La capa de aplicación
Protocolo HTTP - La WWW

Programación y administración de redes - Semana 4

Grado en *Ingeniería Informática*

Departamento de Informática. Universidad de Jaén

Objetivos

General

Conocer el funcionamiento de la web en general y del protocolo HTTP

Específicos

- Saber **qué es la WWW** (la Web) y cuál es su funcionamiento básico
- Identificar de manera única recursos web mediante **URI/URL**
- Conocer los **mensajes y respuestas** del protocolo HTTP
- Diferenciar entre los distintos **tipos de conexiones** HTTP
- Aprender a transferir diferentes **tipos de datos** con MIME
- Comprender el funcionamiento de **las *cookies*** y cómo funciona el almacenamiento temporal de datos

La WWW
Características esenciales y conceptos relacionados

La WWW - Introducción

Definición

La World Wide Web (WWW) es una infraestructura o marco de trabajo que permite acceder a documentos enlazados y distribuidos por Internet

Características

- Interfaz gráfica en los **clientes** (también de texto: Lynx)
- Facilidad de uso
- Acceso a gran cantidad de información
- Facilidad para crear y publicar **contenidos** web
- Ofrece una interfaz relativamente sencilla para la creación de verdaderas **aplicaciones** web
- Al usuario le basta con su **navegador** (cliente web) para acceder fácilmente a estas aplicaciones

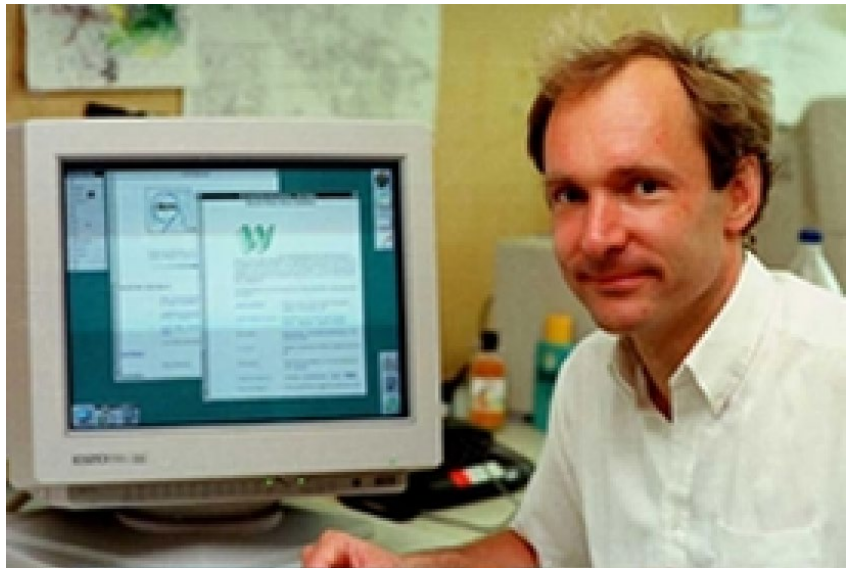
La WWW - Historia

Origen

La WWW, así como el lenguaje HTML para la creación de documentos y el protocolo de transferencia HTTP, fueron creados por Sir Tim Berners-Lee

Características

- **La WWW nace** a principios de los años 90, en el CERN (*Centro Europeo de Investigación Nuclear*) situado en Ginebra (Suiza)
- **Objetivo**: facilitar la colaboración de distintos grupos de científicos
- Su principal apogeo llega cuando en 1993 se desarrolla Mosaic el **primer navegador gráfico**
- En 1994 el CERN y el MIT llegan a un acuerdo para formar el **W3C** (*World Wide Web Consortium*) , dedicado a establecer estándares en la Web. http://www.w3.org
- Actualmente la WWW (o Web) sirve no solo para compartir documentos sino también como plataforma de desarrollo de **aplicaciones web (prácticas 1 a 3)**



La WWW - Funcionamiento

Arquitectura

Como la mayor parte de servicios en Internet, la WWW también emplea el paradigma cliente/servidor como arquitectura básica de funcionamiento

Detalles

- La información se transfiere usando el **protocolo HTTP**, por lo que a los clientes y servidores web también se les conoce como clientes HTTP y servidores HTTP
- **Clientes HTTP:**
  - Conectan con el servidor HTTP mediante el puerto **TCP 80 o 443** (HTTPS)
  - También conocidos como **navegadores**, visores web o *browsers*
  - Indican al servidor HTTP el documento que les interesa mediante un **URL**
  - Interpretan el **código HTML** componiendo y mostrando el documento solicitado
- **Servidores HTTP:**
  - Atienden las peticiones de los clientes **respondiendo** a cada solicitud
  - Los servidores de uso más habitual actualmente son **Apache**, **Nginx** y Microsoft **IIS**

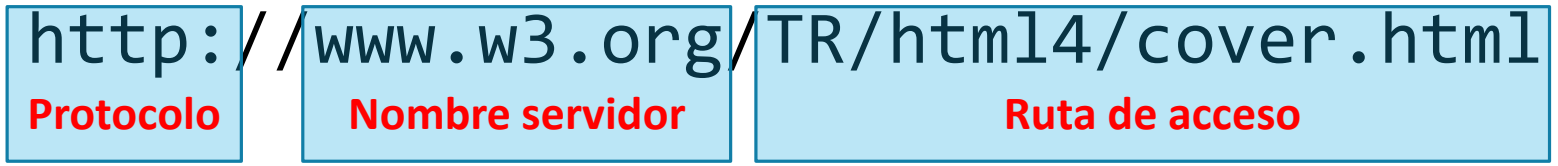
La WWW - Direccionamiento de recursos

Definición

*Todo recurso alojado en la WWW es accesible a través de un **URL (Universal Resource Locator)** que indica su ubicación exacta*

Detalles

– Un URL se compone de tres partes:



- **Protocolo** de acceso al recurso, P.e: `http`, `ftp`, `mailto`, etc.
- **Dirección o nombre** del servidor donde está el recurso, P.e: `www.w3.org`, `ftp.ujaen.es`
- **Ruta de acceso** al recurso relativa al *host* previo, P.e. `TR/html4/cover.html`

El protocolo se separa del nombre/dirección del servidor con **`://`**, mientras que este se separa de la ruta con el carácter **`/`**

La WWW - Hipertextos

Definición

*Se llama hipertextos a los **documentos** (páginas) alojados en un servidor web y que se transfieren al cliente para su visualización*

Características

- Se denominan hipertextos porque constan de un documento base, formado por texto, y un conjunto de **referencias a otros objetos** almacenados en archivos independientes:
  - Imágenes
  - Sonidos
  - Vídeos
  - Hipervínculos a otros documentos
- **Composición:**
  - El cliente web es el encargado de **componer y mostrar** al usuario el documento formado por todos esos elementos

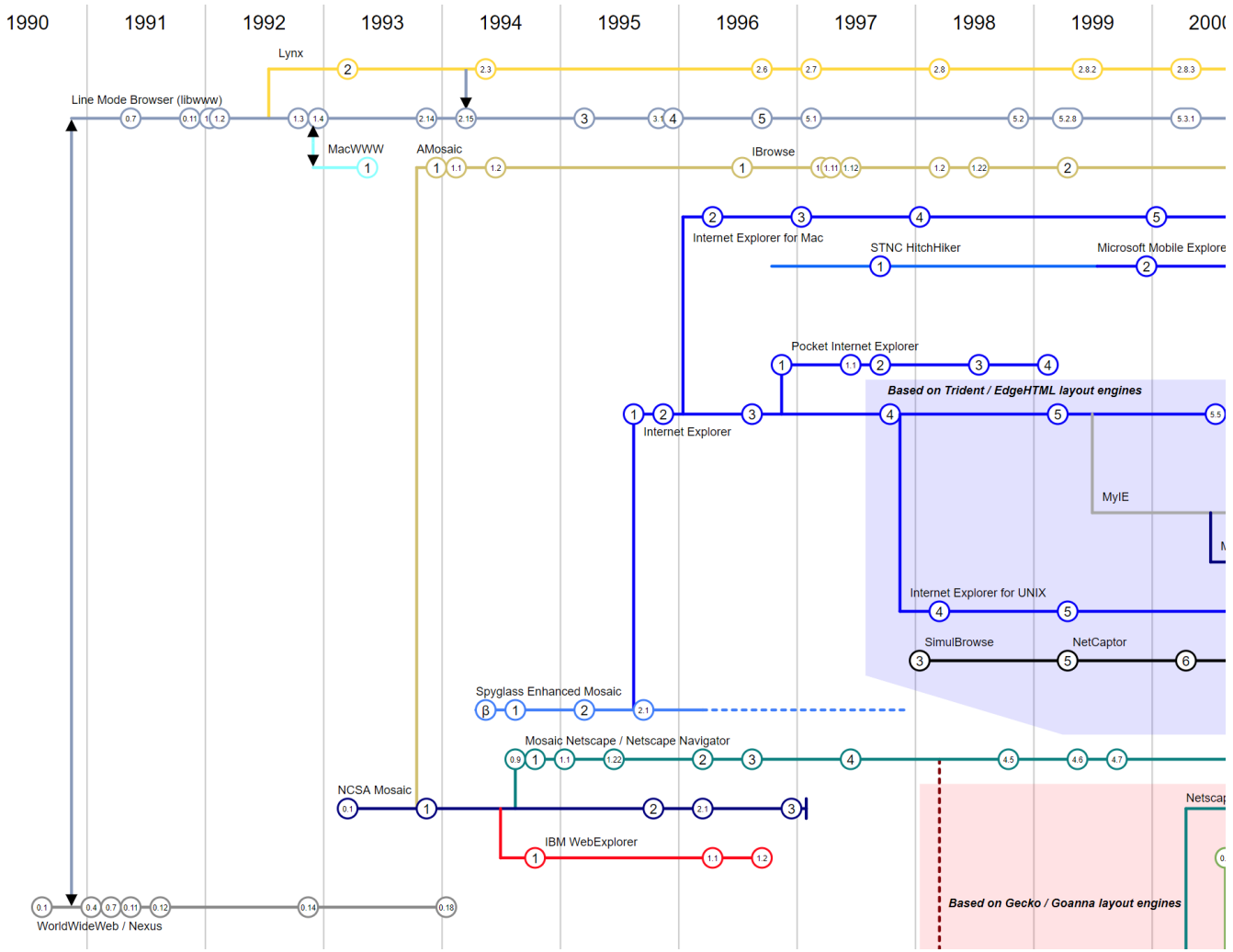
Actividad

Navegadores web

Enumera los clientes HTTP que conoces y sus características principales:

- Sistema operativo
- Versión de HTTP que usan
- Funciones propias que les diferencian de otros navegadores

Puedes buscar información sobre navegadores históricos, como Mosaic, Lynx, WebExplorer, Netscape, etc.



El protocolo HTTP
Introducción

El protocolo HTTP - Introducción

Definición

HTTP (HyperText Transfer Protocol) es un protocolo de la capa de aplicación, al igual que FTP, SMTP, POP3 e IMAP, pero no mantiene información de estado, es un protocolo de tipo **stateless**

Características

- Creado en el CERN (*European Organization for Nuclear Research*) por Tim Berners-Lee.
  Versión actual **HTTP/2** → RFC7540 (https://tools.ietf.org/html/rfc7540)
  Actualmente en borrador **HTTP/3** (https://tools.ietf.org/html/draft-ietf-quic-http-34)
- Es un protocolo para transferir información (texto, audio, imágenes, etc.) y diseñado especialmente para **transferir hipertextos**
- Protocolo de la **capa de aplicación** sin estado (cada solicitud es independiente)
- Sencillez y rapidez para sistemas de información **distribuidos**, colaborativos e hipermedia
- Especificación y negociación de la **representación** de los datos transferidos

El protocolo HTTP - Funcionamiento básico

Arquitectura

HTTP recurre al mismo paradigma cliente/servidor de otros protocolos de la capa de aplicación

Detalles

- El cliente establece una **conexión TCP** con el puerto 80 del servidor
- Establecida la conexión, el cliente envía al servidor **mensajes con peticiones** de recursos
- El servidor responde a las peticiones mediante los correspondientes **mensajes HTTP**
- Entre cliente y servidor pueden existir sistemas **intermediarios**:
  - Servidores proxy
  - Cortafuegos
- Actualmente casi todos los clientes y servidores usan HTTPS, la **versión segura de HTTP**, empleando el puerto **TCP 443** en lugar del 80

El protocolo HTTP - Versiones y características

HTTP/1.0.

- Conexiones **no persistentes**
- **No se puede especificar el dominio**. Si embargo hoy en día, un servidor puede albergar varios dominios
- Las respuestas deben ser de una **longitud conocida**. Sin embargo si se transmite audio o vídeo este dato no se tiene
- Se deben descargar los **archivos completos**. No se permiten rangos

HTTP/1.1.

- Conexiones **persistentes**
- Campo obligatorio host que **especifica el dominio**
- Recepción de flujos de **longitud variable** y no conocida
- Permite la **descarga de un rango** de un archivo (reiniciar transferencia interrumpida)

HTTP/2.

- **Compresión** de cabeceras
- Múltiples comunicaciones concurrentes sobre una sola conexión
- **No sustituye** a HTTP/1.1, es una alternativa más eficiente

Actividad – Acceso a servidores HTTP con `curl`

1. Abre la terminal de GNU/Linux y consulta la documentación del comando `curl` mediante la orden **`man curl`** para comprobar la gran lista de opciones que contempla

```
fcharte@Volstag:~$ curl -I https://www.ujaen.es
HTTP/1.1 200 OK
Date: Wed, 26 Jan 2022 10:17:18 GMT
Server: Apache/2.4.6 (Red Hat Enterprise Linux) OpenSSL/1.0.2k-fips PHP/7.3.11
X-Powered-By: PHP/7.3.11
Cache-Control: max-age=60, public
X-Drupal-Dynamic-Cache: MISS
Link: <https://www.ujaen.es/>; rel="canonical", <https://www.ujaen.es/>; rel="shortlink", <https://www.ujaen.es/en/ho
me>; rel="alternate"; hreflang="en", <https://www.ujaen.es/inicio>; rel="alternate"; hreflang="es", <https://www.ujae
n.es/inicio>; rel="revision"
X-UA-Compatible: IE=edge
Content-language: es
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Expires: Sun, 19 Nov 1978 05:00:00 GMT
Vary: Cookie
X-Generator: Drupal 8 (https://www.drupal.org)
X-Drupal-Cache: HIT
Last-Modified: Wed, 26 Jan 2022 10:17:16 GMT
ETag: "1643192236"
Content-Type: text/html; charset=UTF-8

fcharte@Volstag:~$
fcharte@Volstag:~$ curl google.com
<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">
<TITLE>301 Moved</TITLE></HEAD><BODY>
<H1>301 Moved</H1>
The document has moved
<A HREF="http://www.google.com/">here</A>.
</BODY></HTML>
fcharte@Volstag:~$
fcharte@Volstag:~$
```

2. Mediante la opción `-I` se envía el comando `HEAD` de HTTP a un servidor. Prueba **`curl -I https://www.ujaen.es`** y examina la información que te devuelve sobre la configuración del servidor

3. Cuando se facilita a `curl` solo un URL, usa el comando `GET` de HTTP para obtener el recurso indicado. Prueba a ejecutar la orden **`curl google.com`** y verás la respuesta

4. ¿Qué opción de `curl` habría que usar para enviar los datos de un formulario con el método POST de HTTP? Úsala para simular el envío de un nuevo ticket al *backend* de la práctica de esta semana

El protocolo HTTP
Tipos de conexiones

El protocolo HTTP - Tipos de conexiones

Arquitectura

Dependiendo de la versión HTTP y la solicitud del cliente tenemos diferentes tipos de conexiones posibles

Detalles

Conexiones no persistentes -- Utilizadas por **HTTP/1.0**

- Para cada objeto/archivo que se envía (de una página Web) es necesario establecer **una conexión TCP independiente** que se liberará cuando termine de enviarse el objeto

Conexiones persistentes -- Utilizadas a partir de **HTTP/1.1**

- Se utiliza **una única conexión** para enviar todos los objetos de una página Web. Más eficientes. Tipos:
  - Conexiones persistentes **sin pipeline**: el cliente solo emite una nueva petición una vez que ha recibido la respuesta de la anterior
  - Conexiones persistentes **con pipeline**: por defecto en HTTP/1.1. El cliente hace su petición tan pronto como encuentra un objeto referenciado, habiendo por tanto multiples peticiones paralelas

HTTP - Ejemplo de conexión no persistente

Tiempo

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

1b. El servidor HTTP en el host `www.escuela.edu` espera la conexión TCP del puerto 80 y **acepta** la conexión, notificando al cliente.

2. El cliente HTTP envía un *mensaje HTTP de petición* (que contiene el URL) a través del socket de la conexión TCP donde se indica que se quiere el objeto "`dep/home.index`"

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

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

HTTP - Cálculo del tipo de respuesta a una petición

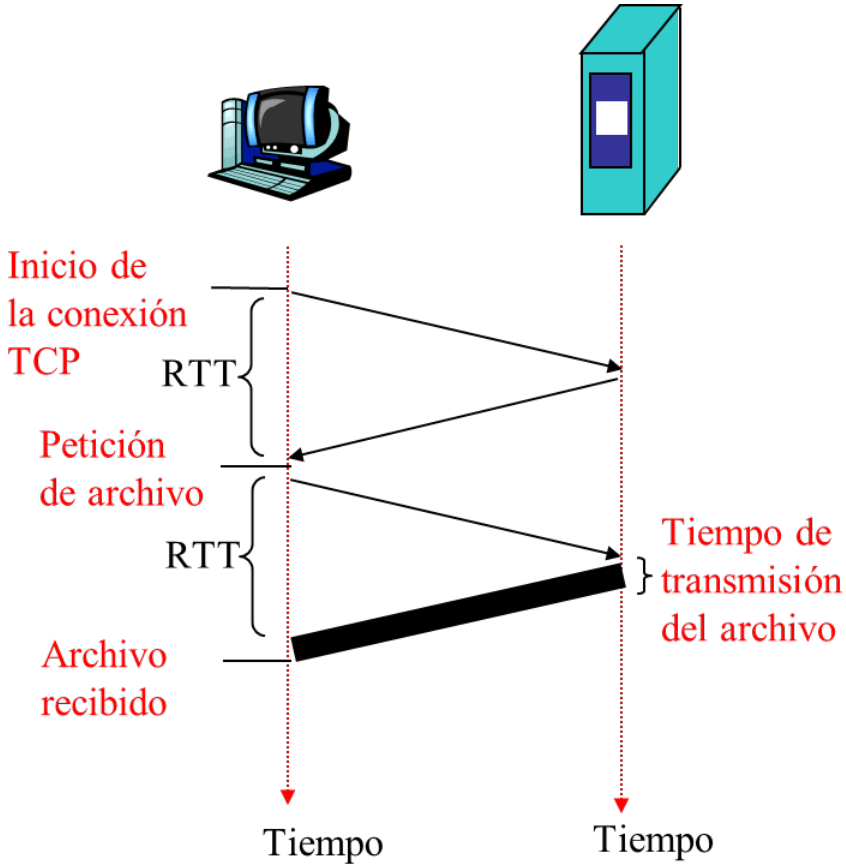
Concepto de RTT

Usamos como unidad de medida el RTT (Round Trip Time), tiempo necesario para enviar una solicitud desde el cliente al servidor y obtener la respuesta

Caso práctico

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

```
Total = 2RTT + tiempo de transmisión
```



HTTP - Eficiencia de los tipos de conexiones

Objetivo

Determinar qué tipo de conexión HTTP resulta más eficiente

Escenarios posibles

Conexiones **no persistentes**

- Requieren **dos RTT por objeto** más lo que tarde en descargarse el objeto
- Además el sistema operativo debe asignar los recursos del host para cada conexión TCP
- Sin embargo, los navegadores suelen abrir **conexiones TCP paralelas** para traer los objetos referenciados

Conexiones **persistentes**

- Conexiones **sin pipeline**:
  - Un RTT de conexión + un RTT y transmisión de página base + un RTT por cada objeto referenciado más lo que tarde este en descargarse
- Conexiones **con pipeline**:
  - Un RTT de conexión + un RTT y transmisión de página base + tan solo un RTT para todos los objetos referenciados más descarga de estos

Actividad - Ejercicio para tutoría colectiva

Qué configuración es más eficiente

Analiza qué configuración HTTP sería la más eficiente contabilizando el número de RTT que se emplearían en cada caso

Caso de estudio

- **El cliente solicita al servidor** una página web que tiene enlazadas tres imágenes. Considera las siguientes configuraciones:
  - Conexión no persistente
  - Conexiones no persistentes en paralelo
  - Conexiones persistentes sin *pipeline*
  - Conexiones persistentes con *pipeline*

Para cada caso dibuja el diagrama de comunicación y calcula el número de RTT empleados

El protocolo HTTP
Formato de los mensajes

El protocolo HTTP - Mensajes

Definición

HTTP es un protocolo del nivel de aplicación y, como otros que ya conocemos, está basado en la arquitectura cliente/servidor

Detalles

Esto implica que existen **dos categorías diferentes** de mensajes:

- **Petición**: mensajes enviados desde el **cliente al servidor**
- **Respuesta**: mensajes que van del **servidor al cliente**

Por defecto se usa la **codificación ASCII** en la transferencia de mensajes HTTP, aunque este aspecto actualmente es configurable

Mensaje de petición - Formato

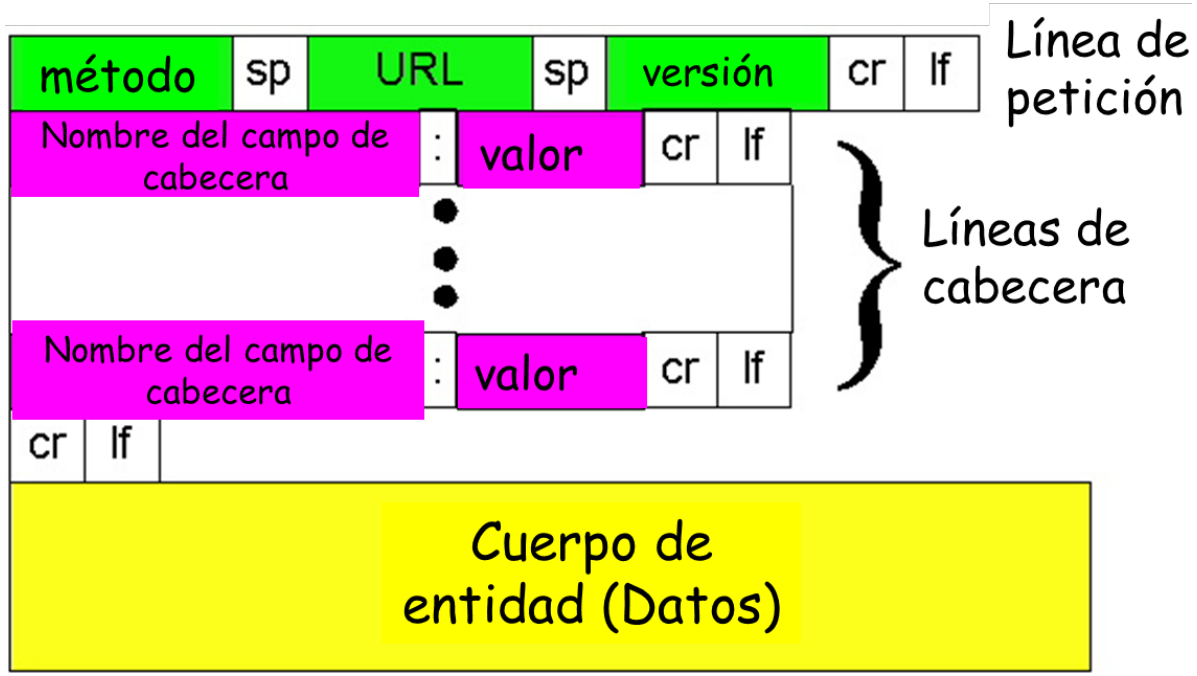
Línea de petición (commandos GET, POST, HEAD)

```
GET /dir/pagina.html HTTP/1.1
Host: www.escuela.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language:fr
```

Líneas de cabecera

Línea en blanco (CR+LF) para indicar el final del mensaje

- **sp**: espacio en blanco
- **cr**: retorno de carro
- **lf**: nueva línea



Mensaje de petición - Línea de petición

Descripción

La línea de petición del mensaje puede usar diferentes comandos del protocolo HTTP a fin de demandar del servidor distintas operaciones

Comandos

- **GET**. Petición de información identificada en el URL
- **HEAD**. Pide campos de cabecera de respuesta, sin incluir cuerpo.
- **POST**. Envío de datos a un servidor.
- **PUT**. Manda datos a un servidor o los sustituye. La diferencia con POST es que puede enviar datos a un recurso que sea capaz de procesarlos
- **OPTIONS**. Pide los métodos válidos en un servidor
- **TRACE**. Obtiene el recorrido de una solicitud, en el caso de que haya pasado por una jerarquía de proxy cachés
- **DELETE**. Solicita que un recurso sea borrado del servidor

Mensaje de petición - Líneas de cabecera

Descripción

Las líneas de cabecera se componen de líneas del tipo ***`clave: valor`*** *y hay entradas de carácter general, específicas para peticiones y para respuestas*

Claves de propósito general

- **`Cache-Control`**: directivas a cumplir por el mecanismo de almacenamiento en caché

  ```
  Cache-Control: no-cache
  ```

- **`MIME-Version`**: se siguen las instrucciones de la versión MIME indicada

  ```
  Mime-Version: 1.0
  ```

- **`Connection`**: configuración para la conexión TCP, por ejemplo para solicitar conexiones no persistentes:

  ```
  Connection: close
  ```

Mensaje de petición - Líneas de cabecera

Claves de solicitud

- **Host**: dominio al que se hace la solicitud. Por ejemplo:

```
Host: www.ujaen.es
```

- **Accept-language**: idiomas aceptados por el cliente. Por si existen diferentes versiones de la página. Por ejemplo:

```
Accept-language: es
```

- **From**: indica la identidad del usuario y se suele utilizar como un sistema de autentificación muy simple. Por ejemplo:

```
From: antonio@ujaen.es
```

- **If-Modified-Since** (se usa con GET): solicita unos datos solo en el caso de que se hayan modificado. Por ejemplo:

```
If-Modified-Since: Fri, 18 Feb 2021 9:28:44 GMT
```

- **Referer**: El cliente indica al servidor la URL del documento donde encontró el enlace al recurso que está solicitando. Por ejemplo:

```
Referer: http://www.cica.es/enlaces/universidades
```

- **User-Agent**: indica el tipo de navegador utilizado, por ejemplo:

```
User-Agent: Mozilla/4.0
```

Mensaje de petición - Líneas de cabecera

Cabeceras de entidad

Son claves que se usan habitualmente cuando el servidor envía la respuesta a fin de facilitar información sobre el cuerpo del mensaje

Claves

- **`Content-Type`**: indica el tipo MIME de los datos solicitados para que el cliente sepa cómo tratarlos

```
Content-Type: text/html
```

- **`Content-Length`**: número de bytes del objeto enviado

```
Content-Length: 682
```

- **`Date`**: fecha en que el mensaje fue originado.

```
Date: Tue, 15 Nov 2018 15:34:33 GMT
```

- **`Expires`**: fecha de caducidad de los datos transmitidos

```
Expires: Tue, 15 Nov 2019 15:34:33 GMT
```

- **`Last-Modified`**: fecha en la que se modificaron los datos

```
Last-Modified: Mon, 22 Nov 2018
```

Mensaje de respuesta - Formato

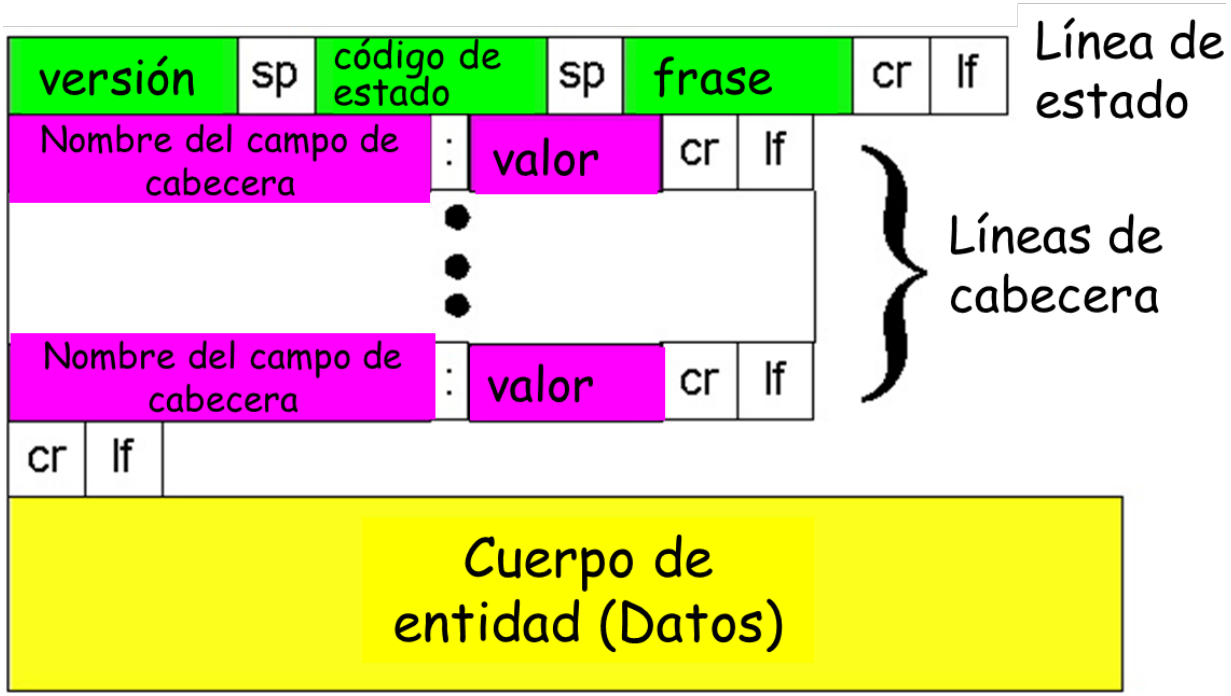
```
HTTP/1.1 200 OK
Connection: close
Date: Thu, 28 Feb 2021 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Feb 2021 …...
Content-Length: 6821
Content-Type: text/html

datos datos datos
```

Línea de estado (version, código de estado y frase de stado)

Líneas de cabecera

Datos, por ejemplo la página web solicitada

- **sp**: espacio en blanco
- **cr**: retorno de carro
- **lf**: nueva línea



Mensaje de respuesta - Línea de estado

Descripción

Indica la versión del protocolo y el código y frase de estado

Códigos de estado

- Sintaxis: **`Versión-HTTP Código_estado Frase_razón`**

  Por ejemplo: `HTTP/1.0 404 Not Found`

Se agrupan según el primer dígito

- 1xx: Información
- 2xx: Éxito
- 3xx: Redirección
- 4xx: Error por parte del cliente
- 5xx: Error por parte del servidor

Ejemplos:

`202` Datos aceptados para procesamiento (POST)
`401` Acceso no autorizado
`403` Acceso prohibido
`405` Método no permitido
`408` Timeout en envío de petición
`501` Método no implementado
`503` Servicio No disponible
`505` Versión de HTTP no soportada

Consultar lista actualizada de códigos en
https://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml

Mensaje de respuesta - Líneas de cabecera

Descripción

Al igual que en los mensajes de petición, pueden usarse claves de propósito general aparte de los específicos para mensajes de respuesta, así como claves de entidad

Claves de respuesta

- **`Server`**: contiene información relativa sobre el software utilizado por el servidor que devuelve la respuesta. Por ejemplo:

```
Server: Apache/1.3.6  (Unix)
```

- **`Location`**: redirecciona al navegador hacia otro documento. Se suele utilizar para atender situaciones erróneas, por ejemplo que no exista un determinado recurso solicitado:

```
Location: http://www.altavista.com
Location: /formularios/email.html
```

Mensaje de respuesta - Tipo MIME

Definición

Se usa MIME (Multipurpose Internet Mail Extensions), como en el correo electrónico, para facilitar la transferencia de datos que no son texto

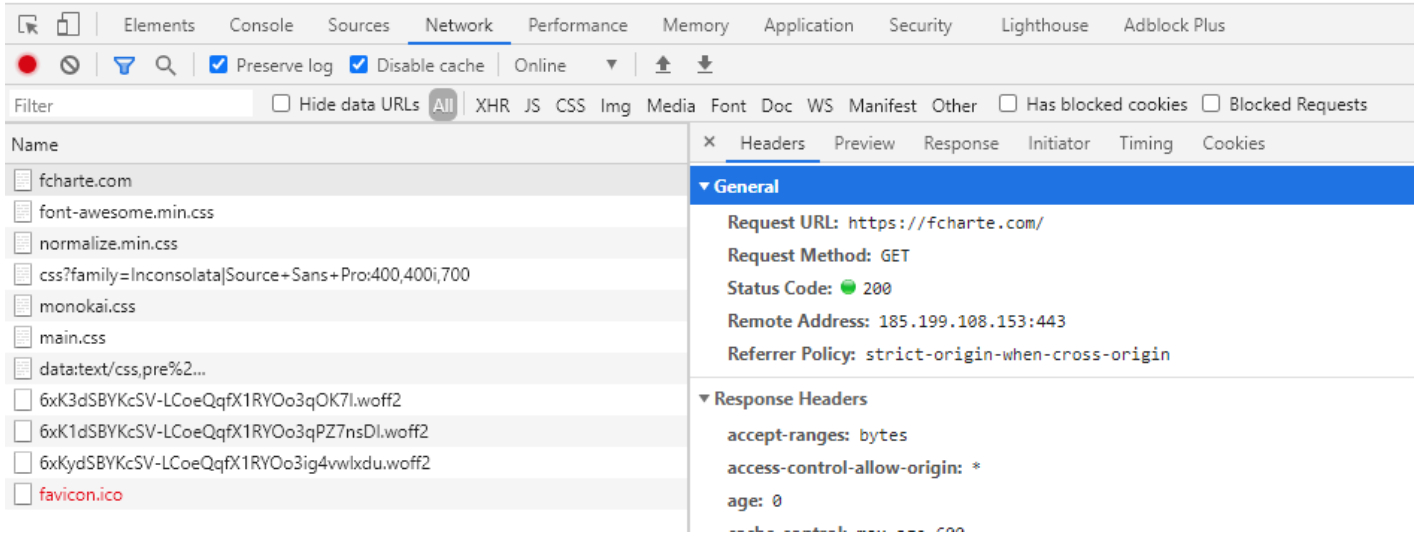
Detalles

- **Sistema de representación** para tipos de datos extensible y abierto
- En un mensaje de respuesta el servidor debe de indicar el **tipo de información** que está enviando para que el cliente sepa representarlo
- Se **asocian extensiones** de archivos a tipos de información
- Ejemplos de tipos MIME y sus extensiones:
  - `application/x-javascript` `js`
  - `audio/mpeg` `mpga mp2 mp3`
  - `image/jpeg` `peg jpg jpe`
  - `text/html` `html htm`

Actividad - Comunicación por HTTP

1. Abre las herramientas de desarrollador de tu navegador, habitualmente se hace con F12, localiza la pestaña **Red/Network** y a continuación accede a alguna página web. Examina las cabeceras de los mensajes de solicitud y de respuesta

```
[root@servidor ~]# telnet fcharte.com 80
Trying 185.199.108.153...
Connected to fcharte.com.
Escape character is '^]'.
GET / HTTP/1.1
Host: fcharte.com
Connection: keep-alive

HTTP/1.1 301 Moved Permanently
Content-Type: text/html
Server: GitHub.com
Location: https://fcharte.com/
X-GitHub-Request-Id: EA02:FAB6:94BD11:9CE936:6023AF5A
Content-Length: 162
Accept-Ranges: bytes
Date: Wed, 10 Feb 2021 10:03:07 GMT
Via: 1.1 varnish
Age: 0
Connection: keep-alive
X-Served-By: cache-mad22064-MAD
X-Cache: MISS
X-Cache-Hits: 0
X-Timer: S1612951387.949939,VS0,VE116
Vary: Accept-Encoding
X-Fastly-Request-ID: c9c8018dea77cefdf7819797f0f89f204809edea

<html>
<head><title>301 Moved Permanently</title></head>
<body>
<center><h1>301 Moved Permanently</h1></center>
<hr><center>nginx</center>
</body>
</html>
```



2. Usa el comando **`telnet`** para conectar con un servidor web y solicita el documento raíz enviando el comando **`GET`** y las cabeceras **`Host`** y **`Connection`**. Examina la respuesta devuelta por el servidor

El protocolo HTTP
Otros aspectos de interés

HTTP - Identificación básica de usuarios

Descripción

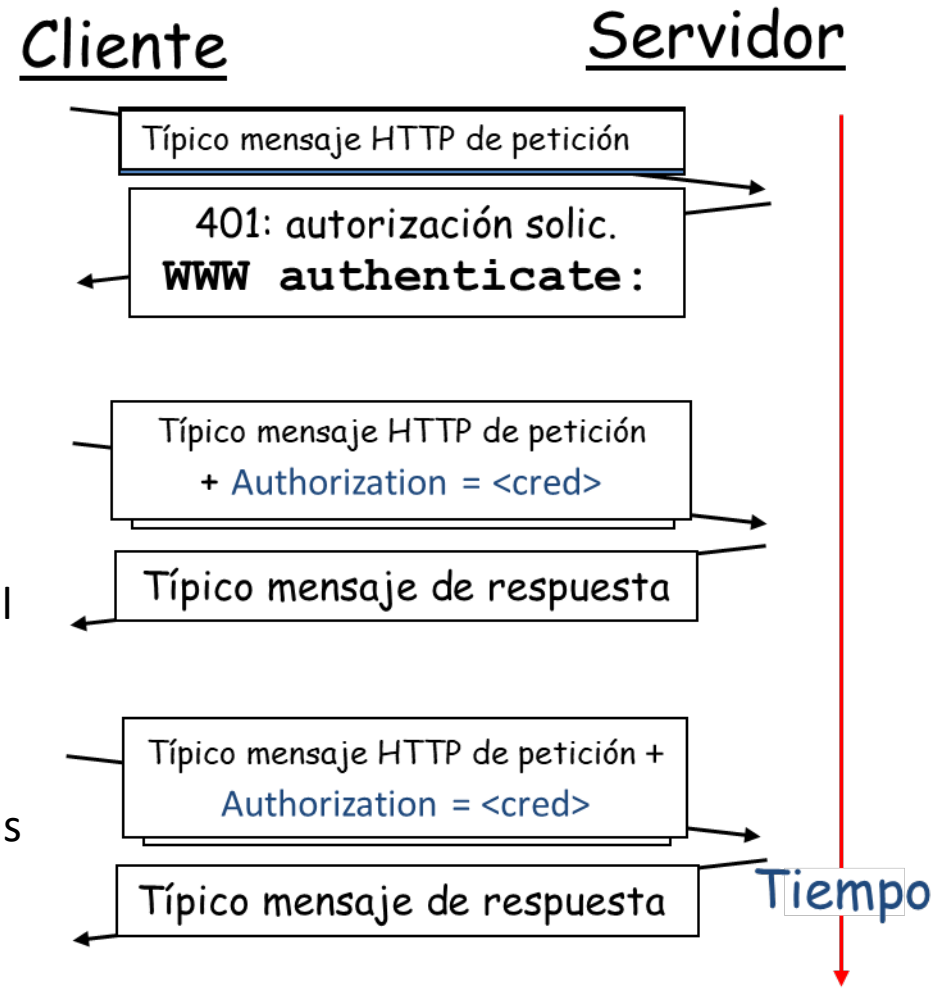
HTTP cuenta con un mecanismo básico para solicitar la identificación de usuarios

Detalles

- Permite la identificación básica de usuarios por parte de un servidor y por tanto el **control de acceso** a este
- **Credenciales** de autorización, normalmente: nombre y contraseña
- Funcionamiento:
  - Ante un típico mensaje de petición del cliente, el servidor responde con el mensaje:

    `401 Unauthorized`

    `WWW-Authenticate:` (en cabecera)
  - Como HTTP no mantienen estado, el cliente debe presentar al servidor las credenciales de autorización para **cada petición**:
    - El usuario solo da al principio el usuario y la contraseña
    - El cliente mantendrá estas credenciales en caché para sucesivas peticiones



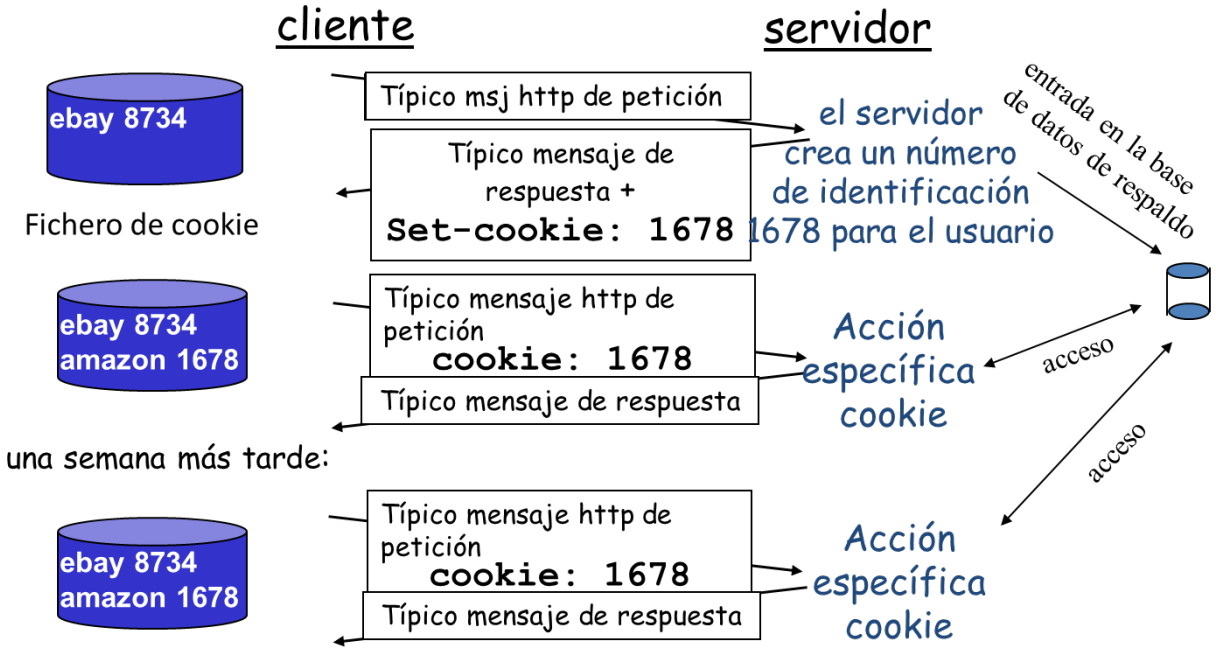
HTTP - Uso de *cookies*

Descripción

Las cookies son datos que un servidor HTTP almacena en el navegador del cliente

Detalles

- Cuando un usuario entra por primera vez en un sitio web el servidor puede crear un nuevo registro en su base de datos al que le asigna un **identificador**
- Este registro puede contener diferente información del usuario:
  - Indirectamente la puede recabar el servidor: dirección IP, datos de la máquina, etc.
  - Directamente puede solicitar esos datos
- El cliente web guarda el identificador en un archivo **en la máquina del usuario** y lo envía siempre que se conecta al sitio web
- A partir de ahí puede ofrecer **información a medida** (ciertas páginas web, etc.) en función de: las página que más visitas, la información que ha podido recolectar del usuario, etc.



HTTP - Uso de cookies

Problemática

Las cookies pueden usarse para vulnerar la privacidad de los usuarios que acceden a un sitio web, razón por la que existe una normativa comunitaria al respecto

Aspectos a tener en cuenta

- Las *cookies* permiten que el servidor obtenga información diversa, más de la que podríamos pensar:
  - Datos personales
  - Gustos, aficiones
  - Dispositivos distintos con los que trabajamos
  - Sistemas que usamos: sistema operativo, navegador, etc.
- Incluso hay empresas que **venden estos datos** a agencias de publicidad

Actividad - Ejercicio para tutoría colectiva

Qué *cookies* crean los sitios web

Identifica las cookies que distintos sitios web crean en tu navegador cuando accedes a ellos

Pasos a seguir

1. Consulta en la documentación del comando `curl` cómo **guardar en un archivo** las *cookies* creadas en la respuesta a una solicitud
2. Ejecuta `curl` con dicha opción para guardar en el archivo `cookies.txt` el resultado de acceder a sitios como `www.Facebook.com`, `www.twitter.com` u otras web que uses habitualmente. **Tras cada acceso examina** el contenido de `cookies.txt`
3. **Crea una lista** con el nombre, expiración y valor de las *cookies* que hayas encontrado para cada sitio con el que pruebes. Intenta deducir cuál es la utilidad que tiene cada una
4. ¿Cómo podrías enviar con `curl` una solicitud, a cualquiera de los sitios que has comprobado, **incluyendo las *cookies*** que interesen?

HTTP - Comando GET condicional

Descripción

Los navegadores almacenan en caché los datos para que el servidor no tenga que enviarlos si ya están disponibles localmente

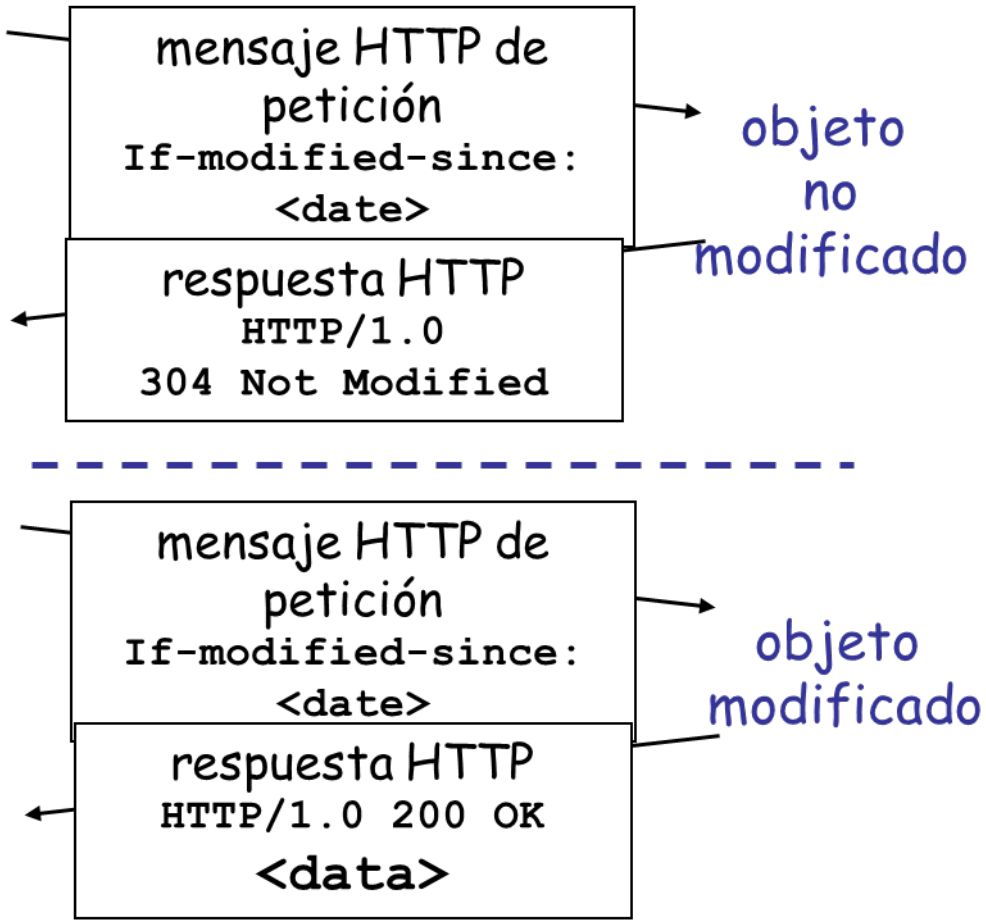
Detalles

- **Objetivo**: no enviar objetos si el cliente tiene una versión actualizada en su caché
- **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`



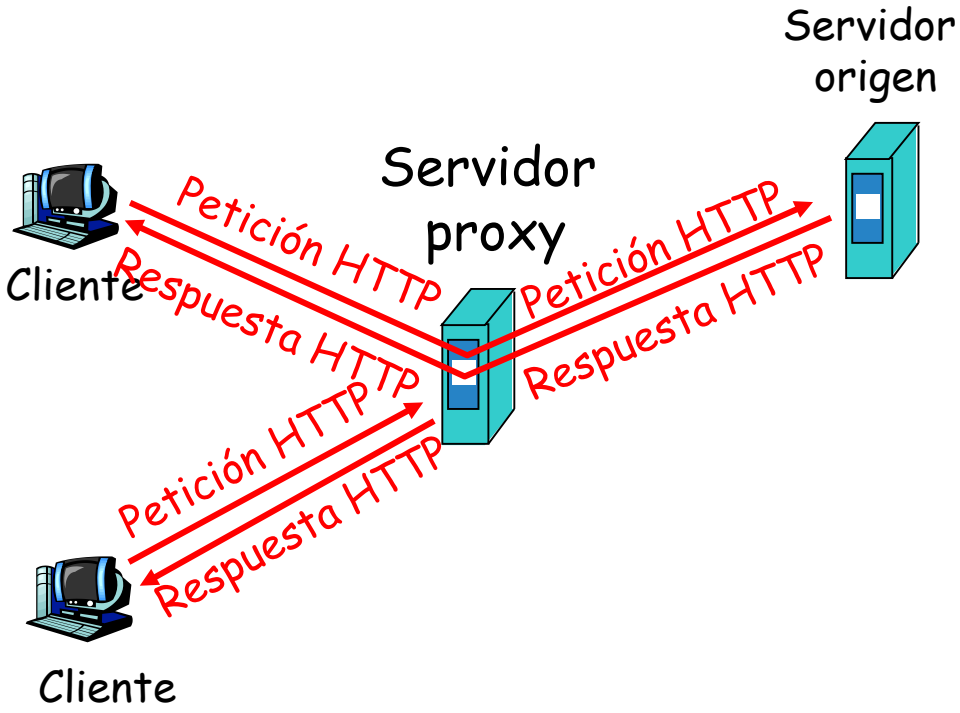
HTTP - Servidores proxy (caché web)

Definición

El servidor proxy es aquél por el que los hosts conectan con la web y tiene en su caché las páginas solicitadas por los clientes

Funcionamiento

- Cuando en un cliente web se especifica un URL, este hace la petición al servidor *proxy*. Pueden darse dos casos:
  - Que el servidor *proxy* tenga en su caché la página, en cuyo caso **se la enviará al cliente** sin solicitarla al servidor
  - Que el servidor *proxy* no la tenga, en cuyo caso él mismo hace la **petición al servidor**, obtiene los datos, los guarda en caché y se los envía al cliente



- **Ventajas** de contar con un servidor proxy:
  - Aumenta la **velocidad** de acceso a la web y también la **seguridad**
  - Reduce el **tráfico** en el enlace de acceso
  - Permite a los provedores de contenidos con bajo ancho de banda distribuir de **manera efectiva** sus contenidos
  - Suelen ser instalados por los ISP o empresas que quieren tener esas ventajas

Cuestiones clave de este tema

Cuestiones clave

Qué deberías saber

Al inicio de este tema se planteaban unos objetivos específicos que deberían permitirte ***responder a las siguientes cuestiones*** *clave*

Cuestiones

- ¿Qué es la WWW y cómo funciona en términos generales?
- ¿Cómo se direccionan los recursos almacenados en los servidores web?
- ¿Cuál es el formato y cabeceras de los mensajes del protocolo HTTP?
- ¿Qué tipos de conexiones HTTP hay cuáles son sus diferencias?
- ¿Cómo pueden transferirse contenidos no textuales mediante HTTP?
- ¿Cuál es la finalidad de las *cookies* y su funcionamiento?
- ¿Cómo funcionan las peticiones condicionales y la autenticación básica?

Material adicional

Descripción

Para ampliar tus conocimientos sobre los contenidos de esta semana te recomendamos que consultes los recursos indicados a continuación.

Recursos

- **Capítulo 2** La capa de aplicación, del libro Redes de computadoras 7ED disponible en formato digital en la BUJA (recuerda identificarte para poder acceder a leerlo desde tu navegador), concretamente la **sección 2.2 La Web y HTTP**
- **World Wide Web (WWW, "The Web") and the Hypertext Transfer Protocol (HTTP)** en el recurso electrónico The TCP/IP Guide, donde encontrarás todos los detalles sobre el funcionamiento de la web y del protocolo HTTP
- **HTTP headers** en MDN Web Docs, la documentación de referencia de Mozilla para todo lo relacionado con la web