CEU

6 Capa de Aplicación

6.3 www

RdE 2014-2015

6 Guión del Tema 6

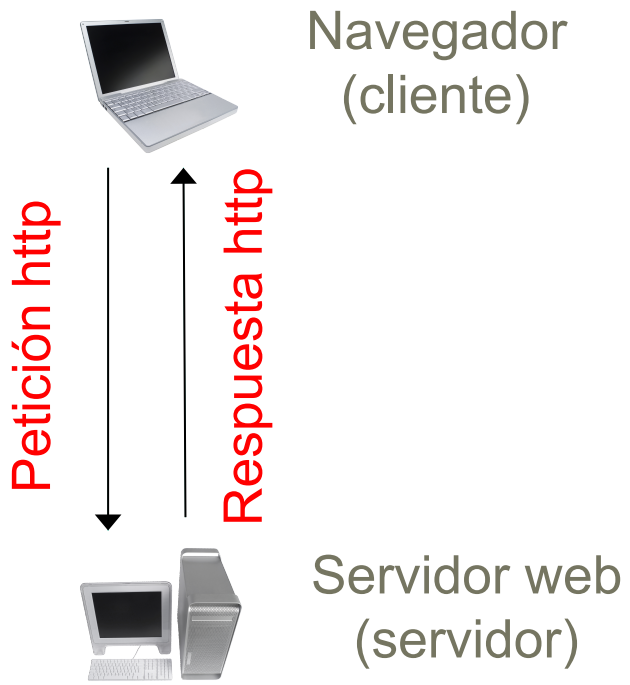
- 6. Capa de transporte:
  - 6.1 DNS.
  - 6.2 Correo electrónico.
  - **6.3 WWW.**
  - 6.4 Multimedia.

6.3 Protocolos www

- WWW proporciona un interfaz común para acceder a diferentes tipos de servicios/documentos a través de un sistema de nombres: ***Universal Resource Locator*** **(URL)**.
- Describe una forma de incluir enlaces a URLs en documentos textuales: ***HyperText Markup Language*** **(HTML).**
- Para acceder a WWW se utiliza un programa navegador (browser) que se encarga de obtener documentos hipermedia desde su lugar de origen, utilizando el protocolo ***HyperText Transfer Protocol*** **(HTTP)**.

6.3 HTTP

1. El cliente abre una conexión TCP con el servidor.
2. El cliente envía un **mensaje de petición.**
3. El servidor responde con un **mensaje de respuesta.**
4. El servidor cierra la conexión TCP.



HTTP no mantiene estado (no se guarda información sobre las peticiones anteriores hechas por el mismo cliente al mismo servidor).

6.3 HTTP

- **HTTP**: *HyperText Transfer Protocol*.
- Es el protocolo que se utiliza para servir páginas web.
- Protocolo cliente-servidor.
- Funciona sobre TCP, con el servidor (normalmente) en el puerto 80.
- No olvidar que HTTP puede servir tanto contenido **estático** (ficheros) como **dinámico** (el resultado de ejecutar programas en el servidor).

6.3 Página web

- Una página web consta de objetos.
- Un objeto es un archivo (un archivo HTML, una foto jpg, applet Java, etc) que es direccionable a través de su URL.
- La mayoría de las páginas web están formadas por un archivo HTML base y diversos objetos referenciados.
  - Ej: una página HTML y 5 objetos jpg es una página formada por 6 objetos.

6.3 Conexiones HTTP

Conexiones HTTP no persistentes	Conexiones HTTP persistentes
• Se envía un objeto como máximo por una conexión TCP. • HTTP/1.0 utiliza sólo conexiones HTTP no persistentes.	• Se pueden enviar múltiples objetos por una sola conexión TCP entre el cliente y el servidor. • HTTP/1.1 utiliza conexiones persistentes por omisión.

6.3 Conexiones HTTP no persistentes

Secuencia de entrada en una página web: www.sample.com/index.html

tiempo

1a. El cliente HTTP inicia la conexión TCP con el servidor HTTP (proceso) de www.sample.com en el puerto 80.

2. El cliente HTTP envía un mensaje HTTP de petición (que contiene la URL) a través de la conexión TCP. El mensaje indica que el cliente quiere el objeto /index.html

1b. El servidor HTTP en el host www.sample.com que estaba esperando conexiones TCP en el puerto 80 y "acepta" esta conexión, y se lo notifica al cliente.

3. El servidor HTTP recibe el mensaje de petición, compone un mensaje de respuesta que contiene el objeto solicitado, y lo envía al cliente.

6.3 Conexiones HTTP no persistentes

Secuencia de entrada en una página web: www.sample.com/index.html

tiempo

4. El servidor HTTP cierra la conexión TCP.

5. El cliente HTTP recibe el mensaje de respuesta que contiene el archivo html. Analizando el archivo html, se encuentran referenciados 10 objetos jpeg.

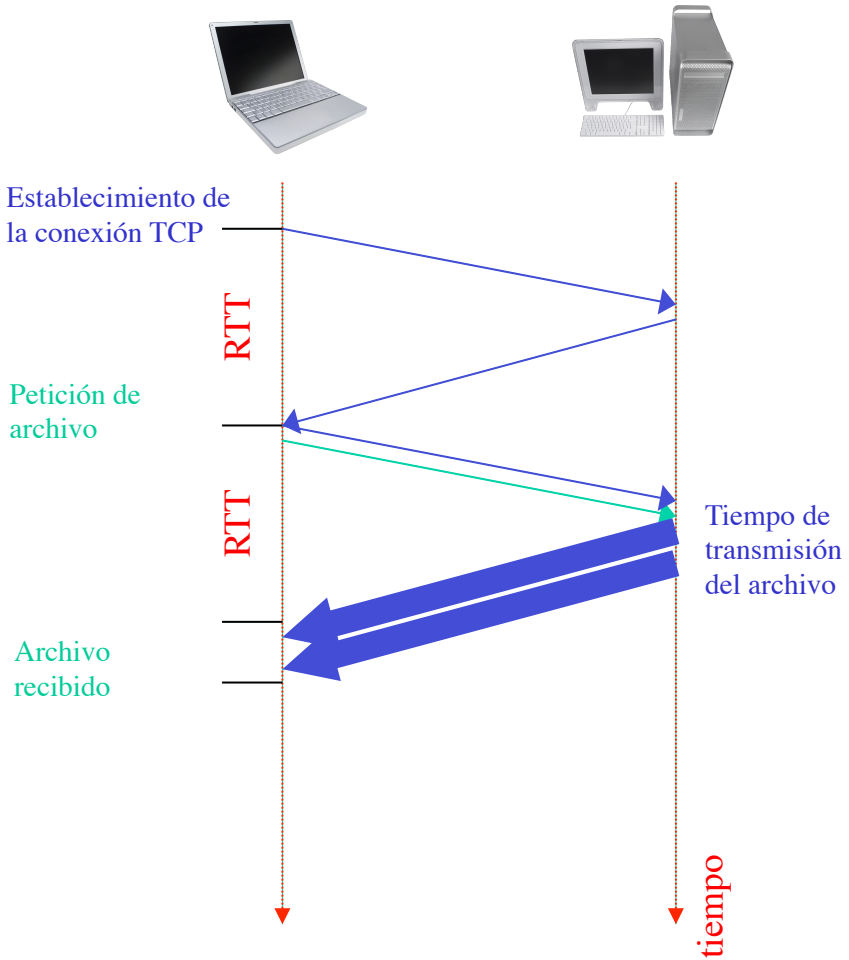
6. Se repiten los pasos del 1 al 5 para cada uno de los 10 objetos jpeg.

6.3 Modelo del tiempo de respuesta

Definición de RTT: tiempo respuesta necesario para enviar un paquete “pequeño” desde el cliente hasta el servidor y después de vuelta al cliente.

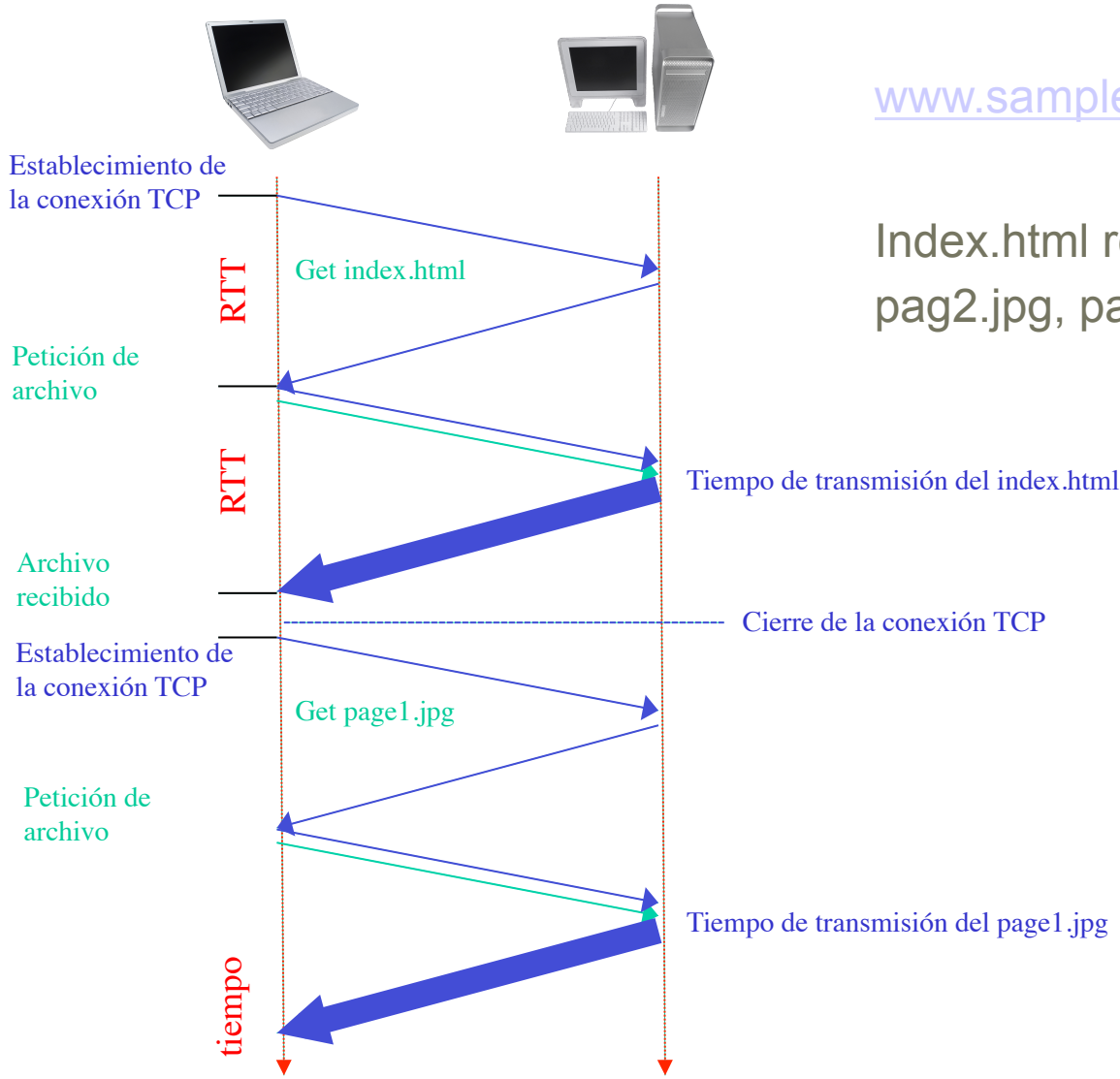
Tiempo de respuesta:

- Un RTT para iniciar la conexión TCP.
- Un RTT para la petición HTTP RTT y los primeros bytes de respuesta HTTP de vuelta.
- Tiempo de transmisión del archivo.



Total = 2RTT+tiempo de transmisión

6.3 HTTP 1.0



www.sample.com/index.html

Index.html referencia a page1.jpg, pag2.jpg, page3.jpg

6.3 Conexiones HTTP persistentes

HTTP no persistente

- Requieren 2 RTT por objeto
- El sistema operativo gasta tiempo en asignar los recursos del host para cada conexión TCP.
- Los navegadores suelen abrir conexiones TCP paralelas para traer los objetos referenciados.

HTTP persistente

- El servidor deja la conexión TCP abierta tras enviar la respuesta.
- Los mensajes HTTP posteriores entre el mismo cliente/servidor se envían por la misma conexión.
- El servidor cerrará la conexión inactiva pasado un plazo.

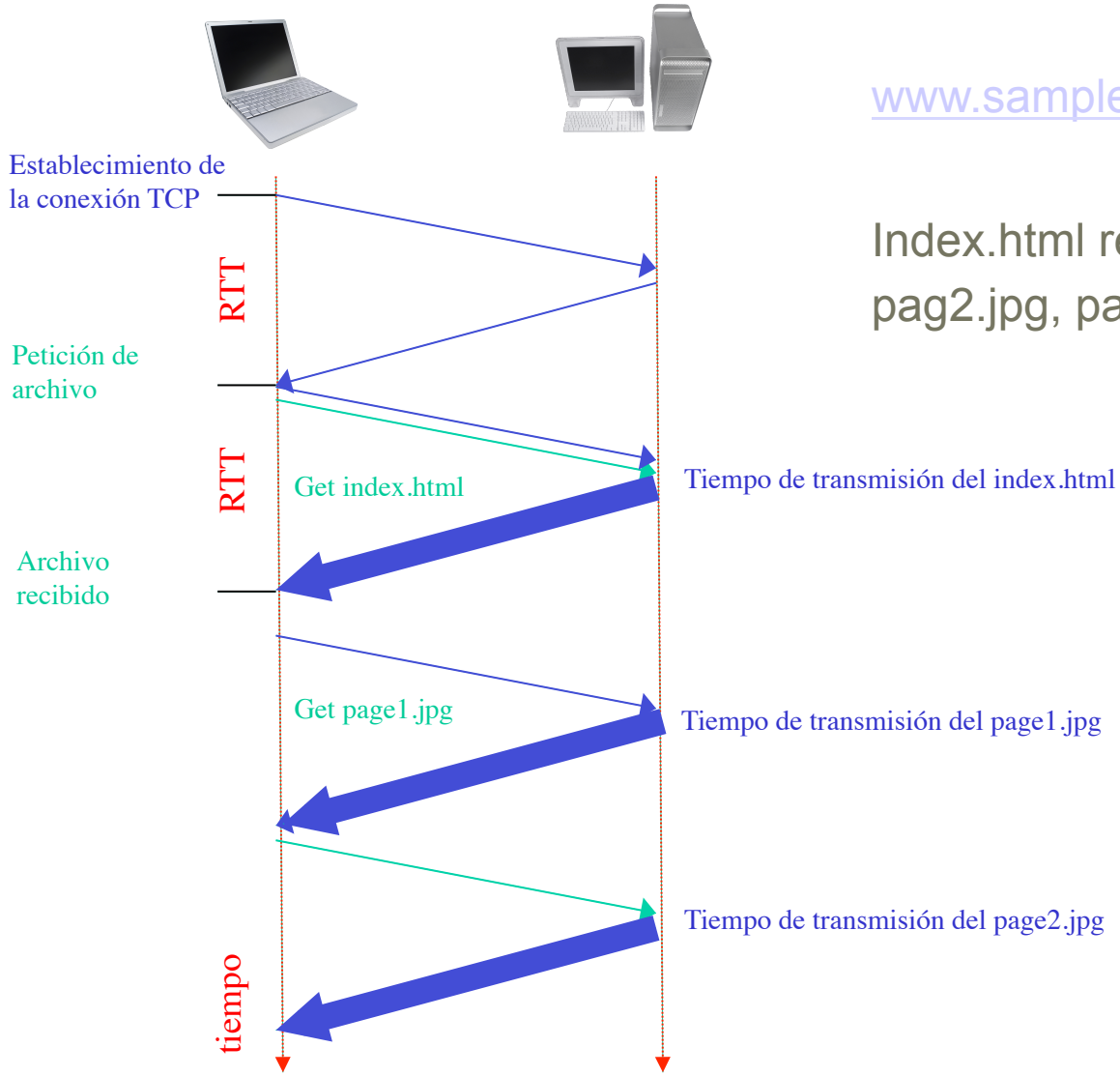
Conexiones persistentes sin *pipelinig*:

- El cliente sólo emite una nueva petición una vez que ha recibido la anterior respuesta.
- Un RTT por cada objeto referenciado.

Conexiones persistentes con *pipelining*:

- Por defecto en HTTP/1.1
- El cliente hace su petición tan pronto como encuentra un objeto referenciado.
- Tan sólo un RTT para todos los objetos referenciados.

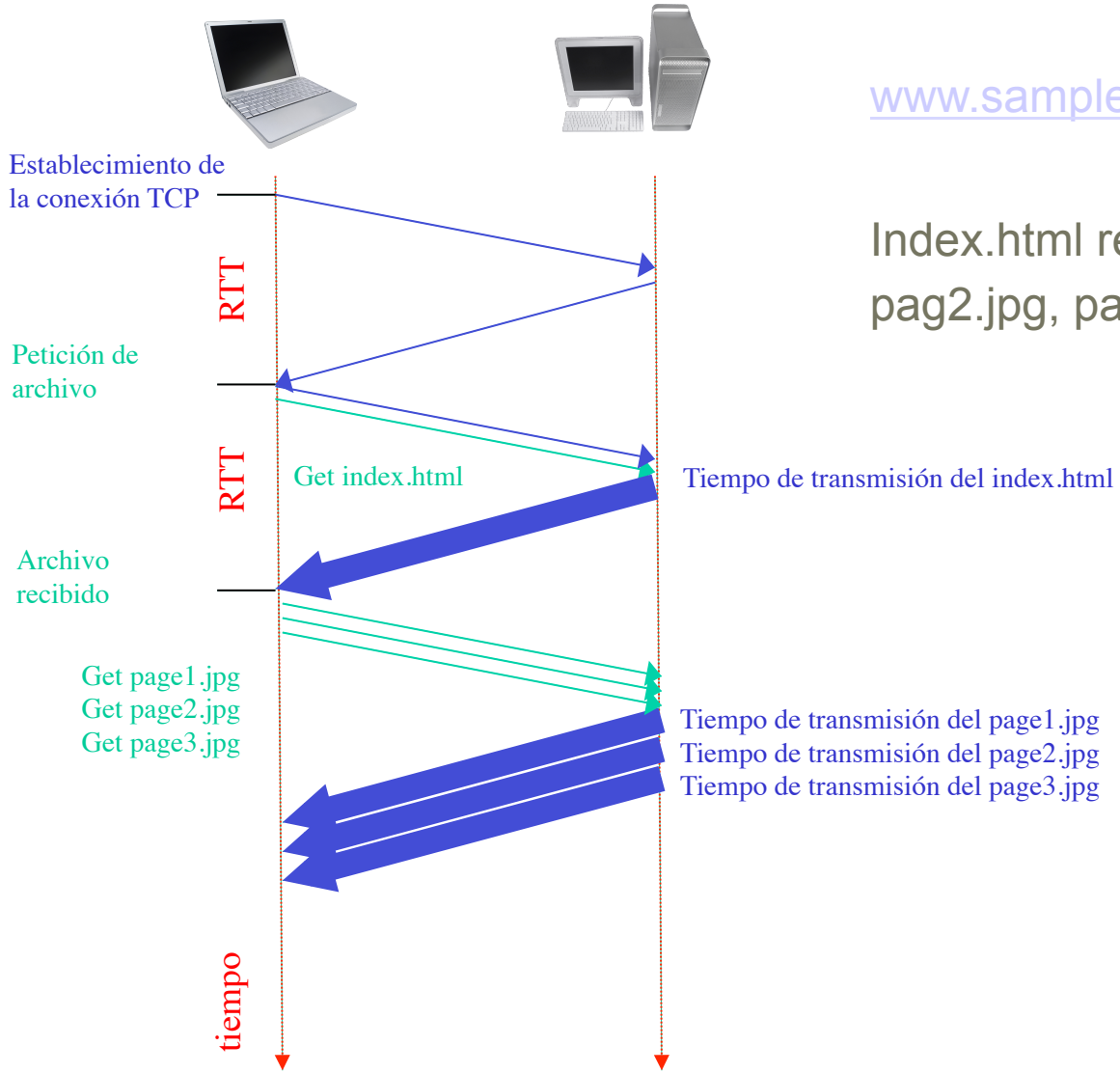
6.3 Conexiones persistentes sin *pipeline*



www.sample.com/index.html

Index.html referencia a page1.jpg, pag2.jpg, page3.jpg

6.3 Conexiones persistentes con *pipeline*



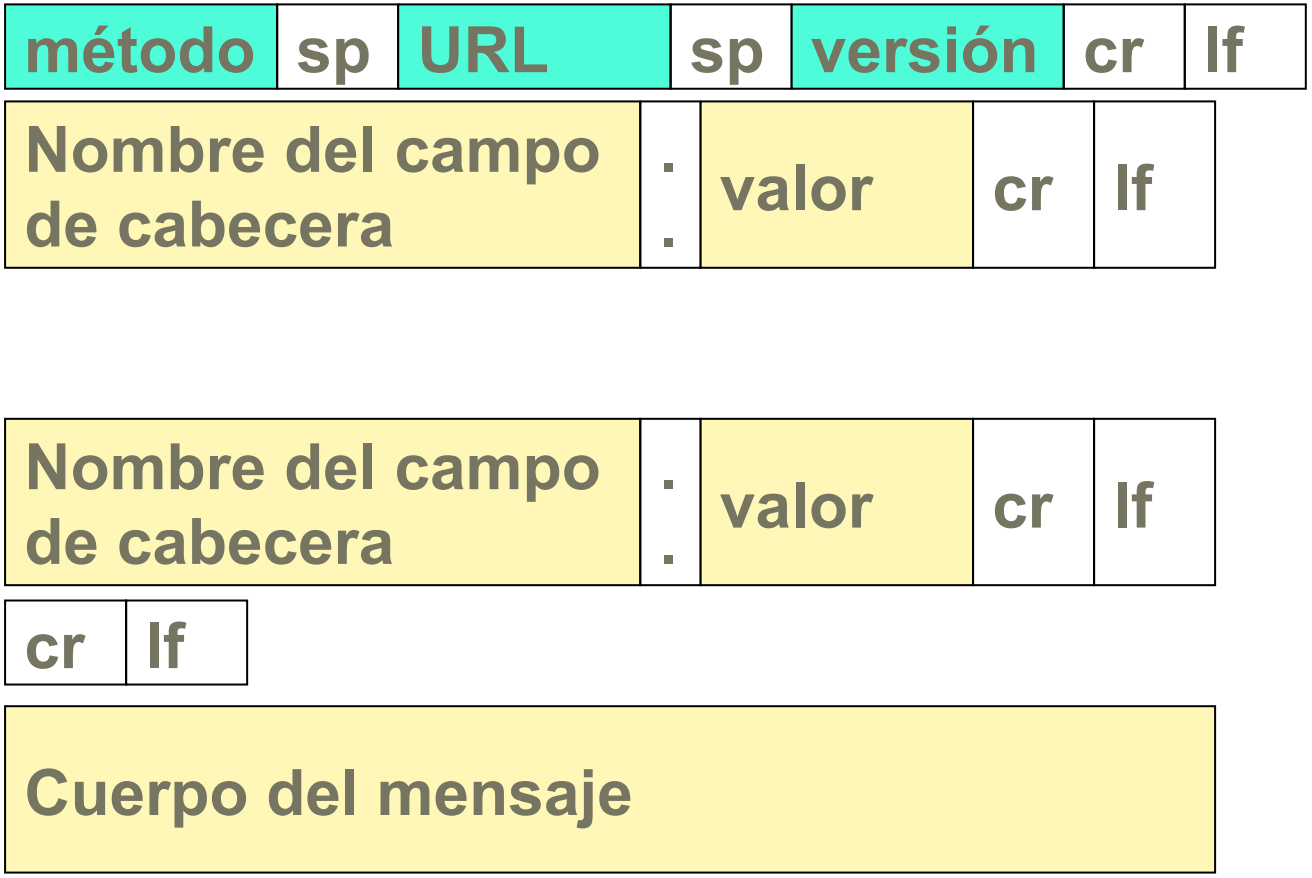
www.sample.com/index.html

Index.html referencia a page1.jpg, pag2.jpg, page3.jpg

6.3 Formato de los mensajes

- Mensajes “legibles” (compuestos por líneas de texto “casi en inglés”):
  - **Línea inicial de petición** (diferente para petición y respuesta), terminada en CRLF (*Carrier Return Line Feed*).
  - **Líneas de cabecera (0 ó más),** cada una terminada en CRLF: Cabecera-X: Valor-XCRLF
  - **Línea en blanco** (CRLF).
  - **Cuerpo del mensaje** (opcional).
- Además de CRLF, deberían tratarse adecuadamente líneas terminadas en LF.

6.3 Mensaje HTTP de petición: formato general



6.3 Línea inicial (peticiones)

- Especifica el recurso que se solicita y qué se quiere de él:
  - **Nombre del método** (GET, POST, HEAD)
  - **Camino de acceso** (path)
  - **Versión de HTTP** (siempre HTTP/x.y)

- Ejemplo:

GET	**/directorio/fichero.html**	**HTTP/1.0**

6.3 Ejemplo de mensaje HTTP de petición

Línea de petición (GET, POST, HEAD)

```
GET /directorio/fichero.html HTTP/1.1
```

Líneas de cabecera

```
Host: www.sample.es
User-agent: Mozilla/4.0
Connection: close
Accept-language: es
```

Retorno de carro y avance de línea que indican el final de la cabecera

6.3 Línea inicial (respuestas)

- Proporciona **información de estado:**
  - ❑ Versión de HTTP (siempre HTTP/x.y).
  - ❑ Código numérico de estado.
  - ❑ Código de estado “en inglés”.

- Códigos de estado:
  - ❑ 1xx: Mensaje informativo.
  - ❑ 2xx: Resultado exitoso (200 OK).
  - ❑ 3xx: Redirección del cliente a otra URL (301 Moved permanently, 303 See Other).
  - ❑ 4xx: Error en el lado del cliente (404 Not Found).
  - ❑ 5xx: Error en el lado del servidor (500 Server Error).

6.3 Ejemplo de mensaje HTTP de respuesta

Línea de información de estado (protocolo, código y el estado)

```
HTTP/1.1  200 OK
```

Líneas de cabecera

```
Connection: close
Date:  Thu, 06 Aug 2006 11:35:01 GMT
Server: Apache/1.3.0 (Unix)
Last-modified: Mon, 22 March 2006
Content-Length: 5422
Content-type: text/html
```

Retorno de carro y avance de línea

Datos, por ejemplo el archivo html solicitado

```
datos
```

6.3 Cabeceras comunes

- **Content-Type:**
  - Descripción MIME de la información contenida en este mensaje.
  - MIME (Multipurpose Internet Mail Extensions): Estándar que especifica como debe un programa transferir archivos multimedia (no ASCII). Los tipos MIME se especifican con “contenido/subtipo”:
    - text/html, text/plain ...
    - image/gif, image/jpeg, image/tiff ...
    - video/mpeg, video/quicktime ...
- **Content-Length:**
  - Longitud en bytes de los datos enviados.
- **Content-Encoding:**
  - Formato de codificación de datos enviados en el mensaje. Para enviar datos comprimidos (z-gzip, o z-compress).
- **Date:**
  - Fecha local de la operación, incluye zona horaria.

6.3 Cabeceras sólo para peticiones

- **Accept:**
  - Lista de tipos MIME aceptados por el cliente. Se puede utilizar * para indicar rangos de tipos de datos; tipo/* indica todos los subtipos de un determinado medio, mientras que */ * representa a cualquier tipo de dato disponible.
- **Authorization:**
  - Clave de acceso que envía un cliente para acceder a un recurso de uso protegido o limitado.
- **From:**
  - Dirección de correo electrónico del usuario del cliente Web que realiza el acceso.
- **If-Modified-Since:**
  - Permite realizar operaciones GET condicionales, en función de si la fecha de modificación del objeto requerido es anterior o posterior a la fecha proporcionada.
- **Referer:**
  - Contiene la URL del documento desde donde se ha activado este enlace. De esta forma, un servidor puede informar al creador de ese documento de cambios o actualizaciones en los enlaces que contiene.
- **User-agent:**
  - Cadena que identifica el tipo y versión del cliente que realiza la petición. Por ejemplo, los browsers de Netscape envían User-Agent: Mozilla/4.5 [en]

6.3 Cabeceras sólo respuestas

- **Allow**:
  - Informa de los comandos HTTP opcionales que se pueden aplicar sobre el objeto al que se refiere la respuesta
- **Expires:**
  - Fecha de expiración del objeto enviado.
- **Last-modified:**
  - Fecha local de modificación del objeto devuelto
- **Location:**
  - Informa sobre la dirección exacta del recurso al que se ha accedido. Cuando el servidor proporciona un código de respuesta de la serie 3xx, este parámetro contiene la URL necesaria para accesos posteriores a este recurso.
- **Server:**
  - Cadena que identifica el tipo y versión del servidor: Server: Apache/1.3.0 (Unix)
- **WWW-Authenticate:**
  - Cuando se accede a un recurso protegido o de acceso restringido, el servidor devuelve un código de estado 401, y utiliza este campo para informar de los modelos de autentificación validos para acceder a este recurso.

6.3 Cuerpo del mensaje

- En las respuestas contiene el recurso pedido o texto explicando un error.
- En las peticiones contiene datos de usuario o ficheros para subir.
- Si hay cuerpo, normalmente hay algunas cabeceras relativas a él:
  - ❑ “Content-Type”: tipo MIME (Multipurpose Internet Mail Extensions) de los datos.
  - ❑ “Content-Length”: número de bytes en el cuerpo.

6 Capa de Aplicación

6.3 www

RdE 2013-2014

6.3 Métodos HEAD, GET y POST

- **GET**:
  - Solicita un objeto al servidor especificando su URL.
- **HEAD:**
  - Igual que un GET, pero sólo pide las cabeceras.
    Se pueden consultar las características sin bajarse el fichero:
  - Permite que los clientes puedan verificar un link o comprobar si ha habido modificaciones en cierto objeto, sin necesidad de transferir todo el objeto.
- **POST**:
  - Hay datos en el cuerpo (que se "suben" al servidor). Se especifica en la cabecera Content-Type y *Content-Length*.
  - La URL "pedida" es normalmente el programa que trata los datos enviados. También se pueden enviar datos con un GET (codificándolos en la URL pedida).

6.3 Ejemplo de envío de datos GET/POST

GET /index.jsp?name=Mr+Curiosity&OK=1 HTTP/1.0 Host: www.sample.com
User-Agent: Mozilla/4.5 [en]
Accept: image/jpeg, image/gif, text/html
Accept-language: en
Accept-Charset: iso-8859-1

?: separación entre el recurso de los parámetros

+: espacio

&: separación entre parámetros

POST /index.jsp HTTP/1.0
Host: www.sample.com
User-Agent: Mozilla/4.5 [en]
Accept: image/jpeg, image/gif, text/html
Accept-language: en
Accept-Charset: iso-8859-1
Content-Type: application/x-www-form-urlencoded
Content-Length: 26

Nombre=Mr+Az&OK=1

6.3 Otros métodos

- **PUT:**
  - Actualiza información sobre un objeto del servidor. Similar a POST, pero el servidor debe almacenar en URL que acompaña el comando el contenido del mensaje.
  - Originalmente para subir a un servidor páginas WWW.
- **DELETE:**
  - Elimina en el servidor el documento especificado.
- **LINK:**
  - Crea en el servidor una relación entre documentos.
- **UNLINK:**
  - Elimina una relación existente entre documentos del servidor.
- …

6.3 Prueba de cliente HTTP

1. Telnet al servidor

2. Escribir petición HTTP

3. Ver la respuesta del servidor

1. **Telnet localhost 80**

   Abre conexión TCP por el puerto 80. Lo que se envíe irá al servidor del puerto 80.

2. **GET / HTTP/1.0**

   Pide la página /. Necesario dar dos veces al retorno (¿pq?).

3. xp apache.txt

6.3 HTTP 1.1

- Es una evolución de HTTP 1.0.
- Facilidades específicas para máquinas virtuales (**virtual hosts).**
- **Codificación por trozos,** para respuestas
- Uso de **conexiones persistentes** que permiten varios envíos sucesivos (se evitan establecimientos de conexión).
- Facilidades específicas para **cachés** (“If-Modified-Since”, “If-Unmodified-Since”).

6.3 Virtual Host

- Permite tener más de un sitio web en una sola máquina.
- Es necesario indicar en cada petición a cuál de ellos se dirige. Es obligatorio el uso de la cabecera “Host”.
  - Los sitios web virtuales pueden estar "basados en direcciones IP", lo que significa que cada sitio web tiene una dirección IP diferente.
  - O "basados en nombres diferentes", lo que significa que con una sola dirección IP están funcionando sitios web con diferentes nombres (de dominio).
- Si un servidor recibe una petición sin “Host”, debe devolver un mensaje de error (400 *Bad Request*).
- Los servidores también han de aceptar primeras líneas de petición con URLs completas, en lugar de caminos (será lo habitual en versiones futuras).
- Ejemplo de petición “mínima”:
  - GET /index.html HTTP/1.1\r\n
  - Host: www.sample.com\r\n \r\n

6.3 Codificación por trozos

- Cabecera "*Transfer-Encoding: chunked*". Usada por un servidor que quiere enviar trozos que ya tiene listos antes de tener el total de la petición.
- El cuerpo de cada mensaje "por trozos" contiene una serie de pedazos, cada uno:
  - Comienza por el tamaño de los datos del trozo, en hexadecimal, seguidos por ";", quizás algo más, y CRLF.
  - Los datos, terminados por CRLF.
  - Finaliza con una línea con "0CRLF" (longitud 0, se han terminado los trozos)
  - Seguido de "footers" (como cabeceras).
  - Acabado en una línea en blanco (CRLF).

6.3 Ejemplo de una respuesta por trozos

```
HTTP/1.1 200 OK\r\n
Date: Fri, 06 Apr 2012 14:00:00 GMT\r\n
Content-Type: text/plain\r\n
Transfer-Encoding: chunked\r\n
\r\n
```

```
1a; comentario sobre los datos\r\n
abcdefghijklmnopqrstuvwxyz\r\n
10;\r\n
1234567890abcdef\r\n
0\r\n
a-footer: su-valor\r\n
another-footer: otro-valor\r\n
\r\n
```

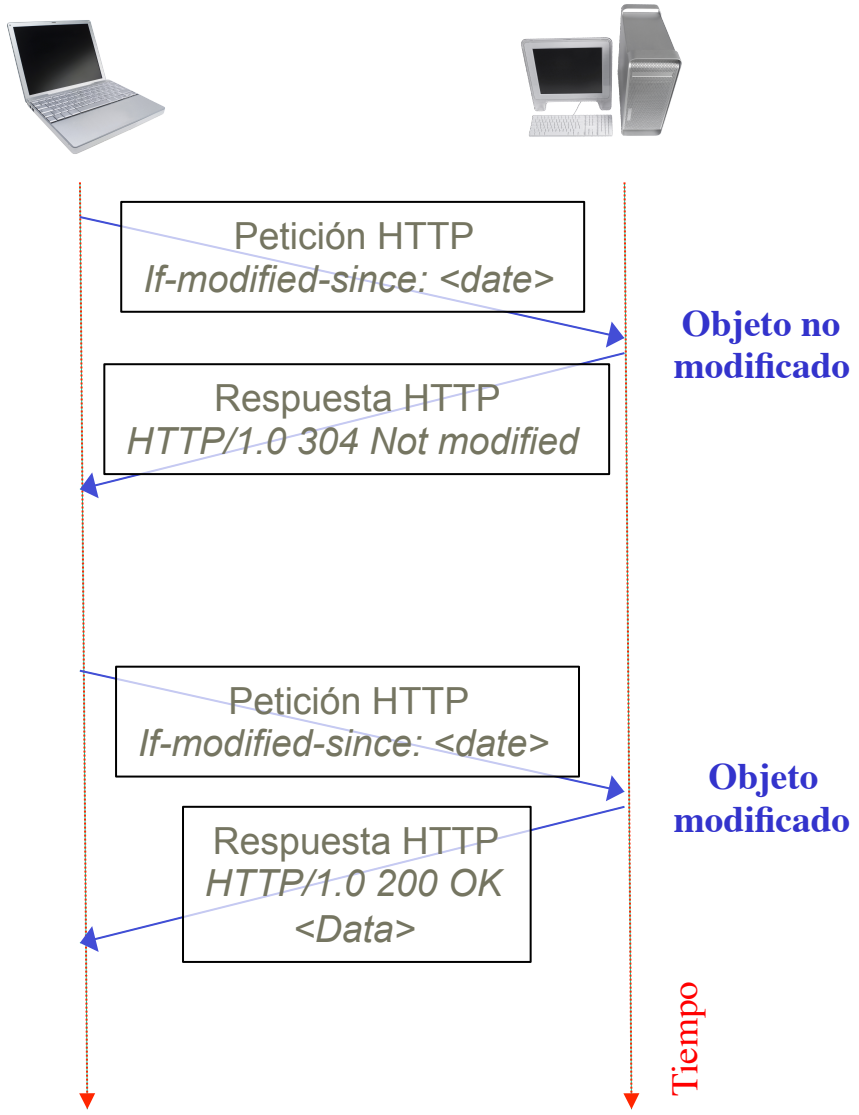
- Longitud total dos trozos = 42 = 0x1a + 0x10

6.3 Conexiones persistentes

- Permiten que varias peticiones y respuestas sean transferidas usando la misma conexión TCP.
- Se usan por omisión en HTTP 1.1.
- Si se envía la cabecera “Connection: close” el servidor cerrará la conexión después de cada respuesta.
- Un servidor puede cerrar la conexión antes de enviar todas las respuestas.
- El servidor cerrará las conexiones inactivas pasando un plazo (ej: 10 segundos).

6.3 Caché en el cliente

- **Objetivo:** no enviar objetos si el cliente tiene una versión caché actualizada.
- Cliente: especifica la fecha de la copia en caché en la petición HTTP: **If-modified-since: <date>**
- Servidor: su respuesta no contiene ningún objeto si la copia en caché está actualizada: **HTTP/1.0 304 Not Modified**



6.3 Cachés

- Los servidores deben responder siempre con la cabecera "*Date*" (con la fecha actual, en GMT).
- Los servidores han de entender "*If-Modified- Since*" y "*If-Unmodified-Since*" (los clientes pueden usarlos).
- Respuesta a "*If-Modified-Since*": "*Not Modified*".
- Respuesta a "*If-Unmodified-Since*": "*Precondition Failed*".

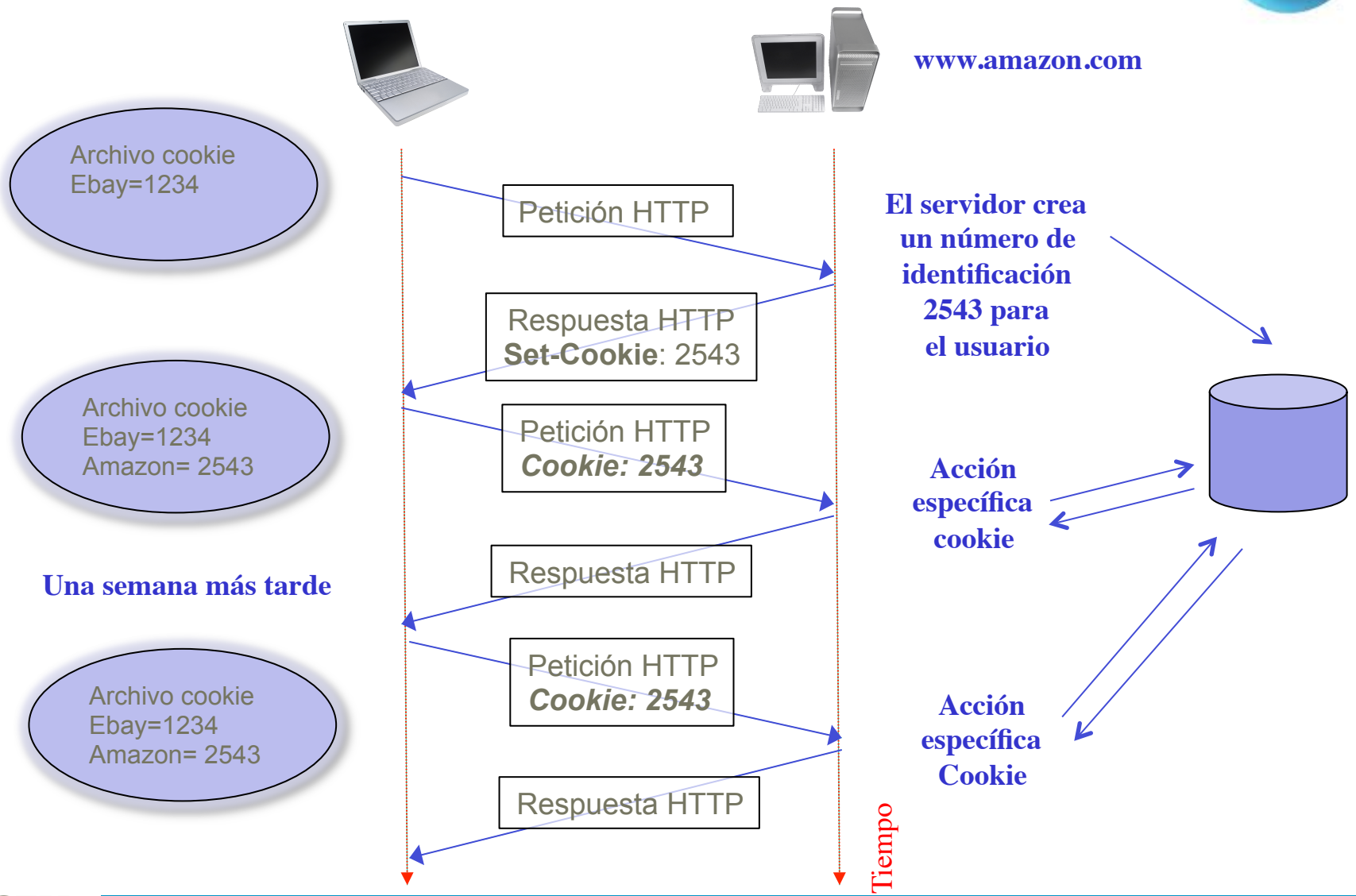
6.3 HTTPS

- HTTP sobre SSL (Secure Socket Layer).
- La conexión TCP está cifrada, de forma que una tercera parte no puede conocer su contenido.
- Permite enviar datos “sensible” a un servidor web, y recibirlos de él.
- Necesita de otros mecanismos (certificados, etc.) para ofrecer un nivel de seguridad razonable.
- Las URLs son https://.

6.3 Cookies

- Sirven para asociar estado a un conjunto de operaciones (peticiones/respuesta).
- Normalmente son datos asociados a un usuario (carro de la compra, cuenta de usuario, etc.)
- Las cookies son generadas por los servidores, y presentadas por los clientes en ciertas ocasiones.
- Especificación original de Netscape, luego propuesta como RFC 2109.

6.3 Cookies mantenimiento del estado



6.3 Cabecera “Set-Cookie”

- Cabecera puesta por un servidor cuando quiere enviar una galletita. Formato:
  - “Set-Cookie:”
  - Nombre de la cookie y valor (“nombre=valor”).
  - Fecha de caducidad (“expires=fecha”).
  - Dominio, camino (“domain=dominio path=camino”). Para decidir más tarde si se envía una cookie o no.
  - “secure”: si está marcada así, sólo se transmitirá sobre canales seguros (HTTPS).
- Ejemplo:
  - Set-Cookie: nombre=eci; expires=Mon, 30-May-2012 12:35:23 GMT; path=/dir; domain=sample.com; secure

6.3 Cabecera Cookie

- Cuando un cliente pide una URL, buscará en su lista de cookies si hay alguna que tenga que enviar (según domain y path).
- Enviará todas las cookies en una única cabecera (“Cookie”).
- Dentro de esta cabecera, las cookies se ordenarán de más a menos específicas (según su “path”).
- No se consideran las cookies con caducidad en el pasado (de hecho, se eliminan).

- Ejemplo:
  - ❑ Cookie: unnombre=unvalor; otronombre=otrovalor

6.3 Ejemplo Cookie

- **Cliente solicita documento y obtiene respuesta:**
  - Set-Cookie: CUSTOMER=WILE_E_COYOTE; path=/; expires=Wednesday, 09-Nov-12 23:12:40 GMT
- **El cliente solicita URL con path="/" en ese servidor y envía:**
  - Cookie: CUSTOMER=WILE_E_COYOTE
- **El cliente solicita documento y obtiene respuesta::**
  - Set-Cookie: PART_NUMBER=ROCKET_LAUNCHER_0001; path=/
- **El cliente solicita URL con path="/" en ese servidor y envía:**
  - Cookie: CUSTOMER=WILE_E_COYOTE; PART_NUMBER=ROCKET_LAUNCHER_0001
- **El cliente recibe:**
  - Set-Cookie: SHIPPING=FEDEX; path=/foo
- **El cliente solicita URL con path="/" en ese servidor y envía:**
  - Cookie: CUSTOMER=WILE_E_COYOTE; PART_NUMBER=ROCKET_LAUNCHER_0001
- **El cliente solicita URL con path="/foo" en ese servidor y envía:**
  - Cookie: CUSTOMER=WILE_E_COYOTE; PART_NUMBER=ROCKET_LAUNCHER_0001; SHIPPING=FEDEX

6.3 Bibliografía

- Tanenbaum, A. S., Computer Networks, 4ª Ed Pearson 2003, apartado 7.3.
- J.F Kurose y K.W. Ross, Redes de Computadores: un enfoque descendente basado en Internet, 2002.
- “Definition of URL/URI syntax, RFC 2396”. http://www.ietf.org/rfc/rfc2396.txt
- “HTTP Made Really Easy. A Practical Guide to Writing Clients and Servers”, por James Marshall. http://www.jmarshall.com/easy/http/

6.3 Bibliografía

[1] “HTTP 1.0, RFC 1945”. http://www.ietf.org/rfc/rfc1945.txt

[2] “HTTP 1.1, RFC 2068”. http://www.ietf.org/rfc/rfc2068.txt

[3] “HTTP State Management Mechanism, RFC 6265” http://tools.ietf.org/html/rfc6265