Arquitecturas Distribuidas

TEMA 2. La Web

Tema 2. La Web

1. ¿Qué es la Web?
2. Cronología
3. Comparación entre servidores web
4. Funcionamiento del WWW
   - 4.1. Identificadores URL
   - 4.2. Protocolo HTTP
   - 4.3. Lenguaje HTML
5. Mejoras actuales para el desempeño del WWW
   - 5.1. Mejoras en el lado del servidor
     - 5.1.1. Caché de disco
     - 5.1.2. Sistemas multi-disco
     - 5.1.3. “Granjas” de servidores
   - 5.2. Mecanismos de caché
     - 5.2.1. Caché jerárquica
     - 5.2.2. Caché proactiva
   - 5.3. *Mirroring* y equilibro por DNS
   - 5.4. Redes de entrega de contenido
   - 5.5 Optimización del *Front-End*

¿Qué es la Web? (I)

- World Wide Web
- La Web es un conjunto de tecnologías y protocolos, que funcionan sobre la infraestructura física y los protocolos de Internet, usados para acceder e intercambiar recursos **vinculados** entre sí. Está en continua evolución.
- La Web, como sistema, no impone ninguna restricción al tipo de recursos que se pueden albergar en ella
- Simplemente define como esos recursos pueden ser intercambiados entre ordenadores (y, por tanto, entre personas)
- Basada en tres tecnologías muy simples cuyo objetivo es respectivamente:
  - Nombrar los recursos
  - Representar los recursos
  - Transferir los recursos
- La clave:
  - Apertura: el sistema es ampliado e implementado de diferentes formas sin modificar su funcionalidad, abierto respecto a los tipos de recursos

¿Qué es la Web ? (II)

- Concepción inicial: ¿Cómo compartir documentos científicos con colegas?
- Concepto de HIPERTEXTO: los documentos se enlazan con otros documentos
  - idea de Vannevar Bush en 1945.
  - estructura de los enlaces puede ser arbitrariamente compleja y el conjunto de recursos añadidos ilimitado
  - Claves: interfaz intuitiva y fácil.
- Tecnologías para implementar la idea anterior:
  - HTML: formato en el que se almacenan los documentos
  - HTTP: protocolo de nivel de aplicación, tipo petición/respuesta, para la transferencia de documentos hiperenlazados

¿Cómo se trabaja en la web?

- Los documentos se almacenan en servidores HTTP (servidores Web), es decir, que implementan la interfaz del protocolo HTTP.
- Los documentos (o páginas Web) se visualizan mediante "navegadores" (*browser*): aplicación software
  - Programa cliente de HTTP e intérprete de HTML
  - Mediante "extensiones" (*plugins*) se amplía el abanico de recursos representables

Cronología

- Comienza en marzo de 1989 en el CERN
  - Propuesta original del físico T. Berners-Lee.
  - Para intercambio de información entre grupos de trabajo.
  - Utiliza el concepto de Hipertexto
- 18 meses después → Primer prototipo (basado en texto)
- Diciembre de 1991 → Demostración pública.
- Febrero de 1993 → Marc Andreesen (univ. De Illinois) libera Mosaic, el primer navegador gráfico.

Cronología

- 1994, Andreesen funda Netscape, compañía cuya meta era desarrollar clientes, servidores y otro tipo de *software* web.
- 1995, liberación de Netscape
- 1995 – 1998: "Guerra de navegadores" entre Netscape y Microsoft Explorer
  - Nuevas características
  - Muy mal programados (ambos contenían muchos errores).

Cronología

- 1998 → AOL compra Netscape
- 1998 + → Nace Mozilla, navegador de código abierto, para cualquier plataforma.
- 1998 → Nace Google
- 2003 + → Versiones avanzadas de Mozilla (Firefox, Mozilla v1.7)

Cronología

- Otros hitos:
  - 1994 → CERN y MIT fundan el Consorcio del World Wide Web, con el objetivo:
    - Impulsar el desarrollo del Web
    - Estandarizar protocolos
    - Fomentar de la interoperabilidad entre múltiples fabricantes.
    - http://www.w3c.org

Funcionamiento del WWW

- WWW se asienta sobre tres pilares, que *curiosamente* son independientes entre sí, es decir, se pueden utilizar por separado y en otros ámbitos:
  - Los **identificadores URL/URI** (Localizadores Universales de Recursos). **¿Cómo nombrar los recursos?**
  - El **protocolo HTTP** (Protocolo de Transferencia de Hipertexto). **¿Cómo intercambiar los recursos?**
  - El **lenguaje HTML** (Lenguaje de Marcas de Hipertexto). **¿Qué intercambiamos? ¿En qué formato expresamos los recursos?** (originalmente)

Tema 2. La Web

1. ¿Qué es la Web?
2. Cronología
3. Comparación entre servidores web
4. Funcionamiento del WWW
   - 4.1. **Identificadores URL**
   - 4.2. Protocolo HTTP
   - 4.3. Lenguaje HTML
5. Mejoras actuales para el desempeño del WWW
   - 5.1. Mejoras en el lado del servidor
     - 5.1.1. Caché de disco
     - 5.1.2. Sistemas multi-disco
     - 5.1.3. “Granjas” de servidores
   - 5.2. Mecanismos de caché
     - 5.2.1. Caché jerárquica
     - 5.2.2. Caché proactiva
   - 5.3. *Mirroring* y equilibro por DNS
   - 5.4. Redes de entrega de contenido
   - 5.5 Optimización del *Front-End*

Identificadores URL

- Surgen de la necesidad de tener un mecanismo para nombrar y localizar recursos de manera uniforme. Tal mecanismo debe responder a:
  - ¿Cómo se llama el recurso?
  - ¿Dónde está el recurso?
  - ¿Cómo se puede acceder al recurso?

Identificadores URL

- Analogía, si cada página tuviera un identificador único, ¿bastaría?
  - NO! → Analogía) Casi todos los europeos poseemos un número de identificación, que nos distingue, pero conociéndolo, no sabemos dónde está la persona, ni en qué idioma podemos hablar con ella.

Identificadores URL

- Solución: Uso de URL (Uniform Resource Locator), cadena de caracteres que identifica un recurso en la Web
- Tienen 3 partes:

http://www.upct.es/documentos/indice.html

http://	www.upct.es	/documentos/indice.html
• Protocolo o esquema	• Nombre DNS de la máquina • Puede incluir el puerto: www.upct.es:8080	• Nombre local del recurso

Identificadores URL

- PROTOLO O ESQUEMA: El protocolo nativo para la Web es el HTTP (http://), sin embargo, los navegadores soportan otros protocolos adicionales.
- De hecho, las URL se utilizan en muchos otros ámbitos, no son específicas de la Web.
- Lista de esquemas en:

  http://www.iana.org/assignments/uri-schemes

Ejemplos de Esquemas

Nombre del esquema	Descripción	Referencia
ftp	File Transfer Protocol	RFC 1738
http	Hypertext Transfer Protocol	RFC 2616
mailto	Electronic mail address	RFC 2368
file	Acceso local a archivo	
news	USENET news	RFC 1738

URL relativas y absolutas

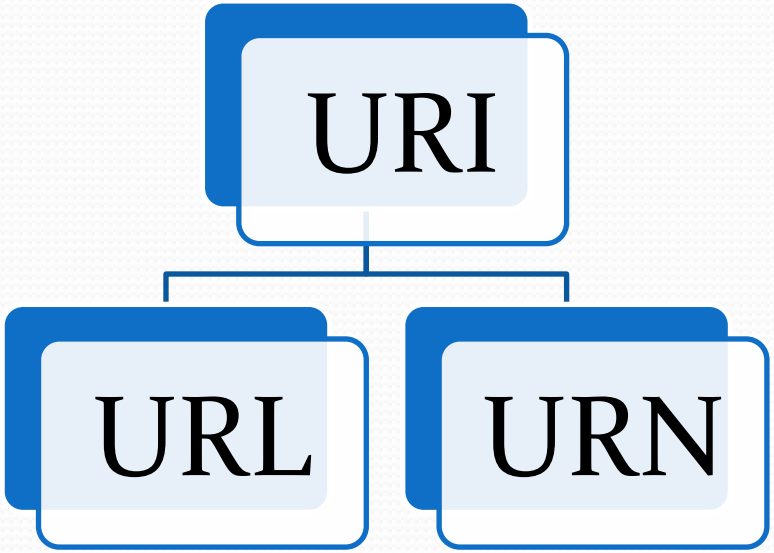
- Cuando una URL aparece dentro de un documento HTML ésta puede ser relativa a la URL actual o absoluta.
- Ventaja de las relativas: se puede mover toda la estructura del árbol de documentos sin tener que cambiar todas las URL
- Ejemplo: Dentro de un documento HTML en http://www.upct.es/ad/index.html aparecen las siguientes URLs
  - *imagenes/a.gif* indica que la imagen está en un subdirectorio que cuelga del directorio de la URL actual, su equivalente absoluta sería: http://www.upct.es/ad/imagenes/a.gif
  - *../index.html*, su equivalente absoluta sería http://www.upct.es/index.html
  - *a.gif,* su equivalente absoluta sería http://www.upct.es/ad/a.gif

Identificadores URL

- Los URLs sirven no sólo para su uso en web, sino para permitir a los usuarios acceder a una multitud de servicios, como correo electrónico, noticias, etc.
- Problemas con los URL: apuntan siempre a un host específico. Para páginas muy visitadas sería bueno tener varias copias (réplicas) diferentes. Pero los URL no ofrecen ningún mecanismo para referirse a un documento sin decir simultáneamente donde está.

URI y URN

- En realidad las URL formalmente son un subconjunto de los identificadores que se denominan URI (Uniform Resource Identifier)
- URN (Uniform Resource Name) describen recursos de acuerdo a otras propiedades



Tema 2. La Web

1. ¿Qué es la Web?
2. Cronología
3. Comparación entre servidores web
4. Funcionamiento del WWW
   - **4.1.** Identificadores URL
   - 4.2. **Protocolo HTTP**
   - 4.3. Lenguaje HTML
5. Mejoras actuales para el desempeño del WWW
   - 5.1. Mejoras en el lado del servidor
     - 5.1.1. Caché de disco
     - 5.1.2. Sistemas multi-disco
     - 5.1.3. “Granjas” de servidores
   - 5.2. Mecanismos de caché
     - 5.2.1. Caché jerárquica
     - 5.2.2. Caché proactiva
   - 5.3. *Mirroring* y equilibro por DNS
   - 5.4. Redes de entrega de contenido
   - 5.5 Optimización del *Front-End*

Protocolo HTTP

- HTTP (HyperText Transfer Protocol), Protocolo de Transferencia de HiperTexto.
- Protocolo de **petición/respuesta** para el intercambio de bloques de información
- Es el protocolo de transferencia de datos/documentos en la Web
- Especifica **qué mensajes** pueden intercambiarse entre clientes y servidores, y su **formato**.
- Especificación contenida en el RFC 2616.

Protocolo HTTP: Conexiones

- Es un **protocolo de nivel de aplicación**
- Protocolo de texto: Datos codificados en ASCII
- Conexiones HTTP:
  - Se realizan sobre un canal de transporte fiable (casi siempre TCP).
  - En HTTP 1.0 la conexión se establecía, se enviaba la solicitud, y se obtenía la respuesta. Después se liberaba dicha conexión. → Adecuado al principio del WWW, cuando las páginas contenían sólo texto.

Protocolo HTTP: Conexiones

- Pb) Con el tiempo partes significativas de las páginas eran gráficos, iconos, y otros elementos no textuales → Establecer una conexión por cada elemento era MUY COSTOSO.
  - Sol) HTTP 1.1 → SOPORTE PARA CONEXIONES PERSISTENTES
    - ABRIR CONEXIÓN CON UN SERVIDOR PARTICULAR
    - MULTIPLES SOLICITUDES (ASÍNCRONAS o SÍNCRONAS)
    - CERRAR CONEXIÓN

Protocolo HTTP: puerto

- En HTTP por defecto, las comunicaciones van al puerto 80 del servidor, pero puede cambiarse en el URL:

http://www.upct.es:81/pagina.html

Protocolo HTTP: formato petición

- Petición/respuesta. Directamente se hace una petición y se recibe una respuesta
- Sin estado (a diferencia de TCP, por ejemplo). Cada petición es completamente independiente de la anterior:
  - ¿problemas?
  - ¿Ventajas?
- Cada solicitud del cliente es una cadena ASCII (7 bit), donde la primera palabra (en MAYÚSCULAS) indica la **operación** a realizar. A continuación se identifica **localmente** el recurso sobre el que se realiza la operación

GET /directorio/paginaweb.html HTTP/1.0

Protocolo HTTP: formato respuesta

- Respuesta del Servidor:
- De nuevo, cabeceras con texto en ASCII indicando:

  Línea de estado (Código + Información Código)
  Cabeceras adicionales
  +
  Línea en blanco OBLIGATORIA
  +
  MIME objeto respuesta
- Finalmente, los datos. Normalmente ,HTML

Protocolo HTTP: ejemplo

- Ejemplo de solicitud: GET de / a www.google.com:

```
localhost:~> telnet www.google.com 80
Trying 216.239.53.99...
Connected to 216.239.53.99. Escape character is '^]'.
GET / HTTP/1.0

HTTP/1.0 202 Found
Connection: Keep-Alive
Date: Mon, 06 Oct 2003 17:27:28 GMT
Content-length: 203
Server: GWS/2.1
Content-Type: text/html
Location: http://www.google.es/cxfer?c=PREF%3D:TM%3D1065461248:S%3DCZzXP1rRkCtrYqL8
Set-Cookie:
PREF=ID=6a0fb58773f1a46e:CR=1:TM=1065461248:LM=1065461248:S=Uvz892UeiF3F815E;
expires=Sun, 17-Jan-2038 19:14:07 GMT; path=/; domain=.google.com

<HTML><HEAD>
<TITLE></TITLE>
</HEAD><BODY> CONTENIDO </BODY></HTML>
```

Protocolo HTTP: ejemplo

- Ejemplo de solicitud: GET de / a www.google.com:

```
localhost:~> telnet www.google.com 80
Trying 216.239.53.99...
Connected to 216.239.53.99. Escape character is '^]'.
GET / HTTP/1.0

HTTP/1.0 202 Found
Connection: Keep-Alive
Date: Mon, 06 Oct 2003 17:27:28 GMT
Content-length: 203
Server: GWS/2.1
Content-Type: text/html
Location: http://www.google.es/cxfer?c=PREF%3D:TM%3D1065461248:S%3DCZzXP1rRkCtrYqL8
Set-Cookie:
PREF=ID=6a0fb58773f1a46e:CR=1:TM=1065461248:LM=1065461248:S=Uvz892UeiF3F815E;
expires=Sun, 17-Jan-2038 19:14:07 GMT; path=/; domain=.google.com

<HTML><HEAD>
<TITLE></TITLE>
</HEAD><BODY> CONTENIDO </BODY></HTML>
```

En este momento se ha establecido la conexión TCP

Petición que recibe el servidor HTTP

Protocolo HTTP: ejemplo

- Ejemplo de solicitud: GET de / a www.google.com:

```
localhost:~> telnet www.google.com 80
Trying 216.239.53.99...
Connected to 216.239.53.99. Escape character is '^]'.
GET / HTTP/1.0

HTTP/1.0 202 Found
Connection: Keep-Alive
Date: Mon, 06 Oct 2003 17:27:28 GMT
Content-length: 203
Server: GWS/2.1
Content-Type: text/html
Location: http://www.google.es/cxfer?c=PREF%3D:TM%3D1065461248:S%3DCZzXP1rRkCtrYqL8
Set-Cookie:
PREF=ID=6a0fb58773f1a46e:CR=1:TM=1065461248:LM=1065461248:S=Uvz892UeiF3F815E;
expires=Sun, 17-Jan-2038 19:14:07 GMT; path=/; domain=.google.com

<HTML><HEAD>
<TITLE></TITLE>
</HEAD><BODY> CONTENIDO </BODY></HTML>
```

Código de estado

Protocolo HTTP

Códigos de respuesta en HTTP

Code	Meaning	Examples
1xx	Information	100 = server agrees to handle client's request
2xx	Success	200 = request succeeded; 204 = no content present
3xx	Redirection	301 = page moved; 304 = cached page still valid
4xx	Client error	403 = forbidden page; 404 = page not found
5xx	Server error	500 = internal server error; 503 = try again later

Protocolo HTTP

- Ejemplo de solicitud: GET de / a www.servidor.com:

```
localhost:~> telnet www.google.com 80
Trying 216.239.53.99...
Connected to 216.239.53.99. Escape character is '^]'.
GET / HTTP/1.0

HTTP/1.0 202 Found
Connection: Keep-Alive
Date: Mon, 06 Oct 2003 17:27:28 GMT
Content-length: 203
Server: GWS/2.1
Content-Type: text/html
Location: http://www.google.es/cxfer?c=PREF%3D:TM%3D1065461248:S%3DCZzXP1rRkCtrYqL8
Set-Cookie:
PREF=ID=6a0fb58773f1a46e:CR=1:TM=1065461248:LM=1065461248:S=Uvz892UeiF3F815E;
expires=Sun, 17-Jan-2038 19:14:07 GMT; path=/; domain=.google.com

<HTML>
<HEAD><TITLE></TITLE>
</HEAD><BODY> CONTENIDO </BODY></HTML>
```

Cabeceras

Protocolo HTTP

- Ejemplo de solicitud: GET de / a www.google.com:

```
localhost:~> telnet www.google.com 80
Trying 216.239.53.99...
Connected to 216.239.53.99. Escape character is '^]'.
GET / HTTP/1.0

HTTP/1.0 202 Found
Connection: Keep-Alive
Date: Mon, 06 Oct 2003 17:27:28 GMT
Content-length: 203
Server: GWS/2.1
Content-Type: text/html
Location: http://www.google.es/cxfer?c=PREF%3D:TM%3D1065461248:S%3DCZzXP1rRkCtrYqL8
Set-Cookie:
PREF=ID=6a0fb58773f1a46e:CR=1:TM=1065461248:LM=1065461248:S=Uvz892UeiF3F815E;
expires=Sun, 17-Jan-2038 19:14:07 GMT; path=/; domain=.google.com

<HTML><HEAD>
<TITLE></TITLE>
</HEAD><BODY> CONTENIDO </BODY></HTML>
```

Objeto de información

Cabecera Content-Type

- Para poder desplegar una página el navegador ha de entender su formato.
- Todos los navegadores han de entender "de igual forma" las mismas páginas.
- Para ello se usa un **lenguaje estandarizado** llamado Lenguaje de Marcado de Hipertexto (HTML). Es el tipo de respuesta por defecto.
- Otros **tipos distintos pueden descargarse** indicando al navegador su tipo de Extensión de Correo Electrónico Multipropósito (MIME).

Cabecera Content-Type

- .html → “text/html”
- .gif → “image/gif”
- …

- El navegador soporta ciertos tipos MIME (distintos de HTML) como tipos integrados. Los tipos no soportados se visualizan a través de *plug-ins*, o aplicaciones auxiliares.

Cabecera Content-Type

- *Plug-ins* → Aplicaciones que se ejecutan a través de una API dentro del navegador (acceso directo a sus contenidos).
  - Ejemplo: Java o Flash
- Aplicaciones → Aplicaciones que se ejecutan externamente al navegador.
  - Ejemplo: Video MPEG4
  - Ejemplo: PDF de Adobe originalmente, después pasa a ser un *plug-in*

Protocolo HTTP: Interfaz

- Otros métodos (comandos/órdenes) HTTP:
  - HTTP se diseño para la WWW, pero se hizo mucho más general de lo necesario para dar soporte a futuras aplicaciones.
  - **Interfaz**: Conjunto de operaciones ofrecidas por el protocolo
  - HTTP soporta distintos métodos u operaciones diferentes a la mera solicitud de una página web.

Protocolo HTTP: Interfaz

Método	**Descripción**
GET	Solicita la lectura de una página web
HEAD	Solicita la lectura del encabezado de una página web
PUT	Solicita el almacenamiento de una página web
POST	Inserta algo a un recurso con nombre
DELETE	Elimina la página Web
TRACE	Repite la solicitud entrante
CONNECT	Reservado para uso futuro
OPTIONS	Consulta ciertas opciones

Información de estado

- HTTP no mantiene información de estado
- **Ventaja**: es mucho más **escalable**. El servidor no tiene que almacenar información de estado.
- No es posible saber qué ha hecho ("qué ha visitado") un cliente previamente, en qué estado está.
- No es posible realizar aplicaciones tipo "carrito de la compra", y muchas otras, que requerían información de qué había realizado antes el cliente.
  - SOLUCIÓN → EXTENSIÓN DE HTTP Y NAVEGADORES → COOKIES.

Tema 2. La Web

1. ¿Qué es la Web?
2. Cronología
3. Comparación entre servidores web
4. Funcionamiento del WWW
   - **4.1.** Identificadores URL
   - 4.2. Protocolo HTTP
   - 4.3. **Lenguaje HTML**
5. Mejoras actuales para el desempeño del WWW
   - 5.1. Mejoras en el lado del servidor
     - 5.1.1. Caché de disco
     - 5.1.2. Sistemas multi-disco
     - 5.1.3. “Granjas” de servidores
   - 5.2. Mecanismos de caché
     - 5.2.1. Caché jerárquica
     - 5.2.2. Caché proactiva
   - 5.3. *Mirroring* y equilibro por DNS
   - 5.4. Redes de entrega de contenido
   - 5.5 Optimización del *Front-End*

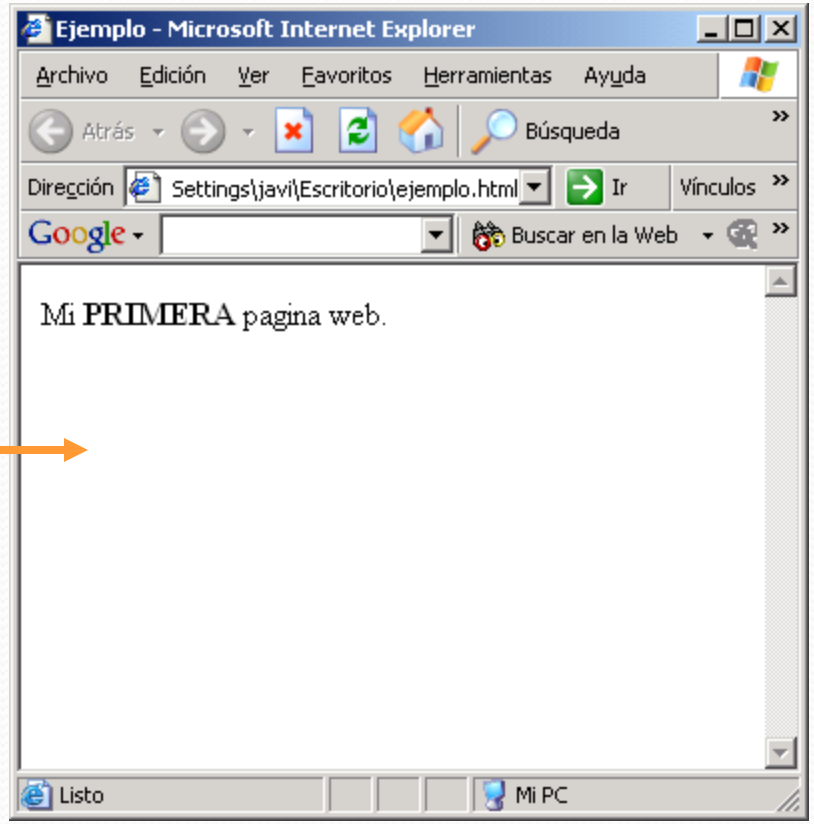
Lenguaje HTML

- Es el lenguaje de las páginas web.
- Son páginas con texto, imágenes, enlaces, etc.
- Es un lenguaje estructurado: incluye datos y metadatos: “datos sobre los datos”
- Es un lenguaje de MARCAS que describen como se deben representar los elementos.
- A diferencia de los URL y el HTTP, HTML no aportó (ya desde un primer momento) toda la generalidad necesaria, y ha sido (y es) objeto de revisiones.

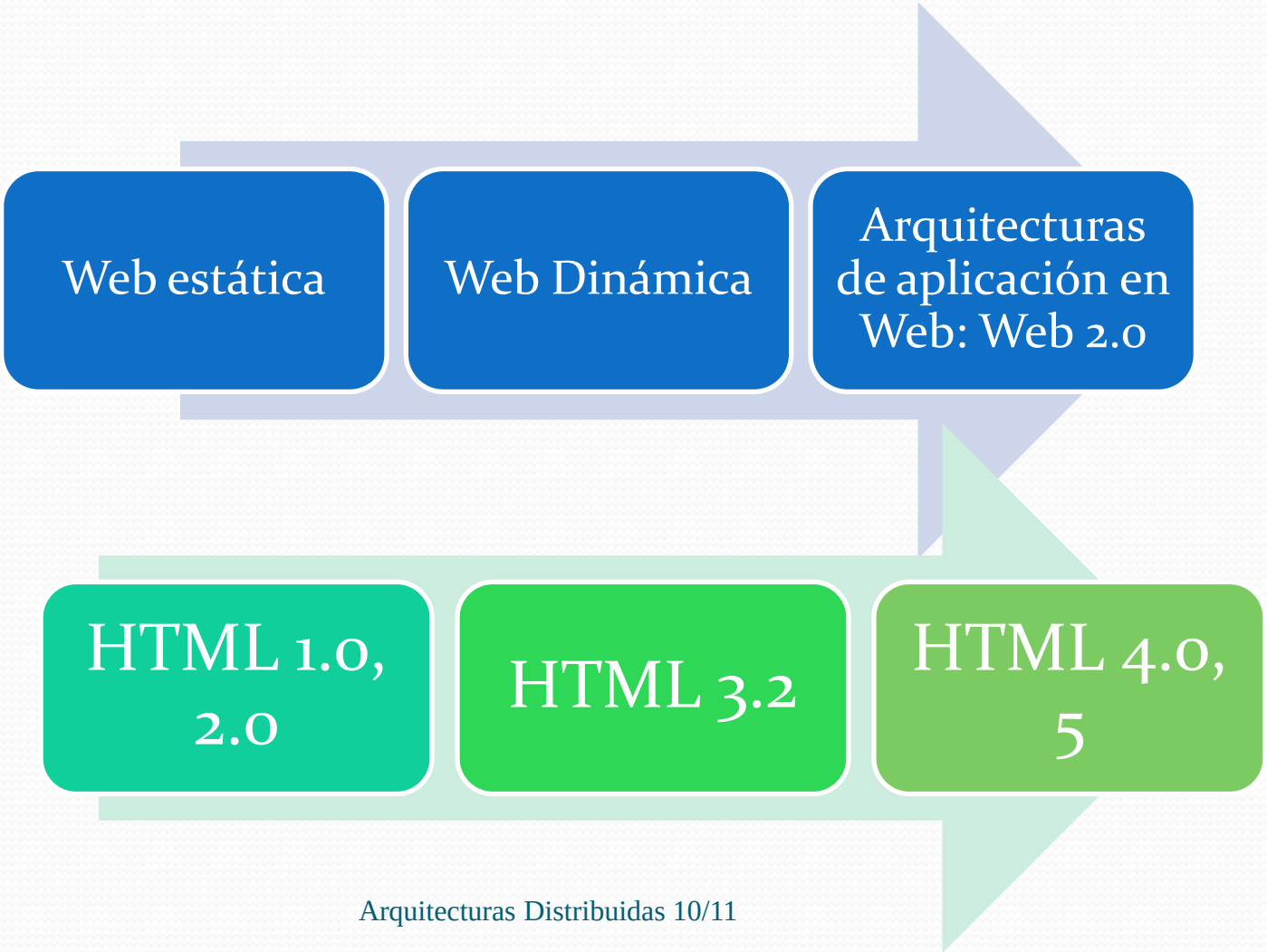
Lenguaje HTML: ejemplo

- Ejemplo página web:

```
<HTML>

<HEAD>
<TITLE> Ejemplo </TITLE>
</HEAD>


<BODY>
Mi <B> PRIMERA </B> pagina web.
</BODY>

</HTML>
```



Lenguaje HTML: evolución



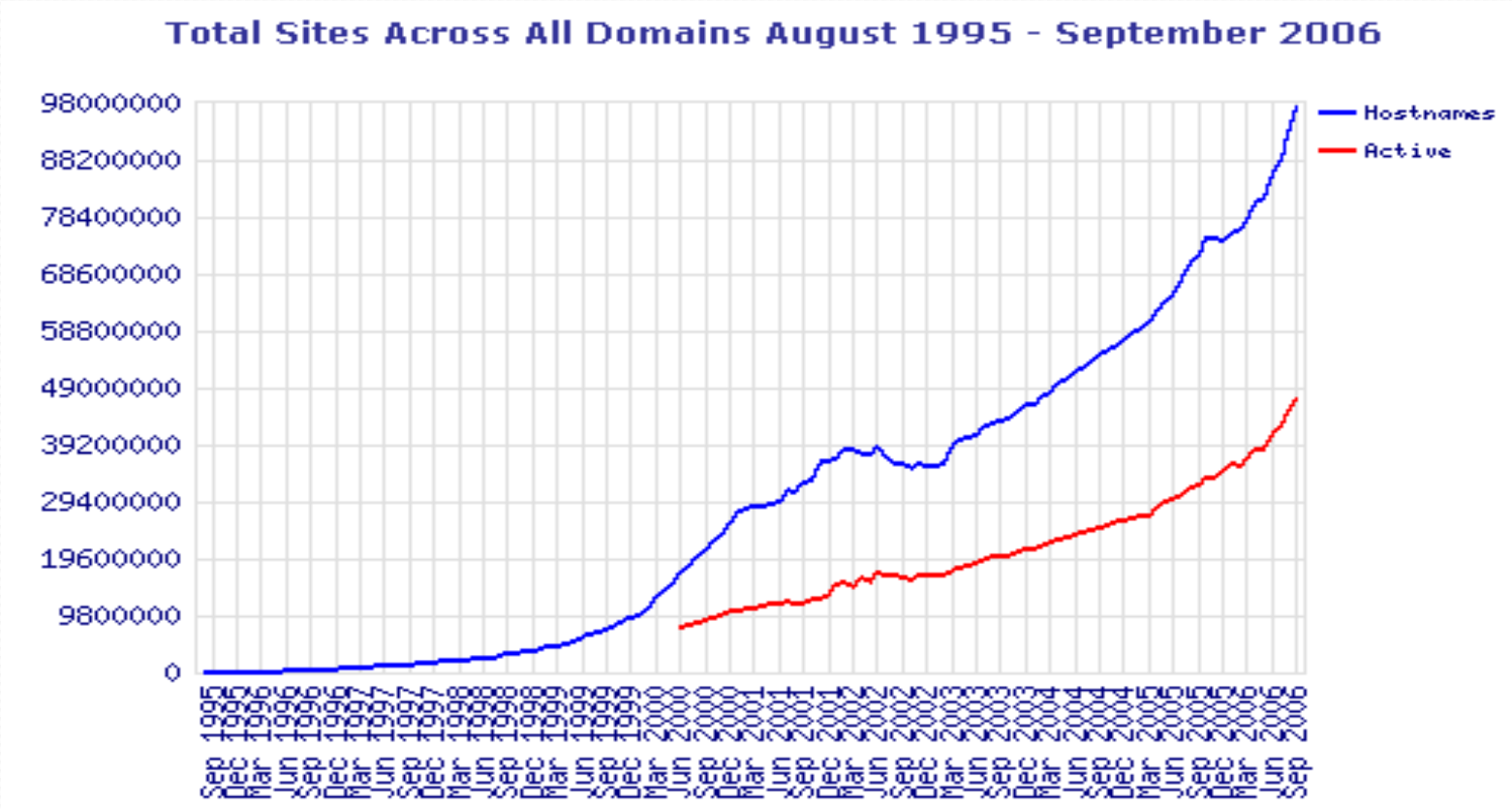
Tema 2. La Web

1. ¿Qué es la Web?
2. Cronología
3. Comparación entre servidores web
4. Funcionamiento del WWW
   - 4.1. Identificadores URL
   - 4.2. Protocolo HTTP
   - 4.3. Lenguaje HTML
5. **Mejoras actuales para el desempeño del WWW**
   - 5.1. Mejoras en el lado del servidor
     - 5.1.1. Caché de disco
     - 5.1.2. Sistemas multi-disco
     - 5.1.3. “Granjas” de servidores
   - 5.2. Mecanismos de caché
     - 5.2.1. Caché jerárquica
     - 5.2.2. Caché proactiva
   - 5.3. *Mirroring* y equilibro por DNS
   - 5.4. Redes de entrega de contenido
   - 5.5 Optimización del *Front-End*

Mejoras actuales para el desempeño del web

- El problema de la Web es: ¡su éxito!
- Su popularidad lleva a la sobrecarga a múltiples "sitios".
- ¿Cómo se puede aliviar el problema de la sobrecarga?
- ¿Cómo podemos organizar la distribución de contenidos?
- ¿Cómo podemos optimizar el sitio desde el punto de vista del contenido?
- Extra: http://cs193h.stevesouders.com/

Crecimiento de la Web



Número máximo de peticiones atendidas

- Problema:

  Número Máximo de Solicitudes ≤
  Número máximo de accesos a disco
  (Nmáx)

ESCASO PARA GRANDES WEBS!!!

$T_{acceso} \approx 1$ mseg

$\Rightarrow$ Nmáx $\approx$ 1000 accesos/seg

Mejoras en el lado del servidor: caché de disco

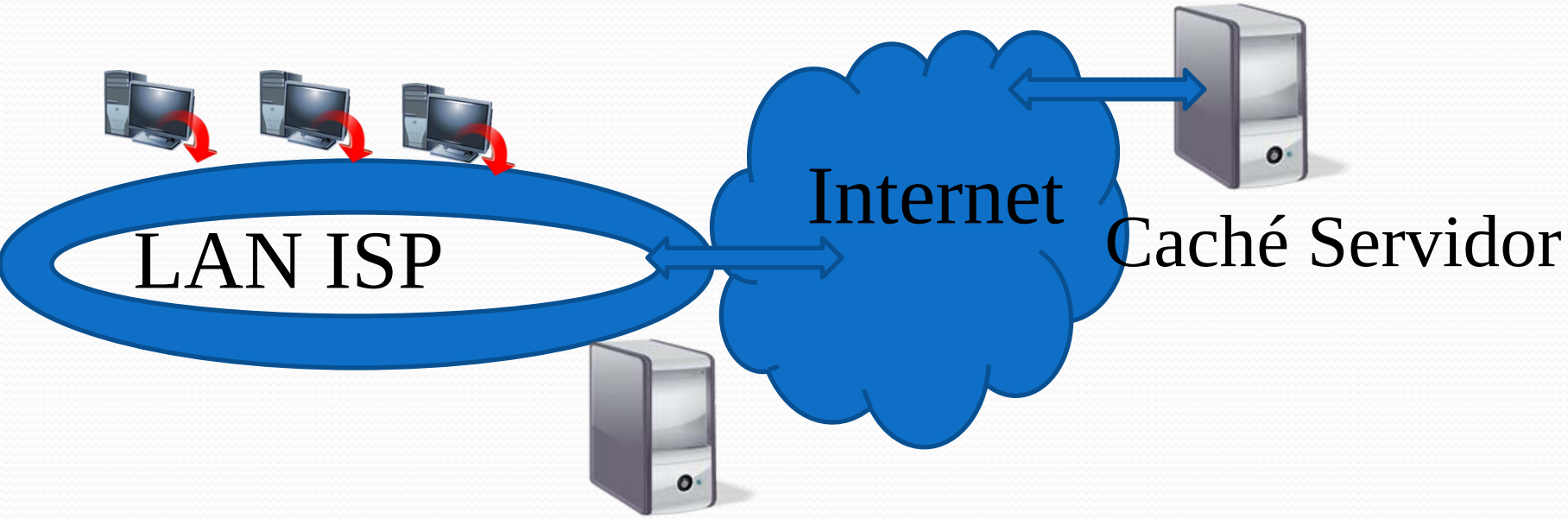
- ¿Cómo mejorar esta cifra?
- Mejora: (en el servidor) caché de las últimas N páginas visitadas.
- No es necesario acceder a disco y el acceso a la memoria RAM es mucho más rápido
  - Pb) Necesidad de mucha memoria RAM

Mecanismos de caché: externa al servidor

- Posible mejora: guardar en puntos diferentes al servidor las páginas con un mecanismo de caché.
  - Dicho proceso es realizado por algún proceso llamado *proxy*.
  - Pueden actuar como proxy el cliente, o elementos intermedios entre cliente y el servidor.
  - Pueden establecerse relaciones jerárquicas entre los proxies.

Mecanismos de caché: caché jerárquica

Caché jerárquica con ISP:



Mecanismos de caché: caché jerárquica

- Pb) “Envejecimiento” de las páginas en los proxies.
  - Sol) Actualización periódica de la caché con heurísticos que determinan tiempo de actualización de las páginas (complejo).
  - Sol) Consulta periódica al servidor para saber si ha cambiado la página (complejo).
- Pb) No funcionan con páginas con contenido generado dinámicamente

Mecanismos de caché: caché proactiva

- Caché proactiva: cuando el proxy obtiene una página web del servidor puede inspeccionarla y pre-cargar en caché las páginas enlazadas en los hipervínculos.
  - Puede reducir el tiempo de acceso, pero también inundar el proxy con material inservible.
  - Mecanismo complejo.

Mejoras en el lado del servidor: sistemas multi-disco

- Mejora: múltiples subprocesos servidores con K discos y memoria compartida de caché.
  - Una sola máquina
  - Aprovechamiento del tiempo: se procesan datos mientras se esperan lecturas desde disco.
  - Necesaria programación eficaz para acceso múltiple a los K discos.
- En la actualidad: in-memory databases
  - Los datos se encuentran en memoria RAM distribuida en múltiples servidores

Mejoras en el lado del servidor: "granjas" de servidores

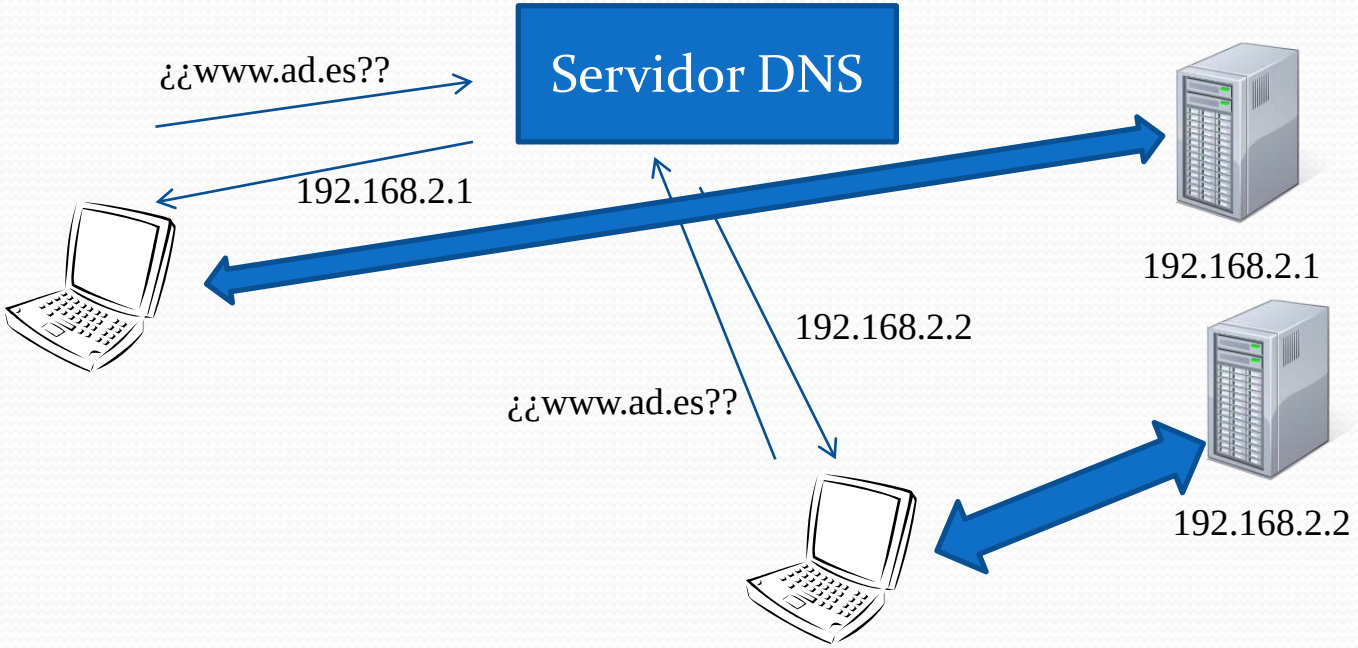
- Mejora: "granja" de servidores (server-farm)
  - Si hay demasiadas peticiones, la CPU no es capaz de manejar la carga de procesamiento, sin importar el número K de discos usados en paralelo.
  - Solución: usar múltiples servidores, pero hay que enrutar las peticiones hacia los distintos servidores
  - Usar un *dispositivo intermedio* que acepta solicitudes y las enruta hacia multiples nodos independientes (cada uno puede contener discos replicados).
    - *PB) SE PIERDE LA CACHÉ COMPARTIDA.*
      - *Sol) dispositivo inteligente.*
    - *PB) RESPUESTA PASA A TRAVÉS DEL DISPOSITIVO INTERMEDIO*
  - Sobrecarga del *dispositivo intermedio:* uso de múltiples servidores que alojan un mismo servicio => Mirroring

Mirroring

- Replicación del servidor:
  - Método muy común usado por los servidores para mejorar su desempeño.
  - Consiste en replicar la información en múltiples ubicaciones separadas considerablemente.
    - En la web principal, el usuario elige manualmente su zona geográfica, y se redirige al "espejo" situado "más cerca".
  - La gestión del mirroring o replicado puede hacerse automáticamente sin intervención del usuario
    - Redirección de URL
    - Equilibrio de carga por DNS

Equilibrio de carga por DNS

- Un mismo dominio tiene asociadas diferentes IPs. Es decir, el servidor/información está replicado en múltiples servidores
- El servidor DNS las sirve según un algoritmo (round robin, por ejemplo)
- Permite equilibrar la carga de los servidores. Muy utilizado



Redes de entrega de contenidos

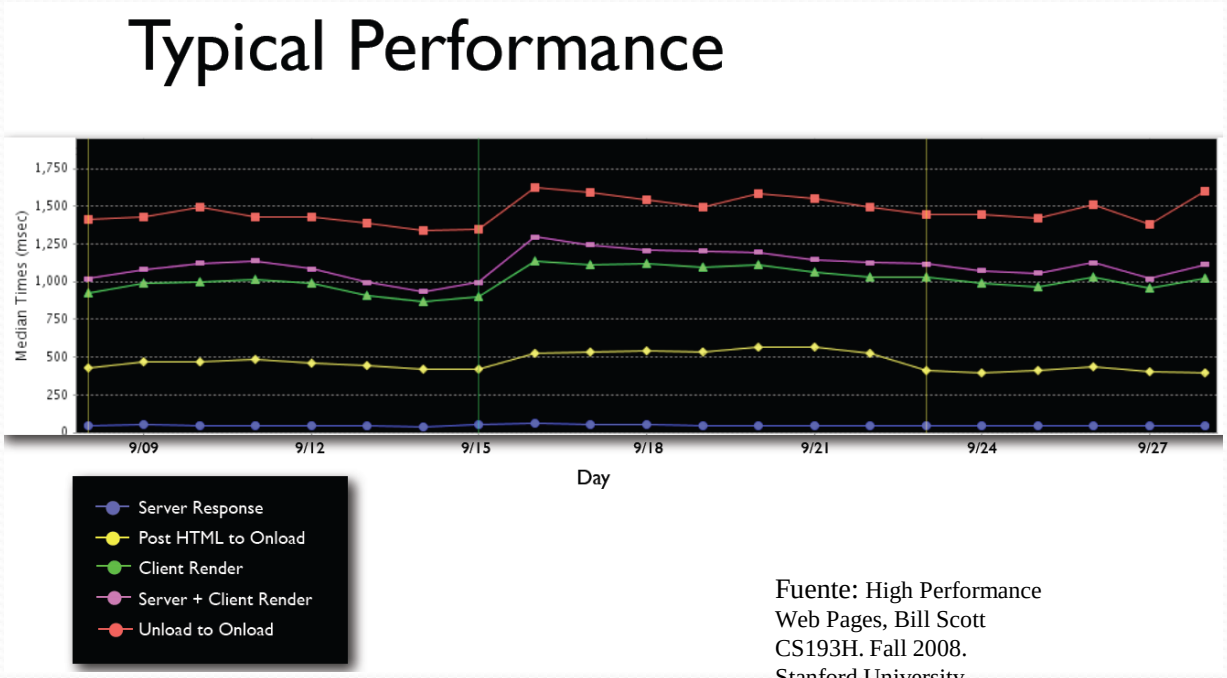
- Redes de entrega de contenido
  - *Content Delivery Network* (CDN)
  - Una CDN es una infraestructura de red especialmente diseñada para servir grandes volúmenes de información
  - Proveedores de contenido contratan con las CDN la entrega de sus contenidos en web: imágenes, programas, mp3, videos, etc.
  - Después, la CDN contrata con ISPs para que le dejen poner su contenido directamente en sus redes locales, en servidores controlados remotamente por la CDN.
  - Los clientes descargan el contenido de los servidores contratados por la CDN, no del proveedor de contenido
    - ¿Cómo se implementa?
    - Ejemplo: nslookup a www.elpais.es

Redes de entrega de contenidos

- Ventajas:
  - Proveedor de contenidos no necesita invertir en infraestructura de red o comunicaciones, sólo genera contenido
  - ISP gestiona todo el tráfico localmente, con lo que no satura sus routers de interconexión con otros ISP
  - Usuarios perciben una mejora en la calidad del servicio
- Ejemplo de CDN: AKAMAI → DECENAS DE MILES DE SERVIDORES.
- ¿Qué es youtube?

Optimización del *front-end*

- En realidad, la respuesta del servidor no es la más influyente en el rendimiento que observa el usuario
- Tiene más peso el procesado de los elementos por parte del navegador



Fuente: High Performance Web Pages, Bill Scott CS193H. Fall 2008. Stanford University

Referencias y bibliografía

- En la Web:
  - www.netcraft.com → Web con estadísticas sobre los servidores
  - http://www.w3schools.com/browsers/browsers_stats.asp → Estadísticas del uso de navegadores
  - www.w3c.org → Consorcio del WWW. Desde aquí es posible consultar todos los estándares relacionados con Web: HTML, HTTP, XML, URI/URL, …

Referencias y bibliografía

- En la Web:
  - www.ietf.org → *Internet Engineering Task Force*. Grupo que trabaja en los protocolos usados en Internet: HTTP, FTP, SIP
  - www.iana.org → Encargados de la asignación de direcciones en Internet
  - Referencias históricas:
    - www.wikipedia.org/wiki/WWW → Muy recomendable

Referencias y bibliografía

- Libros:
  - "Sistemas Distribuidos, Concepto y Diseño", tercera edición, *George Coulouris, Jean Dollimore, Tim Kindberg*, Ed. Addison Wesley, 2001 → Capítulos 1 y 2
  - I. Taylor y A. Harrison, "From P2P and Grids to Services on the Web", 2º Ed., Springer
  - "Redes de Computadoras", cuarta edición, Andrew S. Tanenbaum, 2003 → Sección 7.3