SISTEMAS WEB
CURSO 2022/2023

HTTP - HyperText Transfer Protocol

Codificación de la respuesta

María Luz Álvarez y Oskar Casquero
Dpto. Ingeniería de Sistemas y Automática

FUNCIONAMIENTO DE HTTP:

CUERPO DEL MENSAJE

- En el cuerpo del mensaje se envían octetos (bytes) que pueden representar:
  - Texto (texto plano, HTML, CSS, XML, JSON, CSS, …)
  - Contenido binario (PDF, vídeo, ejecutables, etc.)

- El contenido binario está destinado a aplicaciones o plataformas particulares. Por ejemplo, los PDF solo pueden ser entendidos por un lector de PDF. Si se abre un PDF con un editor de texto plano o se envía a una salida estándar, se muestra una representación incorrecta de su contenido, con base de texto.

- El texto puede ser visualizado en multitud de programas, pero no todos usan la misma configuración para decodificar los octetos.
  - Por ejemplo, si un fichero de texto fue codificado originalmente en UTF-8, el símbolo ”ñ” se grabará como C3 B1.
  - Si intentamos abrir el mismo fichero con un editor de textos configurado en ISO-8859-1, codificación en la que cada carácter se codifica con un solo octeto, la decodificación dará como resultado dos símbolos: Ã±.

FUNCIONAMIENTO DE HTTP:
ACLARACIONES SOBRE CODIFICACIÓN

Código	octeto 4	octeto 3	octeto 2	octeto 1
US-ASCII (7 bits)				
ISO-8859-1 (1 octeto)				
UTF-8 (1 a 4 octetos)				0
			1 1 0	1 0
		1 1 1 0	1 0	1 0
	1 1 1 1 0	1 0	1 0	1 0

- Latin-1 (IS0-8879-1) y UTF-8 llevan dentro la tabla del código US-ASCII.
- En latin-1 los símbolos añadidos a US-ASCII tiene n “1” en el bit mas significativo (MSB - Most Significant Bit).
- En UTF-8 los símbolos añadidos a US-ASCII por Latin-1 se codifican con dos octetos.

FUNCIONAMIENTO DE HTTP:
ACLARACIONES SOBRE CODIFICACIÓN

Código	Carácter Gráfico	Hex		binario								
US-ASCII (7 bits)	Z	--	5A			1	0	1	1	0	1	0
	ñ	--	--									
	á	--	--									
ISO-8869-1 (1 octeto)	Z	--	5A		0	1	0	1	1	0	1	0
	ñ	--	F1		1	1	1	1	0	0	0	1
	á	--	E1		1	1	1	0	0	0	0	1
UTF 8 (1- 4 octetos)	Z	--	5A		0	1	0	1	1	0	1	0
	ñ	C3	B1	1 1 0 0 0 0 1 1	1	0	1	1	0	0	0	1
	á	C3	A1	1 1 0 0 0 0 1 1	1	0	1	0	0	0	0	1

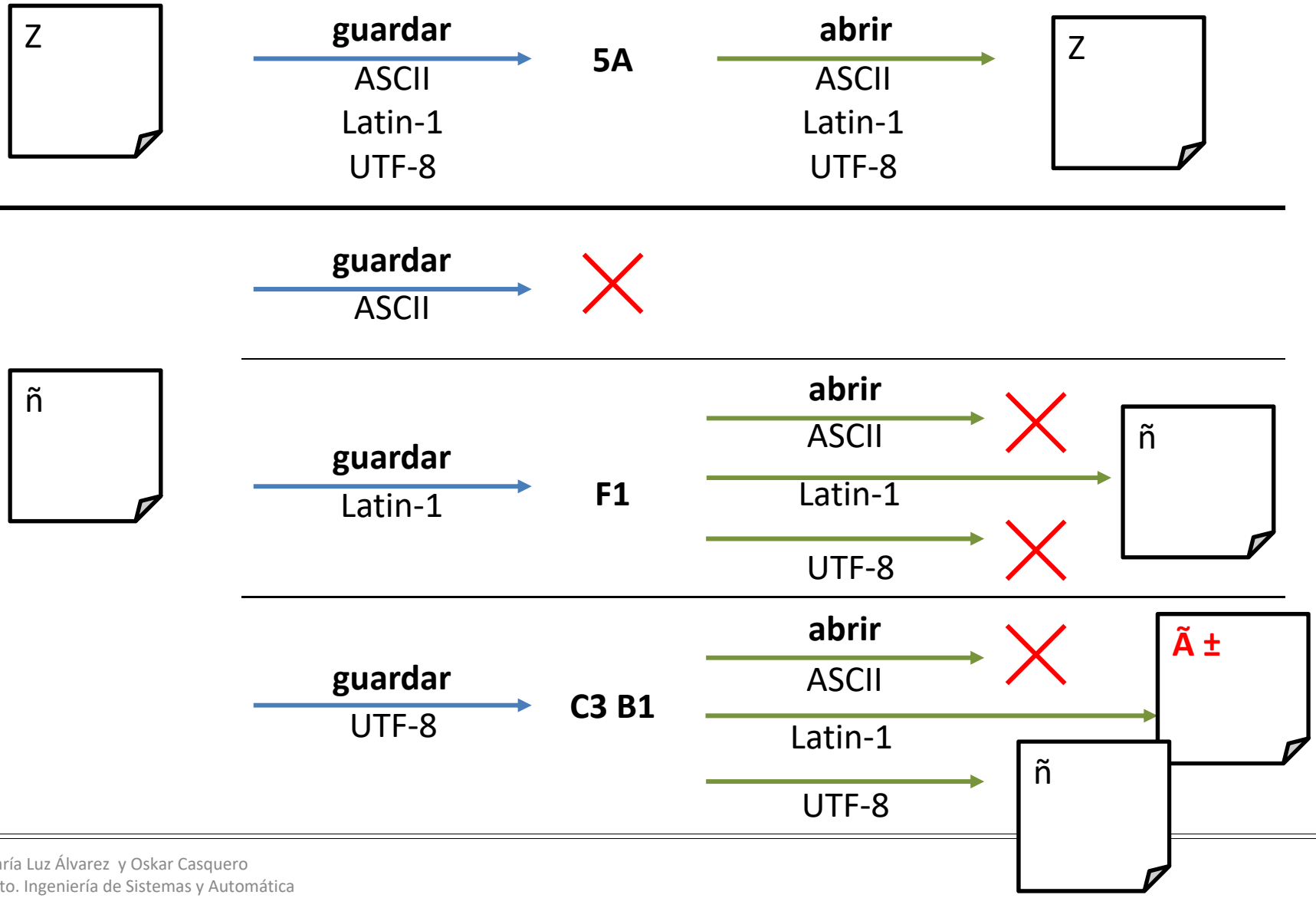
* Supongamos que se desea codificar el siguiente texto: Hola Iñaki Pérez!

US-ASCII: No se puede codificar, porque los símbolos “ñ” y “é” no están recogidos en el alfabeto US-ASCII.

Latin-1: 48 6f 6c 61 20 49 f1 61 6b 69 20 50 e9 72 65 7a 21

UTF-8: 48 6f 6c 61 20 49 c3 b1 61 6b 69 20 50 c3 a9 72 65 7a 21

FUNCIONAMIENTO DE HTTP:
ACLARACIONES SOBRE CODIFICACIÓN



FUNCIONAMIENTO DE HTTP:
CUERPO DEL MENSAJE

- El servidor web puede indicar a través de la cabecera “Content-Type” el tipo de contenido que está devolviendo: text/html, application/pdf, …
- Cuando el navegador lee la cabecera “Content-Type”, comprueba si su valor es de tipo texto o de otro tipo.
  - Si el contenido es de otro tipo, comprueba si tiene algún plugin configurado para abrirlo. Si no lo tiene, ofrece al usuario la opción de guardar el contenido o abrirlo con la aplicación correspondiente.
  - Si el contenido es de tipo texto, él mismo trata de interpretarlo (visualizarlo). Para ello, necesita saber cómo está codificado*.
    - HTTP permite especificar la codificación del contenido en la cabecera “Content-Type”:

      Content-Type: text/html; charset=ISO-8859-1 (Latin-1)

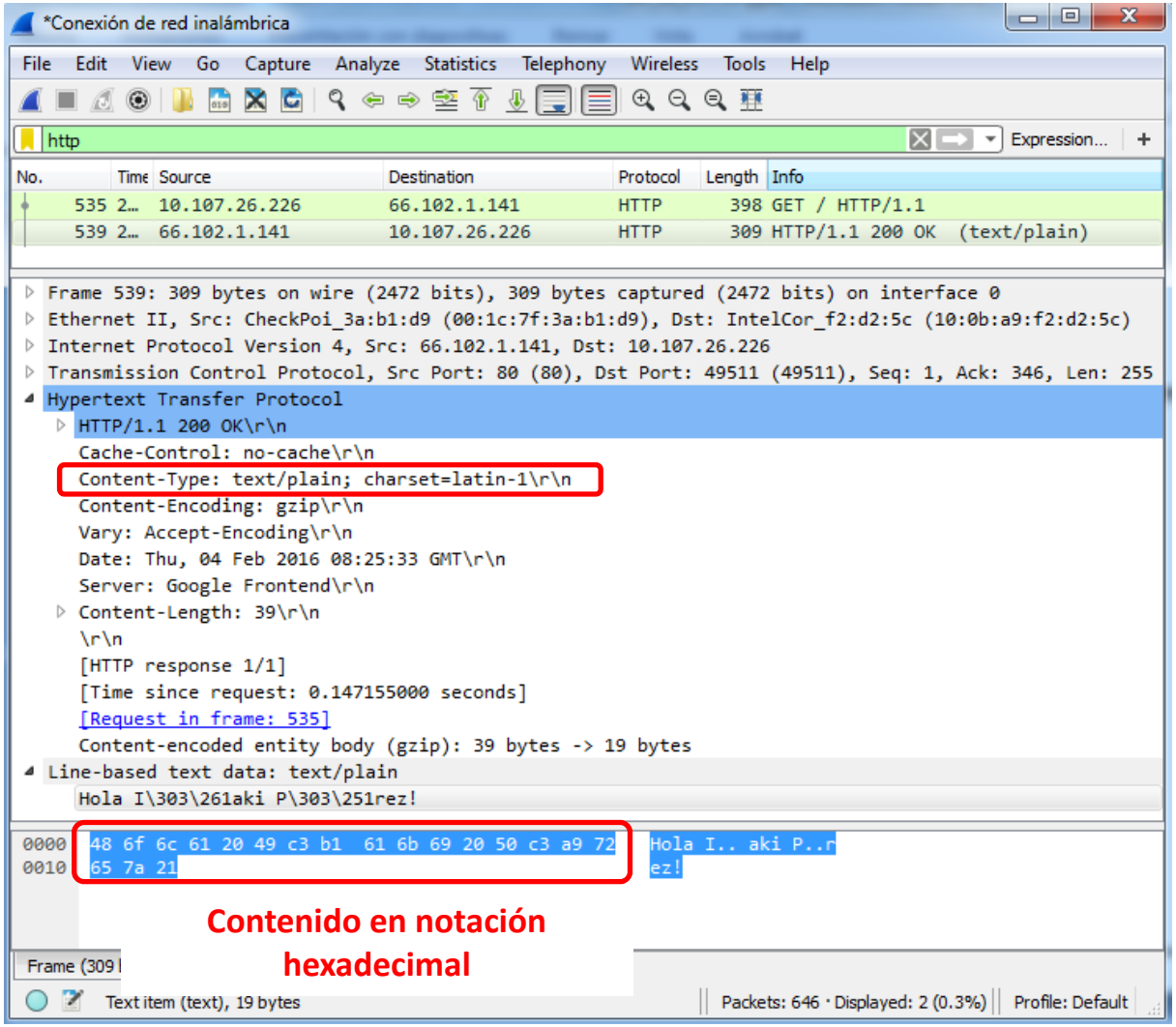
EJEMPLO:
ANÁLISIS DE UN PROBLEMA DE CODIFICACIÓN

Usando Wireshark, abre la siguiente URI en un navegador y encuentre la causa de su problema de codificación

http://ws-responsecontentcoding.appspot.com/

EJEMPLO:

ANÁLISIS DE UN PROBLEMA DE CODIFICACIÓN



- El navegador ve el contenido como "text/plain", por lo que es responsable de interpretar el contenido.
- ¿Cómo debe ser interpretado el texto por el navegador? Es decir, ¿qué tabla de codificación debería usar? Como el parámetro "charset" indica; en este caso: latin-1.

```
48 → H
6F → o
6C → l
61 → a
20 →
49 → I
C3 → Ã
B1 → ±
```

EJEMPLO:
ANÁLISIS DE UN PROBLEMA DE CODIFICACIÓN

- ¿Por qué ocurre esto?
  - Porque quien ha programado el servidor ha escrito una línea que indica que el contenido está codificado en latin-1:

```
self.response.charset = 'latin-1'
```

  - pero no ha tenido en cuenta que el servidor, cuando utiliza el método ***write,*** codifica el contenido en UTF-8

```
self.response.write("Hola Iñaki Pérez!")
```

Content-Type: text/plain; charset: latin-1

```
import webapp2class MainHandler(webapp2.RequestHandler):
    def get(self):
        self.response.content_type = 'text/plain'
        self.response.charset = 'latin-1'
        self.response.write("Hola Iñaki Pérez!")

app = webapp2.WSGIApplication([('/', MainHandler)], debug=True)
```