Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

Departamento de
Informática e Ingeniería
de Sistemas
Universidad Zaragoza

Tema 4: Servicio web

Garantía y Seguridad (30242)

4º curso, Grado en Ingeniería Informática

especialidad en

Ingeniería de Computadores

J. Alastruey, P. Ibáñez, V. Viñals
Área de Arquitectura y Tecnología de Computadores (ATC)

Guión del tema

- Introducción
- Protocolo HTTP
- Web dinámica
- Cache web
- 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
  - Compartir información
- 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

Protocolo HTTP

Protocolo

Versiones

Formato mensajes

Peticiones

Respuestas

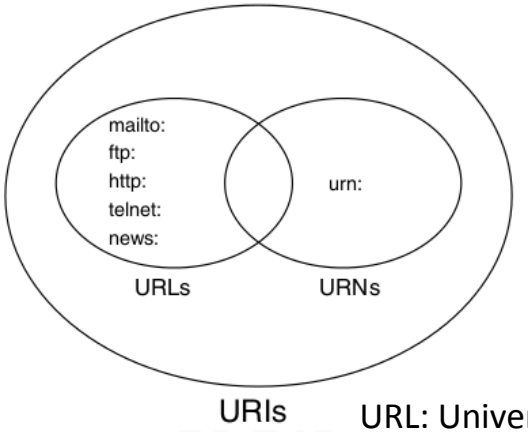
HTTPS

2. HTTP

- Protocolo de Transferencia de Hipertexto
  - RFCs 1945, 2817
  - Entrega **recursos** identificados por **URI**
- Simplifica y toma ideas de SMTP, NNTP, FTP, Gopher
- **Métodos**: órdenes de petición o envío de **recursos**
  - Cliente a servidor
- Recursos: ficheros (HTML, imágenes, resultados de consultas, salida CGI...)
- Protocolo **sin estado**
  - Cada orden HTTP se ejecuta independientemente de las anteriores
  - Dificultad sitios web “inteligentes”
    - Java, JavaScript, cookies, ActiveX ...
- Formato de mensajes: usa MIME

2. Identificación de recursos: URIs

- Recursos especificados por ***Uniform Resource Identifiers*** (URIs)
  - http://www.unizar.es/index.html
  - RFC 1630



URL: Universal Resource Locator

URN: Universal Resource Name

- ***URL encoding***
  - RFC 1738
  - "Octets must be encoded if they have no corresponding graphic character within the US-ASCII coded character set, if the use of the corresponding character is unsafe, or if the corresponding character is reserved for some other interpretation within the particular URL scheme"

Computer Architecture, A Quantitative Approach.pdf

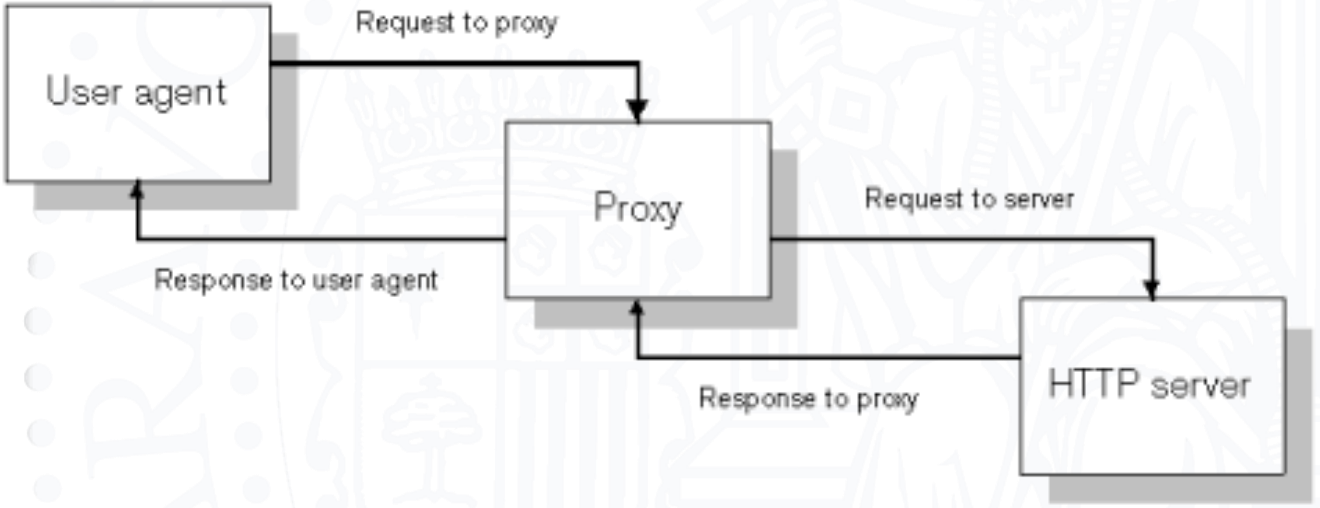
⇩

Computer%20Architecture%2C%20A%20Quantitative%20Approach.pdf

2. HTTP

- Paradigma cliente-servidor

- Comunicación
  - Directa
  - Proxy



2. Proxy 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
- Tipos
  - Transparente: el cliente no es consciente del proxy
  - No transparente

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 datos de cliente a servidor
  - Cada petición $\Rightarrow$ conexión
    - *3-way-handshake* + *slow-start* reducen velocidad

2. HTTP. Versiones

- HTTP/1.1 (RFC 2616)
  - Superconjunto v1.0
  - **Conexiones persistentes**: varias peticiones/conexión
    - 20% más prestaciones
  - Servicio de múltiples dominios desde una sola @IP (***virtual hosting***)
  - Nuevos métodos petición: PUT, DELETE, OPTIONS, TRACE
  - **Peticiones condicionales**
  - Mayor soporte **caches**: latencia y BW
  - Autentificación *digest*: MD5 ...
  - Codificación por trozos –*chunked*-
    - Mayor rapidez en servicio de páginas dinámicas
  - ...

2. HTTP. Versiones

- HTTP/2 (RFC 7540)
  - Compresión de cabeceras

2. Transacción HTTP

- Esquema

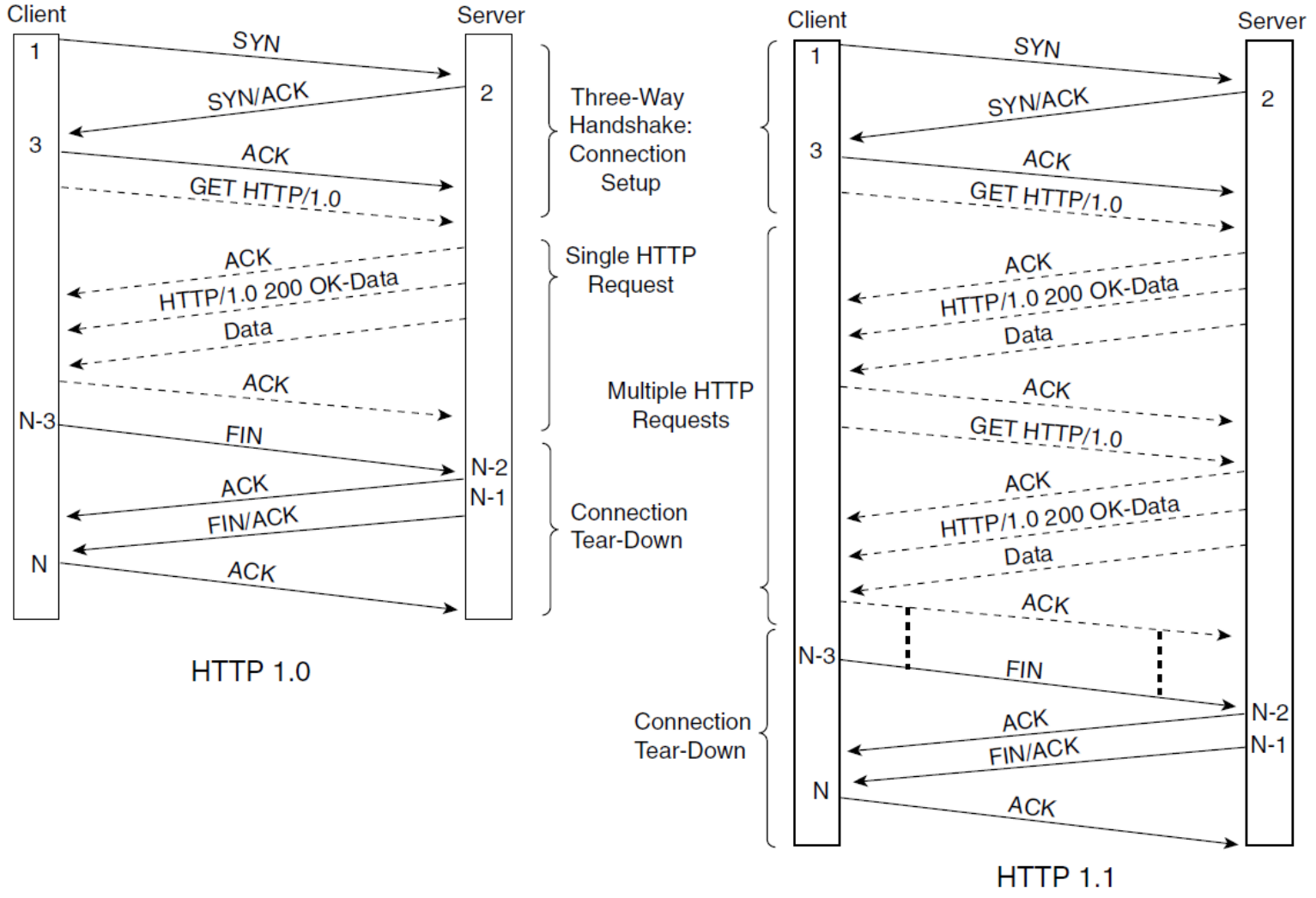


Figura: Data Center Fundamentals. Cisco Press.

2. Transacción HTTP

- HTTP/1.1 *connection pipelining*

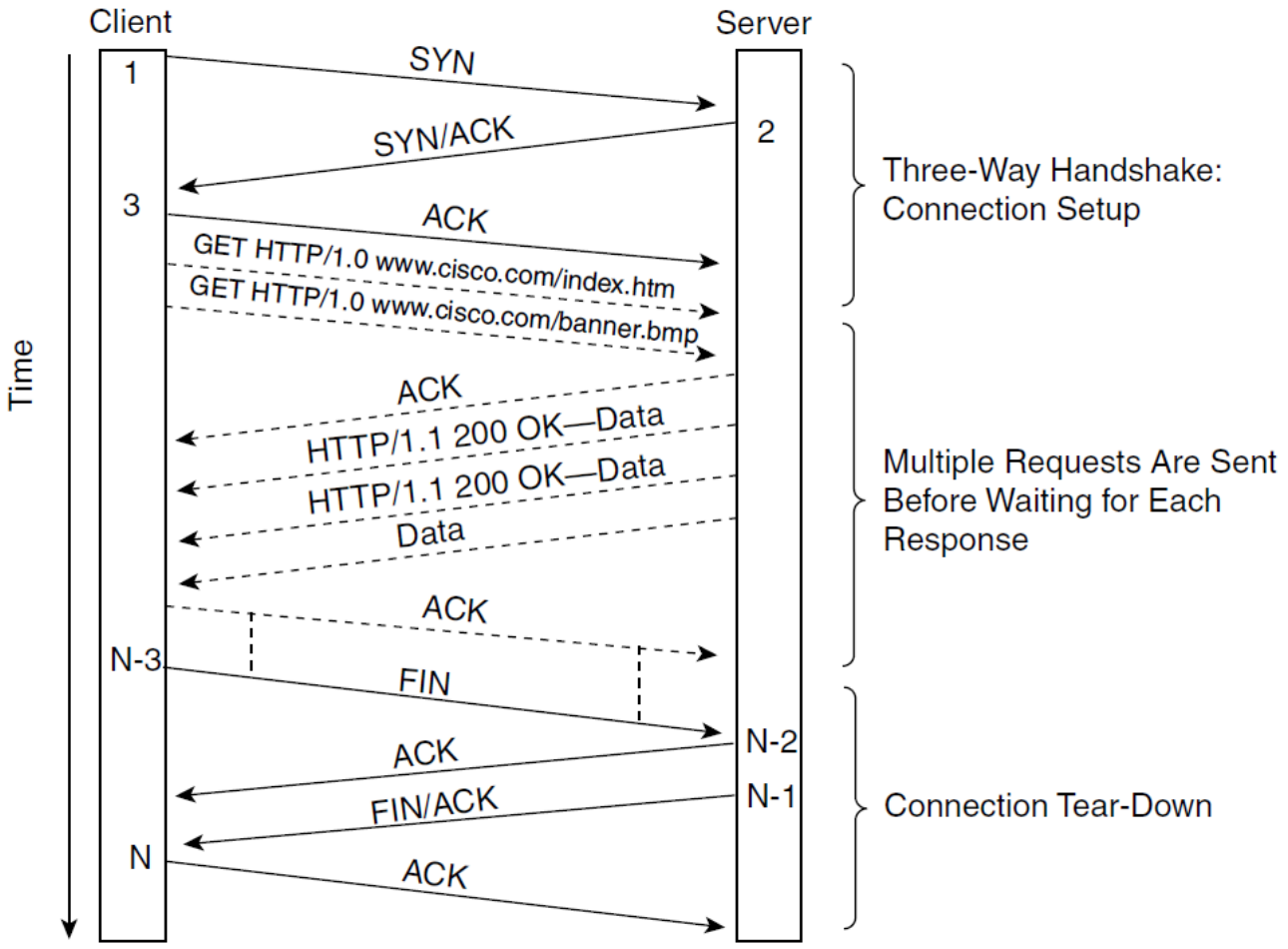


Figura: Data Center Fundamentals. Cisco Press.

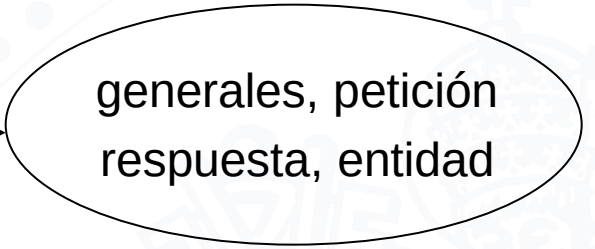
2. Virtual Hosting

- Gestión de múltiples dominios por parte de un **único servidor** HTTP **en una máquina**
  - `www.dominio1.com`
  - `www.dominio2.com`
- Implementación
  - Múltiples direcciones IP: *IP-based virtual web hosting*
  - Múltiples puertos: *port-based virtual hosting*
  - Cabecera HTTP `Host`: *name-based virtual hosting*
- Apache
  - Los procesos hijos del demonio http escuchan múltiples sockets, cuando una nueva conexión llega, solamente uno la gestiona

2. Formato mensajes HTTP

- Mensaje genérico RFC 822

```
<línea inicial, distinta para peticiones y respuestas>
Cabecera1: valor1 (opcionales)
Cabecera2: valor2
Cabecera3: valor3
Cabecera4: valor4
                          // línea en blanco (CRLF)
<cuerpo del mensaje (opcional), por ejemplo
   contenido de un fichero o datos de consultas;
   pueden ser líneas de texto o datos binarios
   $&*%@!^$@>
```



2. Tipos de cabeceras HTTP

- General: conexión
  - Connection, Date, Pragma, Cache-Control, Transfer-Encoding
- Petición: info sobre el recurso solicitado o el cliente
  - If-Modified-Since, Accept-Language, User-Agent
- Respuesta: info sobre el recurso, su ubicación real o el servidor
  - Location, Server
- Entidad: descripción atributos cuerpo del mensaje
  - Content-Length, Content-Type

- HTTP 1.0: 16 cabeceras
  - Opcionales
- HTTP 1.1: 46
  - `Host` requerida en las peticiones

2. Cuerpo mensaje HTTP

- 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
- Descrito 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. 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. 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: OPTIONS /cgi-bin/order HTTP/1.1
S: 200 Ok
S: Allow: POST
S: Accept-Encoding:
```

2. Métodos peticiones HTTP

- **GET**: petición recurso especificado por URI
  - 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. Métodos peticiones HTTP

- **HEAD**: petición cabeceras de URI
  - Implementación obligatoria

```
C: HEAD /index.html HTTP/1.1
S: (cabeceras respuesta index.html)
```

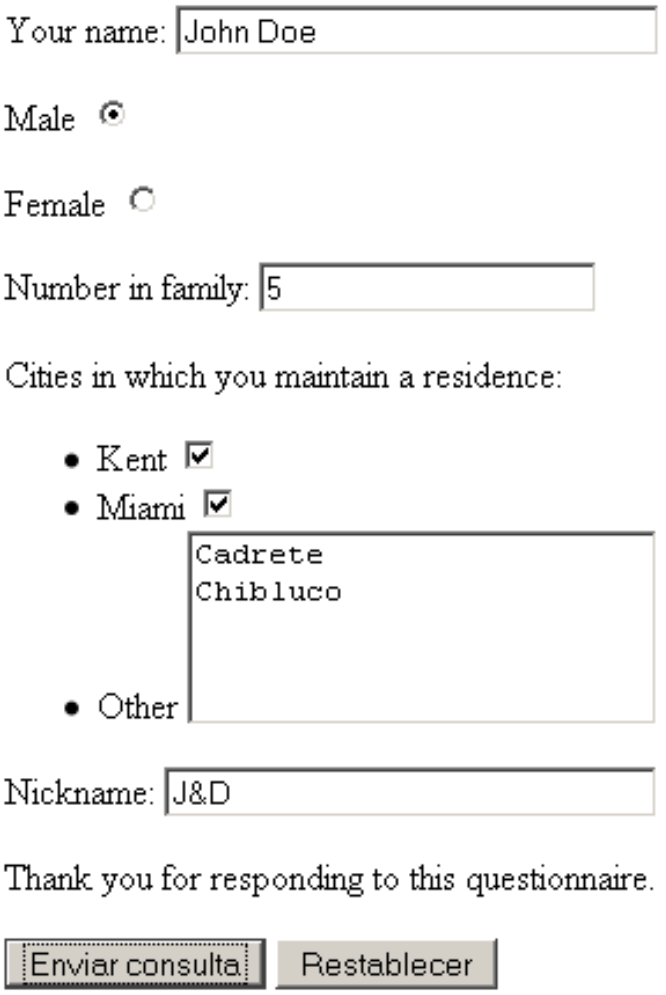
2. Métodos peticiones HTTP

- **POST**: envío datos al servidor en el cuerpo del mensaje
  - URI: programa procesado datos enviados (Perl, C ...)
  - Datos codificados URL
    - Fichero o formulario
  - 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]
```

2. GET vs.POST: Formulario

- Ejercicio




```
<FORM METHOD="POST" ACTION="http://www.w3.org/sample">
<P>Your name: <INPUT NAME="name" size="48">
<P>Male <INPUT NAME="gender" TYPE=RADIO VALUE="male">
<P>Female <INPUT NAME="gender" TYPE=RADIO VALUE="female">
<P>Number in family: <INPUT NAME="family" TYPE=text>
<P>Cities in which you maintain a residence:
<UL>
<LI>Kent <INPUT NAME="city" TYPE=checkbox VALUE="kent">
<LI>Miami <INPUT NAME="city" TYPE=checkbox VALUE="miami">
<LI>Other <TEXTAREA NAME="other" cols=48 rows=4></textarea>
</UL>
Nickname: <INPUT NAME="nickname" SIZE="42">
<P>Thank you for responding to this questionnaire.
<P><INPUT TYPE=SUBMIT> <INPUT TYPE=RESET>
```

```
name=John+Doe&gender=male&family=5&
city=kent&city=miami&
other=Cadrete%0D%0AChibluco&nickname=J%26D
```

Referencia: http://www.w3.org/MarkUp/html-spec/html-spec_8.html#SEC8.2.1

2. GET vs.POST

- Escenario 1: página web fabricante, búsqueda de modelo de producto
- Escenario 2: página para acceso a información bancaria

- **GET**
  - Interacción tipo lectura, consulta datos (“idempotente”)
  - Permite compartir URLs
- **POST**
  - Interacción tipo orden, almacenamiento o actualización de datos
  - Interacción cambia el estado del recurso (por ejemplo, suscripción a un servicio)
  - Permite cifrar los datos enviados

Referencia: http://www.w3.org/2001/tag/doc/whenToUseGet.html

2. Métodos peticiones HTTP

- **PUT**: almacena cuerpo petición en el URI indicado

```
C: PUT /users/phethmon/welcome.html HTTP/1.1
   Content-Type: text/html
   Content-Length: 3109
   [CR + 3109 bytes de datos ]
S: HTTP/1.1 204 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
   [CR + 4526 bytes de datos ]
S: HTTP/1.1 501 Not Implemented
   Server: 3wd/1.1
```

2. Métodos peticiones HTTP

- **DELETE**: elimina URI indicado
  - Seguridad, autentificación

```
C: DELETE /catalog/sales/oct96.html HTTP/1.1
S: HTTP/1.1 204 No Content

C: DELETE /company/about.html HTTP/1.1
S: HTTP/1.1 202 Accepted Pending Approval
```

2. Métodos peticiones HTTP

- **TRACE**: petición cabeceras recibidas por el servidor

```
C: TRACE / HTTP/1.1
   Host: www.unizar.es
   Connection: close

S: HTTP/1.1 200 OK
   Date: Mon, 24 May 2010, 09:23:20 GMT
   Server: Apache
   Connection: close

   TRACE / HTTP/1.1
   Host: www.unizar.es
   Connection: close
```

- Vulnerabilidad
  - http://www.kb.cert.org/vuls/id/867593
  - http://www.cgisecurity.com/whitehat-mirror/WhitePaper_screen.pdf

2. Respuestas HTTP

```
Versión protocolo + código éxito/error + descripc.
Cabeceras general|respuesta|entidad

[Cuerpo mensaje]
```

- Ejemplos

```
HTTP/1.1 200 OK
Server: 3wd/1.1
Content-Type: text/html
Content-Length: 200

[200 bytes de datos]
```

```
HTTP/1.0 404 Not Found
```

2. Respuestas HTTP

- **Códigos respuesta** definidos en RFC 2616
  - **1xx**: informativo
  - **2xx**: éxito
  - **3xx**: redirección cliente a otro URL
  - **4xx**: error del cliente
  - **5xx**: error del servidor

2. HTTP 1.1. Clientes

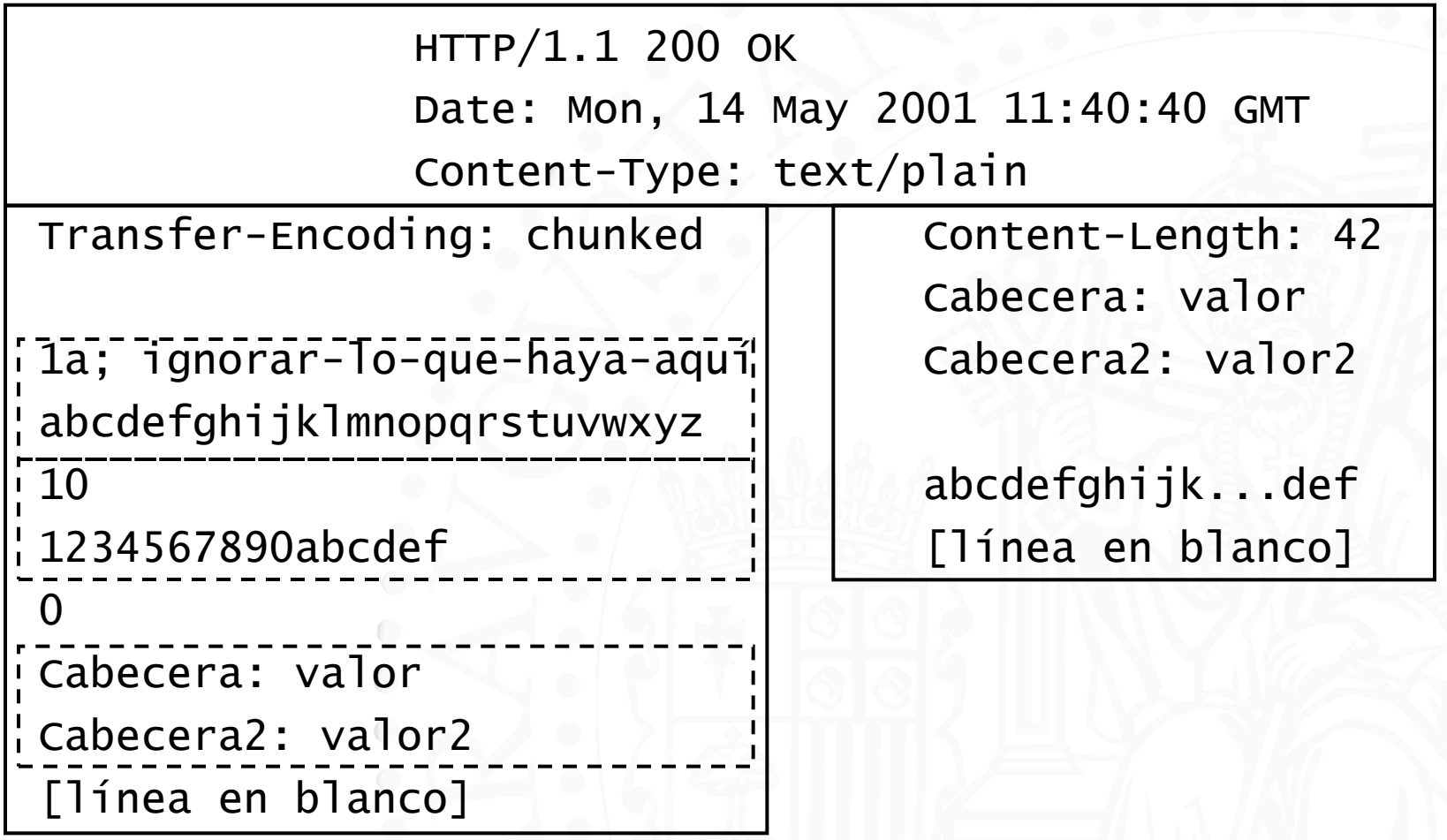
- Cabecera `Host`
  - www.kk1.com y www.kk2.com pueden estar en el mismo servidor
  - Hay que especificar destinatario petición:

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

- 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
  - http://developers.sun.com/mobility/midp/questions/chunking/

2. HTTP 1.1. Clientes

- Ejemplo codificación por trozos



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 aboluto

```
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 conexión
  - 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.
```

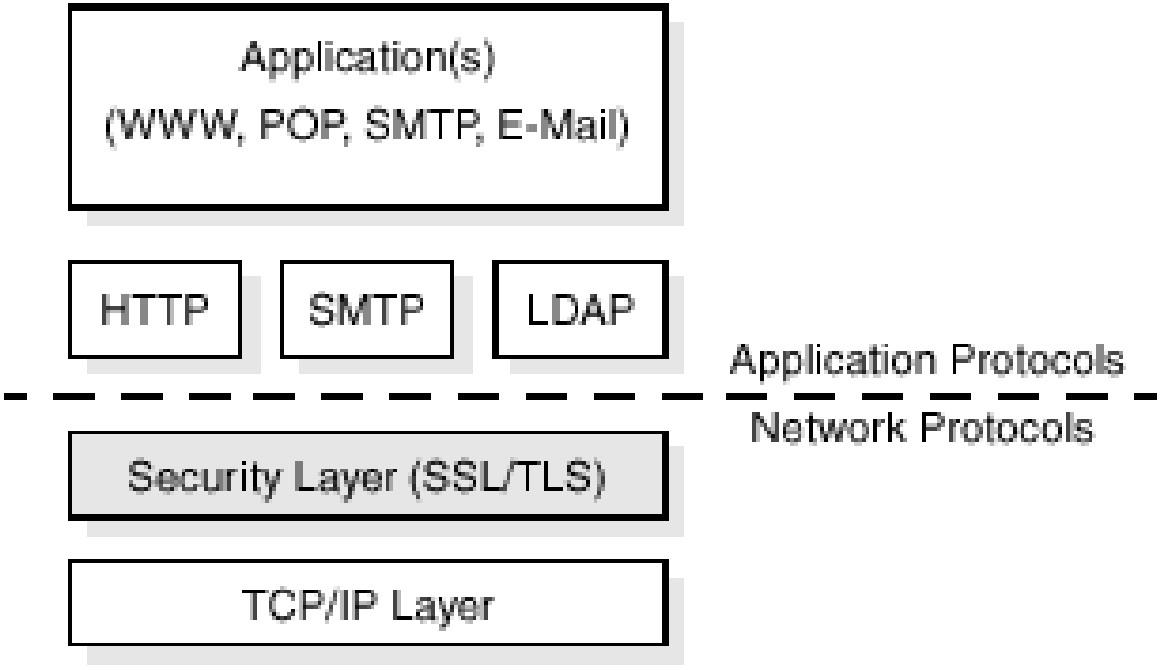
2. HTTP

- Descarga de documentos con wget

```
hendrix:~> wget diis.unizar.es

--13:03:15--  http://diis.unizar.es:80/
           => `index.html.1'
Connecting to diis.unizar.es:80... connected!
HTTP request sent, fetching headers... 1 2 3 4 5 6 7 8 9 done.
Length: 1,085 [text/html]

    OK -> .

13:03:15 (529.79K/s) - `index.html.1' saved [1085/1085]
```

2. HTTPS

- HTTPS = HTTP sobre SSL (Secure Socket Layer)
- Extensión segura de HTTP
- Cifrado y autentificación extremo a extremo
  - Tráfico no puede ser leído por intermediarios (ISPs …)
- Puerto TCP 443



2. HTTPS: autentificación

- Verificación de identidad
  - Servidor
  - Cliente (menos habitual)
- Basada en certificado digital
  - Documento electrónico que asocia una clave pública con la identidad de su propietario
  - Autofirmado (ventana emergente que provoca molestia, desconfianza, riesgo seguridad) o
    emitido por autoridad de certificación (CA, Certificate Authority)
    - VeriSign, Dirección General Policía, FNMT, Banesto, Telefónica, RedIris

2. HTTPS: certificados de servidor

- Servicio de Certificados de Servidor (SCS) de RedIris
  - Proporciona certificados a servidores
- Ventajas de este certificado:
  - CA reconocida por los principales clientes (web, correo)
    - NO hay que instalar ningún certificado en los clientes
  - Procedimiento simple para obtener los certificados
  - Gratuito para toda la comunidad de RedIRIS
- Enlaces
  - https://www.rediris.es/tcs/
  - https://sicuz.unizar.es/comunicaciones/certificados-servidor/certificados-de-servidor-inicio

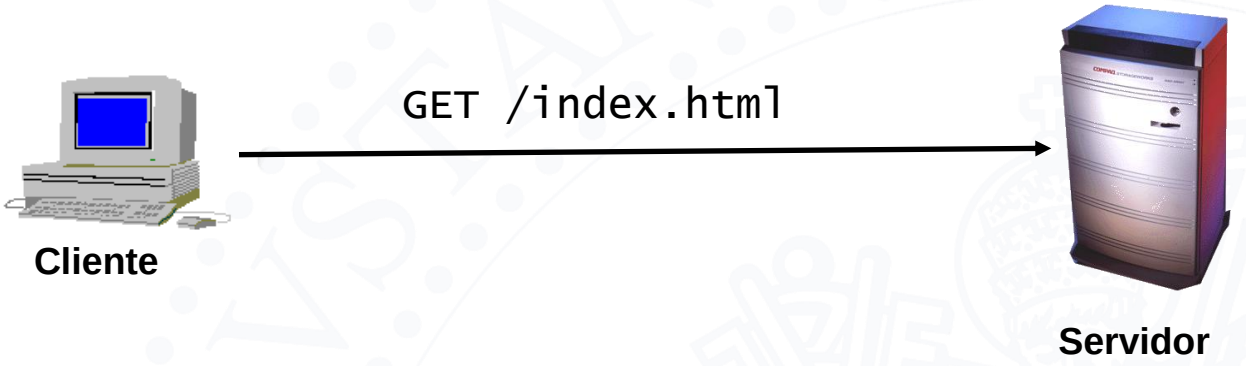
Web dinámica

Programación en el lado del cliente

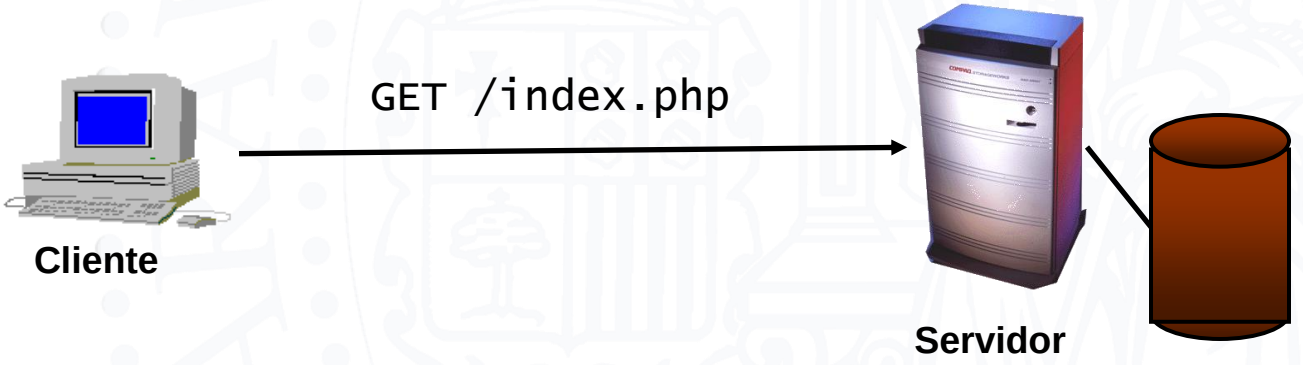
Programación en el lado del servidor

3. Web dinámica

- Programación en el lado del cliente
  - *client-side*



- Programación en el lado del servidor
  - *server-side*



3. Programación en el lado del cliente

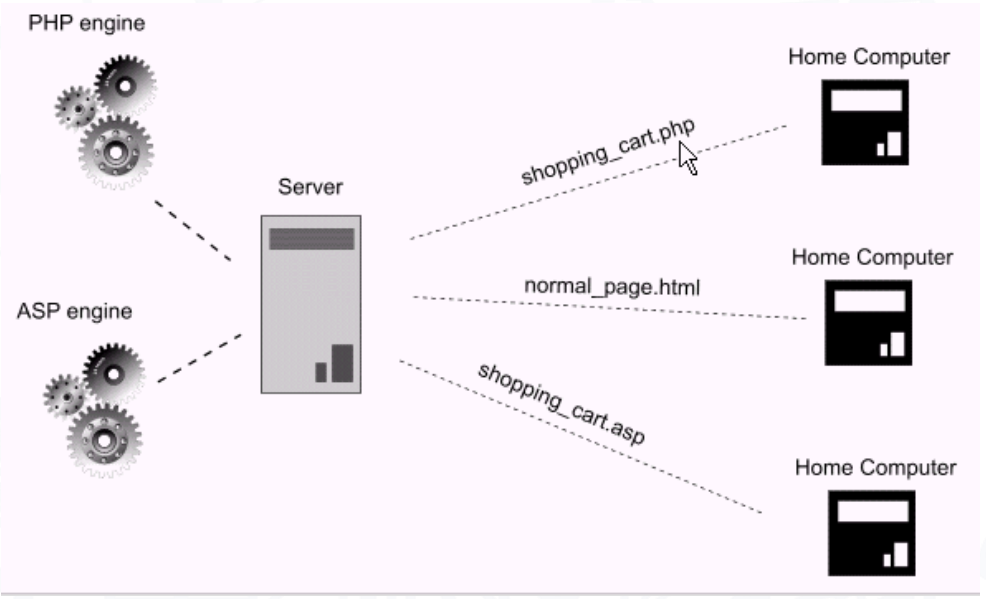
- *client-side programming*
- Ejecución de código en el cliente
  - navegador [+extensiones]
- Interacción con página web
- Pros
  - Respuesta rápida (con el código descargado)
  - Servicio relativamente seguro (no hay envío de info al servidor)
- Contras
  - Variabilidad sistema (navegadores, extensiones, SO)
  - Limitación a interacción sencilla (por ej. validación)

3. Programación en el lado del servidor

- *server-side programming*
- Ejecución de código en el servidor
- Procesado de info recibida del cliente
- Pros
  - Ejecución eficiente de procesos complejos
    - El cliente no tiene que descargar y ejecutar código y datos
  - Independencia de configuración del cliente (navegador, SO ...)
- Contras
  - Seguridad
  - Sobrecarga servidor

3. Alternativas

- Programación en el lado del cliente
  - JavaScript: ejec. en navegador
  - Applets Java: ejec. en JVM
  - ActiveX
- Programación en el lado del servidor
  - CGI
    - Shell script, C, Perl
  - PHP, ASP, .NET
  - Java Server Pages (JSP)
  - Django
  - ColdFusion
- Cliente + servidor
  - AJAX



3. Ejemplos

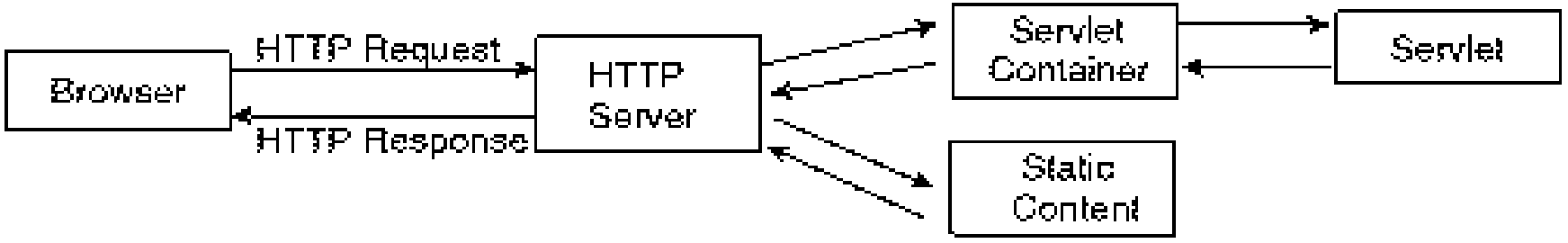
- Servlet
  - Componente web basado en tecnología Java
  - Gestionado por *container* o *servlet engine*
  - Genera contenido dinámico
  - Clases Java independientes plataforma
    - bytecodes

- Servlet container
  - Ejecuta servlets
  - Se ejecuta en un servidor de aplicaciones o dentro de un servidor web
  - Ejemplos: Tomcat, WebSphere

3. Ejemplos

- Java Servlet specification 2.4, servlet-2_4-fr-spec.pdf

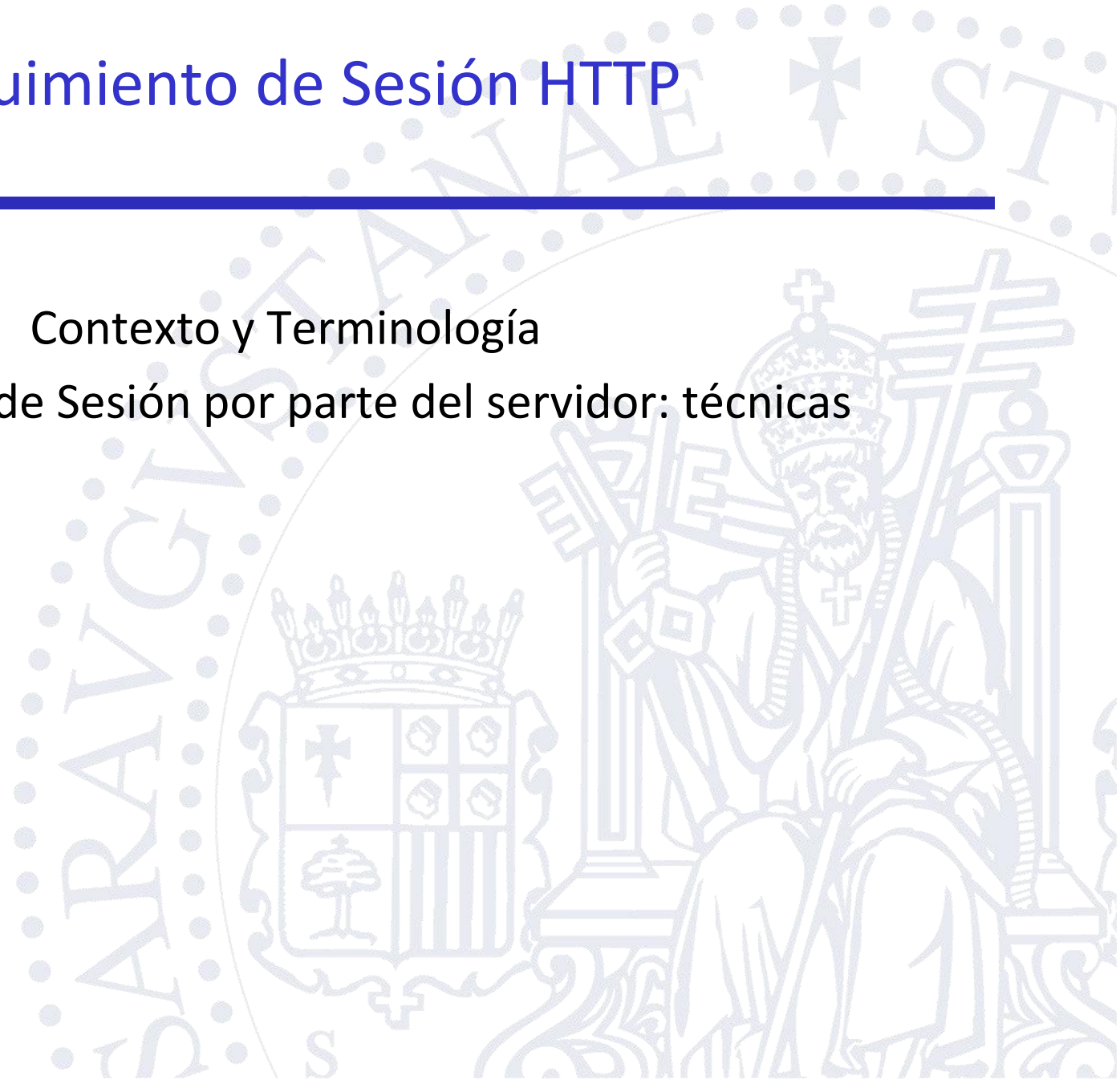
1. Cliente realiza petición HTTP a un servidor web (WS)
2. La petición es recibida por el WS y enviada para su procesamiento al servlet container
   - en la misma máquina o en otra distinta del WS
3. El servlet container determina a qué servlet debe invocar
4. El servlet se ejecuta, genera datos según los parámetros POST que han podido ser enviados en la petición y los envía al cliente
5. El servlet container devuelve el control al WS



Seguimiento de Sesión HTTP

Contexto y Terminología

Seguimiento de Sesión por parte del servidor: técnicas



4. Contexto y terminología

- Contexto
  - Cliente accede a aplicación web en un servidor a través de HTTP
    - Correo electrónico
    - Tienda
- Sesión
  - Conexión + secuencia de transacciones HTTP + desconexión
- Seguimiento de sesión por parte del servidor
  - Identificación de clientes
  - Mantenimiento de información asociada a cada cliente
- Pero … HTTP es un protocolo sin estado
  - Una petición HTTP no puede asociarse a otra previa

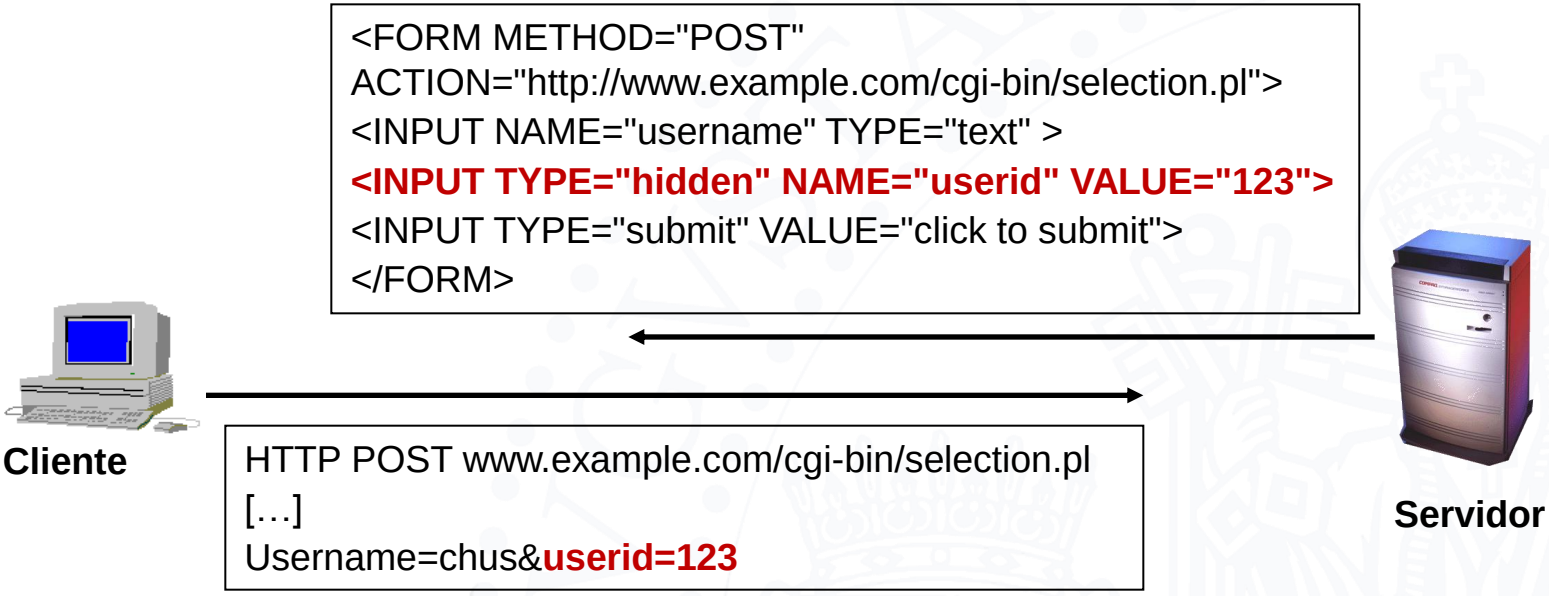
4. Seguimiento de sesión por parte del servidor

- Acciones
  - Generar identificador de sesión
  - Gestionar peticiones del cliente
  - Al cierre de sesión, eliminar datos asociados

- Distintas técnicas
  - Campos ocultos de formulario
  - Reescritura de URL
  - Cookies
  - Combinación de métodos

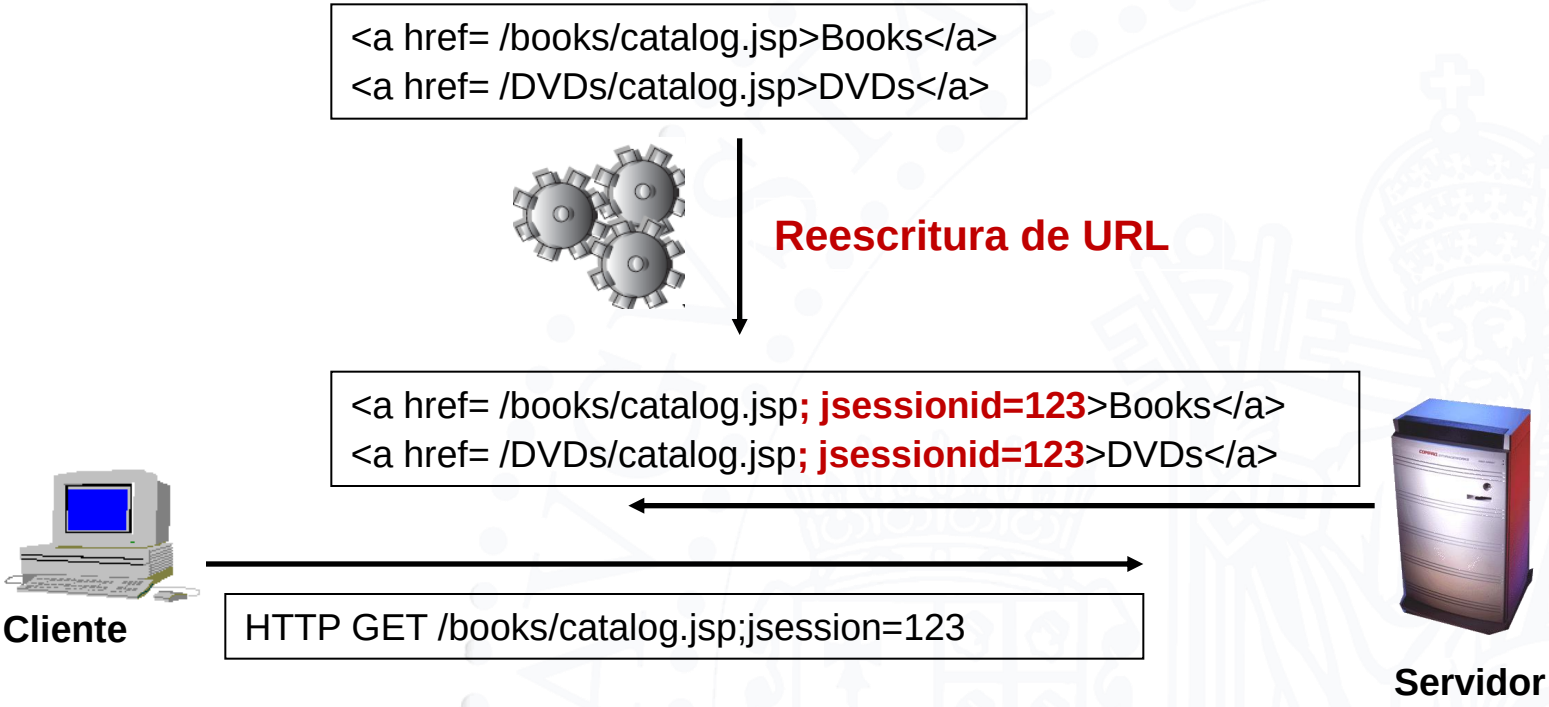
4. Campos ocultos de formularios

- Campos ocultos en formularios HTML



4. Reescritura de URL

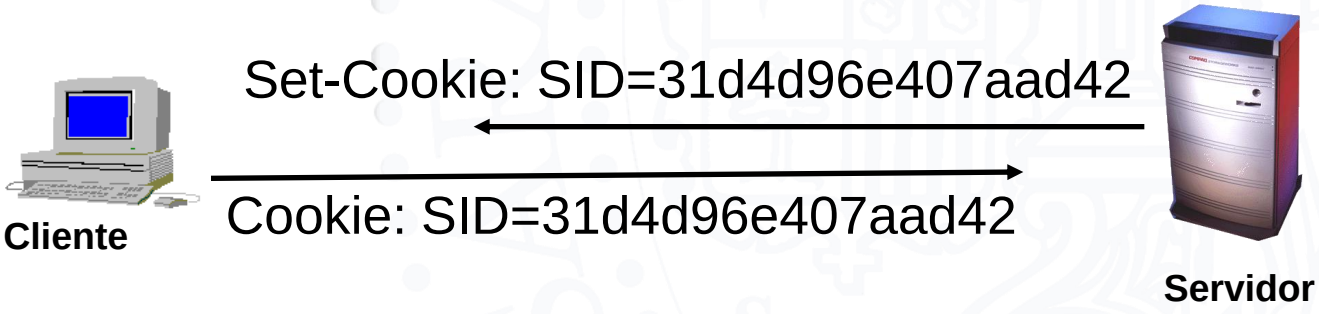
- El servidor cambia la URL en el código HTML para incluir el identificador de sesión



- Apache: mod_rewrite
  - http://httpd.apache.org/docs/2.4/mod/mod_rewrite.html

4. Cookies

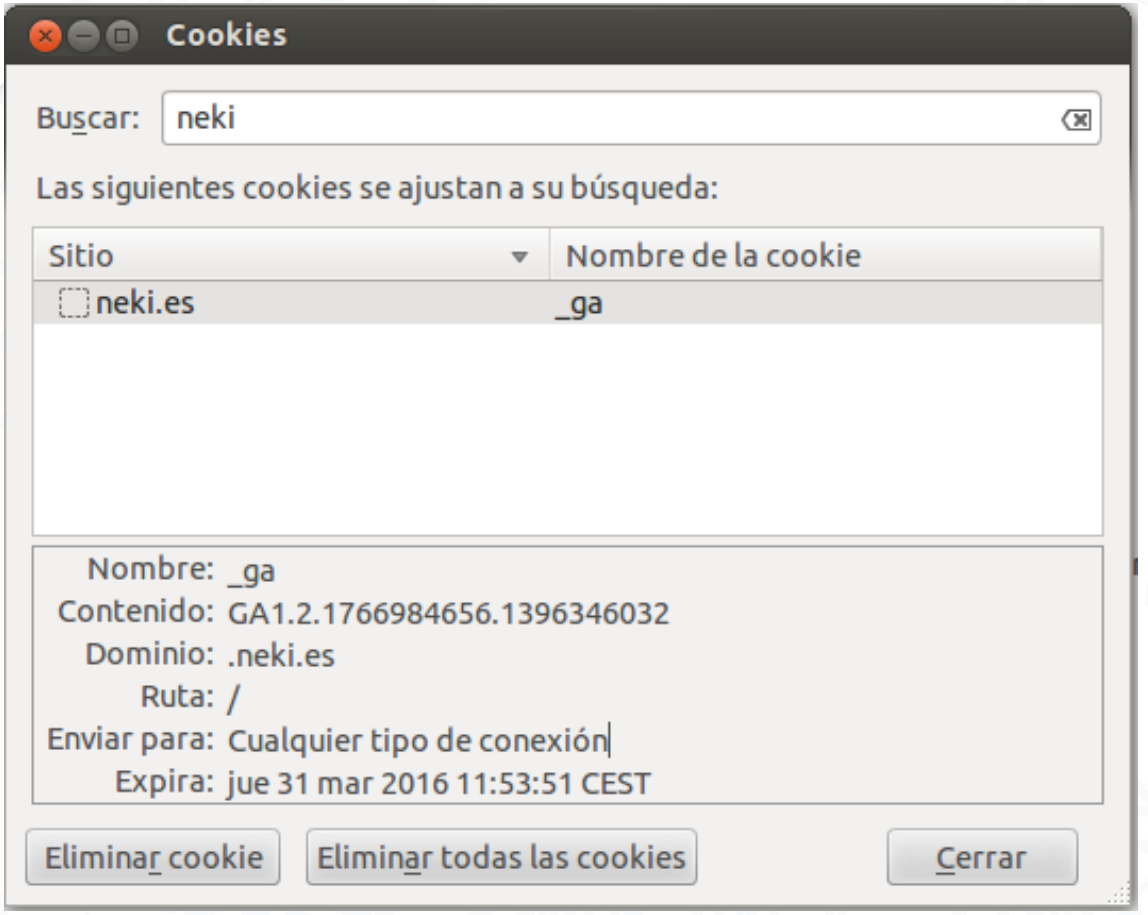
- Historia
  - Netscape cookies (1994, Lou Montulli)
    - http://curl.haxx.se/rfc/cookie_spec.html
  - HTTP State Management Mechanism
    - RFC 2109 (1997), RFC 2609 (2000), RFC 6265 (2011)
- Cookie
  - Texto que el servidor manda al cliente en una cabecera HTTP
  - Parejas nombre-valor y metadatos asociados
  - Fichero almacenado por el navegador
  - Tamaño máximo recomendado: 4096 bytes
    - http://browsercookielimits.squawky.net/



4. Formato de cookies

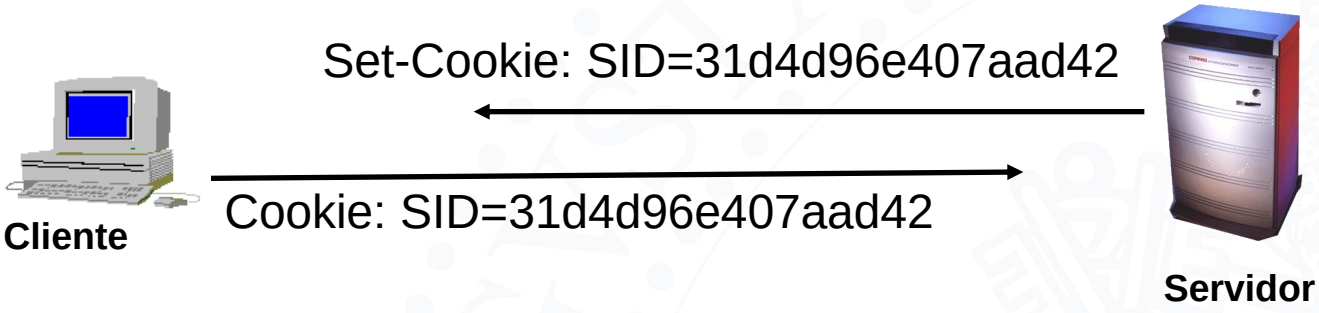
- Set-Cookie: nombre=valor; atributos

- Atributos
  - Domain
    - Amplía o acota dominio de uso de cookie
      - eina.unizar.es
  - Path
    - Acota ruta de uso
  - Expire
    - Sin expires, la cookie caduca con la sesión
  - Secure
    - Sólo si HTTPS

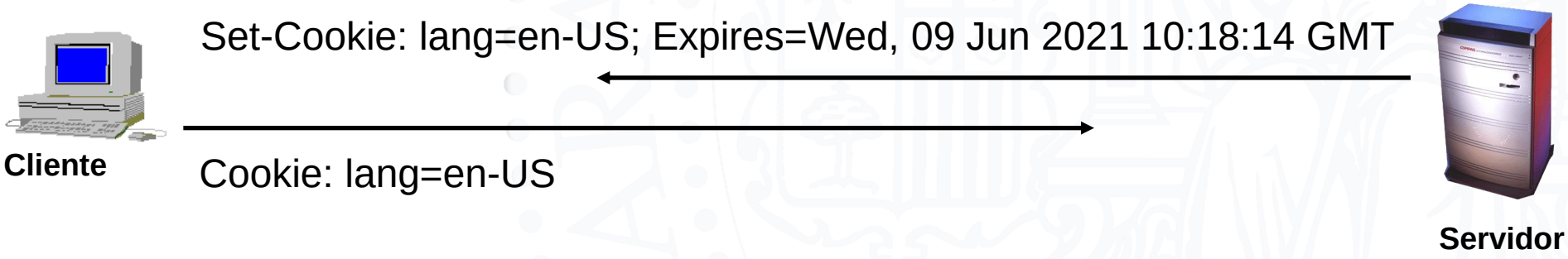


4. Cookies de sesión y cookies persistentes

- Sesión o temporal
  - Asociadas a sesión entre cliente y servidor
    - Webmail, tienda

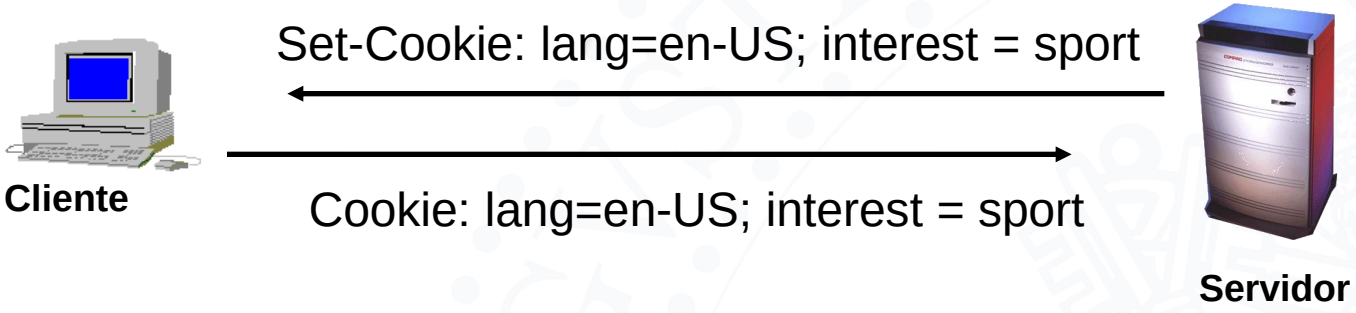


- Persistente
  - Almacenamiento de preferencias

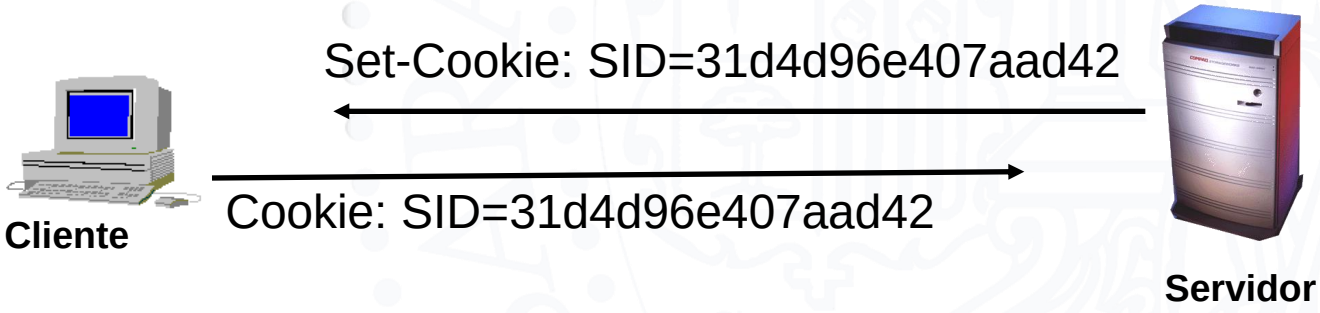


4. Cookies

- Dos formas de uso
  - Almacenar la información del cliente en la cookie
    - Generalmente desaconsejado por pesado para el navegador y por seguridad

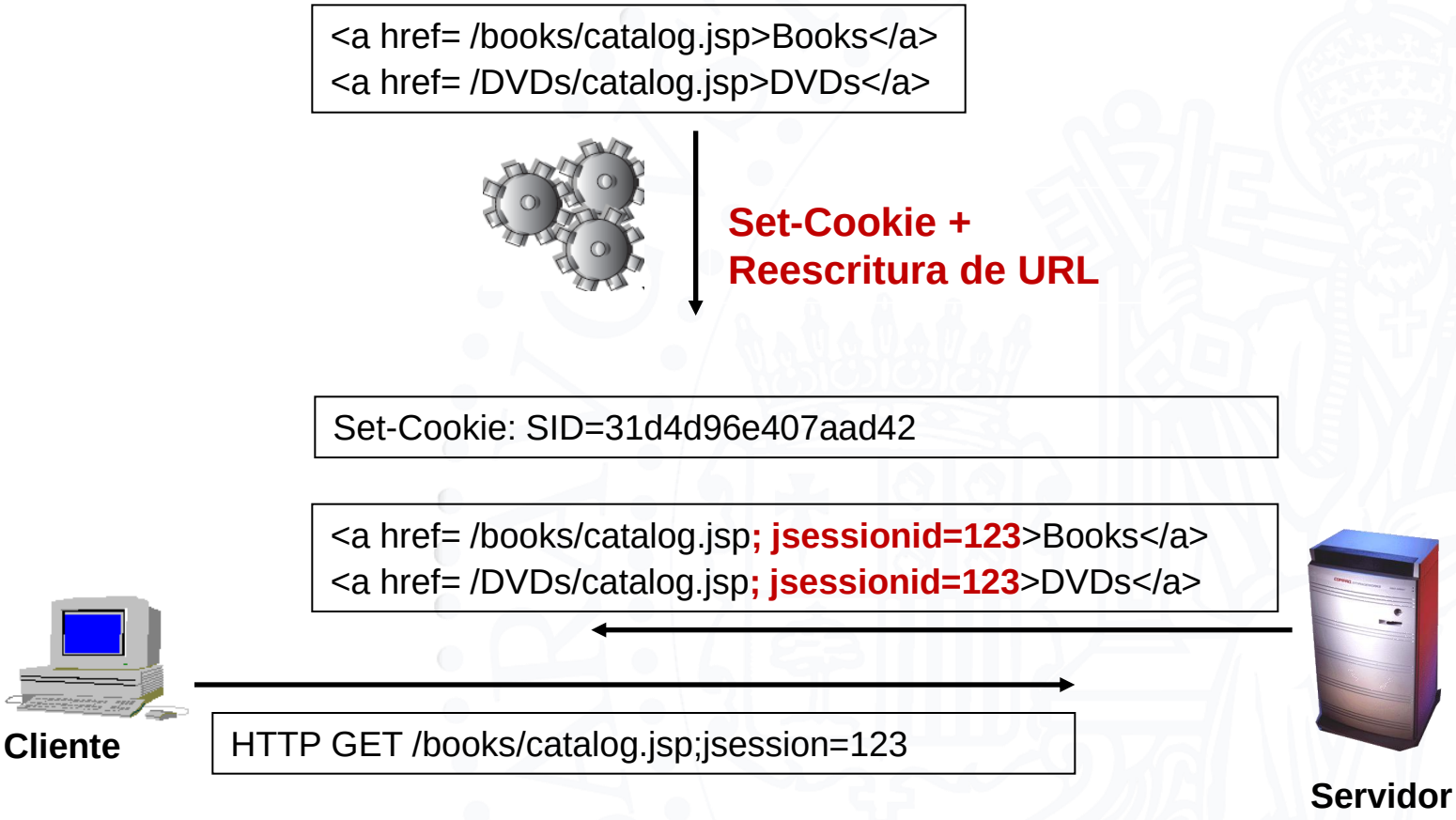


  - Almacenar la información en el servidor, cookie: puntero
    - El método más habitual



4. Combinación de métodos

- Combinación típica: cookies + reescritura de URLs
  - Por defecto cookies
  - Reescritura de URLs si el cliente tiene desactivadas cookies



Cache web

Proxy cache

Proxy cache inverso

Redes de distribución de contenidos

Agrupación de caches

Control de caches

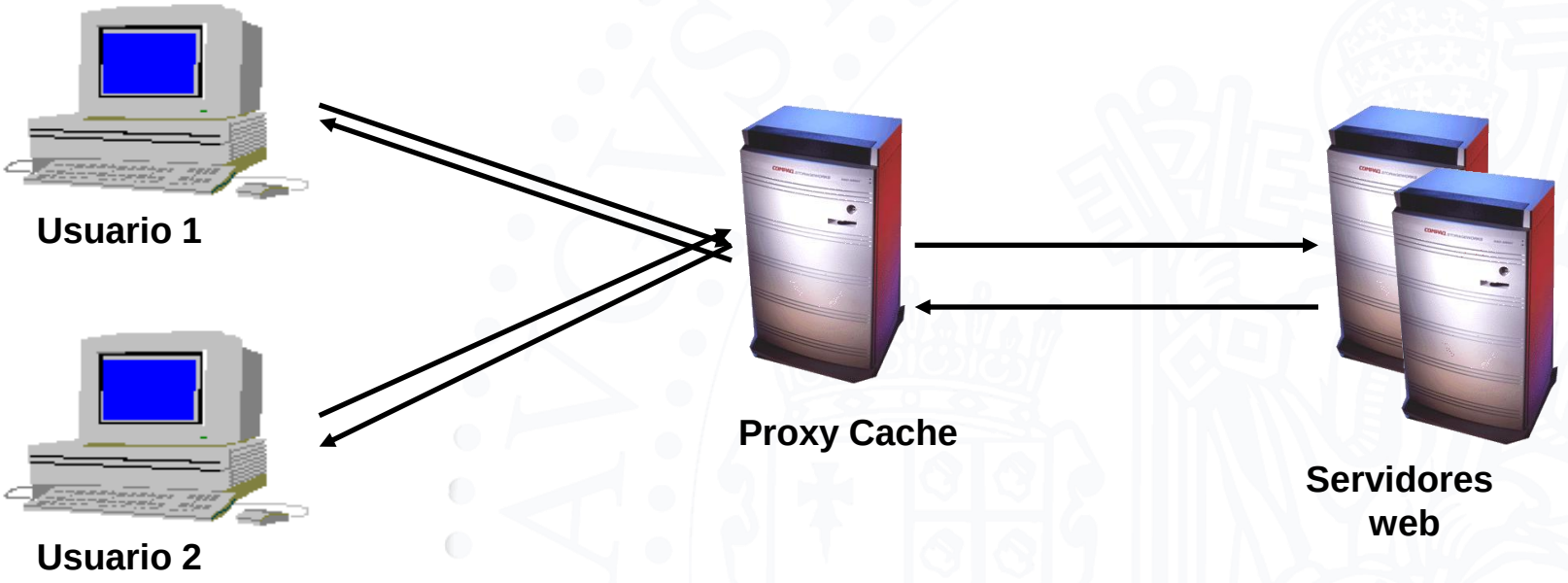
Squid

5. 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)

5. Cache web

- Funcionamiento

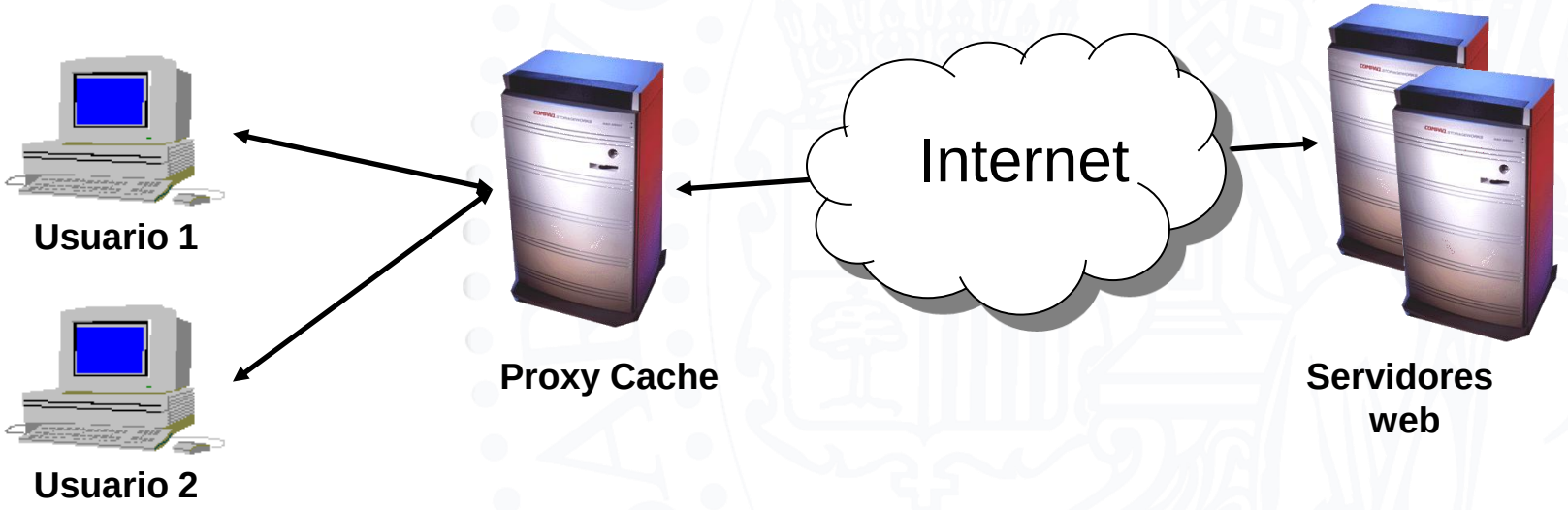


5. Tipos de cache web

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

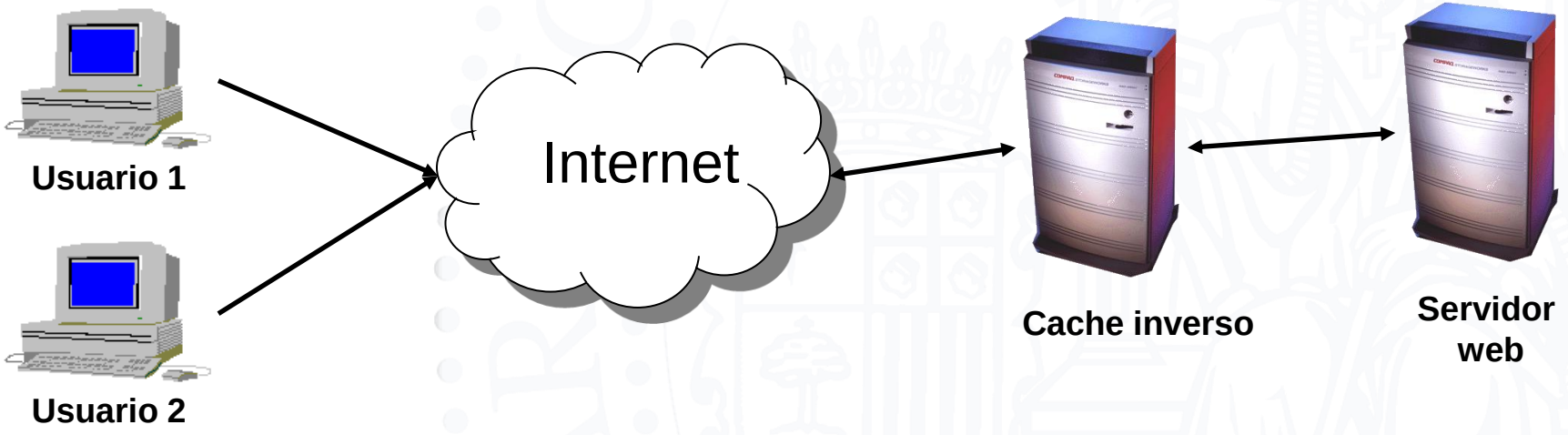
5. Tipos de cache web

- 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
  - Otras funciones: autenticación, filtrado de contenido



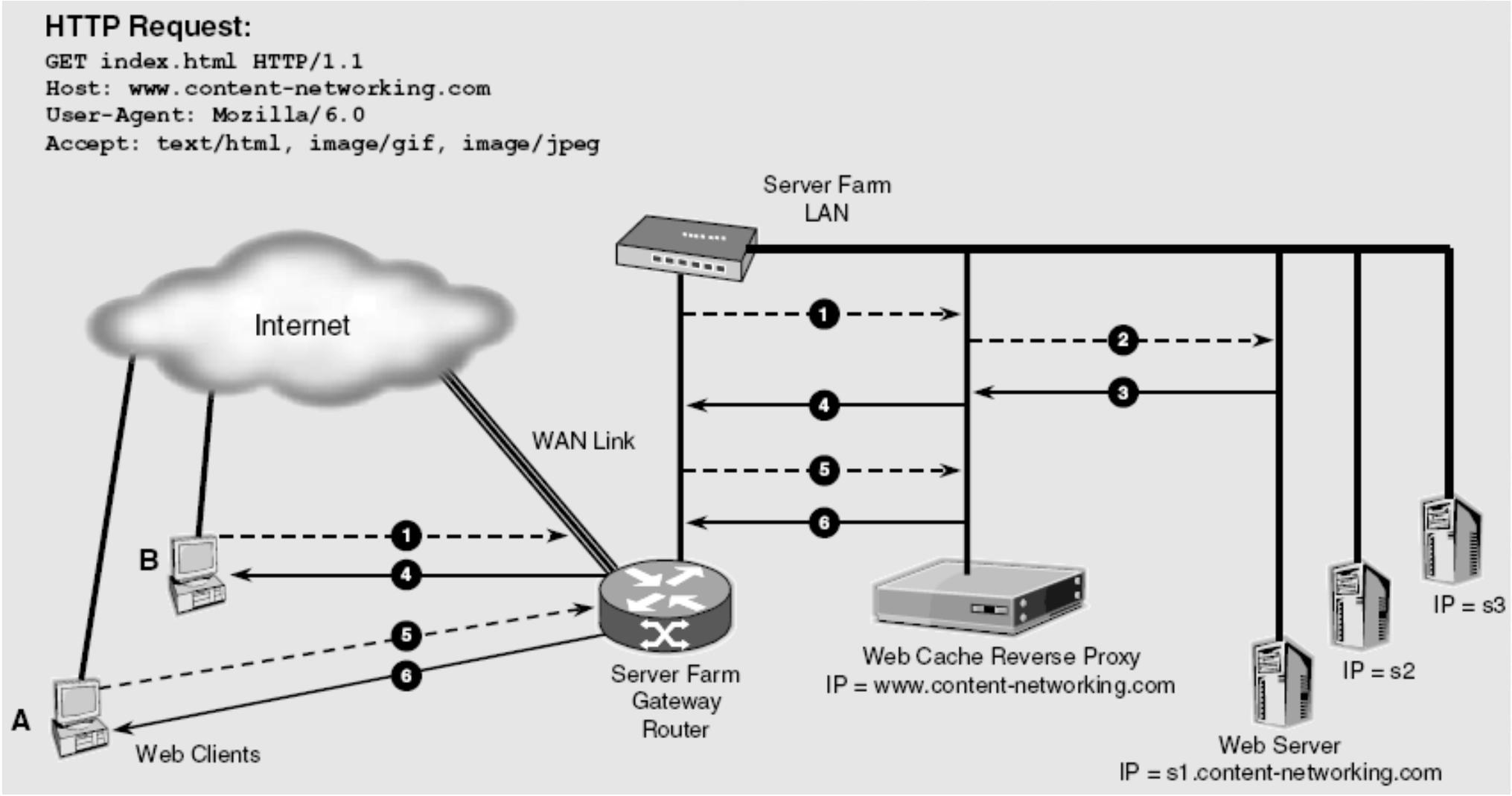
5. Tipos de cache web

- Proxy cache inverso (*reverse proxy, gateway)*
  - Entrega de contenidos más rápida (estáticos)
  - Disminución carga servidores
  - Respaldo de un servidor
  - Un servidor, varios usuarios



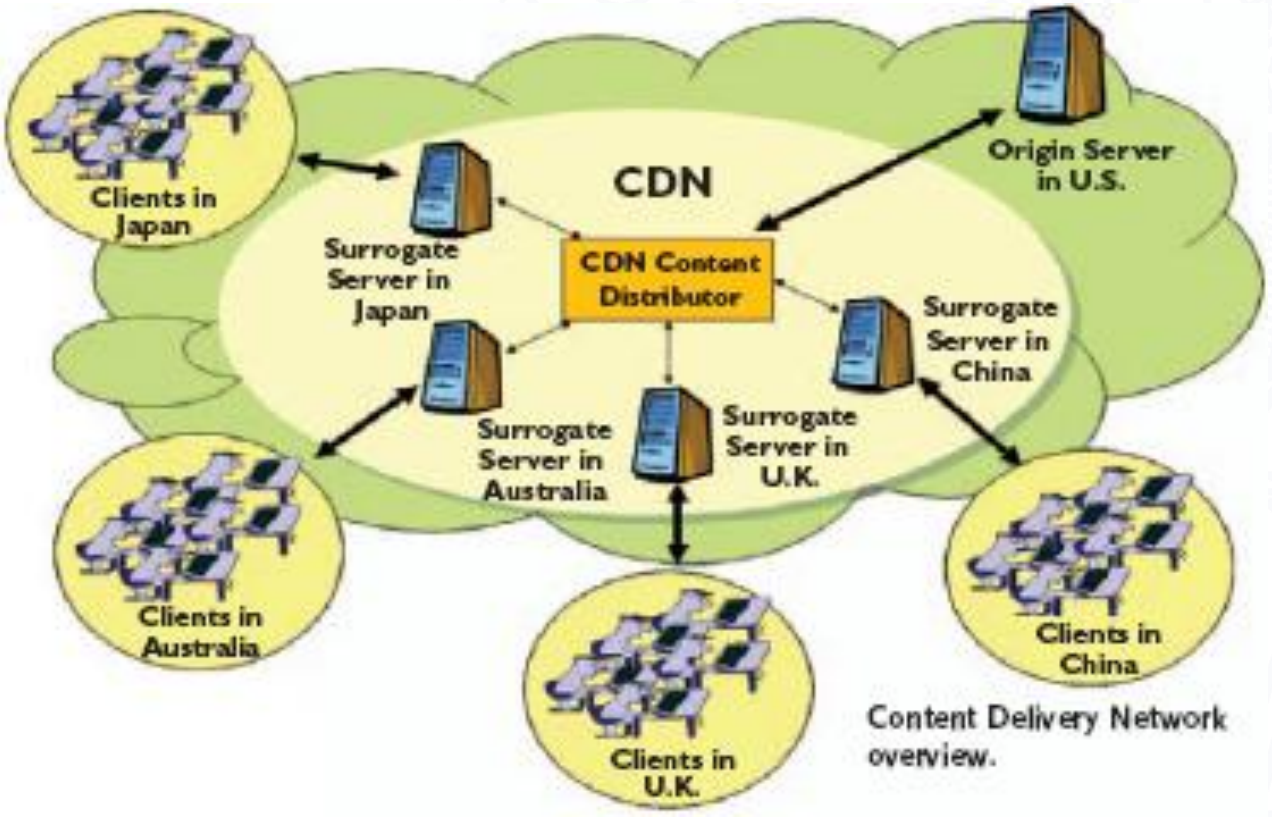
5. Proxy cache inverso

- Ejemplo de instalación
  - *Content Networking: Architecture, Protocols, and Practice*. Markus Hofmann and Leland Beaumont. Morgan Kaufmann



5. Redes de distribución de contenidos

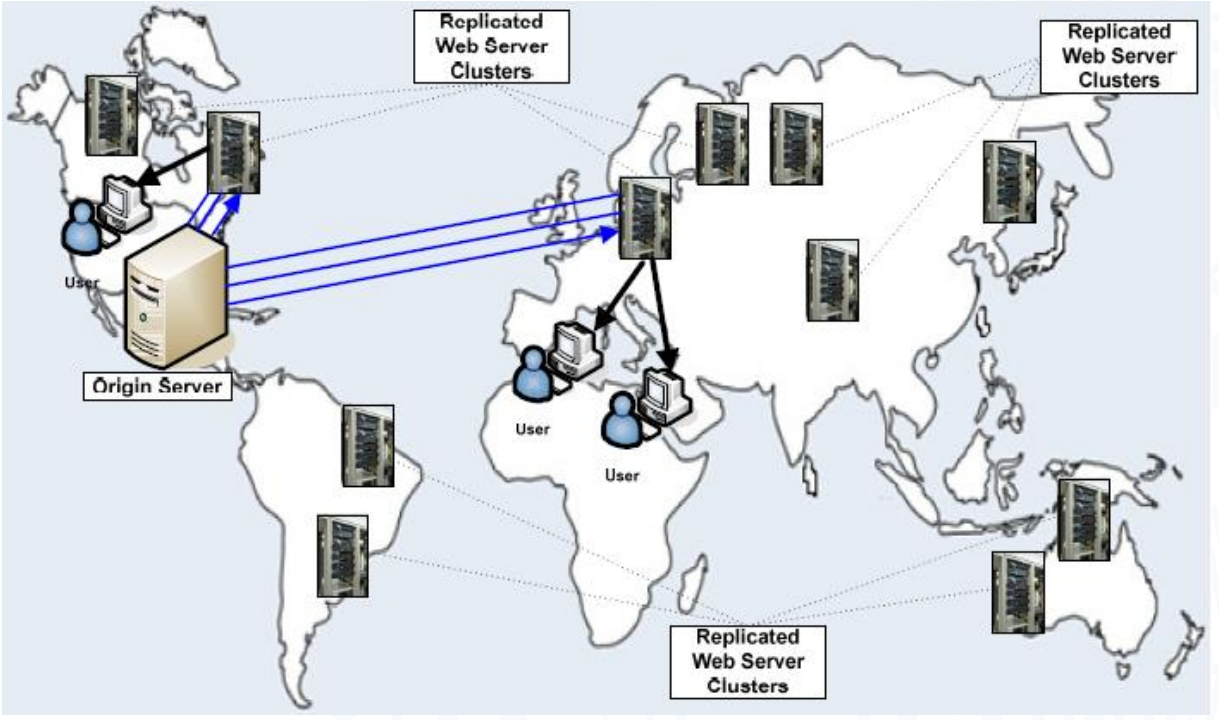
- CDN: *Content Delivery Network*
- Agrupación de proxys cache inversos
- Replicación de contenidos del servidor original en servidores repartidos por todo el mundo



Content Delivery Network overview.

5. Redes de distribución de contenidos

- Actores
  - Suministradores de contenidos
    - anuncios por Internet, data centers, ISPs, …
  - Suministradores de CDN
    - Speedera Networks, Akamai, Limelight Networks, CDNetworks
  - Usuarios finales

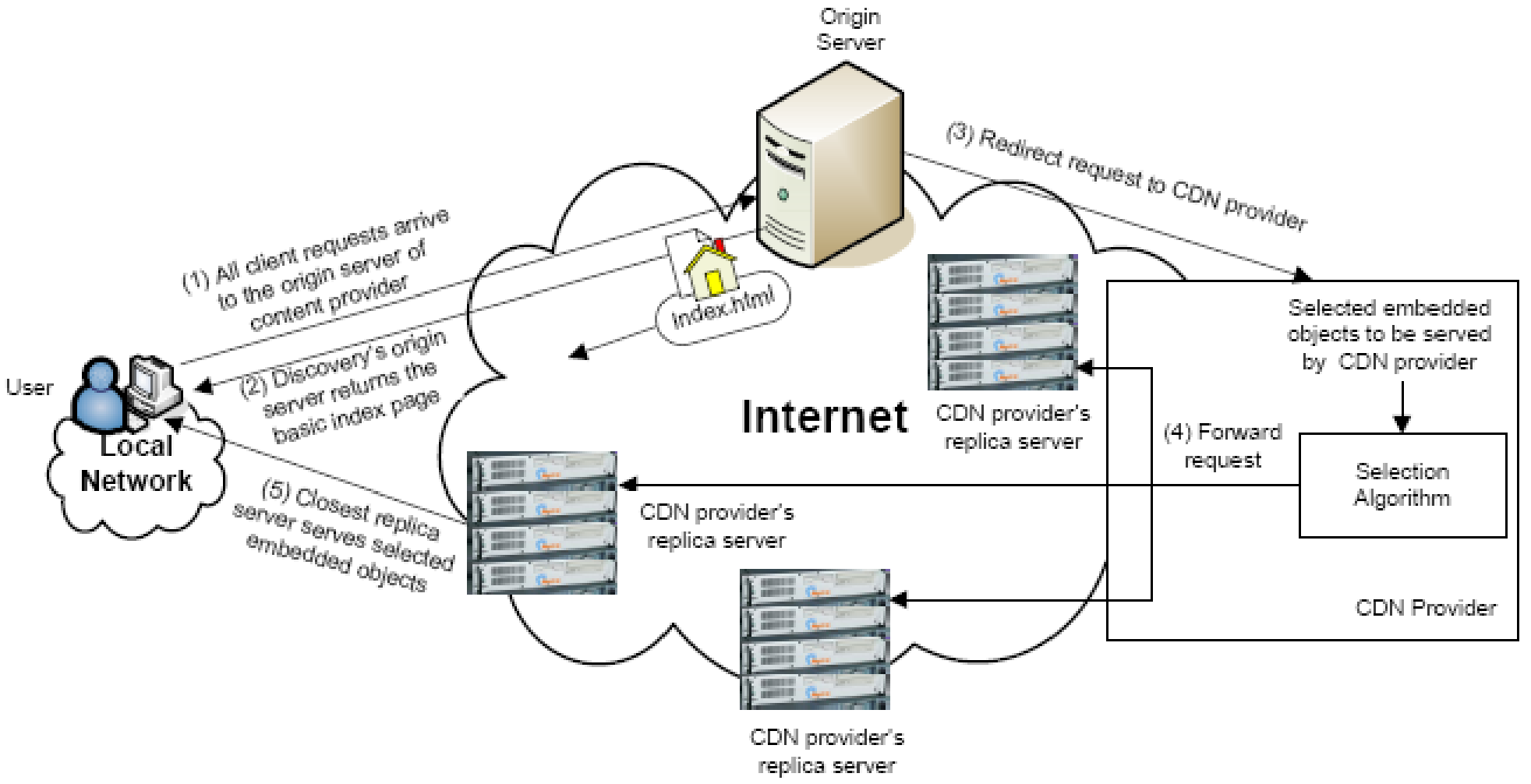


5. Redes de distribución de contenidos

- Infraestructura CDN
  - Entrega de contenido
    - De los servidores finales a los usuarios
  - Encaminamiento petición
    - Del servidor original al servidor más adecuado
  - Distribución y gestión de contenidos
    - De servidor original a servidores finales
  - Registro: para cobrar por servicio CDN

5. Redes de distribución de contenidos

- Encaminamiento de la petición
  - Servidor más próximo, con menos carga …



5. Redes de distribución de contenidos

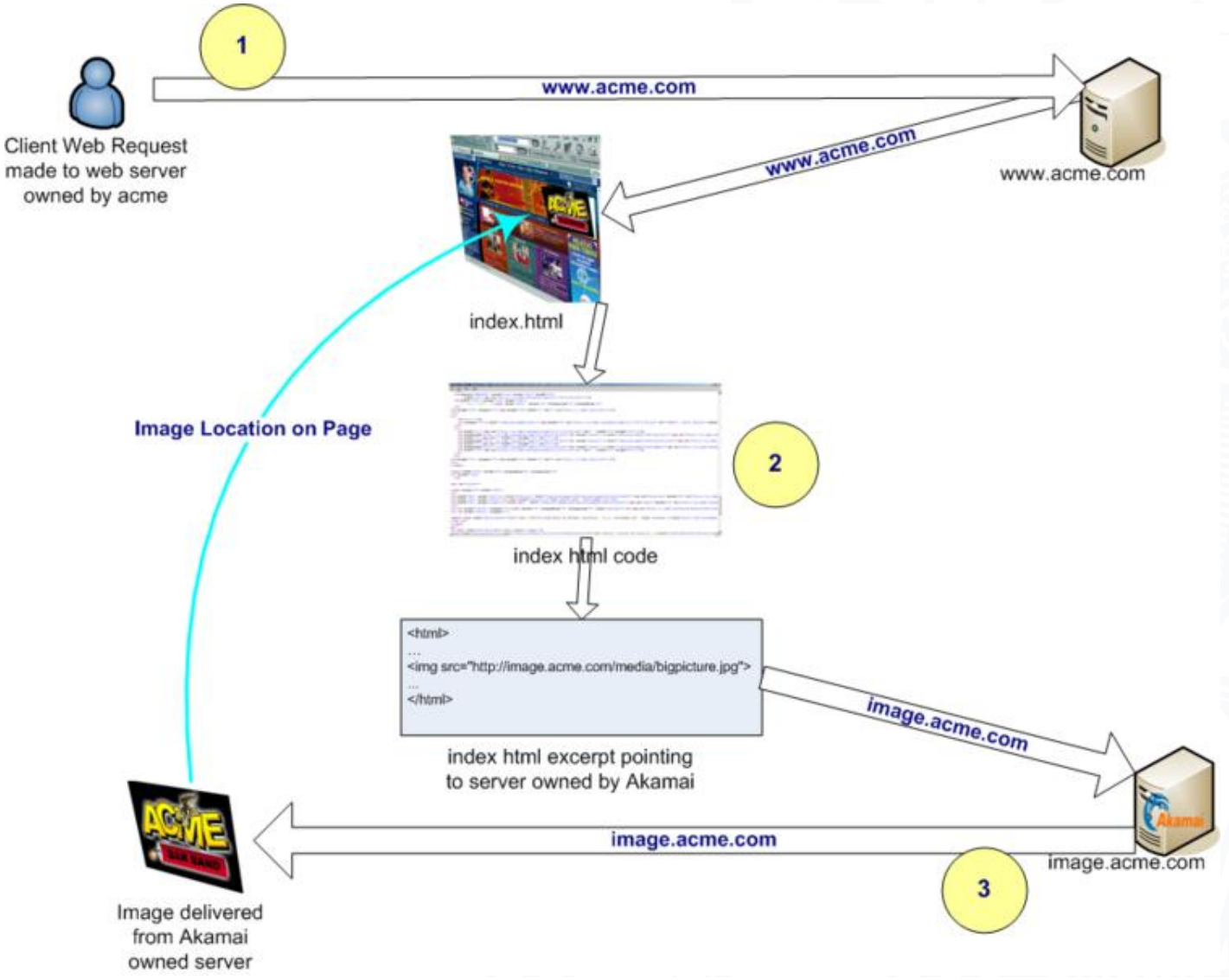
- Encaminamiento de la petición
  - Global Server Load Balancing (GSLB)
    - Cada servidor conoce el estado del resto
  - DNS-based request routing - Redirección HTTP
    - Info sobre servidores réplica en cabeceras HTTP
  - Reescritura URL
    - En el servidor
  - Anycast – RFC 1546
    - Una misma IP es asignada a varias máquinas
    - Los routers tienen el camino a la más cercana
  - CDN peering
    - Modelo P2P aplicado a servidores en CDNs
  - Algunas en RFC 3568

5. Redes de distribución de contenidos

- Distribución de contenidos: protocolos de comunicación
  - entre elementos del CDN
    - Network Element Control Protocol (NECP)
    - Web Cache Control Protocol (WCCP)
    - SOCKS
  - entre caches
    - Cache Array Routing Protocol (CARP)
    - Internet Cache Protocol (ICP)
    - Hypertext Caching Protocol (HTCP)
    - Cache Digest
- Internet Content Adaptation Protocol (ICAP), RFC 3507

5. Redes de distribución de contenidos

- Ejemplo: Akamai



5. Agrupación de caches web

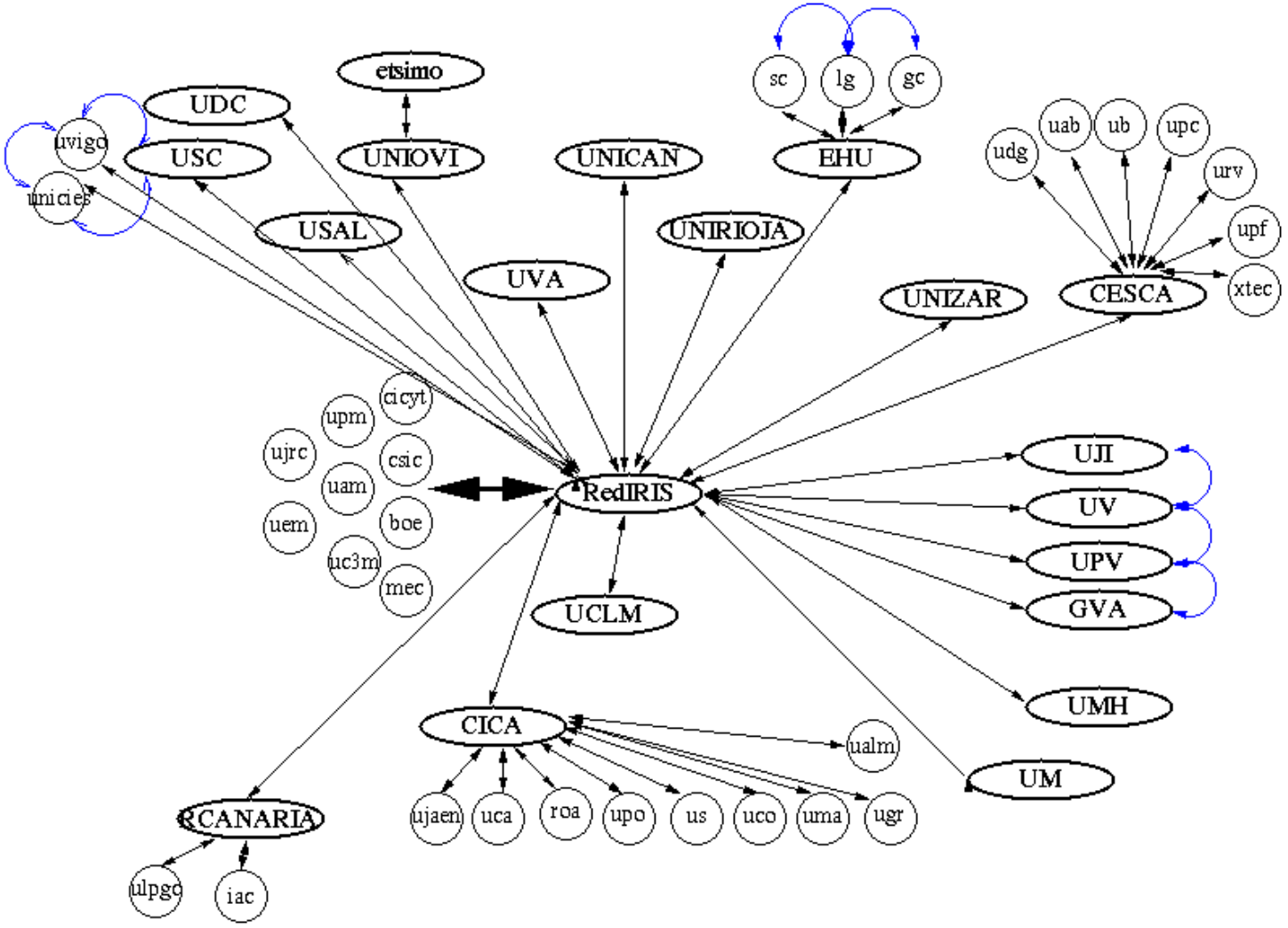
- Comunicación entre caches próximas
- Configuración típica:
  - Jerarquía con cache principal en el límite de una red grande, antes del enlace con otra red más lenta o saturada
  - Gestiona todos los accesos fuera de la red
  - Conexión con cache principal al otro lado de la red
  - Cache principal $\Rightarrow$ Secundarias $\Rightarrow$ 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

5. Agrupación de caches web

- 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

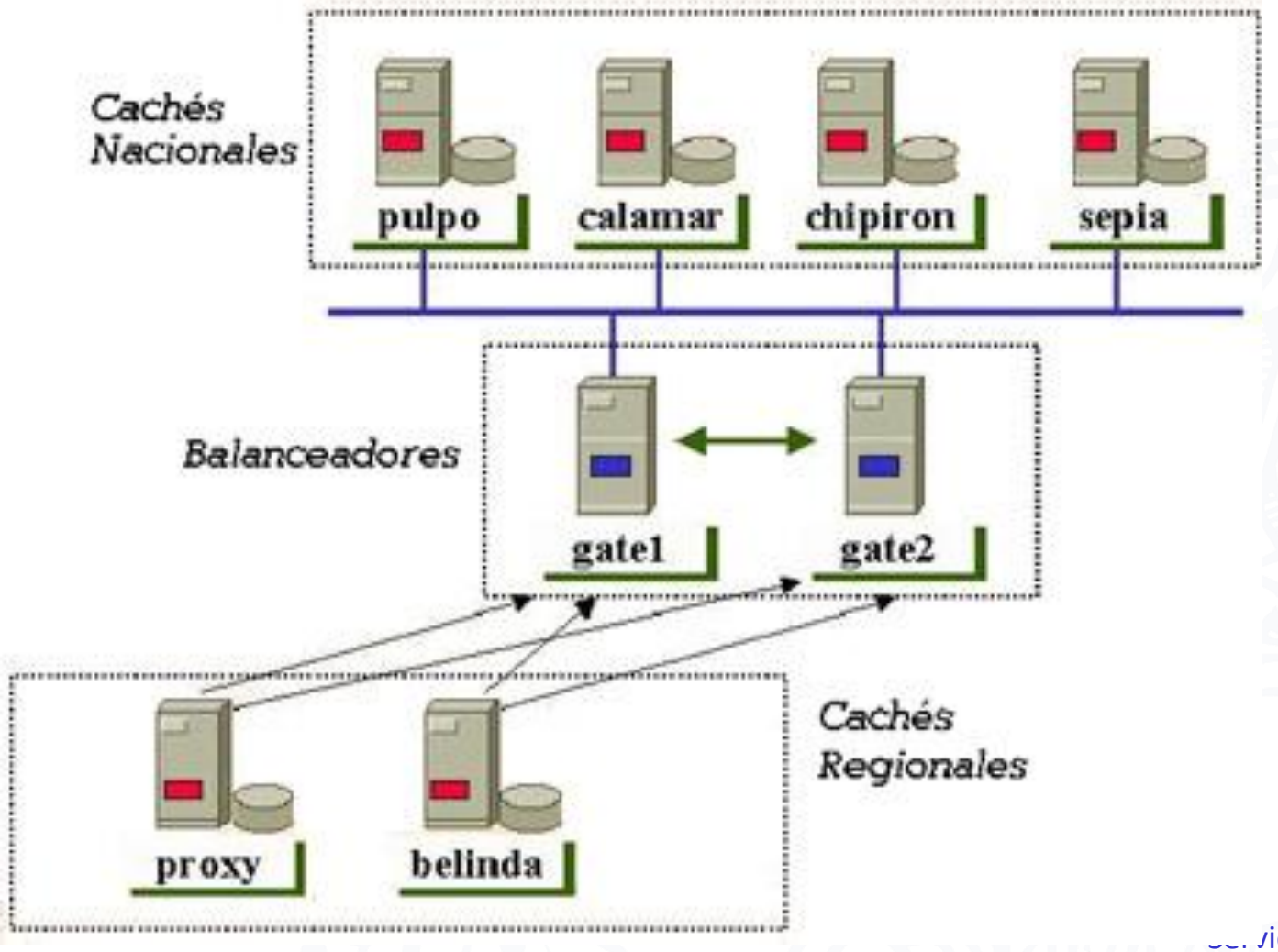
5. Agrupación de caches web

- Mapa caches RedIRIS



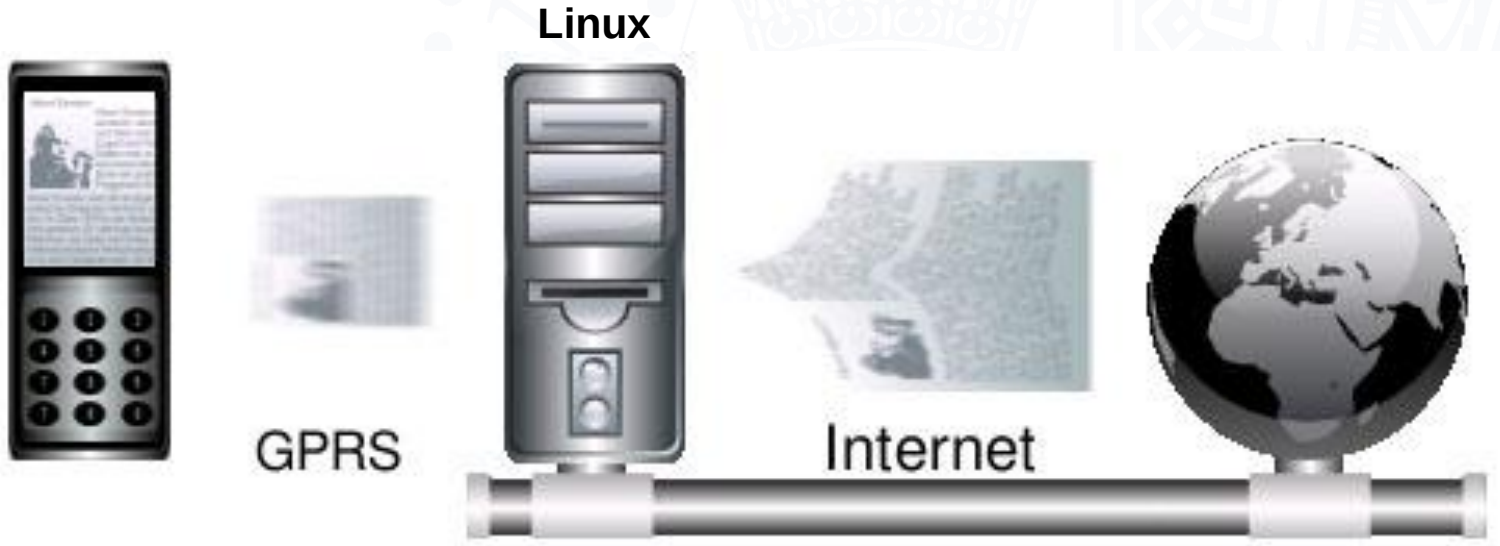
5. Agrupación de caches web

- Configuración caches RedIris



5. Opera

- Móvil pide contenido web a proxy de Opera
- Proxy pide contenido, lo reformatea, comprime y envía a móvil (OBML, Opera Bynary Markup Language)
- CPDs en Noruega e Islandia
  - http://my.opera.com/portalnews/blog/2010/11/01/20-million-opera-mini-users-move-to-iceland



5. Cache web

- Tecnología incomprendida
  - Pérdida control sitio web
  - Servicio de contenidos obsoletos
  - Punto de fallo
  - Estadísticas de acceso poco precisas
  - Recursos adicionales

- Realidad
  - Infraestructura caches

5. Control caches

- HTML Meta Tags
  - Descripción atributos de página web, en la sección <HEAD>
  - No son efectivas: proxy caches no leen HTML, ¿por qué?
  - http://www.w3.org/TR/REC-html40/struct/global.html#h-7.4.4.2

```
<META HTTP-EQUIV="PRAGMA" CONTENT="NO-CACHE">
<META HTTP-EQUIV="CACHE-CONTROL" CONTENT="NO-CACHE">
<META HTTP-EQUIV="Expires" CONTENT="-1">
```

- Cabeceras HTTP
  - Navegador, proxy, servidor
  - HTTP 1.0
    - cabecera `Expires`
  - HTTP 1.1
    - directivas `Cache-Control`

```
HTTP/1.1 200 OK
Date: Tue, 04 Jun 2013 08:08:11 GMT
Server: Apache
Etag: "1370332615-0"
Content-Language: es
Cache-Control: public, max-age=600
Last-Modified: Tue, 04 Jun 2013
07:56:55
Expires: Sun, 19 Nov 1978 05:00:00 GMT
```

5. 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
```

5. Cache web

- ¿Qué objetos no deben almacenarse en una cache?
  - Respuestas a peticiones
    - POST
    - PUT
    - DELETE
    - OPTIONS
    - TRACE
    - Cabecera `Authorization`
      - `Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==`
  - Almacenar respuestas a GET, HEAD
    - hay excepciones ...
    - Cabecera `no-cache`, ausencia validador

5. Políticas de reemplazo

Política de reemplazo	**Objeto reemplazado**	**cache se llena con los objetos**
Least Recently Used (LRU)	más tiempo sin ser requerido	con peticiones más recientes
First In, First Out (FIFO)	primero que se almacenó	más recientemente almacenados
Least Frequently Used (LFU)	menos frecuentemente requerido	más frecuentemente requeridos
Next To Expire (NTE)	primero en caducar	más infrecuentemente cambiados
Largest File First (LFF)	más grande	más pequeños

- Políticas combinadas

5. Reglas cache web

1. Objeto web está en cache navegador
   - Servir si *una vez por sesión*
2. Objeto web en proxy
   - No obsoleto: servir
     - Fecha expiración OK
     - Reciente y modificado hace mucho tiempo
   - 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

5. Control caches

- 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
```

5. Control caches

- 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

5. Control caches

- Peticiones **`Cache-control`**
  - no-cache: cache debe validar
  - no-store: proxy no debe almacenar
  - max-age=[segundos]: máxima edad
  - max-stale=[segundos]: se aceptan objetos [segundos] caducados
  - min-fresh=[segundos]: debe quedar para caducar [segundos]
  - no-transform
  - only-if-cached: enviar sólo si está en cache
  - cache-extension

- Más info en el draft HTTP 1.1

5. Control caches

- 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 almacener
  - 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
```

5. Control caches

- 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>
```

5. Control caches

- Validación: verificación de cambio en un objeto
  - cache-cache, cache-origen
- Validadores
  - Ausencia ⇒ 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

5. Control caches

- 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
```

5. Control caches

- Ejemplo 1

```
P: GET /resource HTTP/1.1
   Cache-Control: no-cache
R: La cache debe validar el recurso

P: GET /resource HTTP/1.1
   Cache-Control: max-age=0
R: La cache debe validar el recurso

P: GET /resource HTTP/1.1
   Cache-Control: max-age=500
R: La cache utiliza la copia almacenada
```

5. Control caches

- 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

5. Control caches

- Cabeceras `Expires` y `Last-Modified`
  - Pueden verse en *Page-Info* (Netscape), extensión LiveHTTPHeaders (Firefox)

Caching Tutorial for Web Authors and Webmasters has the following structure:
http://www.mnot.net/cache_docs/

Location: http://www.mnot.net/cache_docs/

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.

5. Control caches

- 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>
...
```

5. Cache web: trucos sitio web

- Utilizar servidores web que soporten peticiones GET IMS
- Consistencia enla 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

5. 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

5. 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.

5. 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

5. Squid

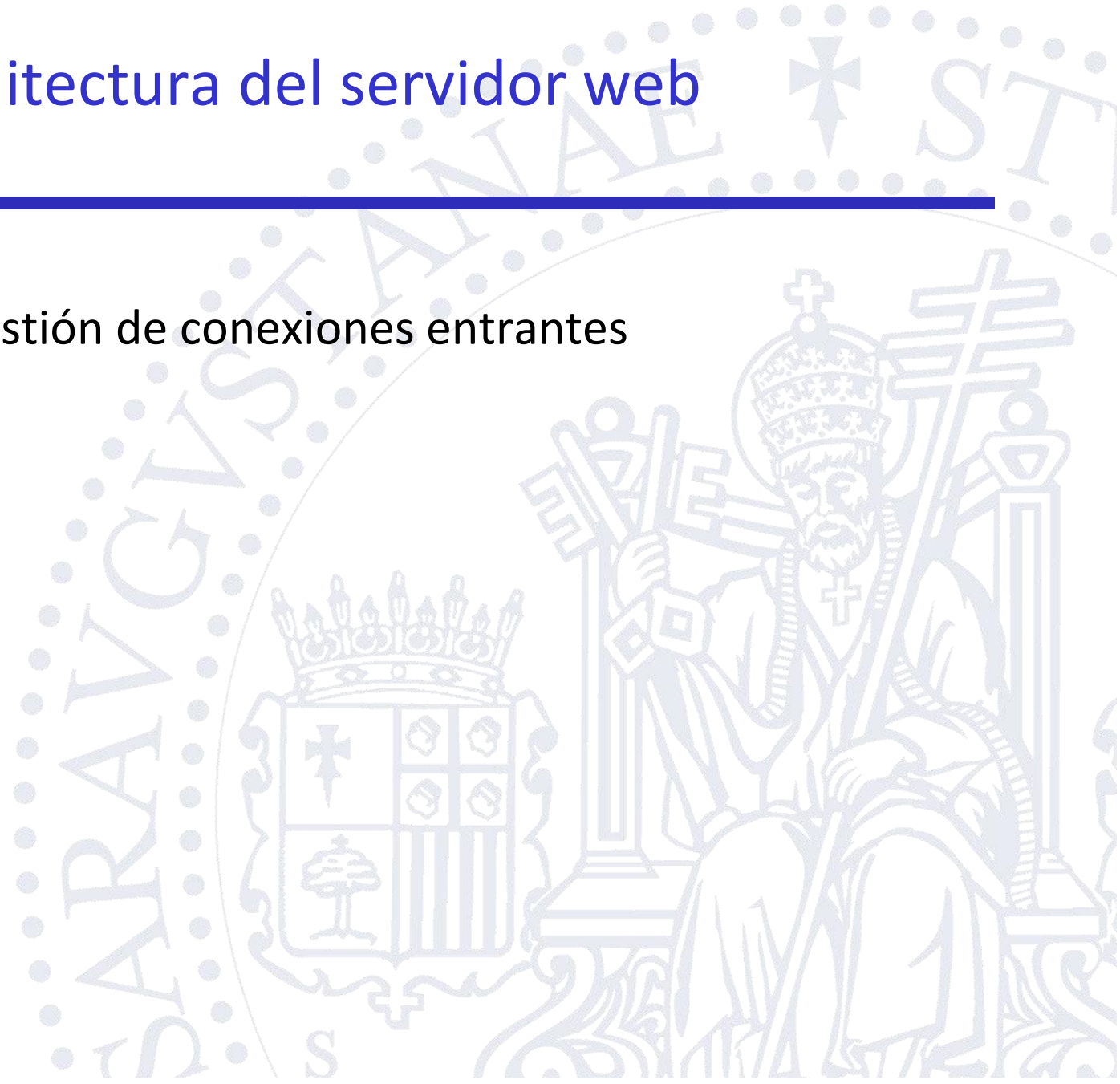
- Libre distribución
  - Gratuito + código fuente
- Rendimiento
- ICP (jerarquía)
- Cache-Digest: intercambio de URLs (cada hora)
- Módulos adicionales
  - Estadísticas, administración web ...
- Utilizado por agrupaciones mundiales de caches (NLANR)
- Unix

Bibliografía

- RFC 2616
- *Illustrated Guide to HTTP*, P. Hethmon
- Taxonomía de CDNs
  - http://www.gridbus.org/reports/CDN-Taxonomy.pdf
- Tutorial sobre caches para autores web y webmasters
  - http://www.mnot.net/cache_docs

Arquitectura del servidor web

Gestión de conexiones entrantes



5. Arquitectura del servidor web

- Servidor web: proceso escuchando en un puerto (nivel 4)
- Gestión conexiones entrantes: nuevo proceso generado por
  - un proceso que escucha varios puertos (inetd en Unix)
  - propio servidor web (standalone)
- Servidores
  - Forking
  - Threaded

```
unix$ ps -ef | grep httpd
…
tcp   0   0   0.0.0.0:22   0.0.0.0:*   LISTEN   23858/sshd
udp   0   0   0.0.0.0:53   0.0.0.0:*            30016/named
…
```

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/>
- Web dinámica
  - http://tomcat.apache.org/