

# Tema 6: World Wide Web

---

Introducción

HTTP

Cache web

WebDAV

Bibliografía

# 1. Servicio web

---

- El servicio más extendido de Internet
- World Wide Web -Tim Berners Lee (CERN)
  - 1990: Primer cliente (navegador-editor) y servidor web
  - 1991: URI, HTML, HTTP
  - 1993: Mosaic
- Espacio de comunicación común
  - Compartición info.
- Otros usos: distribución de información
  - Intranets
  - Documentación
  - Acceso datos aplicaciones

Aprovechan interfaz web y navegadores

# 1. Servicio web

---

- Hipertexto, hipermedia
- Arquitectura cliente-servidor
- HTTP
  - Protocolo intercambio de mensajes entre clientes y servidores
  - Formato de mensajes
- HTML define
  - Formato y visualización de páginas web
    - ◆ Contenido respuestas HTTP

## 2. HTTP

---

- Protocolo de Transferencia de Hipertexto
  - RFCs 1945, 2068 (obsoleto), 2616, 2817
  - Entrega recursos web identificados por URI (URL)
- Capa de aplicación
- Sistemas de información
  - Hipermedia, distribuidos, colaborativos
- Simplifica y toma ideas de SMTP, NNTP, FTP, Gopher

## 2. HTTP

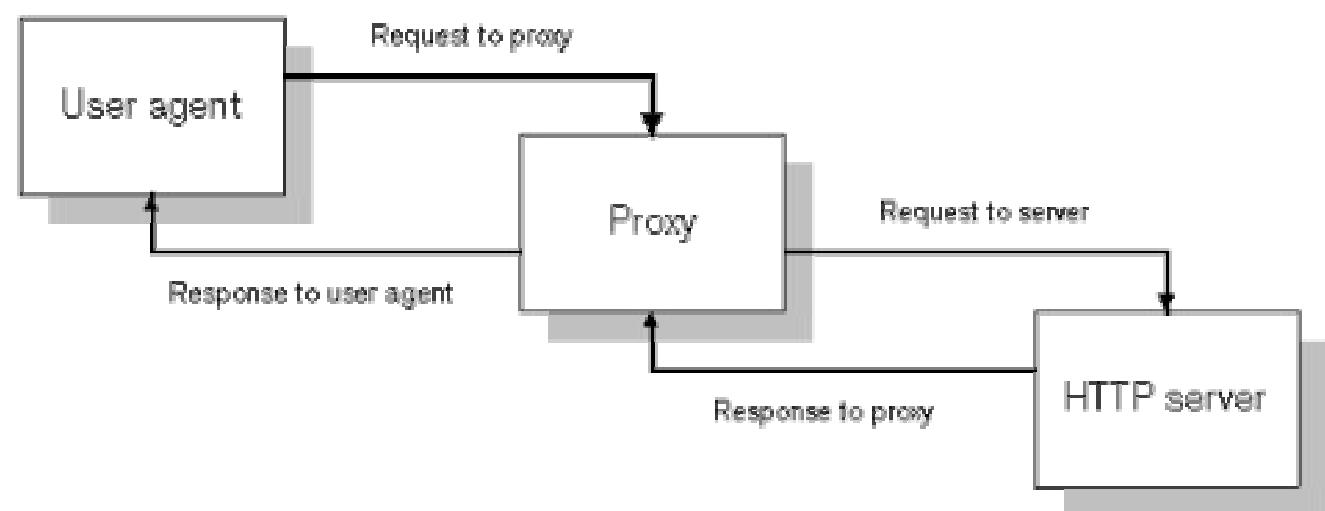
---

### ■ Paradigma cliente-servidor



### ■ Comunicación

- Directa
- Proxy



## 2. HTTP

---

### ■ Proxies HTTP

- Intermediario entre cliente y servidor
- Funciona como cliente y servidor
- Cortafuegos, caches LAN, ...
- Cliente con proxy: peticiones a éste en lugar de al servidor
  - ◆ El proxy sabe dónde tiene que redirigir la petición

## 2. HTTP

---

- Métodos de petición y envío de datos
- Entrega de recursos identificados por URLs
  - Ficheros (HTML, imágenes, resultados de consultas, salida CGI...)
- Protocolo sin estado
  - Cada orden se ejecuta independientemente de los anteriores
  - Dificultad sitios web “inteligentes”
    - ◆ Java, JavaScript, cookies, ActiveX ...
- Usa MIME

## 2. HTTP. Versiones

---

- HTTP/0.9
  - *httpd* CERN, Mosaic
  - Sencillo, sólo transmisión de datos
  - GET
- HTTP/1.0 (RFC 1945)
  - MIME
  - Metainfo datos: descripción contenido del mensaje
  - Envío info al servidor
  - Cada petición ⇒ conexión
    - ◆ *3-way-handshake + slow-start* reducen velocidad

## 2. HTTP

---

- HTTP/1.1 (RFC 2616)

- Superconjunto v1.0
- Conexiones persistentes: varias peticiones/conexión
  - ◆ 20% más prestaciones
- Peticiones condicionales
- Mayor soporte caches: latencia y BW
- Servicio de múltiples dominios desde una sola @IP
- Nuevos métodos petición: PUT, DELETE, OPTIONS, TRACE
- Autentificación *digest*: MD5 ...
- Codificación por trozos –*chunked*–
  - ◆ Mayor rapidez en servicio de páginas dinámicas
- ...

- HTTP-NG

## 2. HTTP

---

### ■ Transacción HTTP

- Cliente establece conexión con servidor
  - ◆ TCP 80
- Cliente envía petición
- Servidor responde
- Servidor cierra conexión

HTTP/1.1

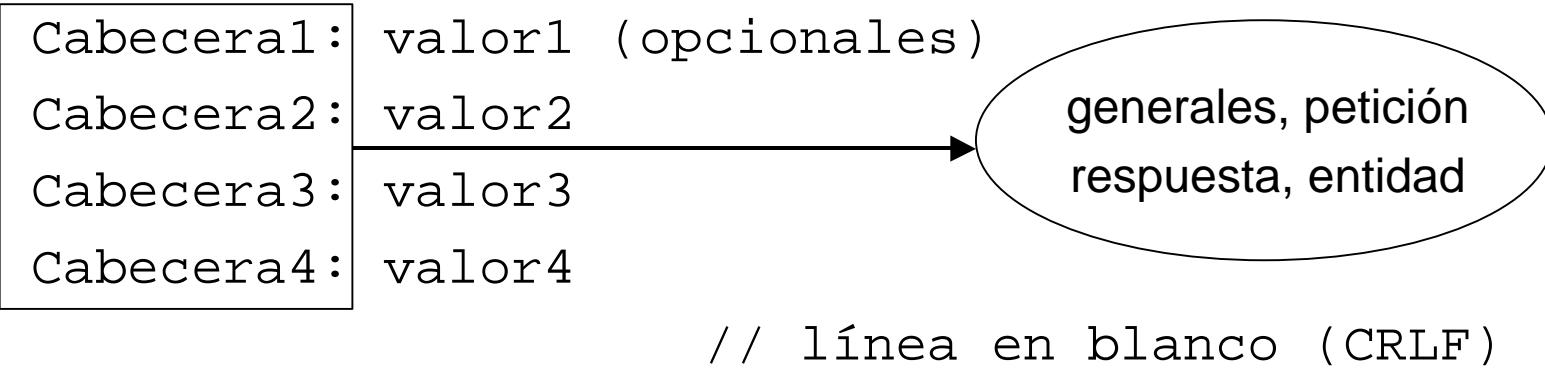


## 2. HTTP

---

- Formato mensajes
  - Mensaje genérico RFC 822

<línea inicial, distinta para peticiones y respuestas>



<cuerpo del mensaje (opcional), por ejemplo  
contenido de un fichero o datos de consultas;  
pueden ser líneas de texto o datos binarios  
\$&\*%@!^\$@>

## 2. HTTP

---

### ■ Cabeceras

- General: conexión
- Petición
- Respuesta
- Entidad: descripción atributos cuerpo del mensaje
  
- HTTP 1.0: 16 cabeceras
  - ◆ Opcionales
- HTTP 1.1: 46
  - ◆ Host requerida en las peticiones

## 2. HTTP 1.0

---

### ■ Cuerpo del mensaje

- Petición
  - ◆ Datos del usuario, ficheros enviados a un servidor
  - ◆ Anunciado por cabecera Content-Length o Transfer-Encoding
- Respuesta
  - ◆ Recurso enviado al cliente
  - ◆ Texto explicatorio error
- Describo en las cabeceras
  - ◆ Content-Length: número de bytes en el cuerpo
  - ◆ Content-Type: tipo de datos MIME en el cuerpo
    - ◆ text/html, image/gif, ...

## 2. HTTP

---

### ■ Peticiones HTTP

Método petición, URI, versión protocolo

Cabecera general

Cabecera petición

Cabecera entidad

[Cuerpo mensaje]

#### ○ Ejemplo

GET /pub/WWW/TheProject.html HTTP/1.1

Host: www.w3.org

## 2. HTTP

---

### ■ Métodos peticiones HTTP

- OPTIONS: pregunta funcionalidades

- ◆ Servidor

C: OPTIONS \* HTTP/1.1

S: 200 Ok

S: Allow: OPTIONS, GET, HEAD, POST, PUT

S: Accept-Ranges: bytes

S: Accept-Encoding: gzip

- ◆ Recurso

C: Request: OPTIONS /cgi-bin/order HTTP/1.1

S: Response: 200 Ok

S: Allow: POST

S: Accept-Encoding:

## 2. HTTP

---

### ■ Métodos peticiones HTTP

- GET: petición URI indicado
  - ◆ Implementación obligatoria

C: GET /index.html HTTP/1.1

S: (fichero index.html)

C: GET /index.html HTTP/1.1

    If-Modified-Since: Wed, 5 Sep 1996 09:45:23 GMT

S: (fichero index.html)

C: GET /index.html HTTP/1.1

    If-Modified-Since: Tue, 1 Oct 1996 14:09:34 GMT

S: A 304 not modified (no se envía cuerpo)

C: GET /cgi/busca?nom=jesus&pass=j2s5s HTTP/1.1

S: Salida recurso busca con parámetros pasados

## 2. HTTP 1.0

---

C: GET /path/fichero.html HTTP/1.0

From: someuser@jmarshall.com

User-Agent: HTTPTool/1.0

[línea en blanco]

S: HTTP/1.0 200 OK

Date: Mon, 14 May 2001 11:05:25 GMT

Content-Type: text/html

Content-Length: 1354

```
<html>
<body>
<h1>Hola troncos!</h1>
(más contenidos del fichero)
.
</body>
</html>
```

## 2. HTTP

---

- Métodos peticiones HTTP
  - HEAD: petición cabeceras URI indicado
    - ◆ Implementación obligatoria

C: HEAD /index.html HTTP/1.1

S: (cabeceras respuesta index.html)

## 2. HTTP

---

- Métodos peticiones HTTP

- POST

- ◆ Envío datos al servidor en el cuerpo del mensaje
    - ◆ Datos codificados URL
    - ◆ URI: programa procesado datos enviados (Perl, C ...)
    - ◆ Salida del programa: respuesta HTTP

C: POST /cgi-bin/submit HTTP/1.1

Content-Length: 3819

[ 3819 bytes de datos ]

S: [Salida del proceso submit]

C: POST /cgi-bin/order HTTP/1.1

Content-Length: 6082

[ 6082 bytes de datos ]

S: [Salida del proceso order]

```
graph LR; A[Content-Length: 3819<br>[ 3819 bytes de datos ]] --> B([Obligatorio]); C[Content-Length: 6082<br>[ 6082 bytes de datos ]] --> B;
```

## 2. HTTP

---

- Métodos peticiones HTTP
  - PUT: almacenar cuerpo petición en el URI indicado

C: PUT /users/phethmon/welcome.html HTTP/1.1

Content-Type: text/html

Content-Length: 3109

[ 3109 bytes de datos ]

S: 204 No Content

Server: 3wd/1.1

C: PUT /catalog/sect1/pg34.html HTTP/1.1

Content-Type: text/html

Content-Length: 4526

Content-Encoding: gzip

[ 4526 bytes de datos ]

S: 501 Not Implemented

Server: 3wd/1.1 SAN - 6. WWW

## 2. HTTP

---

- Métodos peticiones HTTP

- DELETE: eliminar URI indicado
    - ◆ Seguridad, autentificación

C: DELETE /catalog/sales/oct96.html HTTP/1.1

S: 204 No Content

C: DELETE /company/about.html HTTP/1.1

S: 202 Accepted Pending Approval

## 2. HTTP

---

### ■ Métodos peticiones HTTP

- TRACE: petición cabeceras recibidas por el servidor

C: TRACE / HTTP/1.1

Host: www.utk.edu

Max-Forwards: 10

User-Agent: JoeBrowser/10.0

S: 200 OK

Content-Type: message/http

Content-Length: 84

TRACE / HTTP/1.1

Max-Forwards: 10

User-Agent: JoeBrowser/10.0

Host: www.utk.edu

## 2. HTTP

---

### ■ Respuestas HTTP

Versión protocolo + código éxito/error + descripc.

Cabeceras general|respuesta|entidad

[Cuerpo mensaje]

- Ejemplos

HTTP/1.1 200 OK

HTTP/1.0 404 Not Found

Server: 3wd/1.1

Content-Type: text/html

Content-Length: 200

[200 bytes de datos]

## 2. HTTP

---

- Respuestas HTTP

- Códigos respuesta
  - ◆ 1xx: informativo
  - ◆ 2xx: éxito
  - ◆ 3xx: redirección cliente a otro URL
  - ◆ 4xx: error del cliente
  - ◆ 5xx: error del servidor

## 2. HTTP 1.1

---

### ■ Clientes

#### ○ Cabecera Host

- ◆ www.kk1.com y www.kk2.com pueden estar en el mismo servidor
- ◆ Hay que especificar destinatario petición:

```
GET /ruta/fichero.html HTTP/1.1  
Host: www.kk1.com:80  
[Línea en blanco]
```

#### ○ Aceptar codificación por trozos

- ◆ Servidor envía la respuesta sin saber su longitud
- ◆ Cabecera Transfer-Encoding: chunked
- ◆ Varios trozos seguidos por una línea con un “0” y cabeceras
- ◆ Cada parte
  - ◆ Tamaño de los datos –hexa- ( + “;” + info no estándar)+ CRLF
  - ◆ datos

## 2. HTTP 1.1

---

### ■ Clientes

#### ○ Ejemplo codificación por trozos

```
HTTP/1.1 200 OK
Date: Mon, 14 May 2001 11:40:40 GMT
Content-Type: text/plain

Transfer-Encoding: chunked          Content-Length: 42
Cabecera: valor
Cabecera2: valor2

1a; ignorar-lo-que-haya-aquí      abcdefghijklmn...
abcdefgijklmnopqrstuvwxyz        [línea en blanco]
10
1234567890abcdef
0
Cabecera: valor
Cabecera2: valor2
[línea en blanco]
```

## 2. HTTP 1.1

---

### ■ Clientes

- Conexiones persistentes y cabecera `Connection: close`
  - ◆ HTTP 1.0: una conexión/recurso ⇒ CPU, BW, memoria, ...
  - ◆ HTTP 1.1: conexión persistente por defecto
    - ◆ `Connection: close`
- Respuesta “100 Continue”
  - ◆ Para enlaces lentos
  - ◆ Capacidad para manejar esta respuesta del servidor

## 2. HTTP 1.1

---

### ■ Servidores

- Requerir la cabecera Host
  - ◆ Petición HTTP 1.1 sin cabecera Host ⇒ 400 Bad Request
- Aceptar URLs absolutos
  - ◆ Cabecera Host ⇒ ñapa temporal
  - ◆ Futuras versiones HTTP ⇒ URL absoluto
- GET http://www.maquina.com/ruta/fichero.html HTTP/1.2
- Aceptar codificación por trozos
- Conexiones persistentes
  - ◆ Respuestas en el mismo orden que peticiones de una misma conex
  - ◆ Temporizador cierre conexiones inactivas (~10 segs)
  - ◆ Connection: close

## 2. HTTP 1.1

---

### ■ Servidores

- Cabecera Date
  - ◆ Caches
  - ◆ Respuestas del servidor con sello de tiempo GMT
- Date: Mon, 14 May 2001 11:40:40 GMT
- Peticiones con cabeceras If-Modified-Since/If-Unmodified-Since
  - ◆ Envío sólo si ha habido /o no cambio desde la fecha indicada
  - ◆ GET / todos métodos
  - ◆ Recurso - 304 Not Modified / Recurso – 412 Precondition Failed
- Respuesta “100 Continue”

## 2. HTTP

---

### ■ Navegación con telnet

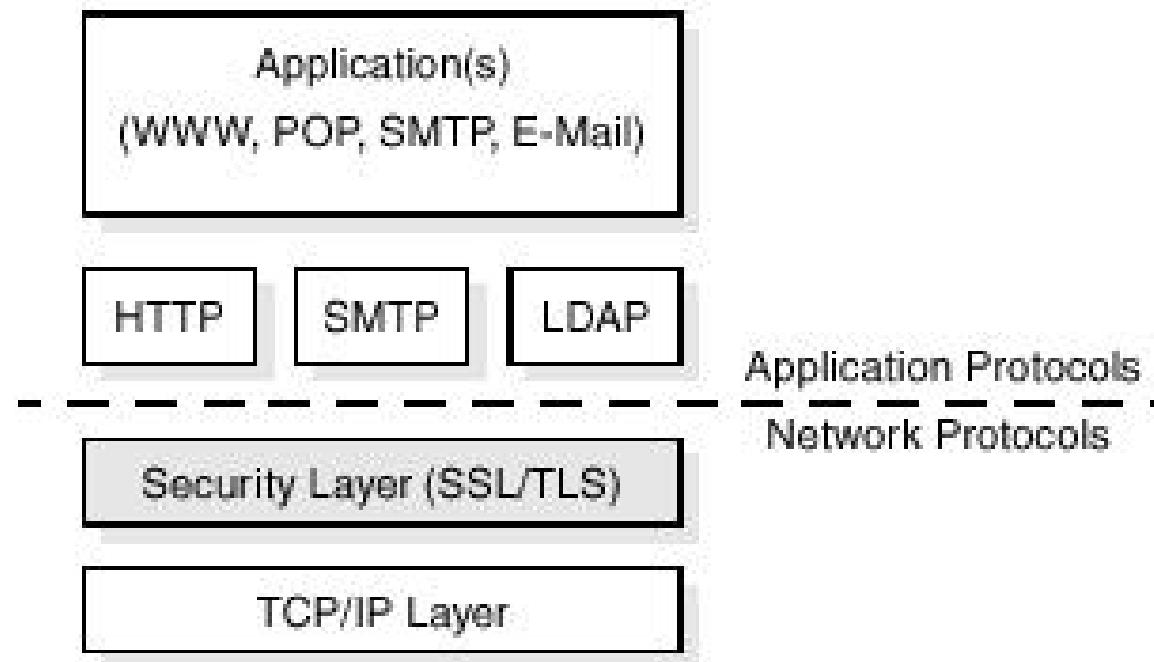
```
% telnet apollo.cps.unizar.es 80
Trying 155.210.152.13...
Connected to apollo.cps.unizar.es.
Escape character is '^]'.

GET /
< contenidos del fichero index.html>
Connection closed by foreign host.
```

## 2. HTTPS

---

- HTTPS = HTTP sobre SSL
- Extensión segura de HTTP
- Cifrado y autentificación de los sistemas finales
- Puerto TCP 443



## 2. HTTP-NG

---

- HTTP 1.x: falta de modularidad
  - Complejo (v1.1)
  - Extensibilidad pobre: interacciones entre extensiones compleja
  - No hay modelo de utilización de aplicaciones
    - ◆ *Tunneling* (GET, POST) en lugar de sistema de objetos distribuido
    - ◆ Difícil aplicar políticas de seguridad (cortafuegos)
  - Tecnologías alternativas no integran bien
    - ◆ DCOM, CORBA, Java RMI utilizan HTTP como servicio inicialización
  - Escalabilidad pobre
    - ◆ Sobrecarga de cabeceras

## 2. HTTP-NG

---

### ■ Requerimientos

- Simplicidad del núcleo
  - ◆ Arquitectura modular
- Uso eficiente de recursos de Internet (carga HTTP)
- Eficiencia en comunicaciones banda estrecha (*wireless*)
- Escalabilidad global
- Prestaciones a usuarios y aplicaciones finales
- Extensibilidad distribuida
- Flexibilidad de transporte

## 2. HTTP-NG

---

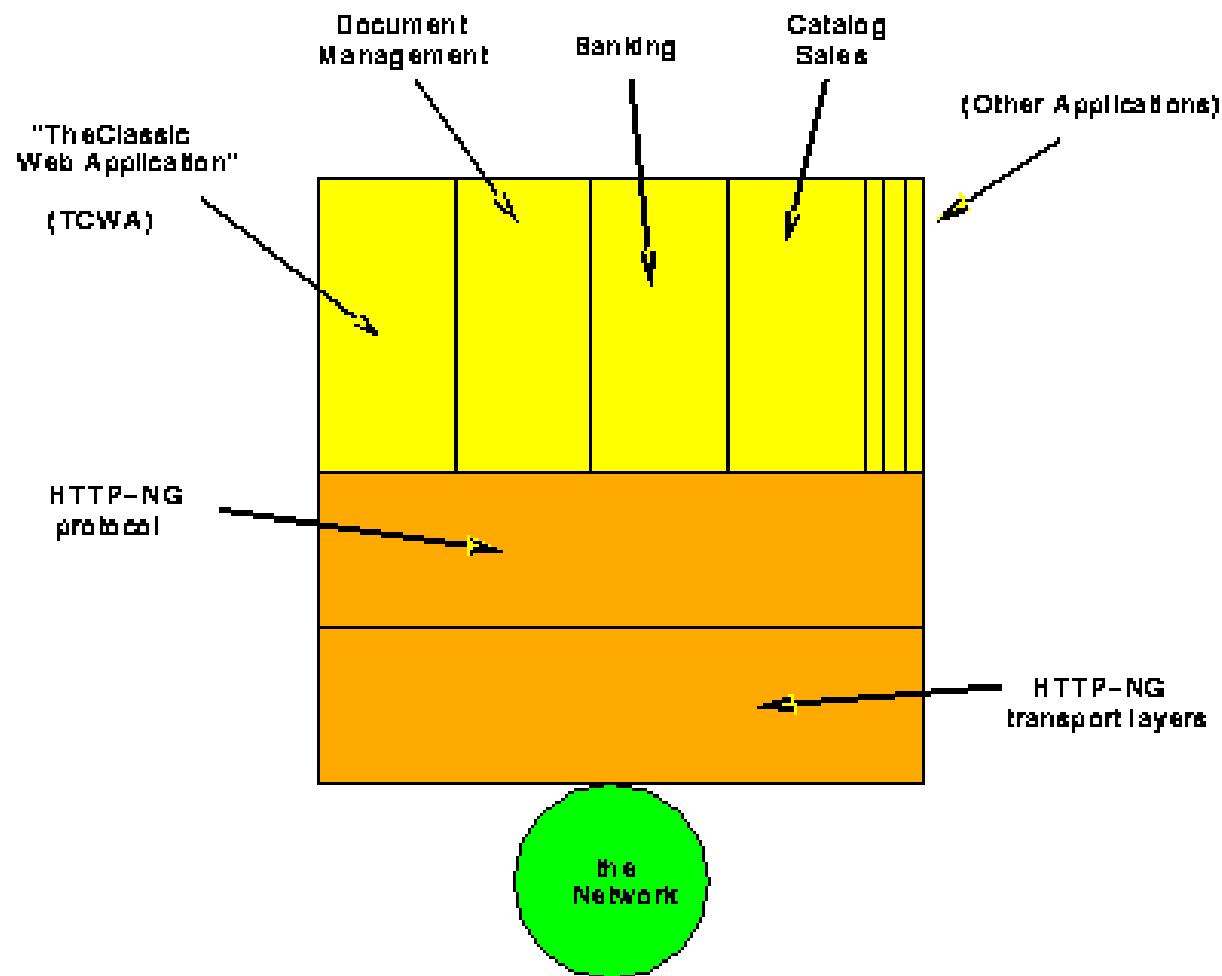
- Arquitectura de capas

- Capa de transporte
    - ◆ Webmux
      - Múltiples conexiones sobre una conexión TCP
      - conexión / puerto webmux = sesión / canal
  - Capa de invocación de operaciones
    - ◆ Protocolo de mensajes orientado a objetos
  - Capa de aplicación
    - ◆ Cambia según el tipo de aplicación
      - TCWA: web clásica, WebDAV

## 2. HTTP-NG

---

### ■ Arquitectura de capas



### 3. Cache web

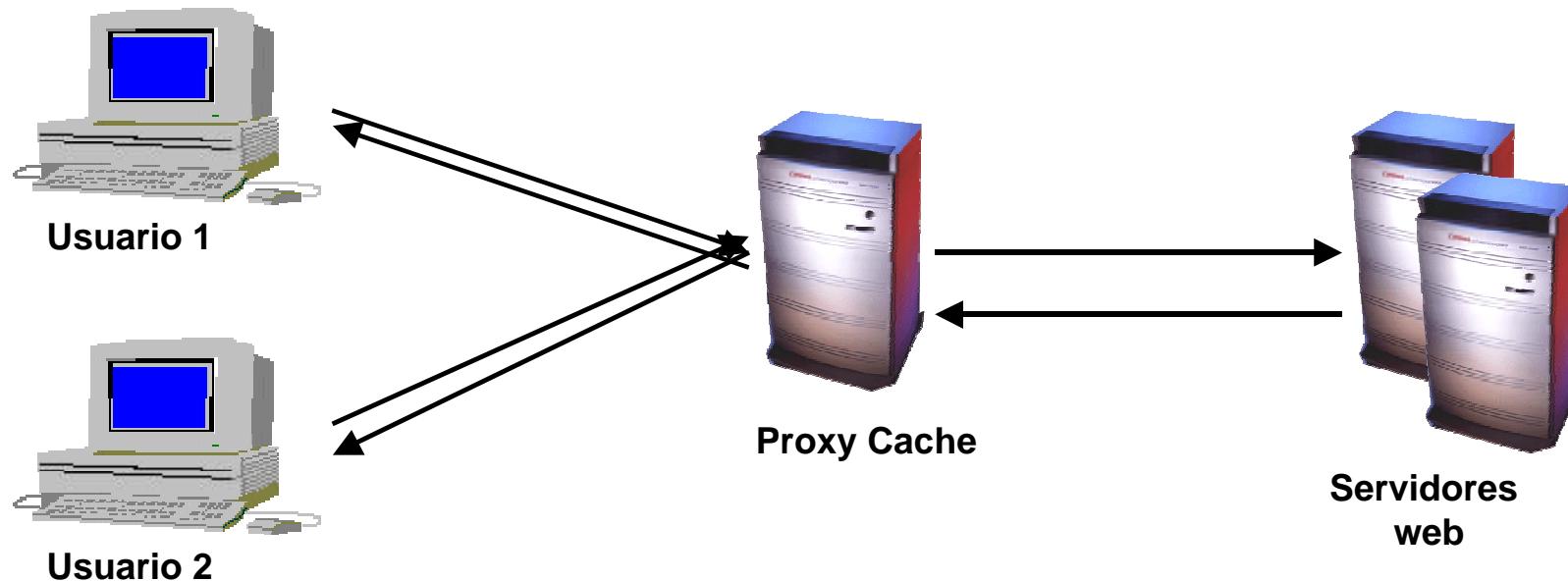
---

- Almacena contenidos web accedidos
  - Docs HTML, imágenes, ficheros
- Si se repite petición a un contenido, sirve copia local
- Ventajas
  - Menor consumo de ancho de banda
    - ◆ Menos peticiones y respuestas al exterior
  - Reducción de la carga de los servidores
    - ◆ Menos peticiones que gestionar
  - Reducción latencia
    - ◆ Respuestas más rápidas (más cerca)

# 3. Cache web

---

## ■ Funcionamiento

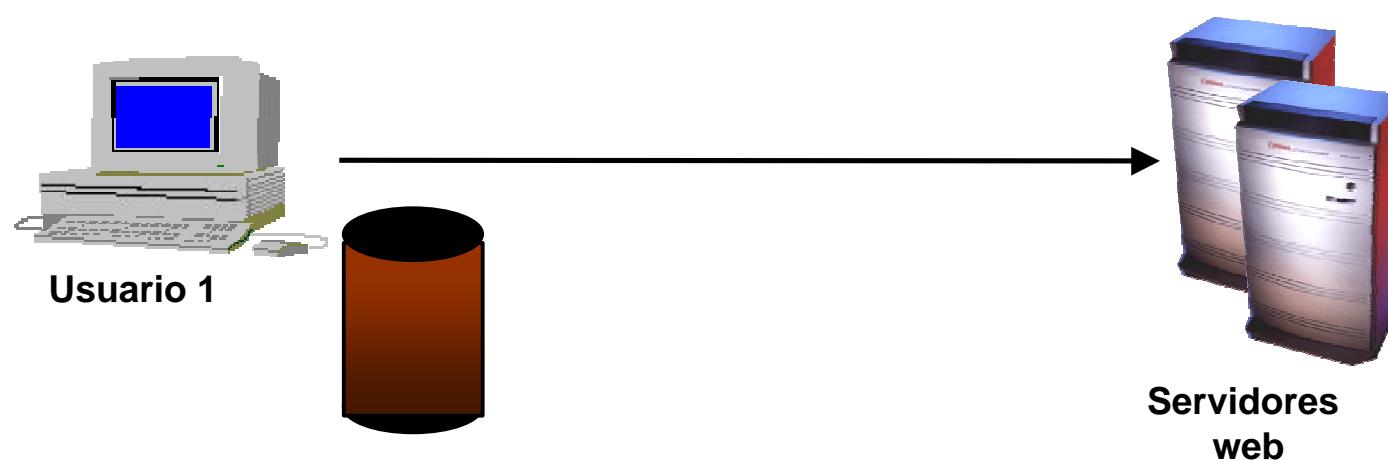


# 3. Cache web. Tipos

---

## ■ Cliente

- Cache navegador
  - ◆ Comprobación datos actualizados
  - ◆ Info varios servidores, un usuario

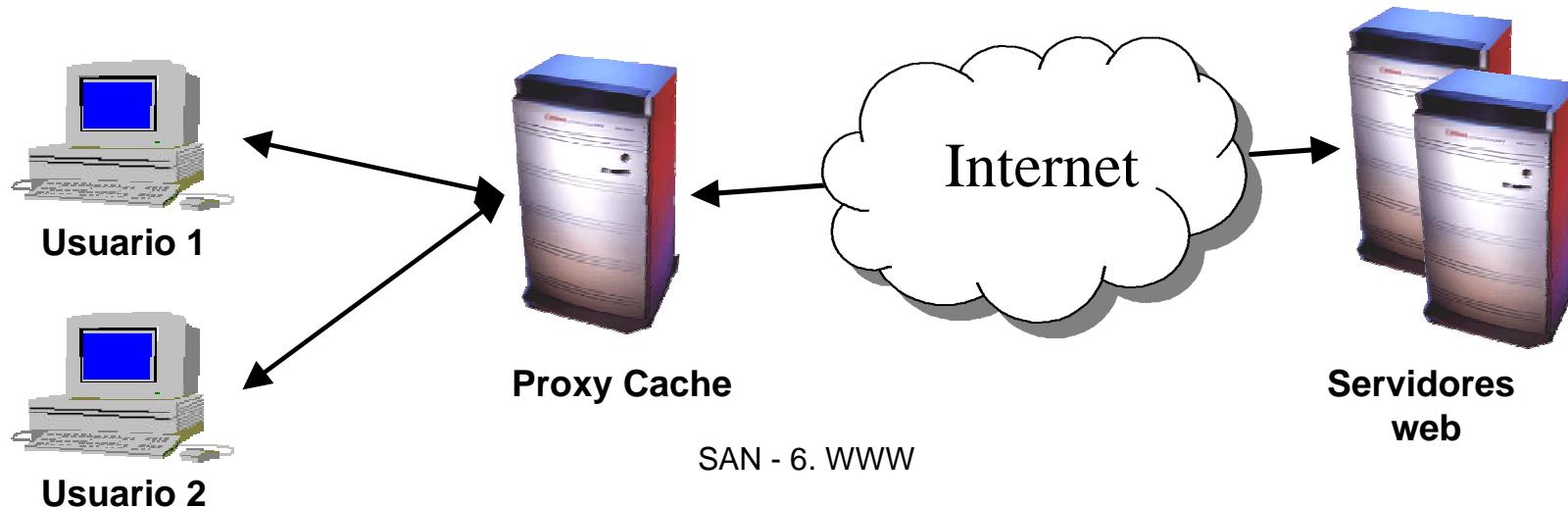


# 3. Cache web. Tipos

---

## ■ Cliente

- Proxy cache, forward proxy
  - ◆ Entre cliente y servidor: discontinuidad de redes
    - ◆ ISPs, grandes entidades, cortafuegos
  - ◆ Info varios servidores, varios usuarios
  - ◆ Acierto: 50-80%
  - ◆ Jerarquía para cooperación
  - ◆ Pasivos, activos



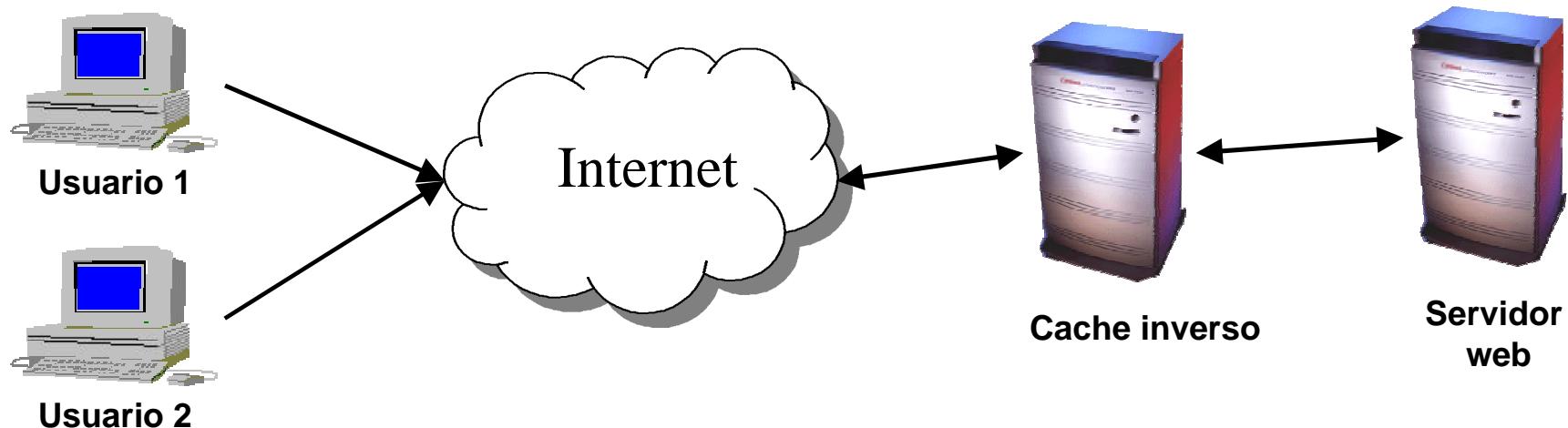
# 3. Cache web. Tipos

---

## ■ Servidor

### ○ Proxy cache Inverso

- ◆ Entrega de contenidos más rápida (estáticos)
- ◆ Disminución carga servidores
- ◆ Respaldo de un servidor
- ◆ Un servidor, varios usuarios



# 3. Cache web

---

## ■ Agrupación caches

- Comunicación entre caches próximas
- Configuración típica:
  - ◆ Jerarquía con cache principal en el límite de una red grande, antes del enlace con otra red más lenta o saturada
  - ◆ Gestiona todos los accesos fuera de la red
  - ◆ Conexión con cache principal al otro lado de la red
  - ◆ Cache principal ⇒ Secundarias ⇒ Usuarios
- Relaciones: padre-hijo y hermanos
  - ◆ Cache-padre: recibe petición y la devuelve (local o red)
    - ◆ Proveedor con clientes
  - ◆ Cache-hermana: recibe petición y devuelve contenido o negativa
    - ◆ Entre proveedores distintos

# 3. Cache web

---

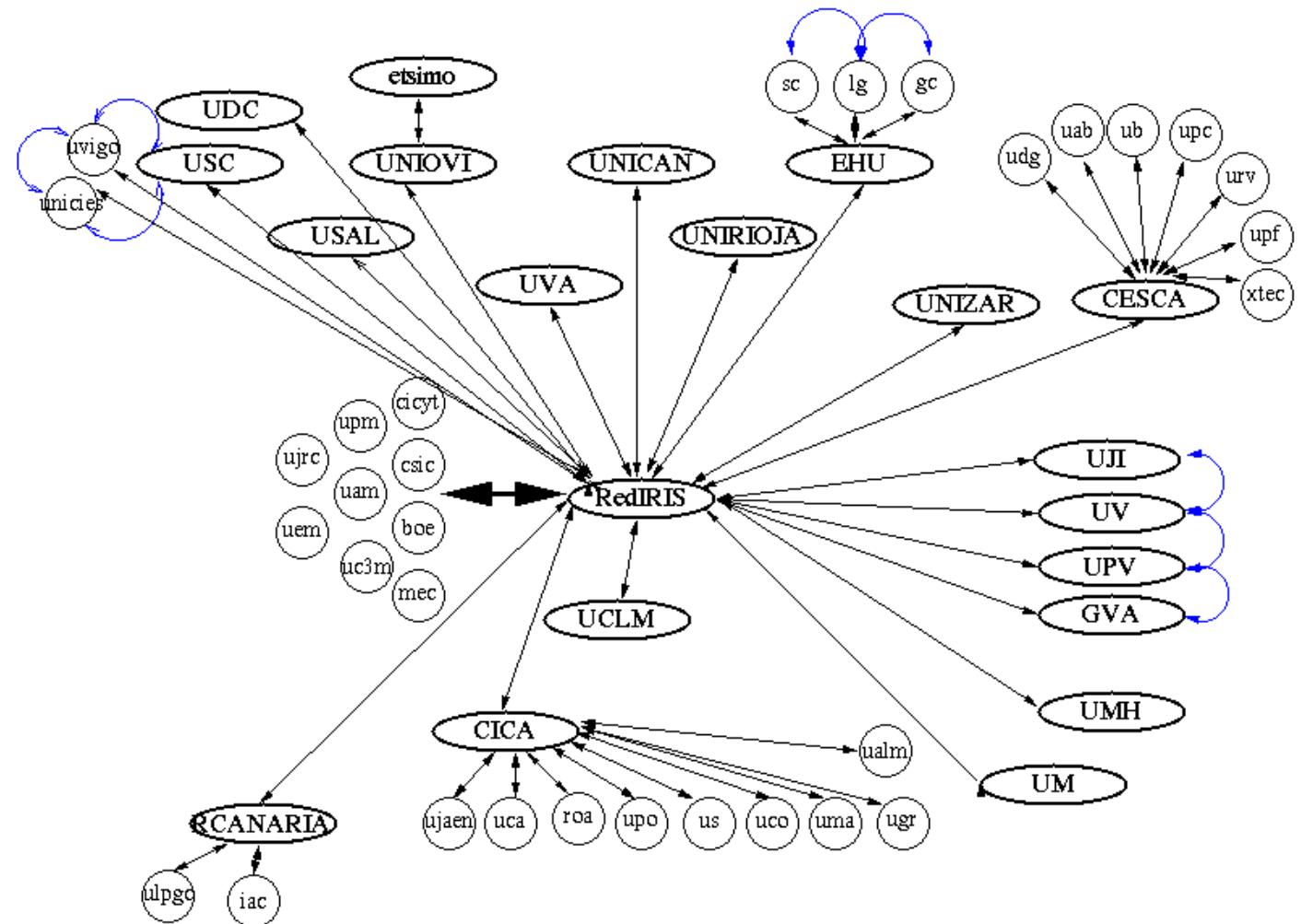
## ■ Agrupación caches

- Comunicación entre caches
  - ◆ ICP: Internet Cache Protocol - RFC2186
    - ♦ Cache pregunta a otra/s si tienen un recurso
  - ◆ Cache-Digest
    - ♦ Intercambio periódico de tablas entre caches
- Configuración de respaldo mutuo
  - ◆ Continuación de servicio en caso de caída de máquina

# 3. Cache web

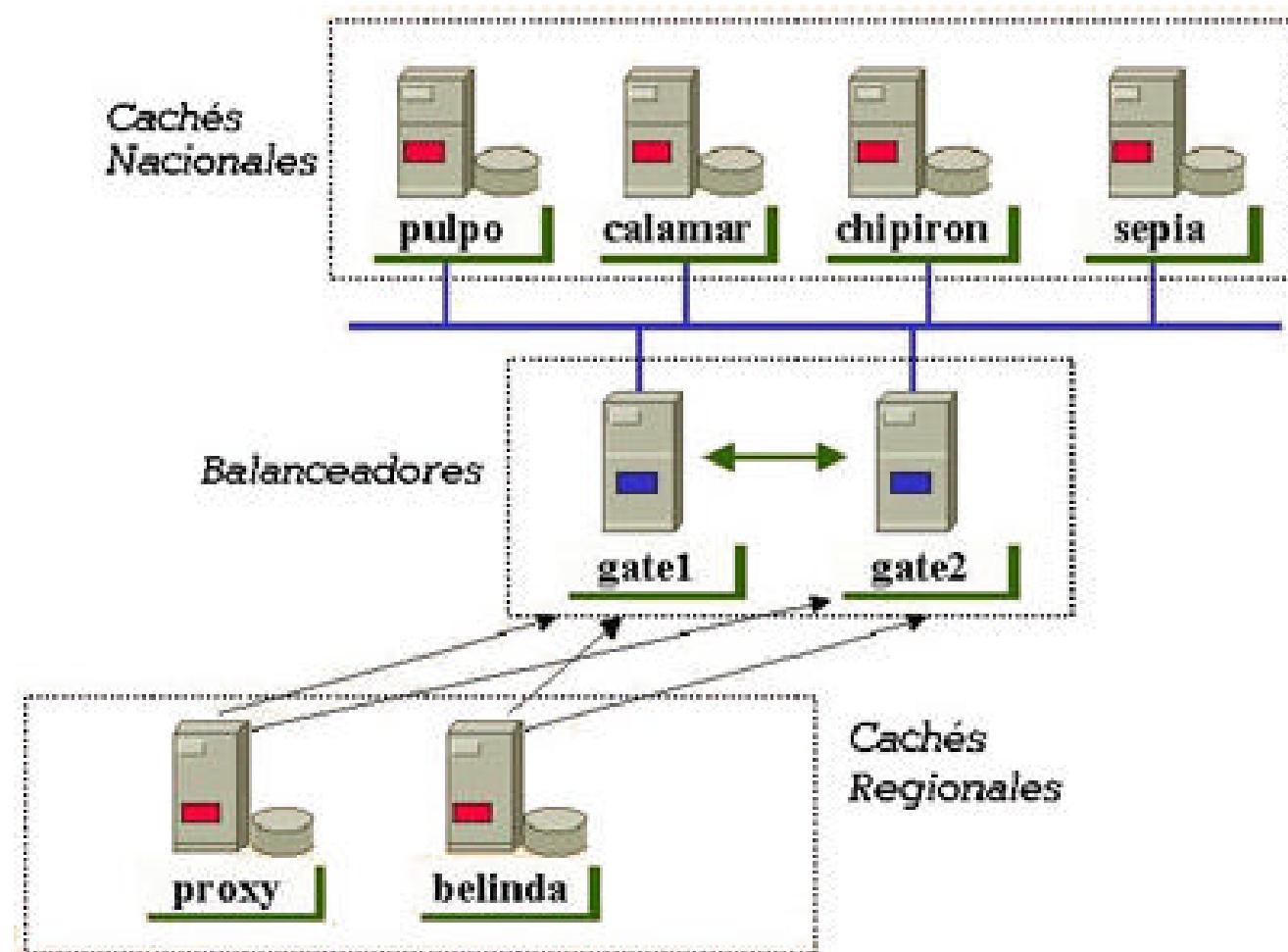
---

## ■ Mapa caches RedIRIS



# 3. Cache web

## ■ Configuración caches RedIris



### 3. Cache web

---

- Tecnología incomprensida
  - Pérdida control sitio web
  - Servicio de contenidos obsoletos
  - Punto de fallo
  - Estadísticas de acceso poco precisas
  - Recursos adicionales
  
- Realidad
  - Infraestructura caches

# 3. Cache web

---

## ■ Control caches

- HTML Meta Tags
  - ◆ Descripción atributos en la sección <HEAD>
  - ◆ No son efectivas: proxy caches no leen HTML
- Cabeceras HTTP
  - ◆ Navegador, proxy, servidor
  - ◆ HTTP 1.0
    - ◆ cabecera Expires
  - ◆ HTTP 1.1
    - ◆ directivas Cache-Control

# 3. Cache web

---

- Control caches
  - Cabeceras HTTP

```
HTTP/1.1 200 OK
Date: Fri, 30 Oct 1998 13:19:41 GMT
Server: Apache/1.3.3 (Unix)
Cache-Control: max-age=3600, must-revalidate
Expires: Fri, 30 Oct 1998 14:19:41 GMT
Last-Modified: Mon, 29 Jun 1998 02:28:12 GMT
ETag: "3e86-410-3596fbbc"
Content-Length: 1040
Content-Type: text/html
```

### 3. Cache web

---

- ¿Qué objetos no deben almacenarse en una cache?
  - Respuestas a peticiones
    - ◆ POST
    - ◆ PUT
    - ◆ DELETE
    - ◆ OPTIONS
    - ◆ TRACE
    - ◆ Cabecera Authorization
  - **Almacenar respuestas a GET, HEAD**
    - hay excepciones ...
      - ◆ Cabecera no-cache, ausencia validador

# 3. Cache web

---

## ■ Reglas cache web

1. Objeto web está en cache navegador
  - ◆ Servir si *una vez por sesión*
2. Objeto web en proxy: determinar si es obsoleto
  - ◆ Fecha expiración OK: servir
  - ◆ Reciente y modificado hace mucho tiempo: servir
  - ◆ Obsoleto: petición al servidor origen para **validarlo**
    - No ha cambiado: servir de cache
    - Cambio: pedir al origen
3. Objeto web no en proxy
  - ◆ Pedir objeto al origen

# 3. Cache web

---

## ■ Cabecera HTTP Expires

- Respuestas HTTP 1.0
- Fecha objeto obsoleto
- Pasada fecha: comprobación de cambios con origen
- Indicación fecha
  - ◆ Fecha HTTP (GMT)
  - ◆ Tiempo absoluto, tiempo último acceso, tiempo último cambio
- Empleo
  - ◆ Imágenes estáticas (barras y botones de navegación): t↑↑
  - ◆ Cambios periódicos: hora expiración = hora actualización
- Ejemplo:

Expires: Fri, 30 Oct 1998 14:19:41 GMT

# 3. Cache web

---

## ■ Cabeceras **Cache-control**

- HTTP 1.1
- Tratamiento de los objetos por parte de las caches
- Qué puede ser almacenado en la cache
- Modificaciones del mecanismo de expiración
- Controles de validación y recarga
- Control sobre la transformación de entidades
  - ◆ Ej: cambio formato de imagen
- Extensiones al sistema de cache

# 3. Cache web

---

- Peticiones **Cache-control**
  - no-cache: caches se inhiben
  - no-store: proxy no debe almacenar
  - max-age=[segundos]: máxima edad aceptable objetos en cache
  - max-stale=[segundos]: se aceptan objetos [segundos] viejos
  - min-fresh=[segundos]: objeto debe ser [segundos] joven
  - no-transform
  - only-if-cached: enviar sólo si está en cache
  - cache-extension
- Más info en el draft HTTP 1.1

# 3. Cache web

---

## ■ Respuestas **Cache-control**

- public: fuerza una respuesta almacenable (autentificaciones tb)
- private: la respuesta no debe ser guardada
- no-cache: fuerza caches a validar antes de servir copia
- no-store: proxy no debe almacenar
- no-transform: no transformar cuerpo mensaje (ej: TIFF-JPEG)
- must-revalidate: validar recurso obsoleto en cada petición
- proxy-revalidate: ídem sin aplicar a caches privadas
- max-age=[segundos]: fecha expiración objeto
- s-maxage=[segundos]:
- cache-extension

Cache-Control: max-age=3600, must-revalidate

## 3. Cache web

---

- Ejemplo: Apache 1.3, fichero .htaccess

```
### activate mod_expires
ExpiresActive On
### Expire .gif's 1 month from when they're accessed
ExpiresByType image/gif A2592000
### Expire everything else 1 day from when it's last
modified
### (this uses the Alternative syntax)
ExpiresDefault "modification plus 1 day"
### Apply a Cache-Control header to index.html
<Files index.html>
Header append Cache-Control "public, must-revalidate"
</Files>
```

# 3. Cache web

---

- Validación: verificación de cambio en un objeto
  - cache-cache, cache-origen
- Validadores
  - Ausencia  $\Rightarrow$  No cacheo
  - Last-Modified: último cambio
    - ◆ Validación con peticiones
      - ◆ If-Modified-Since, If-Unmodified-Since
  - E-Tags (1.1): identificadores únicos generados por el servidor
    - ◆ Cambian con el objeto
    - ◆ Validación con petición If-None-Match
  - Contenidos
    - ◆ Estáticos: servidor genera Last-Modified y E-Tags automáticamente
    - ◆ Dinámicos: no genera validadores

## 3. Cache web

---

### ■ Ejemplo 1

- Agente cache tiene copia del recurso solicitado
- Servidor origen no ha indicado restricciones almacenamiento
- Cache ha calculado la edad y expiración del recurso: actual  
age-value = 100  
freshness-lifetime = 300

P: GET /resource HTTP/1.1

R: La cache sirve la copia almacenada

P: GET /resource HTTP/1.1

Cache-Control: min-fresh=250

R: La cache debe validar el recurso

# 3. Cache web

---

## ■ Ejemplo 1

P: GET /resource HTTP/1.1

Cache-Control: no-cache

R: La cache debe recargar el recurso del servidor

P: GET /resource HTTP/1.1

Cache-Control: max-age=0

R: La cache debe validar con el servidor

P: GET /resource HTTP/1.1

Cache-Control: max-age=500

R: La cache utiliza la copia almacenada

## 3. Cache web

---

- Ejemplo 2: política cache RedIRIS
  - Documento con fecha de expiración
    - ◆ Sin alcanzar: se sirve de cache
    - ◆ Alcanzada: comprobación If-Modified-Since (IMS)
  - Imágenes, sonido, animaciones
    - ◆ Durante un día no IMS
  - Resto
    - ◆ IMS si factor tiempo en cache respecto tiempo última modificación supera un %
    - ◆ 50%: un objeto que al meterlo en cache llevaba 6 días sin modificarse se tomará como vigente (No IMS) durante 3 días

### 3. Cache web

---

- Cabeceras Expires y Last-Modified

- Pueden verse en *Page-Info* (Netscape)

Caching Tutorial for Web Authors and Webmasters has the following structure: [http://www.mnot.net/cache\\_docs/](http://www.mnot.net/cache_docs/)

Location:	<a href="http://www.mnot.net/cache_docs/">http://www.mnot.net/cache_docs/</a>
File MIME Type:	text/html
Source:	Currently in disk cache
Local cache file:	M0N0T7UL
Last Modified:	martes 20 de junio de 2000 4:08:25 Local time
Last Modified:	martes 20 de junio de 2000 2:08:25 GMT
Content Length:	44297
Expires:	lunes 21 de mayo de 2001 8:02:43
Charset:	iso-8859-1
Security:	This is an insecure document that is not encrypted and offers no security protection.

# 3. Cache web

---

- Todas Cabeceras

- Conexión telnet servidor web

```
telnet www.cps.unizar.es 80
GET /index.html HTTP/1.1          ;HEAD
Host: www.cps.unizar.es
```

```
HTTP/1.1 200 OK
Date: Sun, 20 May 2001 21:48:34 GMT
Server: Apache/1.3.12 (Unix)
Last-Modified: Wed, 11 Apr 2001 19:05:42 GMT
ETag: "804-270-3ad4ab06"
Accept-Ranges: bytes
Content-Length: 624
Content-Type: text/html
```

<HTML>

...

# 3. Cache web

---

## ■ Trucos sitio web

- Utilizar servidores web que soporten peticiones GET IMS
- Consistencia en la referencia de objetos
  - ◆ Mismos URLs para mismos objetos
- Biblioteca común de imágenes
- Forzar cache de imágenes y contenidos estáticos
  - ◆ Expires largo
- Caches reconozcan contenidos actualizados regularmente
  - ◆ Tiempo expiración
- Si un recurso cambia, cambiar su nombre
  - ◆ Podrá fijarse t de expiración mayor

# 3. Cache web

---

## ■ Trucos sitio web

- No cambiar ficheros innecesariamente
  - ◆ Actualización selectiva
- Usar cookies sólo donde sea necesario
  - ◆ Difíciles de cachear
  - ◆ Sólo en páginas dinámicas
- Minimizar el uso de SSL
  - ◆ No se cachean
  - ◆ Pocas imágenes
- Scripts
  - ◆ Cachear los que dependan sólo del URL
  - ◆ CGIs cacheables con GET en lugar de POST

### 3. Cache web

---

- Herramientas *cacheabilidad*

- <http://www ircache net/cgi-bin/cacheability.py>

**www.unizar.es**

Expires -

Cache-Control -

Last-Modified 2 hr 26 min ago (Wed, 08 May 2002 07:28:16 GMT) validated

Etag -

Content-Length 8.5K (8707)

Server Netscape-Enterprise/3.5.1

This object doesn't have any explicit freshness information set, so a cache may use Last-Modified to determine how fresh it is with an adaptive TTL (at this time, it could be, depending on the adaptive percent used, considered fresh for: 29 min 19 sec (20%), 1 hr 13 min (50%), 2 hr 26 min (100%)). It can be validated with Last-Modified.

# 3. Cache web

---

- Instalación de proxy-cache
  - Máquinas exclusivas
    - ◆ NetCache, Cisco, Inktomi, Novell, DynaCache
  - Software máquinas propósito general
    - ◆ Cooperante, escalable, caídas *elegantes*
    - ◆ Squid: evolución de libre distribución de Harvest.
    - ◆ NetApp evolución comercial de Harvest, corre tanto en Unix como en Windows NT.
    - ◆ Netscape
    - ◆ Microsoft
    - ◆ Apache: módulo de proxy-cache

### 3. Cache web

---

- Squid

- Libre distribución
  - ◆ Gratuito + código fuente
- Rendimiento
- ICP (jerarquía)
- Cache-Digest: intercambio de URLs (cada hora)
- Módulos adicionales
  - ◆ Estadísticas, administración web ...
- Utilizado por agrupaciones mundiales de caches (NLANR)
- Unix

## 4. WebDAV

---

- Web-based Distributed Authoring and Versioning
  - RFC 2518, 3253
- Extensión de HTTP que permite la edición y gestión colaborativa de ficheros en servidores web remotos

## 4. WebDAV

---

### ■ Objetivos

- Edición documentos en el propio URL
- Control de acceso
- Edición por parte de múltiples autores
  - ◆ Infraestructura para edición en grupo asíncrona
  - ◆ Prevención edición simultánea (cerrojos)
- Gestión de versiones de ficheros en servidores web remotos
- Interfaz común para varios tipos de repositorios

## 4. WebDAV

---

- Web = sistema de ficheros distribuido de grano grueso
- Primer navegador (1990): lectura /escritura
- NCSA Mosaic: sólo lectura
- Primeras ediciones web remotas: 1995
  - Navisoft's NaviPress (AOL), Vermeer Technologies FrontPage
  - Extensiones HTTP no compatibles
- W3C – 1997 (Microsoft, Netscape, Novell, Xerox, IBM, ...)
  - Editor-navegador web: Amaya
  - Servidor web: Jigsaw
- 2002
  - Módulo Apache (mod\_dav), Adobe FrameMaker

## 4. WebDAV

---

- Envío de datos por HTTP
  - URL *munging*
    - ◆ Añade orden tras el URL
    - ◆ `http://www.misrecursos.com/indice.html?comando=check`
    - ◆ Desventajas
  - HTTP-via-POST
    - ◆ RPC: ejecución de código en servidor remoto
    - ◆ Envío de función y parámetros al servidor en el cuerpo de la petición HTTP
    - ◆ Problemas de seguridad
  - Válidos para cantidades pequeñas de info
  - Nuevos métodos HTTP para nuevas operaciones

## 4. WebDAV

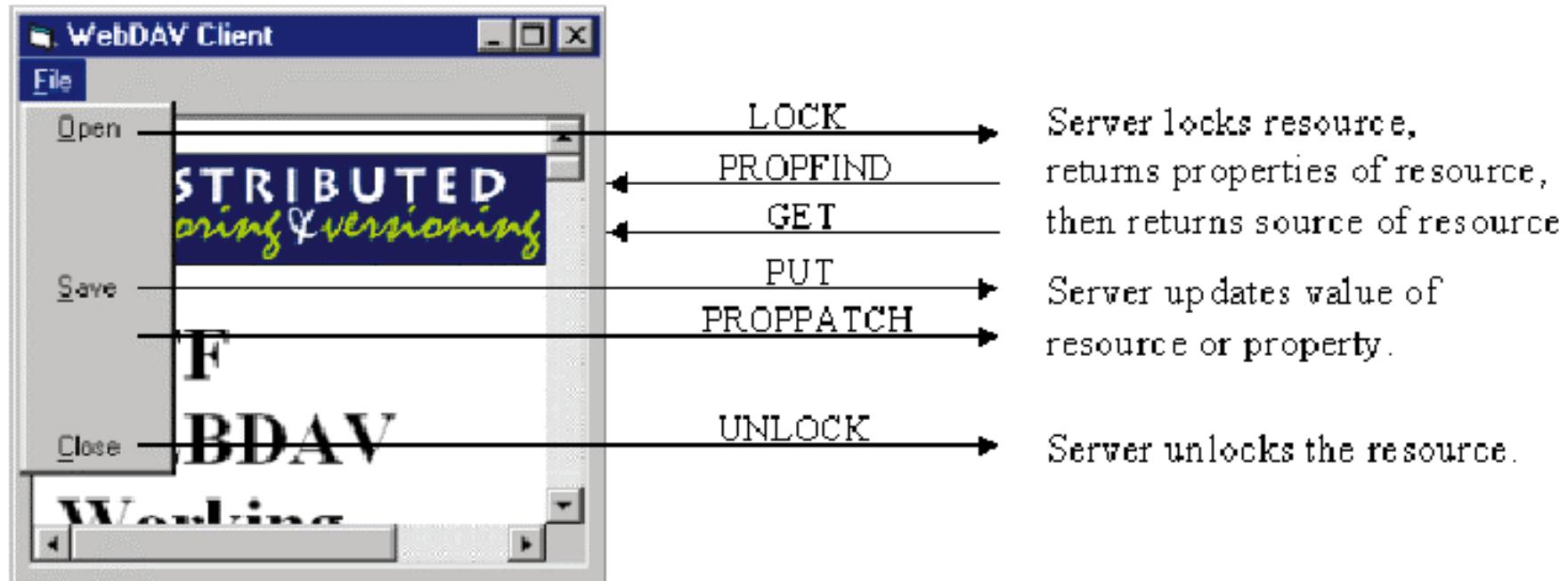
---

- Extensiones WebDAV implementadas (oct-98)
- Comandos
  - PROPFIND: obtención propiedades
  - PROPPATCH: actualización de propiedades
  - MKCOL: creación nuevas colecciones
  - COPY / MOVE: copiar / mover recursos en un nombre de espacio
  - LOCK: cerrojo en un recurso
- Cabeceras http adicionales
- Elementos XML para las propiedades

# 4. WebDAV

---

## ■ Cliente WebDAV



## 4. WebDAV

---

- Extensiones WebDAV futuras
  - Gestión de versiones
    - ◆ Almacenamiento de revisiones
    - ◆ Edición en paralelo
  - Colecciones avanzadas
    - ◆ Referencias
    - ◆ Ordenaciones del cliente
  - Control de acceso
    - ◆ Derechos de acceso
    - ◆ HTTP Digest Authentication (HTTP v1.1)
  - Búsqueda y localización (DASL) de recursos según metadatos y contenido textual

## 4. WebDAV

---

- Colecciones
  - Recursos no aislados
  - Recursos locales y referenciados
- Prevención de sobreescrituras: cerrojo
  - Exclusivo
  - Compartido
  - Sobre un recurso o una jerarquía de recursos (colección)
  - Notificaciones liberación

## 4. WebDAV

---

### ■ Propiedades o metadatos

- Información asociada a una información
- Describen documentos
  - ◆ autor, fecha modificación, tamaño ...
- Útiles en búsquedas
- Nombre (URL) / Valor (secuencia XML)
- Internacionalización
  - ◆ ISO10646, UTF-8, UTF-16
- Gestión metainfo: almacenamiento, petición, obtención, borrado,  
... Propiedades
  - ◆ Neutral: no propone esquema, depende de la comunidad de usuarios
- DASL: búsquedas directamente en el servidor

## 4. WebDAV

---

- Gestión de nombres de espacio
  - Describe la agrupación lógica de recursos y su alcance
  - Directorio y subdirectorios
  - Copiar, mover, listar recursos de la colección
  
- Aplicaciones
  - Edición remota sitios web
  - Entorno de trabajo colaborativo

## 4. WebDAV

---

- Noviembre 98, WebDAV v10: RFC2518
  - Cerrojos, propiedades, gestión de nombres de espacios
  - Microsoft Information Server 5
  - Apache: mod\_dav
  - Servidores y clientes WebDAV
  - Microsoft Internet Explorer 5 y Office 2000 (Web Folders)
- Extensiones futuras
  - Control de acceso, versiones, colecciones avanzadas

# Bibliografía

---

## ■ Introducción

- Historia web: Tim Berners-Lee
  - ◆ <<http://www.w3.org/People/Berners-Lee/>>

## ■ HTTP

- <http://www.w3.org/Protocols/>
- V1.0: RFC1945, V1.1: RFC2616
- *Illustrated Guide to HTTP*, P. Hethmon
- Tutorial HTTP
  - ◆ <<http://www.jmarshall.com/easy/http/>>

# Bibliografía

---

## ■ Cache web

- *Illustrated Guide to HTTP*, P. Hethmon
- RFC 2616
- <<http://www.rediris.es/si/cache/>>
- <<http://www.unizar.es/ccuz/servicios/proxy/inicio.html>>
- Tutorial sobre caches para autores web y webmasters
  - ◆ <[http://www.mnot.net/cache\\_docs](http://www.mnot.net/cache_docs)>

## ■ WebDAV

- <<http://www.webdav.org>>
- WebDAV: IETF Standard for Collaborative Authoring on the Web,  
E. James Whitehead, Jr., Meredith Wiggins
- WebDAV: A Panacea for Collaborative Authoring?, David  
Sussman