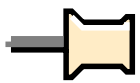
World Wide Web y HTTP

Álvaro González Sotillo

Rey Fernando VI

15 de octubre de 2015

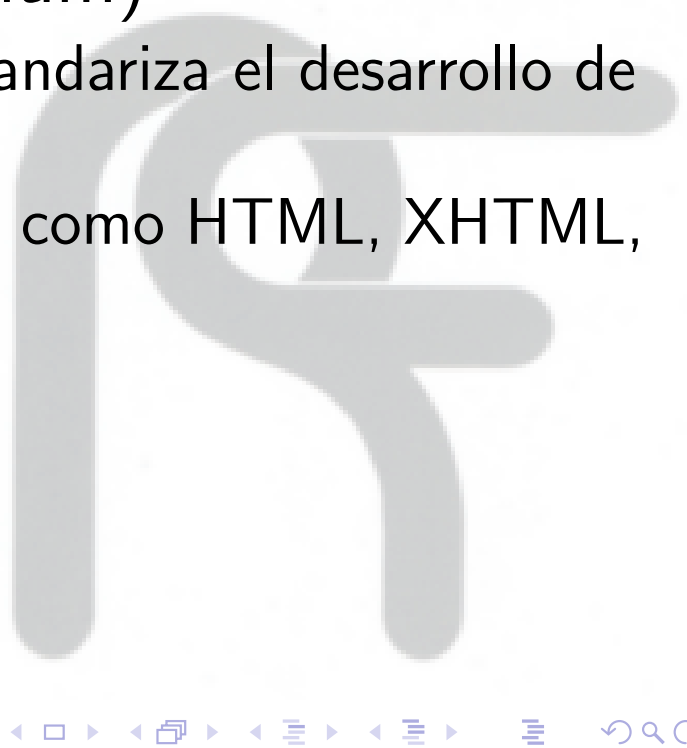


Internet y WWW

- Internet es una red mundial que utiliza IP
- World Wide Web (WWW)
  - Comparte documentos (HTML u otros) con hiperenlaces entre ellos
  - Los enlaces se basan en direcciones (URL y URI)
  - En cualquier servidor HTTP que forme parte de Internet

Estandares Web

- WWW
  - Desarrollada por el CERN en 1989
- W3C (World Wide Web Consortium)
  - Comunidad internacional que estandariza el desarrollo de la WWW
  - Desarrolla y promueve estándares como HTML, XHTML, CSS ...
  - `http://w3.org`
  - `http://w3c.es`

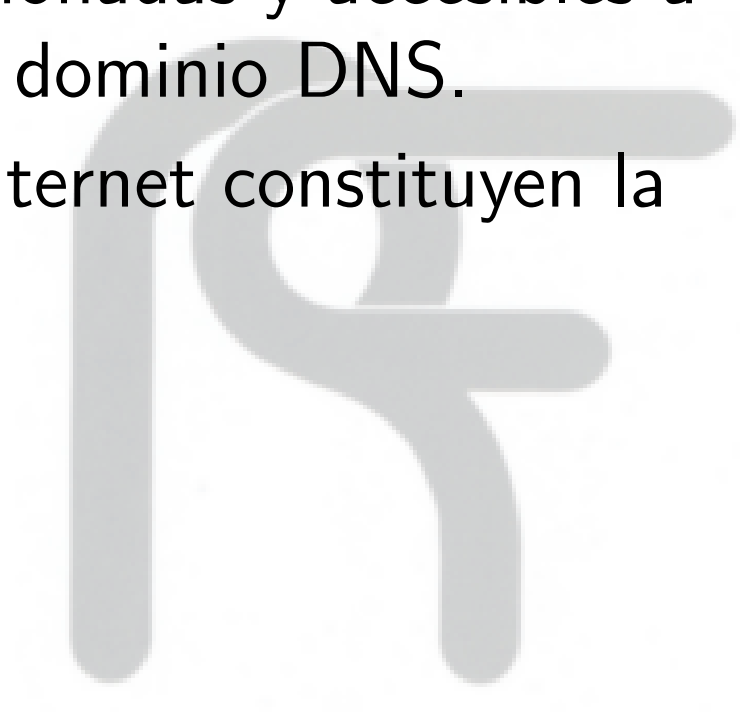


Página web

- Documento hipermedia o conjunto de información electrónica relacionada (texto, audio, imágenes, video, etc.) que normalmente contiene hiperenlaces a otras paginas web o recursos.
- Escrita en lenguajes que son interpretados y/o ejecutados por los navegadores (**XHTML**, **CSS**, **Java Script**, **Flash** . . .)
- Contenidos estático y dinámico
  - Generada dinámicamente por un programa en el servidor
  - Modificada dinámicamente en el cliente

Sitio web

- Conjunto de páginas web relacionadas y accesibles a partir de un mismo nombre de dominio DNS.
- El conjunto de sitios web de Internet constituyen la WWW.



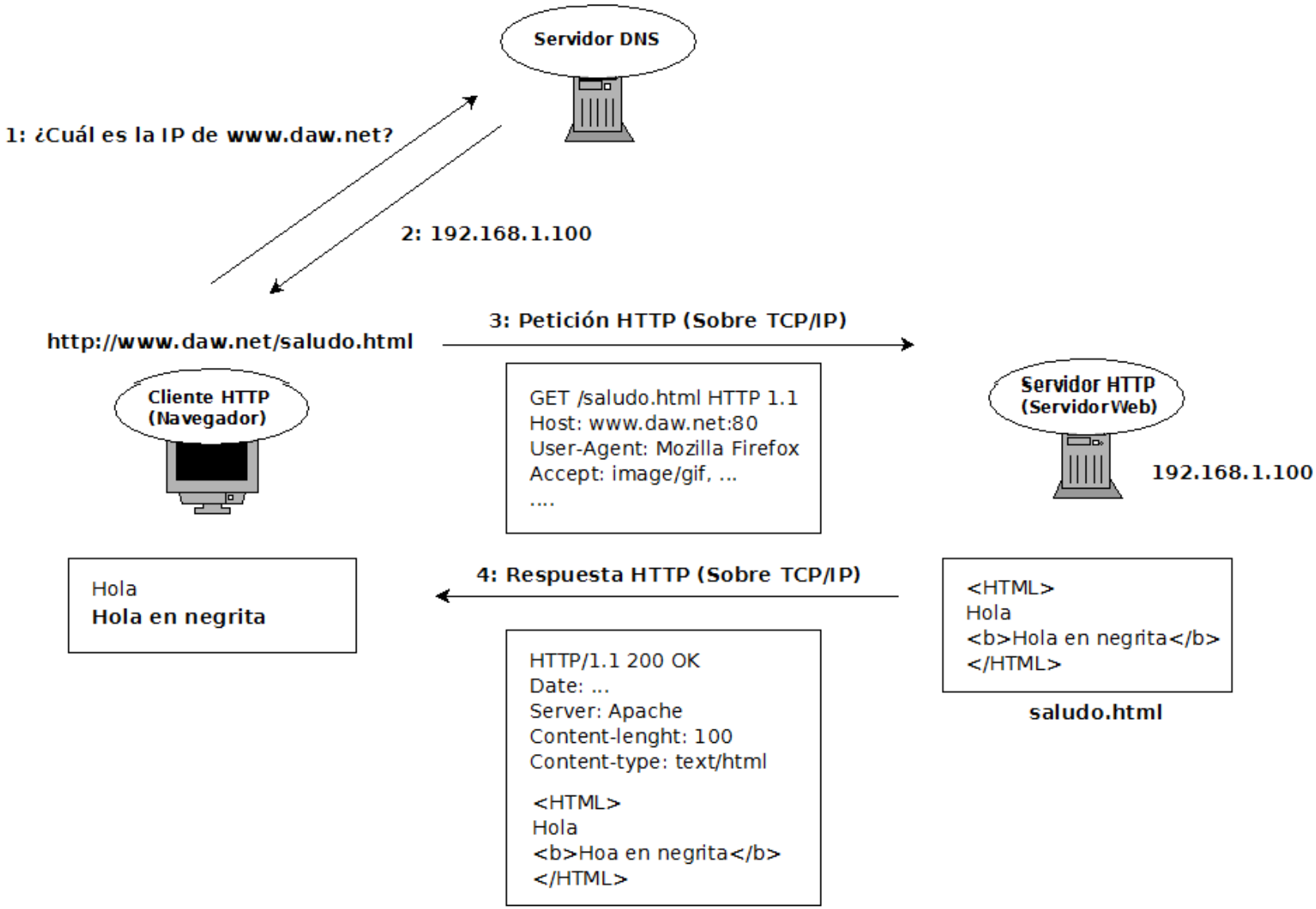
Aplicación web

- Aplicación distribuida cuya interfaz de usuario es accesible desde un navegador web.
- El usuario interactúa con un navegador que accede a los servicios y recursos que ofrece un servidor web (Ejemplo: buscador, una tienda electrónica, un cliente de correo web ...).
- Ejemplos: *Gmail*, *Ebay*, *Facebook* , ...

Componentes y funcionamiento

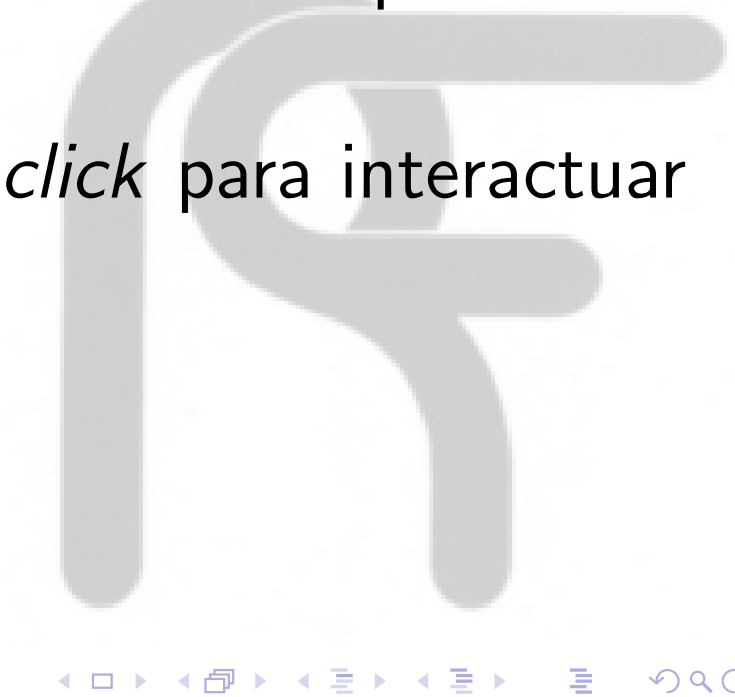
- Recursos (documentos, vídeos, imágenes, etc.) conectados por hiperenlaces.
- Clientes web (clientes HTTP o navegadores).
- Servidores web (o servidores HTTP).
- Nombres y direcciones (URIs y URLs).
- Protocolo HTTP.
- Tecnologías web (*XHTML*, *CSS*, *XML*, *Ajax*, *XPath* ...)

Componentes y funcionamiento



Navegadores

- Programas con los que interactúa el usuario.
- URIs (o URLs) para acceder a recursos disponibles en la red
- Interfaz gráfica tipo *point and click* para interactuar con los enlaces



Navegadores

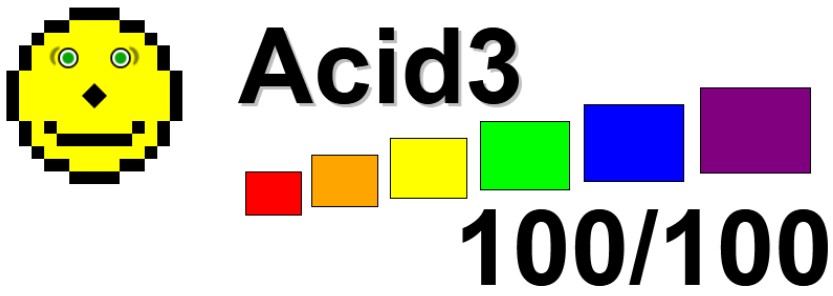
- Clientes de diferentes protocolos, aunque Su función principal es ejercer como clientes HTTP.
- Mantienen una memoria cache
  - Direcciones a las que han accedido (historial)
  - Recursos procesados
  - Contraseñas introducidas por el usuario en las aplicaciones,
- Permiten extender su funcionalidad mediante complementos (*plugins*)

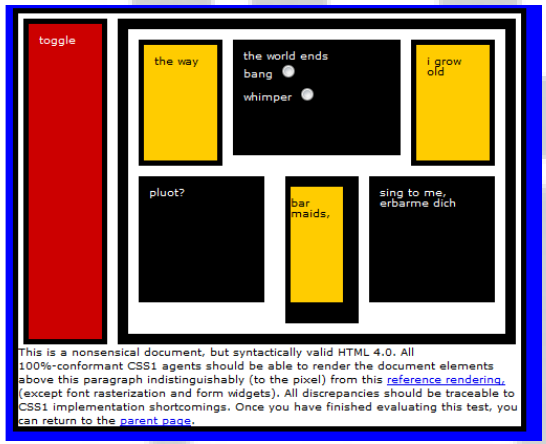
Navegadores

- Internet Explorer
- Mozzila Firefox
- Google Chrome
- Safari
- Opera
- Modo texto: Lynx,

Navegadores

- No todos los navegadores cumplen todos los estándares
- Los test **ACID** evalúan la adecuación de los navegadores a los estándares *W3C*
- `http://www.acidtests.org`





URL (Uniform Resource Locator)

- Usado para identificar un recurso en la Web

```
protocol://[user[:password]@]host[:port]/path-and-name
```

- Ejemplos
  - `http://proxy/maquinas-virtuales`
  - `http://213.0.88.85:8080/`
  - `ftp://root:alumno@debian8`
  - `ssh://root@debian8`

URI (Uniform Resource Identifier)

- Permiten identificar una parte dentro de un recurso.
- Las URLs son tipos de URIs.
- Sintaxis para HTTP
  - **Parámetro**: `nombre=valor`
  - Los parámetros se concatenan con `&`

```
protocol://[user[:password]@]host[:port]/path-and-name[?parameters][#part]
```

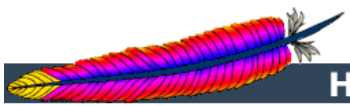
- Ejemplos
  - `http://obelix.dae.es/buscarLibros.php?id=2&tema=Historia`
  - `https://es.wikipedia.org/wiki/URI_scheme#V.C3.A9ase_tambi.C3.A9n`

Servidores Web

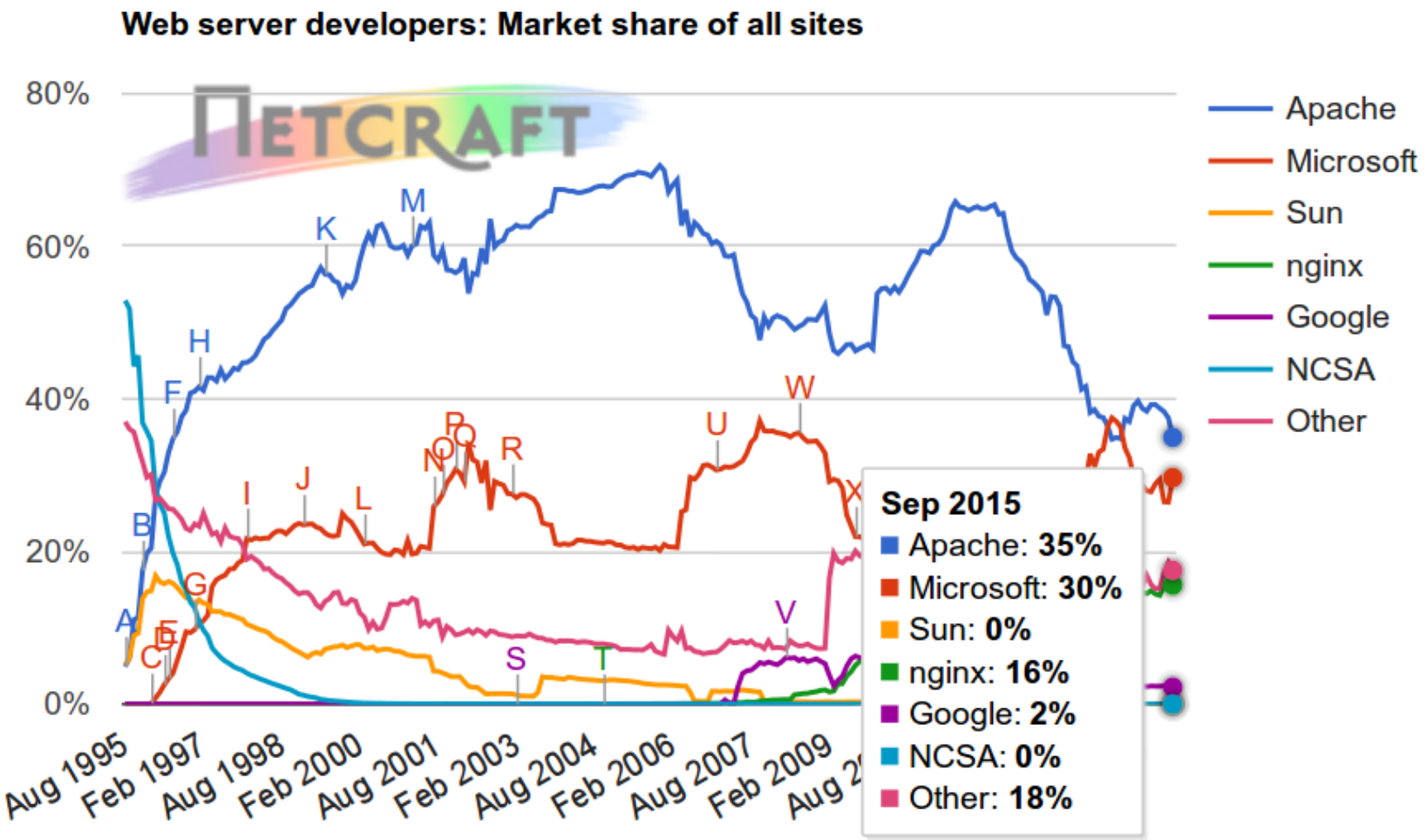
- Atienden peticiones HTTP.
- Procesan e interpretan código escrito en diferentes lenguajes.
- Envían a los clientes los recursos solicitados.
- Arquitectura modular que permite ampliar o quitar funcionalidades fácilmente.
- Peticiones HTTP en el puerto **80/TCP**

Ejemplos de servidores web

- Apache HTTP server.
- IIS de Microsoft
- Nginx
- Lighttpd
- Cherokee



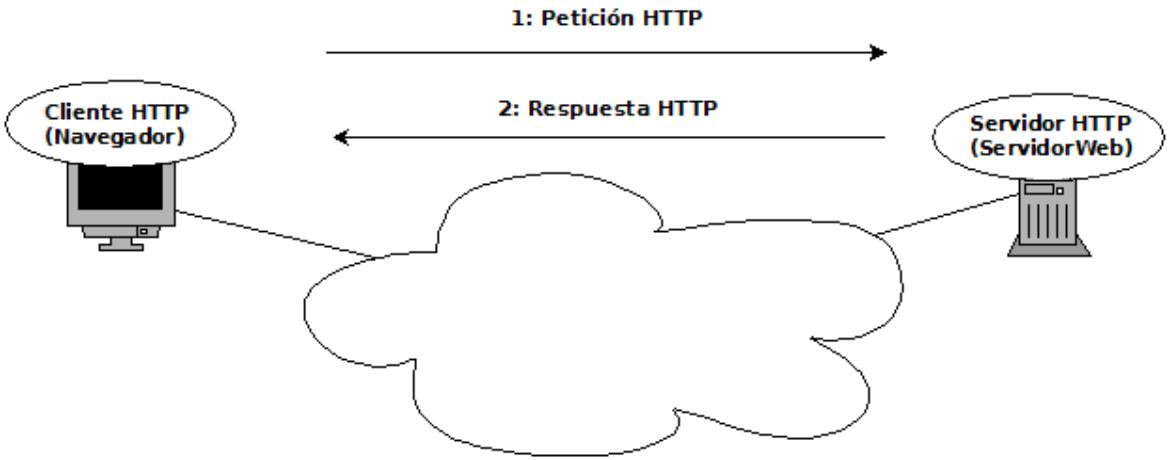
Popularidad de servidores web

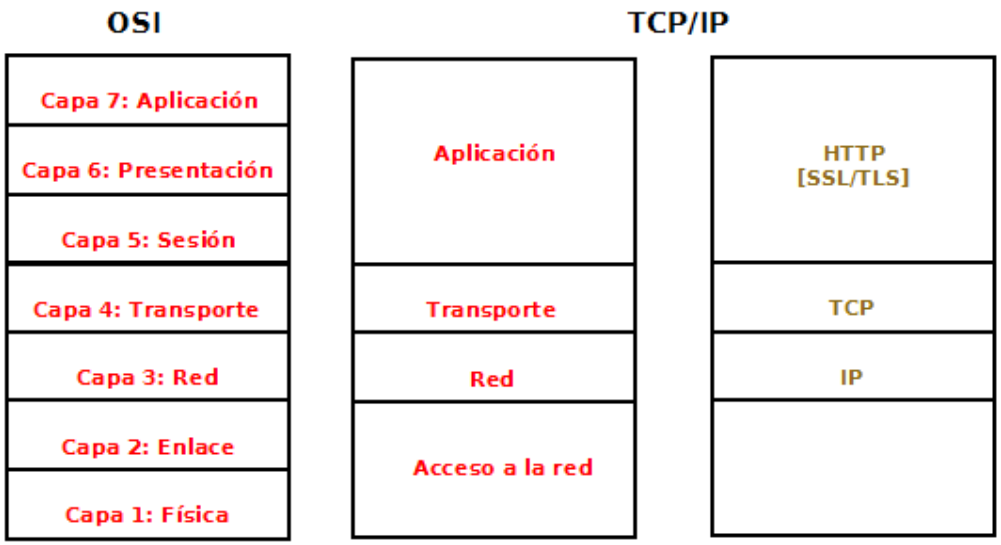


Protocolo HTTP

- *Hyper Text Transfer Protocol*
- Es de facto el protocolo de comunicación en la Web.
- Protocolo sin estado.
- Utiliza TCP como protocolo de transporte.
- `http://www.w3.org/standards/techs/http#w3c_all`

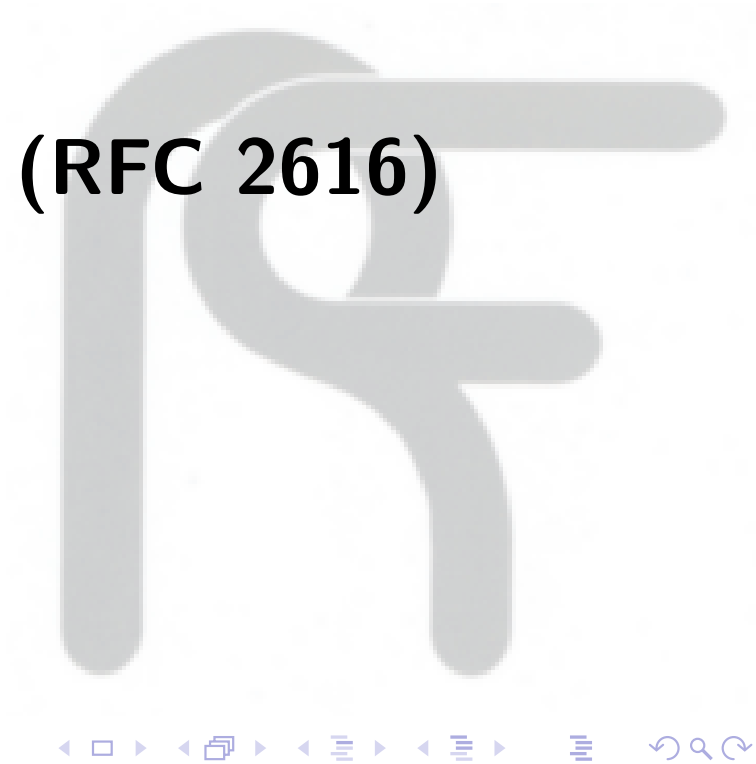
Protocolo HTTP





Versiones HTTP

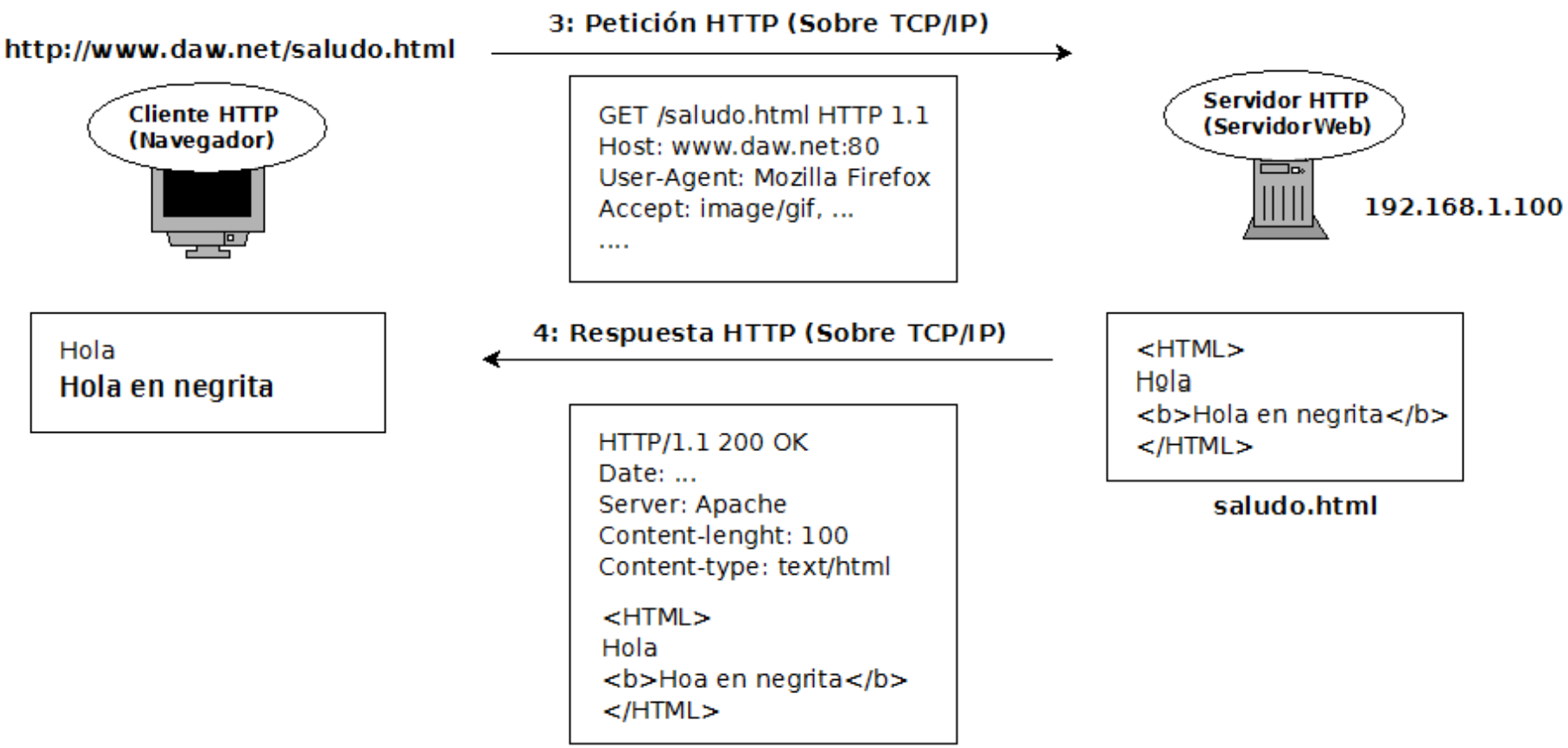
- HTTP/0.9 (Obsoleta).
- HTTP/1.0 (Aún compatible)
- **HTTP/1.1 (versión actual) (RFC 2616)**
- HTTP/1.2 (experimental).



Funcionamiento HTTP

1. El usuario introduce una URI (o URL)
2. URL en la barra de direcciones del navegador o hace clic sobre un hiperenlace.
3. El navegador analiza la URL y establece una conexión TCP con el servidor web.
4. El navegador envía un mensaje HTTP de petición que depende del la URI (o URL).
5. El servidor envía un mensaje de respuesta que depende de la petición enviada y del estado del servidor.
6. Se cierra la conexión TCP (generalmente)

Funcionamiento HTTP



Funcionamiento HTTP

Simular un servidor con **netcat**

Utiliza el comando **netcat** para simular un servidor HTTP corriendo en el puerto 8080. Conecta tu navegador y observa las cabeceras enviadas en la petición.

Simular un cliente con **netcat**

Utiliza el comando **netcat** para simular un cliente HTTP, conectándote a `http://www.marca.es` y consiguiendo su página raíz.

Métodos de petición

- Especifican la operación que quiere realizar el cliente en el servidor.
  - GET
  - POST
  - OPTIONS
  - HEAD
  - PUT
  - DELETE
  - TRACE
  - CONNECT
  - PATH

Método GET

- Se invoca normalmente cuando
  - Se introduce una URL en el navegador. navegador
  - Se pincha sobre un hiperenlace. hiperenlace
  - Se envía un formulario GET.
  - Permite enviar parámetros al servidor en la URI (conocidos como Query String)

`http://datosGET.php?nombre=30&edad=Alicia`

Método GET

- Las peticiones GET no envían cuerpo de mensaje.
- El tamaño de la información enviada estará limitada.
- No se puede usar para subir archivos o realizar otras operaciones que requieran enviar una gran cantidad de datos al servidor.

Método POST

- Para solicitar al servidor que acepte información que se envía adjunta en una petición.
- Las peticiones POST envían el cuerpo de mensaje.
- Los parámetros no son visibles por lo tanto en la URL.
- Se invoca normalmente como consecuencia de enviar un formulario POST.

Método POST

Developer tools de Chrome

Los formularios de *login* suelen utilizar el método POST para que las contraseñas no se vean en la URL.
Inspecciona las cabeceras enviadas y recibidas al introducir una contraseña incorrecta en `https://aulavirtual2.educa.madrid.org/login/index.php`.
Utiliza las *developer tools* de Chrome.

Método POST

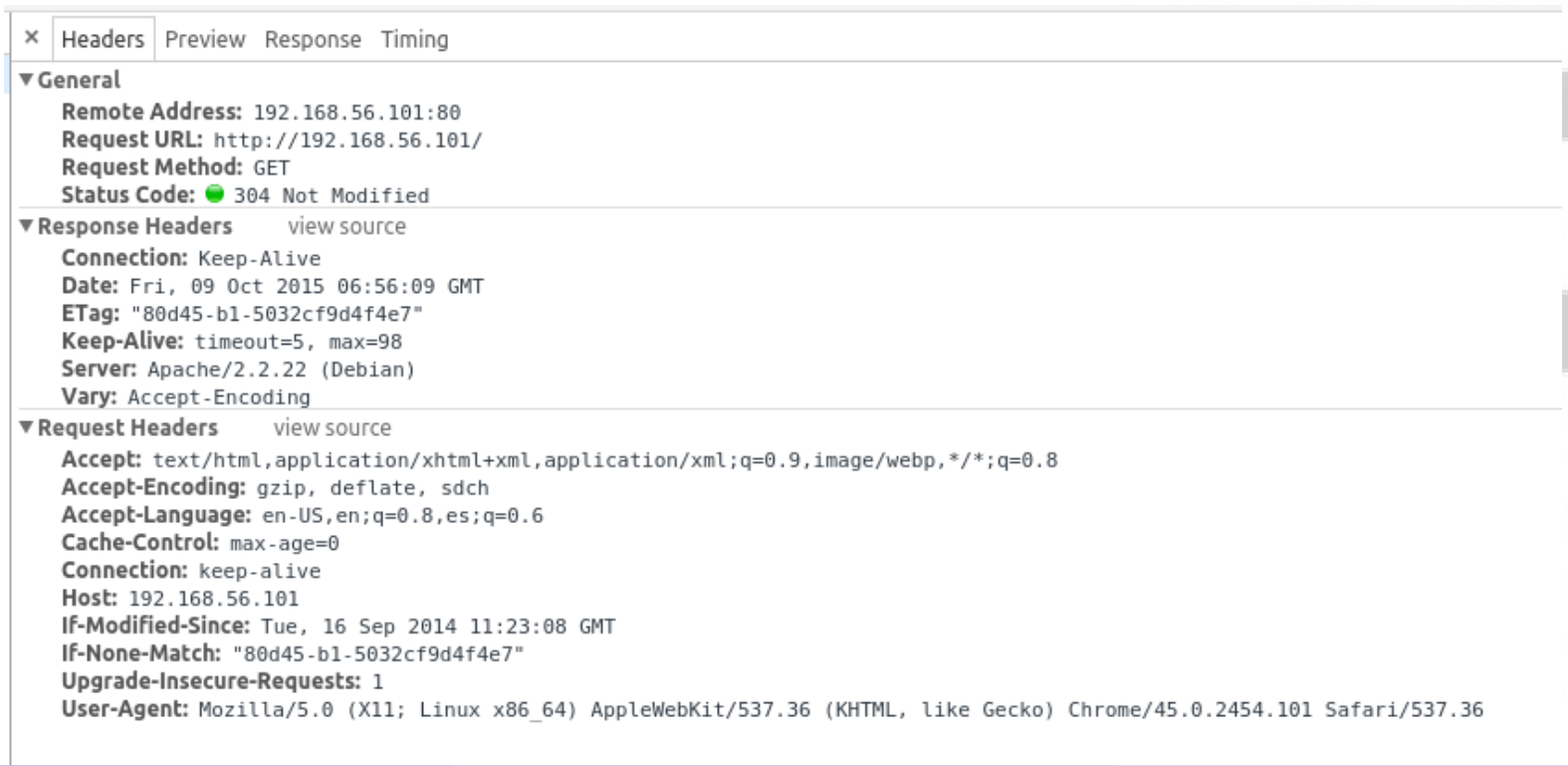
Cliente **curl** y POST

Simula un servidor web con **netcat**. Después utiliza el cliente **curl** para realizar una petición POST, y observa cabeceras y datos enviados

```
curl --data "param1=value1&param2=value2"  http://localhost:8080
```

Cabeceras HTTP

- Son pares nombre-valor que se envían en las peticiones y respuestas HTTP
- Definen información (metadatos) sobre los datos que se intercambian los clientes y servidores.



Cabeceras HTTP

- Tipos de cabeceras:
  - **Generales** (Date, Transfer-Encoding , ...)
  - **De petición** o de cliente (User-Agent, Accept, ...)
  - **De respuesta** o de servidor (Server, Age, ...)
  - **De entidad** (Content-Encoding, Content-Language, Content-Type ...)
- Protocolo HTTP: `http://www.w3.org/Protocols/rfc2616/rfc2616.html`

Cabeceras HTTP

Cliente **curl** y cabeceras HTTP

Realiza una petición web con **curl** y el parámetro **-v** y observa las cabeceras de cliente a servidor, de servidor a cliente, y los datos transferidos

```
curl -v http://www.marca.es
```

Códigos de estado/error

- Códigos que envían los servidores en las respuestas HTTP .
- Informan al cliente de cómo ha procesada la petición.
- Se acompañan de un texto explicativo.

Códigos de estado/error

- Código de 3 dígitos que se clasifican en función del primero.
  - 100-199 (Informativo, Informational).
  - 200-299 (Exito, Successful).
  - 300-399 (Redireccion, Redirection).
  - 400-499 (Errores del cliente, Client Error).
  - 500-599 (Errores en el servidor, Server Error)
- `http://www.w3.org/Protocols/HTTP/HTRESP.html`

Negociación de contenidos

- **Tipos de contenido** MIME-Version, Content-Description, Content-Transfer, Content-Type, ...
- **Lenguaje** Accept-Language
- **Conjunto de caracteres** Accept-Charset
- **Codificación/compresión** Accept-Encoding, Content-Encoding

Negociación de contenidos (MIME)

- Conjunto de especificaciones orientadas a intercambiar, usando protocolos como HTTP y SMTP, todo tipo de recursos (texto, audio, vídeo, imágenes...) de forma transparente a los usuarios.
- Tipos y subtipos (text/html, image/gif...) que determinan el contenido de los recursos enviados a través de la red.
- Lista de tipos MIME: `http://www.iana.org/assignments/media-types/media-types.txt`

Autenticación

- El protocolo HTTP soporta autenticación basada en dos mecanismos
  - **Basic**: Contraseñas en texto plano
  - **Digest**: Intercambio de **hashes** de las contraseñas
- Se consigue mediante
  - Código de estado 401
  - Cabeceras **WWW-Authenticate** y **Authoritation**

Cookies

- Fragmento de información que envía un servidor web en una respuesta HTTP y es almacenada por el navegador.
- El navegador puede enviar la cookie en solicitudes posteriores al mismo servidor.
- Cabeceras: **Cookies** y **Set-Cookie**

Sesiones

- HTTP es un protocolo sin estado
  - Un servidor web atiende las peticiones según llegan
  - Desde su punto de vista no hay relación entre las Peticiones
- Las aplicaciones web suelen necesitar estado
  - Reconocer a un usuario tras una semana sin hacer *login*
  - Reconocer a un usuario entre diferentes webs
  - ... y poder identificarlo con un perfil para enviarle publicidad.

Sesiones: Técnicas

- **Cookies**: Si un usuario no tiene aún una *cookie* de sesión, se crea una y se le asigna
- **URL Rewriting**: un parámetro *GET* único por usuario
- **Hidden fileds**: un párámetro *POST* en un campo tipo *HIDDEN*
- Otros: *fingerprint* de navegador (SVG, Canvas, plugins,...), cabecera **ETAG**
- Los *frameworks* para aplicaciones web suelen incluir API para sesiones (javaEE, PHP, .NET...)

Ampliación

- Almacenamiento en cache: *Cache-Control*, *Last-Modifed*, *Expires*, *Age*, *Etag*, *If-Match*, *IF-Modied-Since*...
- Redirecciones
- Conexiones persistentes: *Connection*, *Content-Length*...
- Para saber más
  - Servicios de Red e Internet (ISBN: 978-84-1622-832-4) Editorial Garceta.