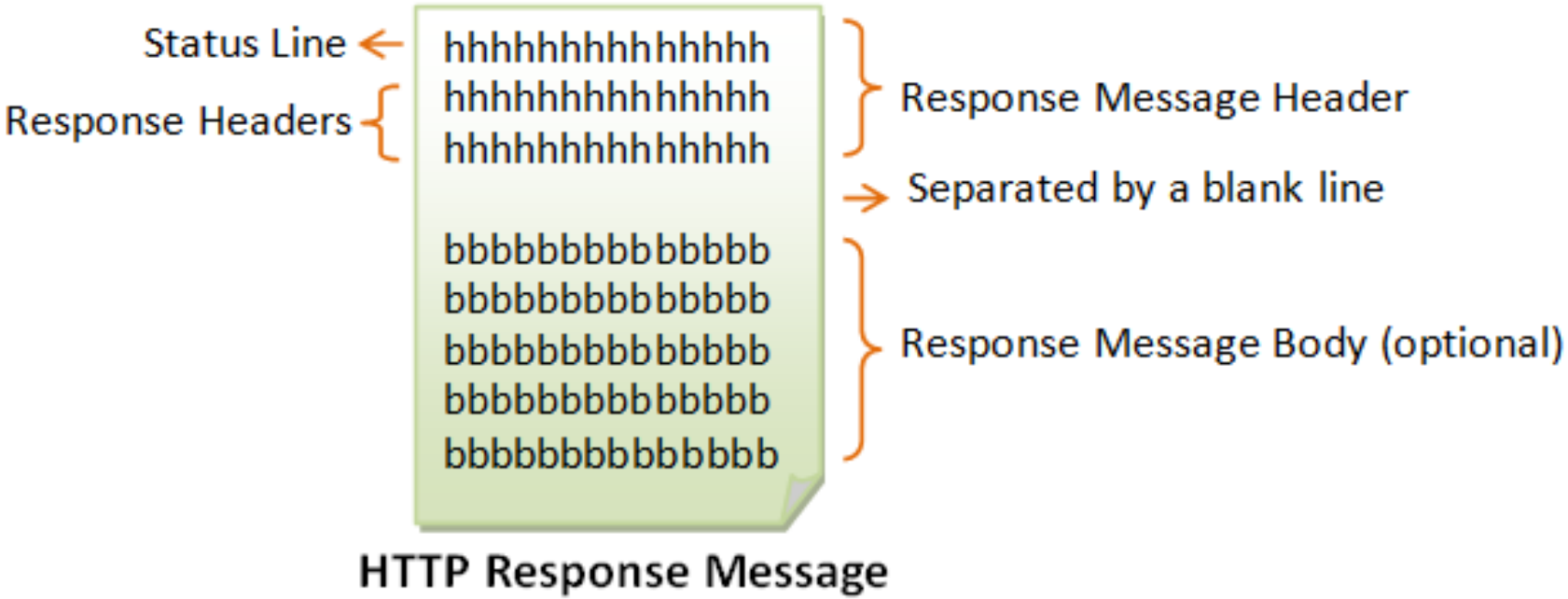
https://www.uninsubria.it/siti-tematici-o-federati/siti-dei-dipartimenti/dipartimento-di-scienze-teoriche-e-applicate-dista

Risultato con argomento
http://www.vattelapesca.uninsubria.it/

I/O problems: terminating

HTTP Response Message

- Il formato del messaggio di risposta HTTP è il seguente:



- Alcuni esempi di Status Line sono:

```
HTTP/1.1 200 OK
HTTP/1.0 404 Not Found
HTTP/1.1 403 Forbidden
```

Risposta

- Una volta ricevuta la richiesta il server risponde con un messaggio di risposta.
- La risposta ha il seguente formato :

Status-Line +

((general-header or response-header or entity header) CRLF)

CRLF

[corpo del messaggio]

- Il corpo del messaggio contiene i dati effettivamente richiesti dal client: i documenti ipertestuali.

```
*** Request To artelab.dista.uninsubria.it:
GET /people.html HTTP/1.1
Host: artelab.dista.uninsubria.it
Connection: close
```

```
*** MESSAGE From artelab.dista.uninsubria.it:
HTTP/1.1 200 OK
Date: Thu, 14 Apr 2016 20:39:11 GMT
Server: Apache
Last-Modified: Fri, 29 Jan 2016 11:44:44 GMT
ETag: "202f6-102d-52a778fc108cf"
Accept-Ranges: bytes
Content-Length: 4141
Vary: Accept-Encoding
Content-Type: text/html
Connection: close

<!DOCTYPE html>
<html lang="en">
```

Risposa: Status Line

- Ha la seguente struttura :

  `Versione-HTPP SP Status-Code SP Reason-Phrase CRLF`

- Lo Status-Code è un codice a tre cifre che ha la funzione di fornire al client delle informazioni di stato riguardo all'esito della ricezione della richiesta.
- Associato ad ogni codice abbiamo la "reason-Phrase" che è una piccola descrizione del significato del codice destinata all'uso umano.
- Il primo digit dello Status-Code definisce 5 classi di risposta
  - 1xx : Informazione - richiesta ricevuta e continuo processo
  - 2xx : Successo - L'azione è stata ricevuta, capita e accettata
  - 3xx : Ridirezione - C'è bisogno di altre informazioni per completare la richiesta
  - 4xx : Client Error - Errori nella richiesta
  - 5xx : Server Error - Il server fallisce

Risposta: Status Line

CODICE	FRASE	CODICE	FRASE
100	Continue	404	Not Found
101	Switching Protocols	405	Method Not Allowed
200	OK	406	Not Acceptable
201	Created	407	Proxy Authentication Required
202	Accepted	408	Request Time-out
203	Non-Authoritative Information	409	Conflict
204	No Content	410	Gone
205	Reset Content	411	Length Required
206	Partial Content	412	Precondition Failed
300	Multiple Choices	413	Request Entity Too Large
301	Moved Permanently	414	Request-URI Too Large
302	Found	415	Unsupported Media Type
303	See Other	416	Requested range not satisfiable
304	Not Modified	417	Expectation Failed
305	Use Proxy	500	Internal Server Error
307	Temporary Redirect	501	Not Implemented
400	Bad Request	502	Bad Gateway
401	Unauthorized	503	Service Unavailable
402	Payment Required	504	Gateway Time-out
403	Forbidden	505	HTTP Version not supported

”GET” Request Method

- GET è il metodo di richiesta HTTP più frequente.
- Un client può utilizzare il metodo richiesta GET per richiedere per un pezzo di risorsa da un server HTTP.
- Un messaggio di richiesta GET prende la seguente sintassi:

```
GET request-URI HTTP-version
( optional request headers )
( blank line )
( optional request body )
```

- La keyword GET è case sensitive e deve essere in maiuscolo.
- request-URI : specifica il percorso della risorsa richiesta, che deve iniziare dalla root ”/”
- HTTP-Version: HTTP/<version>.
  - Il client negozia il protocollo da utilizzare. Ad esempio, il cliente può chiedere di utilizzare HTTP/1.1. Se il server non supporta HTTP/1.1, informa il cliente di utilizzare HTTP/1.0.

Esempio: la richiesta contiene «get» invece di «GET»

- Il server restituisce un errore ”501 Not implemented”.
  Nel Header della risposta, ”Allow” indica al client i metodi consentiti.

```
language=XML,aboveskip=1em,basicstyle=\tiny,numbers=none]
get /test.html HTTP/1.0
(enter twice to create a blank line)

language=HTML,aboveskip=0em,basicstyle=\tiny,numbers=none]
HTTP/1.1 501 Method Not Implemented
Date: Sun, 18 Oct 2009 10:32:05 GMT
Server: Apache/2.2.14 (Win32)
Allow: GET,HEAD,POST,OPTIONS,TRACE
Content-Length: 215
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>501 Method Not Implemented</title>
</head><body>
<h1>Method Not Implemented</h1>
<p>get to /index.html not supported.<br />
</p>
</body></html>
```

Esempio: 404 File Not Found

- In this GET request, the request-URL ”/pathsbagliato” cannot be found under the server’s document directory.
  - The server returns an error ”404 Not Found”.

```
*** MESSAGE From informatica.dista.uninsubria.it:

HTTP/1.1 404 Not Found
Date: Wed, 18 Apr 2018 10:25:19 GMT
Server: Apache/2.2.22 (Debian)
Vary: Accept-Encoding
Content-Length: 308
Connection: close
Content-Type: text/html; charset=iso-8859-1

<fine header>
*** closing...
```

Esempio: Wrong HTTP Version Number

- In questa richiesta GET, HTTP-version è stato scritto male (HTTTP/1.1).
  - Il server restituisce un errore ”404 Bad Request”.

```
*** MESSAGE From informatica.dista.uninsubria.it:

HTTP/1.1 400 Bad Request
Date: Wed, 18 Apr 2018 10:27:13 GMT
Server: Apache/2.2.22 (Debian)
Vary: Accept-Encoding
Content-Length: 301
Connection: close
Content-Type: text/html; charset=iso-8859-1

<fine header>
*** closing...
```