

---

# SISTEMAS WEB

## CURSO 2022/2023

### HTTP - HyperText Transfer Protocol

#### Content Length vs Transfer Encoding



Web Sistemak by [Oskar Casquero](#) & [María Luz Álvarez](#) is licensed under a [Creative Commons Reconocimiento 4.0 Internacional License](#).

# FUNCIONAMIENTO DE HTTP:

## CONTENT LENGTH Y TRANSFER ENCODING

---

- Un cliente web tiene que leer una respuesta enviada desde un servidor web, necesita saber dónde acaba el cuerpo del mensaje.
  - Una forma que tiene el protocolo HTTP de expresar la longitud del cuerpo del mensaje es con la cabecera "Content-Length".
    - NOTA: la cabecera "Content-Length" requiere conocer el tamaño del contenido que se va a incluir en el cuerpo del mensaje de la respuesta antes de enviarlo.
- A veces hay que enviar una gran cantidad de datos en la respuesta, pero no es posible conocer el tamaño de esos datos hasta que la petición ha sido procesada en su totalidad. Ejemplo: supongamos que los resultados de una solicitud a una base de datos se quieren visualizar en una página web.
  - Si se utiliza la cabecera "Content-Length", hay que conocer el tamaño de los datos que van en el cuerpo del mensaje antes de enviar la respuesta HTTP. Esto, por un lado, supone esperar a finalizar el procesamiento de la solicitud de la base de datos. Por otro lado, para guardar los datos se necesita un gran buffer.
  - El retraso generado por el tiempo necesario para procesar la respuesta antes de su envío influye negativamente en la experiencia del usuario.
- El protocolo HTTP proporciona un mecanismo de ir enviando partes del contenido solicitado por el cliente mientras el servidor web procesa la solicitud: cabecera "**Transfer-Encoding**" con valor "**chunked**".

# FUNCIONAMIENTO DE HTTP:

## CONTENT LENGTH Y TRANSFER ENCODING

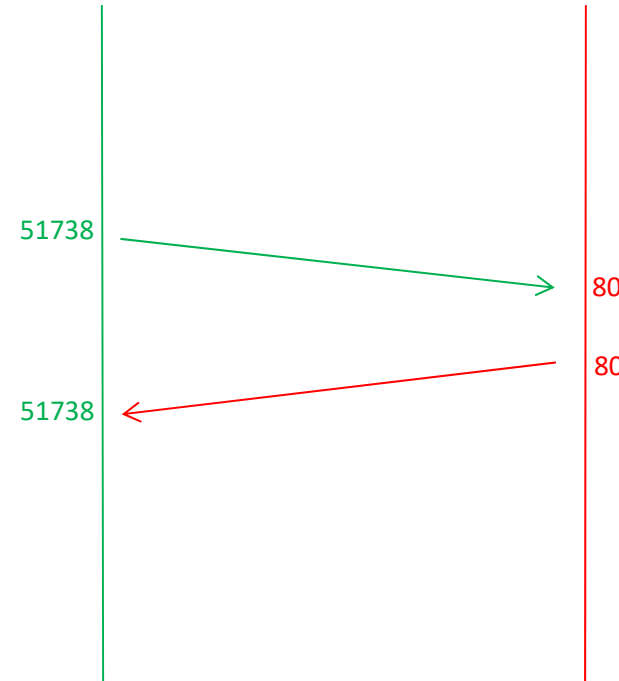
---

- Ejemplo de “**Content-Length**”: supongamos que el servidor tiene que enviar el texto “Hello World!!”

### Respuesta HTTP con “Content-Length”

```
HTTP/1.1 200 OK
Date: Thu, 20 Nov 2015 20:25:52 GMT
Last-Modified: Tue, 17 Sep 2015 13:00:02 GMT
ETag: "1a968-3ec-4e693e61bb8b6"
Content-Length: 13
Content-Type: text/plain

Hello World!!
```

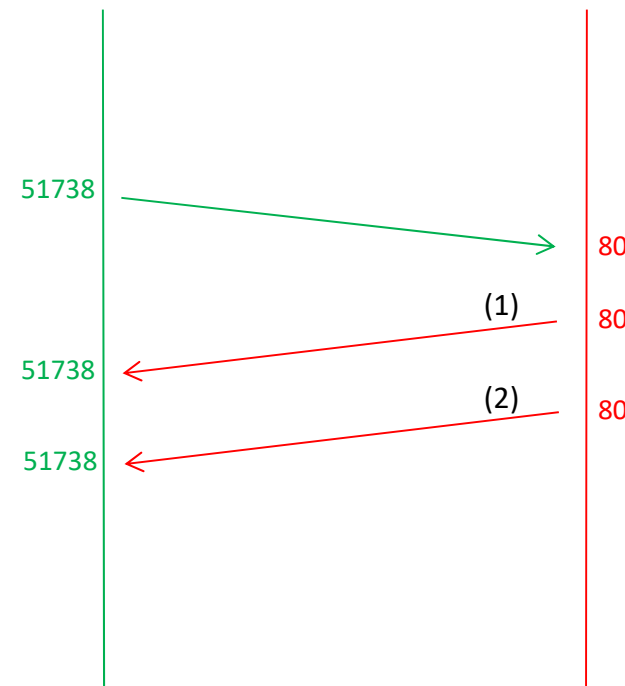
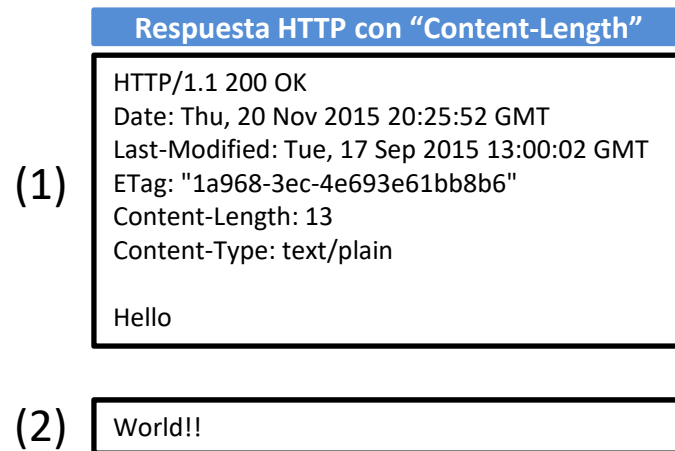


# FUNCIONAMIENTO DE HTTP:

## CONTENT LENGTH Y TRANSFER ENCODING

---

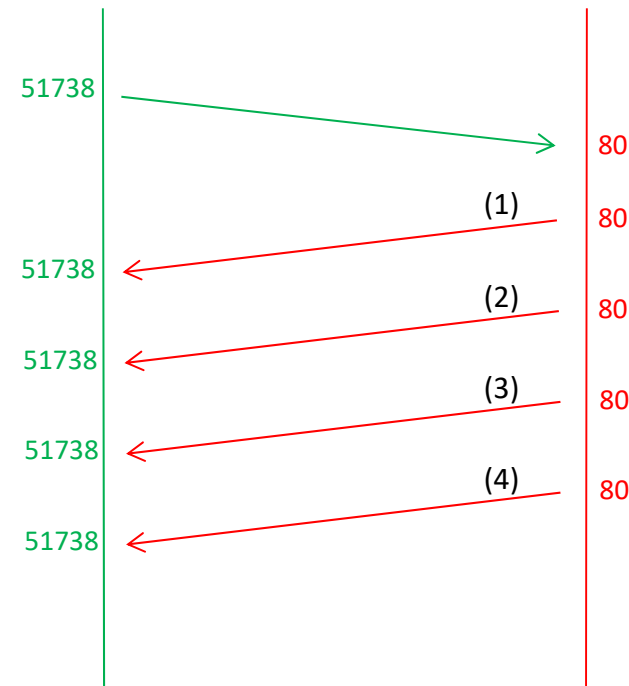
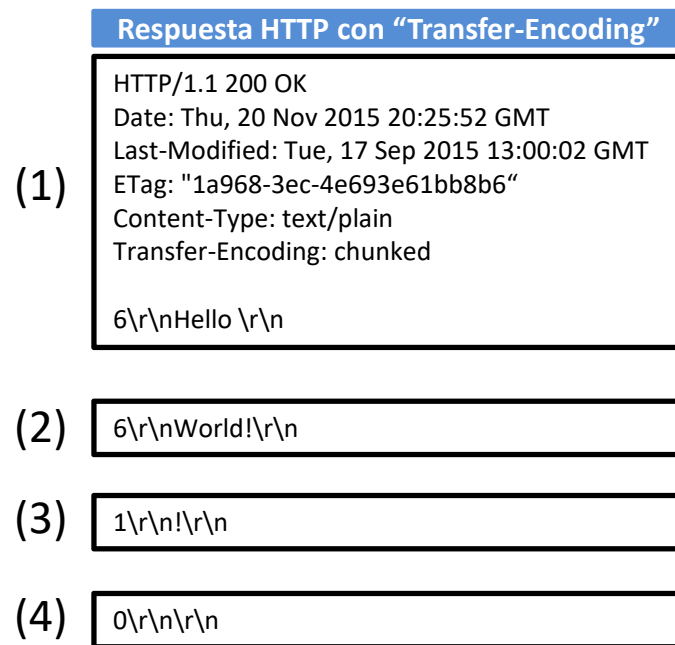
- NOTA: si el contenido de una respuesta es mayor que la MTU (Maximum Transmission Unit) de la capa TCP, el mensaje HTTP se envía en varios segmentos.
- Ejemplo: supongamos que la MTU (Maximum Transmission Unit ) de la capa TCP son 187 octetos.



# FUNCIONAMIENTO DE HTTP:

## CONTENT LENGTH Y TRANSFER ENCODING

- Ejemplo “**Transfer-Encoding: chunked**”: supongamos que el servidor tiene que enviar el texto “Hello World!!”



La longitud de cada trozo se indica en hexadecimal.

Los trozos 2, 3 y 4 no llevan cabeceras HTTP. Van directamente encapsulados en segmentos TCP.

# FUNCIONAMIENTO DE HTTP:

## CONTENT LENGTH Y TRANSFER ENCODING

---

- Diferencias entre ambos mecanismos:
  - con "**Content-Length**" los mensajes no se empiezan a enviar hasta que se ha procesado toda la respuesta (latencia alta), mientras que con "**Transfer-Encoding**" los mensajes se envían según se van procesando los trozos (latencia baja)
  - con "**Content-Length**" el troceado de los mensajes (dependiente de la MTU) lo hace la capa TCP, mientras que con "**Transfer-Encoding**" lo hace la propia capa HTTP.

## EJEMPLO:

### ANÁLISIS DE LA DESCARGA POR “TROZOS” DE UNA IMAGEN

---

- Haciendo uso del navegador descarga la siguiente imagen:  
<http://www.httpwatch.com/httpgallery/chunked/chunkedimage.aspx>
- Utilizando Wireshark analiza la captura
  - ¿Cuál es la relación entre el tamaño del campo de datos del segmento TCP y la parte del mensaje HTTP?
    - $3 + 2 + 1024 + 2 = 1031 \rightarrow$  Explicar de dónde salen estos números

# EJEMPLO:

## ANÁLISIS DE LA DESCARGA POR “TROZOS” DE UNA IMAGEN

The image shows a Wireshark packet capture window titled "\*Conexión de red inalámbrica". The filter bar at the top shows "tcp.port==50021". The packet list pane displays several packets, with packet 162 selected. The packet details pane shows the structure of packet 162, with the "Transmission Control Protocol" section highlighted. The "Len: 1031" field in the TCP section is also highlighted. Below the packet details, the raw packet data is shown in hexadecimal and ASCII format.

No.	Time	Source	Destination	Protocol	Length	Info
146	7.557819	191.236.16.125	10.107.21.103	TCP	1085	[TCP segment of a reassembled PDU]
147	7.557953	10.107.21.103	191.236.16.125	TCP	54	50021 → 80 [ACK] Seq=155 Ack=28144 Win=66048 Len=0
149	7.671041	191.236.16.125	10.107.21.103	TCP	1085	[TCP segment of a reassembled PDU]
151	7.778514	191.236.16.125	10.107.21.103	TCP	1085	[TCP segment of a reassembled PDU]
152	7.778684	10.107.21.103	191.236.16.125	TCP	54	50021 → 80 [ACK] Seq=155 Ack=30206 Win=66048 Len=0
155	7.888757	191.236.16.125	10.107.21.103	TCP	1085	[TCP segment of a reassembled PDU]
159	7.997952	191.236.16.125	10.107.21.103	TCP	1085	[TCP segment of a reassembled PDU]
160	7.998113	10.107.21.103	191.236.16.125	TCP	54	50021 → 80 [ACK] Seq=155 Ack=32268 Win=66048 Len=0
162	8.106597	191.236.16.125	10.107.21.103	TCP	1085	[TCP segment of a reassembled PDU]
164	8.218320	191.236.16.125	10.107.21.103	TCP	946	[TCP segment of a reassembled PDU]
165	8.218500	10.107.21.103	191.236.16.125	TCP	54	50021 → 80 [ACK] Seq=155 Ack=34191 Win=66048 Len=0
168	8.324934	191.236.16.125	10.107.21.103	HTTP	60	HTTP/1.1 200 OK (JPEG JFIF image)
170	8.355338	10.107.21.103	191.236.16.125	TCP	54	50021 → 80 [FIN, ACK] Seq=155 Ack=34196 Win=66048 Len=0
172	8.483130	191.236.16.125	10.107.21.103	TCP	60	80 → 50021 [FIN, ACK] Seq=34196 Ack=156 Win=131328 Len=0
173	8.483458	10.107.21.103	191.236.16.125	TCP	54	50021 → 80 [ACK] Seq=156 Ack=34197 Win=66048 Len=0

Frame 162: 1085 bytes on wire (8680 bits), 1085 bytes captured (8680 bits) on interface 0  
Ethernet II, Src: CheckPoi\_3a:b1:d8 (00:1c:7f:3a:b1:d8), Dst: IntelCor\_f2:d2:5c (10:0b:a9:f2:d2:5c)  
Internet Protocol Version 4, Src: 191.236.16.125, Dst: 10.107.21.103  
Transmission Control Protocol, Src Port: 80 (80), Dst Port: 50021 (50021), Seq: 32268, Ack: 155, Len: 1031

El segmento TCP tiene un tamaño de: **1031** octetos

```
0000  10 0b a9 f2 d2 5c 00 04 2f 51 11 40 00 6a 15 67 00 50 c3 65 71 02 01 cf fa 00 00 3d dd 89 a8 87 21 c5 ee 72 e1 ae c6 7a d2 5f 2c 33 ....r...Z...3
0010  04 2f 51 11 40 00 6a 15 67 00 50 c3 65 71 02 01 cf fa 00 00 3d dd 89 a8 87 21 c5 ee 72 e1 ae c6 7a d2 5f 2c 33 D.eFF.`G..^LS...
0020  15 67 00 50 c3 65 71 02 01 cf fa 00 00 3d dd 89 a8 87 21 c5 ee 72 e1 ae c6 7a d2 5f 2c 33 .....r6..2v...].
0030  02 01 cf fa 00 00 3d dd 89 a8 87 21 c5 ee 72 e1 ae c6 7a d2 5f 2c 33 ~*....].Z....].
0040  dd 89 a8 87 21 c5 ee 72 e1 ae c6 7a d2 5f 2c 33 .....dR..pzWD..
0050  44 dc 65 46 46 f0 60 47 eb cd 5e 4c 53 81 a9 02 0f 9b 95 19 89 72 36 a5 95 32 76 04 b5 81 5d 81 ....49x.....v.
0060  0f 9b 95 19 89 72 36 a5 95 32 76 04 b5 81 5d 81 2d 60 2a d6 04 b5 81 5d 81 5a c8 94 b5 81 5d 81 ...H...Z.3]Z..[
0070  2d 60 2a d6 04 b5 81 5d 81 5a c8 94 b5 81 5d 81 2d 1c 0a d6 02 ad 64 52 91 ea 70 7a 57 44 81 f0
0080  2d 1c 0a d6 02 ad 64 52 91 ea 70 7a 57 44 81 f0 c9 f1 0f 9f 7c e8 34 39 78 f1 8e f1 b3 c6 76 be
0090  c9 f1 0f 9f 7c e8 34 39 78 f1 8e f1 b3 c6 76 be 9f c3 cc 48 e5 2d ff 00 5a 13 33 5d 5a e1 8a 5b
00a0  9f c3 cc 48 e5 2d ff 00 5a 13 33 5d 5a e1 8a 5b
```



# EJEMPLO:

## ANÁLISIS DE LA DESCARGA POR “TROZOS” DE UNA IMAGEN

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp.port==50021

No.	Time	Source	Destination	Protocol	Length	Info
146	7.557819	191.236.16.125	10.107.21.103	TCP	1085	[TCP segment of a reassembled PDU]
147	7.557953	10.107.21.103	191.236.16.125	TCP	54	50021 → 80 [ACK] Seq=155 Ack=28144 Win=66048 Len=0
149	7.671041	191.236.16.125	10.107.21.103	TCP	1085	[TCP segment of a reassembled PDU]
151	7.778514	191.236.16.125	10.107.21.103	TCP	1085	[TCP segment of a reassembled PDU]
152	7.778684	10.107.21.103	191.236.16.125	TCP	54	50021 → 80 [ACK] Seq=155 Ack=30206 Win=66048 Len=0
155	7.888757	191.236.16.125	10.107.21.103	TCP	1085	[TCP segment of a reassembled PDU]
159	7.997952	191.236.16.125	10.107.21.103	TCP	1085	[TCP segment of a reassembled PDU]
160	7.998113	10.107.21.103	191.236.16.125	TCP	54	50021 → 80 [ACK] Seq=155 Ack=32268 Win=66048 Len=0
162	8.106597	191.236.16.125	10.107.21.103	TCP	1085	[TCP segment of a reassembled PDU]
164	8.218320	191.236.16.125	10.107.21.103	TCP	946	[TCP segment of a reassembled PDU]
165	8.218500	10.107.21.103	191.236.16.125	TCP	54	50021 → 80 [ACK] Seq=155 Ack=34191 Win=66048 Len=0
168	8.324934	191.236.16.125	10.107.21.103	HTTP	60	HTTP/1.1 200 OK (JPEG JFIF image)
170	8.355338	10.107.21.103	191.236.16.125	TCP	54	50021 → 80 [FIN, ACK] Seq=155 Ack=34196 Win=66048 Len=0
172	8.483130	191.236.16.125	10.107.21.103	TCP	60	80 → 50021 [FIN, ACK] Seq=34196 Ack=156 Win=131328 Len=0
173	8.483458	10.107.21.103	191.236.16.125	TCP	54	50021 → 80 [ACK] Seq=156 Ack=34197 Win=66048 Len=0

HTTP chunked response

- Data chunk (1024 octets)
  - Chunk size: 1024 octets
  - Data (1024 bytes)
  - Chunk boundary: 0d0a
- Data chunk (1024 octets)
  - Chunk size: 1024 octets
  - Data (1024 bytes)
  - Chunk boundary: 0d0a
- Data chunk (1024 octets)
  - Chunk size: 1024 octets
  - Data (1024 bytes)
  - Chunk boundary: 0d0a

Tamaño del fragmento en notación decimal

Tamaño del fragmento en notación hexadecimal

Longitud de los fragmentos de mensajes HTTP: 1024 octavas

0130 0d 0a 34 30 30 0d 0a ff d8 ff e1 00 18 45 78 69 ..400... Exi

0140 66 00 00 49 49 2a 00 08 00 00 00 00 00 00 00 ...I... ..

0150 00 00 00 ff ec 00 11 44 75 63 6b 79 00 01 00 04 .....D ucky...

0160 00 00 00 3c 00 00 ff ed 00 2c 50 68 6f 74 6f 73 ...<... ,Photos

0170 68 6f 70 20 33 2e 30 00 38 42 49 4d 04 25 00 00 hop 3.0. 8BIM.%..

0180 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00 ..... ..

0190 00 00 00 00 ff ee 00 0e 41 64 6f 62 65 00 64 c0 ..... Adobe.d.

Frame (60 bytes) Reassembled TCP (34195 bytes) De-chunked entity body (33653 bytes)

# EJEMPLO:

## ANÁLISIS DE LA DESCARGA POR “TROZOS” DE UNA IMAGEN

---

- RESUMEN:
  - Como se puede ver en las imágenes anteriores,
    - La longitud del contenido del segmento TCP es: **1031** octetos
    - La longitud del trozo del mensaje HTTP es: **1024** octetos
  - Un fragmento HTTP tiene la siguiente forma:
    - `LONG_FRAGMENTO\r\nCONTENIDO_FRAGMENTO\r\n`
  - El fragmento HTTP se mete en el campo de contenido de TCP, por tanto:
    - $\text{len}("1024") + \text{len}("\r\n") + 1024 + \text{len}("\r\n") = 4 + 2 + 1024 + 2 = \mathbf{1032}$
  - $1031 \neq 1032 \rightarrow$  ¿Por qué no coinciden los números?
    - ¡¡¡La longitud del fragmento debe meterse en notación hexadecimal!!!
      - 1024 en notación decimal = 400 en notación hexadecimal
      - $\text{len}("400") + \text{len}("\r\n") + 1024 + \text{len}("\r\n") = 3 + 2 + 1024 + 2 = \mathbf{1031}$