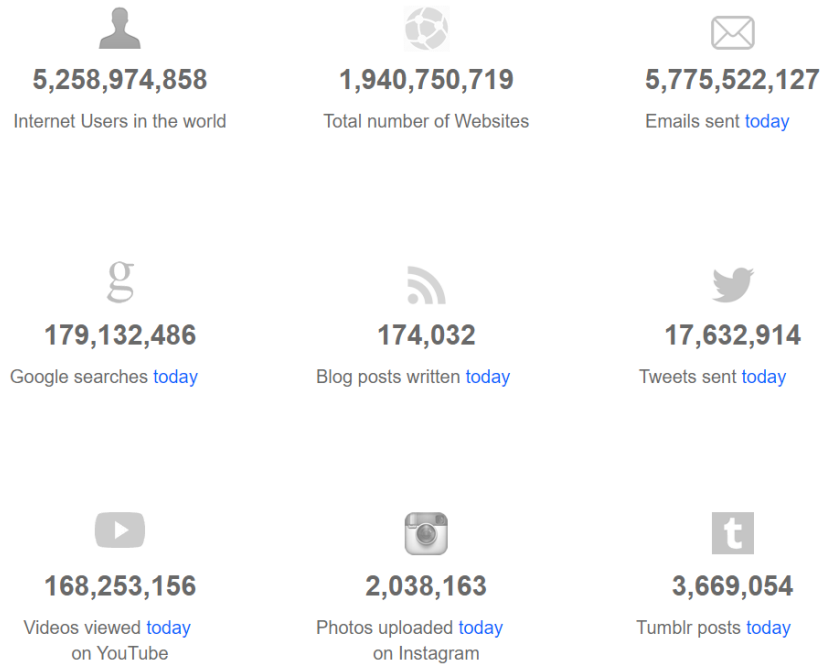
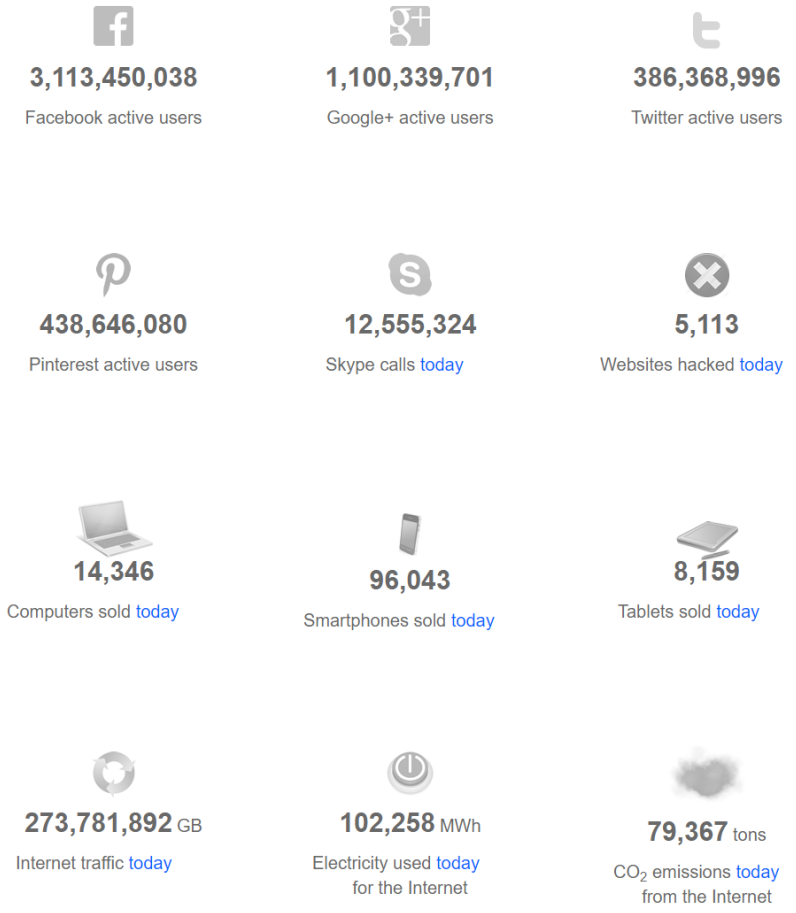
World Wide Web y HTTP

Este material está basado en : Material preparado como apoyo a "*Computer Networking: A Top Down Approach. Featuring the Internet*", 3rd ed. Kurose & Ross Addison-Wesley, July 2004.

World Wide Web y HTTP

- Propuesta por Tim Berners-Lee en 1989.
- Revolucionó la forma de usar Internet.

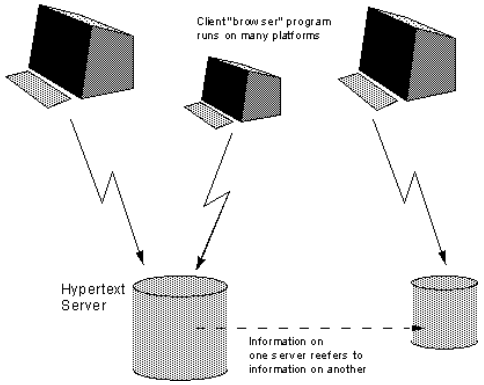




https://www.internetlivestats.com/ (03/04/2022)

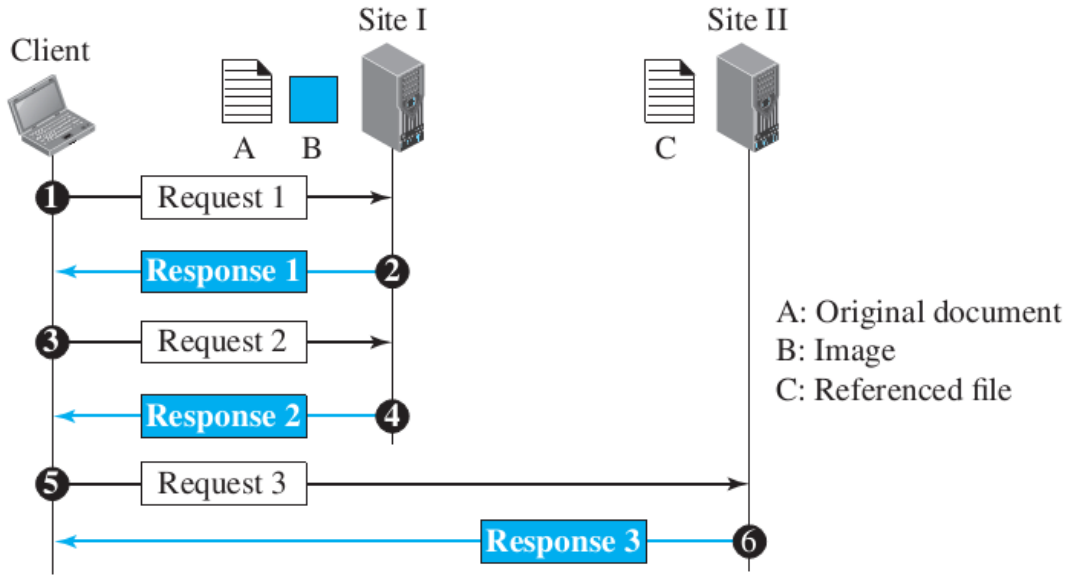
World Wide Web y HTTP

- Consiste de documentos almacenados en servidores que pueden ser accedidos desde equipos clientes.
- Estos documentos se escriben utilizando HTML (Hyper Text Markup Language) que es un lenguaje de etiquetas
- Los documentos web pueden contener referencias a otros documentos u objetos, tanto en el propio servidor como en otros. Tales referencias se denominan hiperenlaces.
- La ubicación de un objeto web se define por su URL (Uniform Resource Locator) que se especifica utilizando el formato:
  - *protocolo://host/ruta*
  - *protocolo://host:puerto/ruta*
- URL se define en los RFC 2396, 3986 y en https://url.spec.whatwg.org/



World Wide Web y HTTP

- Una página Web está compuesta de objetos
- Objetos pueden ser: archivos HTML, imágenes, applets, archivos de audio, vídeo, etc.
- Las páginas Web consisten generalmente de un archivo HTML base que incluye referencias a objetos.
- Cada objeto es referenciable por un Uniform Resource Locator (URL)

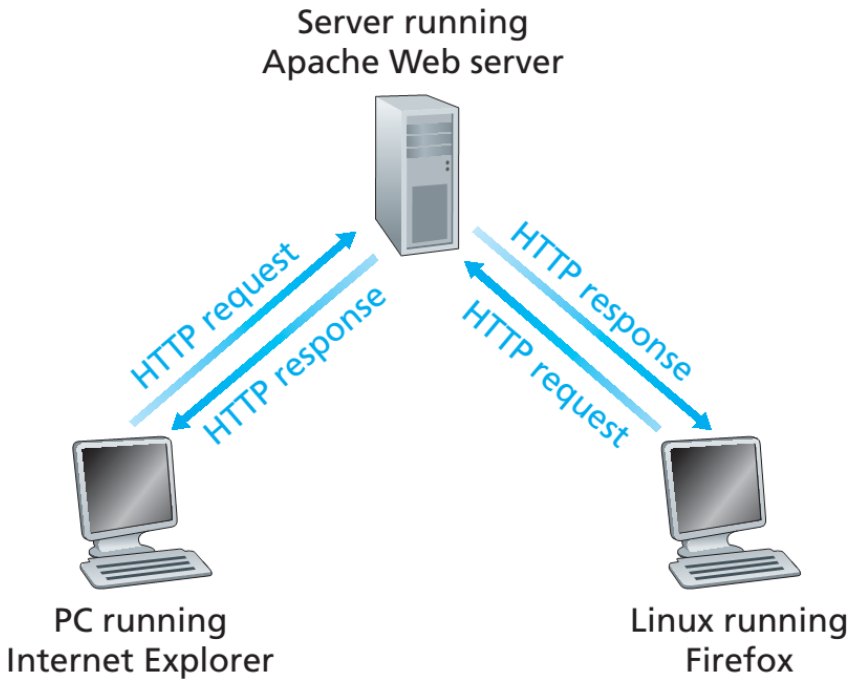


- Ejemplo URL: `www.uandina.edu.pe/dais/index.html`
  - `www.uandina.edu.pe` — Nombre de la máquina
  - `dais/index.html` — Nombre de ruta (path name)

HTTP – Generalidades

- HTTP: HyperText Transfer Protocol
- Protocolo de la capa aplicación de la Web
- Modelo cliente/servidor
  - *cliente:* browser que solicita, recibe, y "despliega" objetos Web
  - *servidor:* Servidor Web envía objetos en respuesta a requerimientos

- HTTP 1.0: RFC 1945 (1996)
- HTTP 1.1: RFC 2068 (1997)



HTTP – Generalidades

Usa TCP:

- Cliente inicia conexión TCP (crea socket) al servidor, puerto 80
- Servidor acepta conexión TCP del cliente
- Mensajes HTTP (mensajes del protocolo de capa aplicación) son intercambiados entre browser (cliente HTTP) y servidor Web (servidor HTTP)
- Se cierra la conexión TCP

HTTP no tiene "estado"

- El servidor no mantiene información sobre los requerimientos del cliente

Protocolos que mantienen "estado" son complejos!

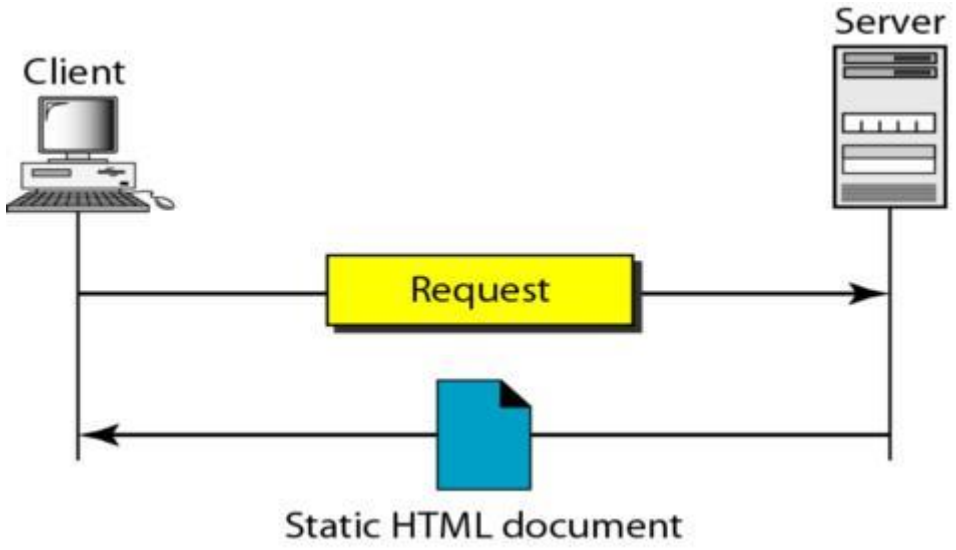
- Historia pasada (estado) debe ser mantenida
- Si servidor o cliente se cae, las vistas del estado pueden ser inconsistentes, deben ser sincronizadas

Tipos de documentos web

- Los documentos web pueden ser:
  - Estáticos
  - Dinámicos
  - Activos

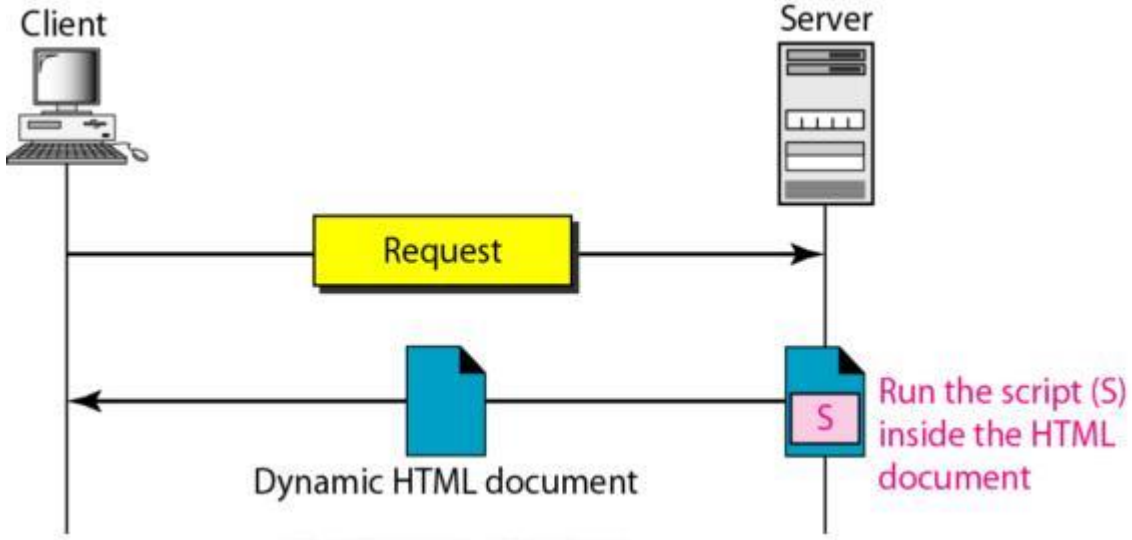
Tipos de documentos web

- Estáticos
  - Documentos de contenido fijo que no se modifica y que solo puede ser recuperado por el cliente.
  - Se redactan en HTML, XML, XSL, XHTML



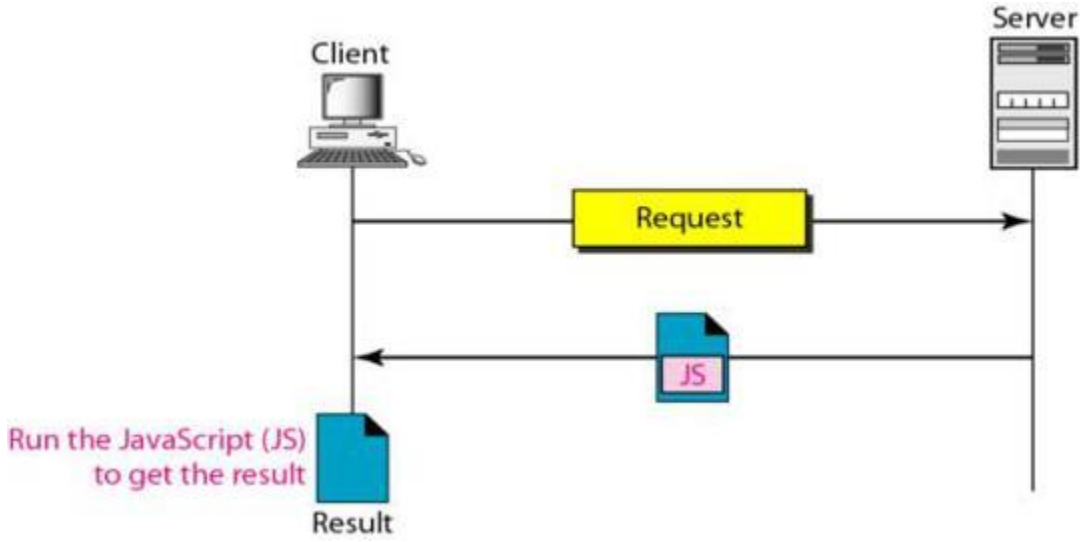
Tipos de documentos web

- Dinámicos
  - Documentos de contenido dinámico que se crean a petición del cliente.
  - Se construyen utilizando CGI, JSP, ASP
  - El código script se ejecuta en el lado del servidor.
  - Se genera información que depende del usuario y la sesión. Por ejemplo, la fecha o datos del usuario



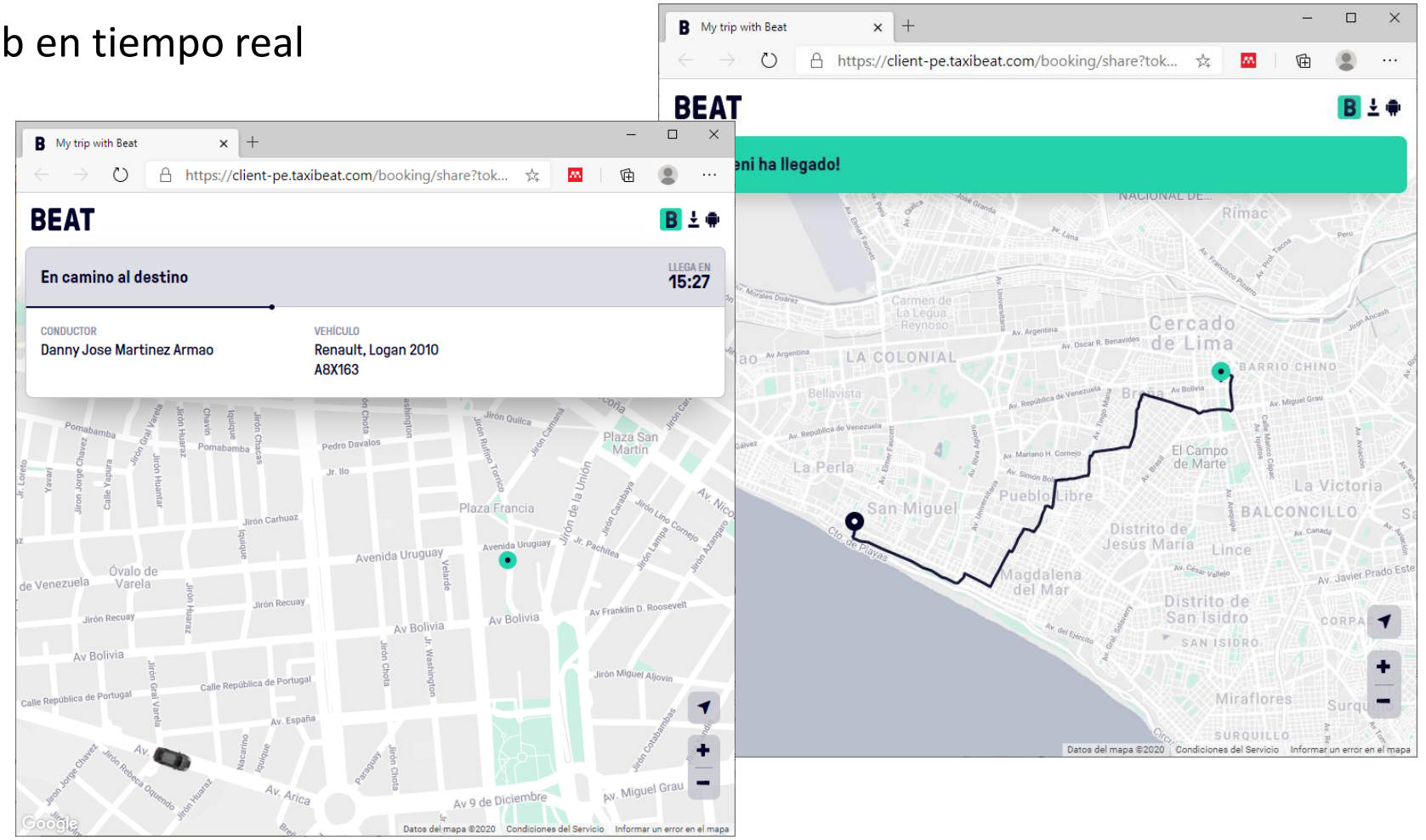
Tipos de documentos web

- Activos
  - Programa o script que se ejecuta en el cliente
  - Se construye utilizando Java Applets, Javascript.
  - También se denominan documentos dinámicos del lado del cliente
  - El código script se envía del servidor al cliente, para que el mismo se ejecute localmente en el cliente.



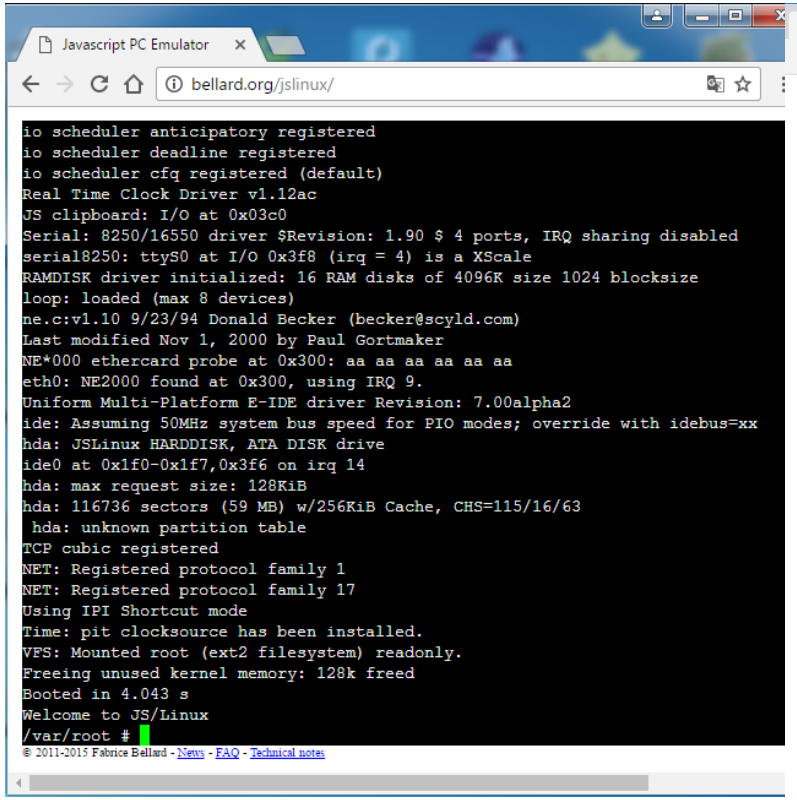
Innovaciones web

- Aplicaciones web en tiempo real



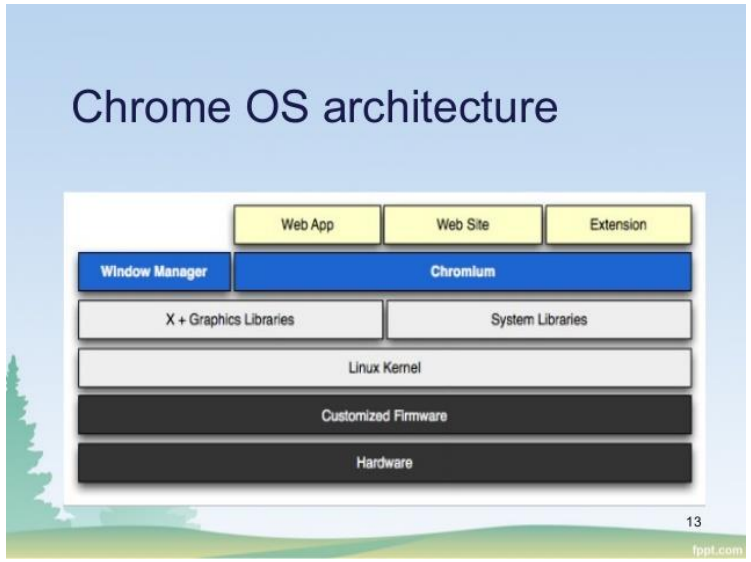
Innovaciones web

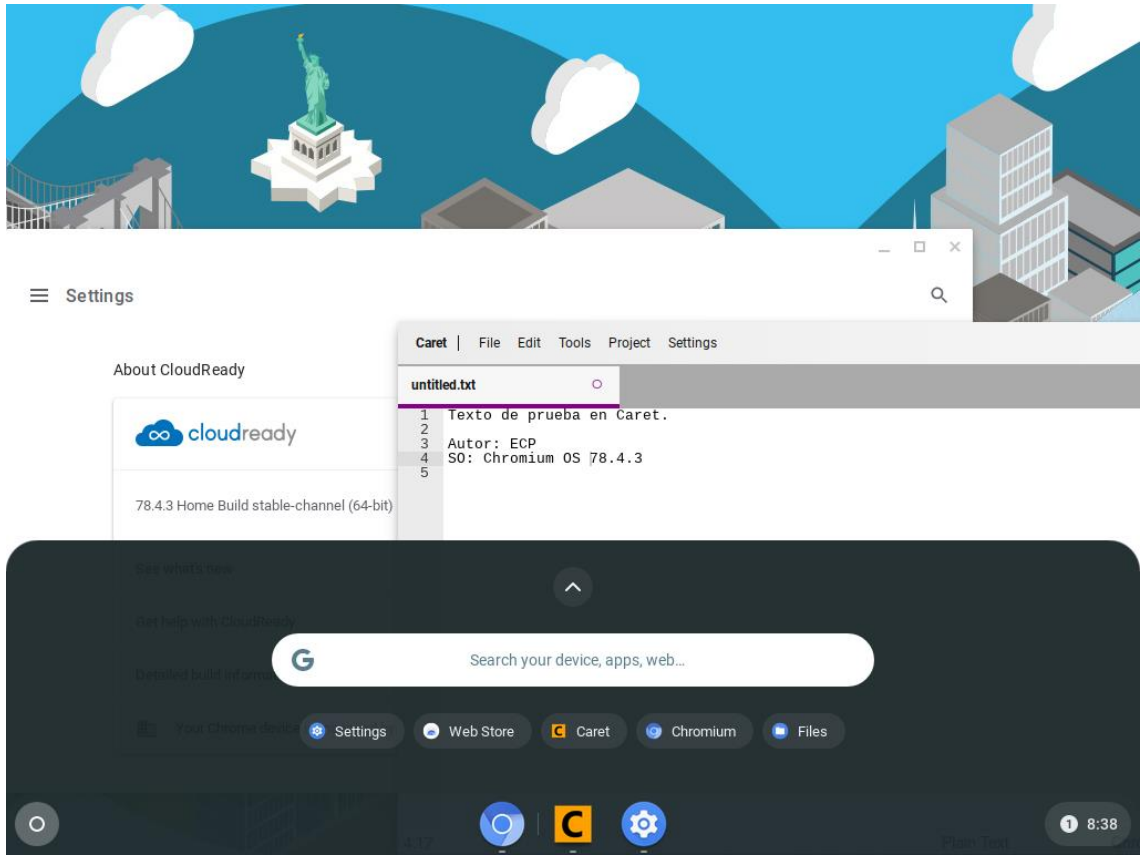
- Simulador de PC escrito en JavaScript, ejecutando Linux y Windows 2000 en navegador web: jslinux



Innovaciones web

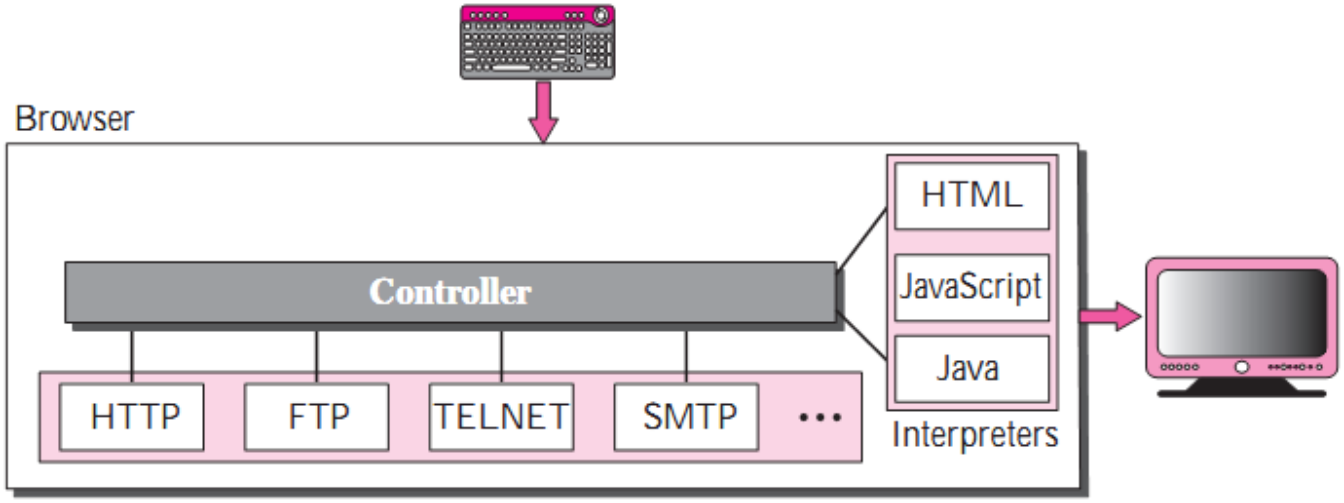
- Sistema operativo basado en tecnología web: Chrome OS





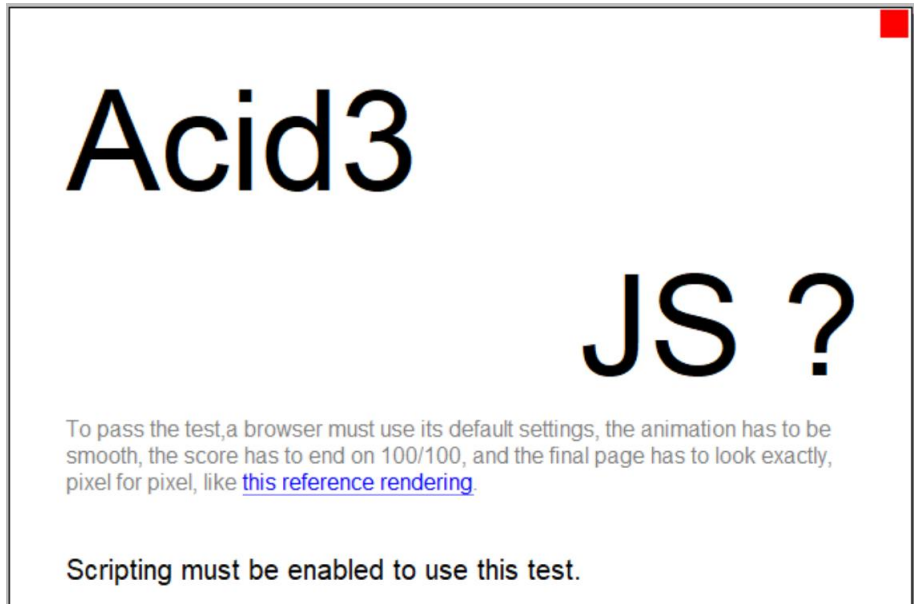
HTTP Generalidades: Arquitectura del Browser

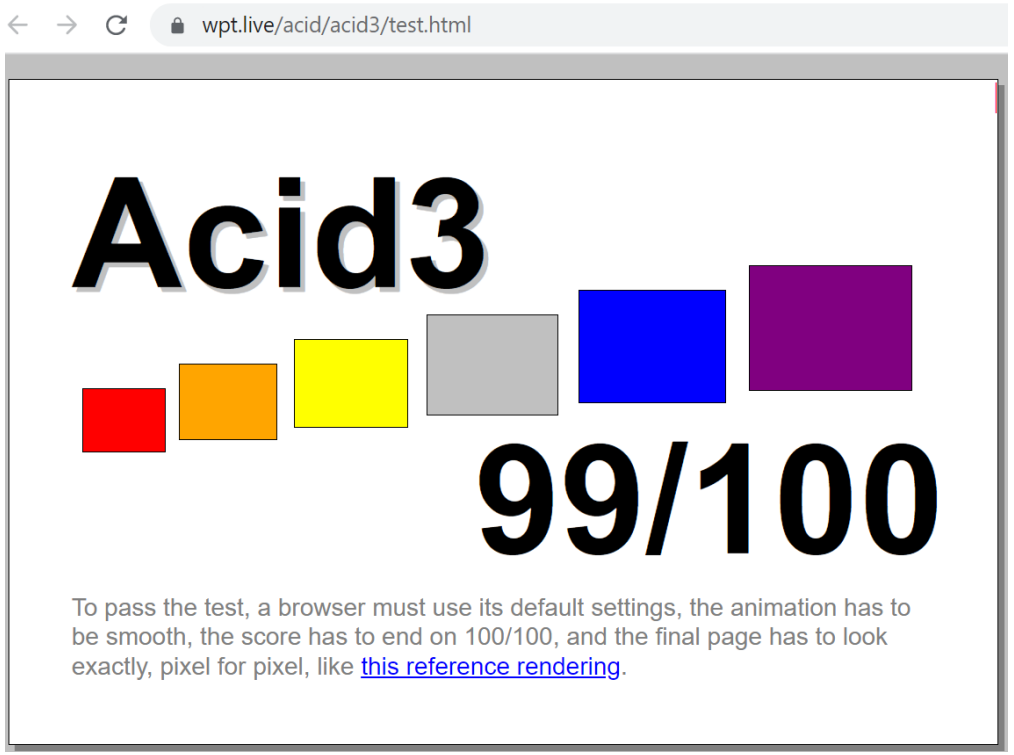
- El controlador recibe una entrada desde el teclado o el mouse y utiliza los programas del cliente para acceder al documento.
- Una vez accedido al documento, el controlador utiliza uno de los interpretes para mostrar el documento en pantalla



HTTP Generalidades: Test del Browser

- El cumplimiento de los estándares HTTP en el navegador web se mide mediante el test denominado Acid3.
- El resultado ideal debería ser 100/100.





Conexiones HTTP

HTTP No-persistente

- A lo más un objeto es enviado por una conexión TCP.
- Es como hacer una llamada por objeto.
- HTTP/1.0 utiliza HTTP no-persistente

HTTP Persistente

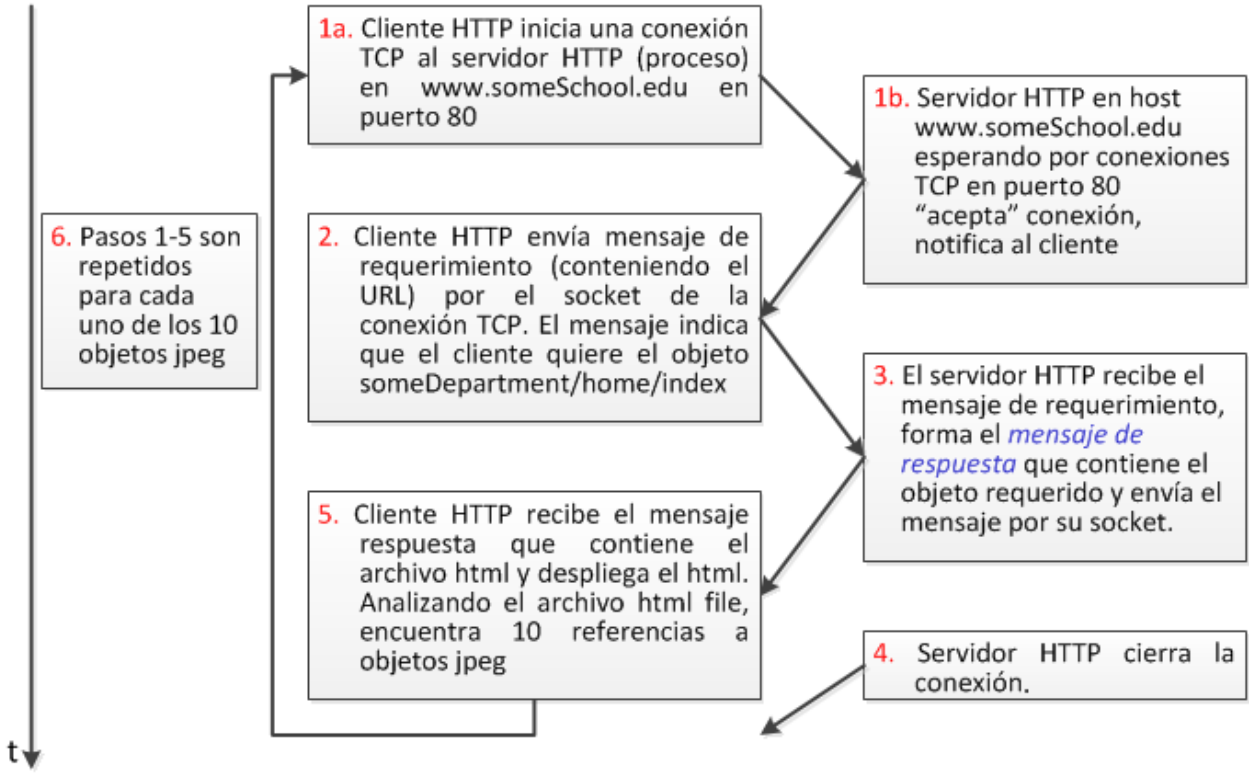
- Múltiples objetos pueden ser enviados por una única conexión TCP entre el cliente y servidor.
- HTTP/1.1 usa conexiones persistentes en su modo por defecto

HTTP no-persistente

Supongamos que el usuario ingresa URL

`www.someSchool.edu/someDepartment/home/index`

(Contiene texto y referencia a 10 imágenes jpeg)



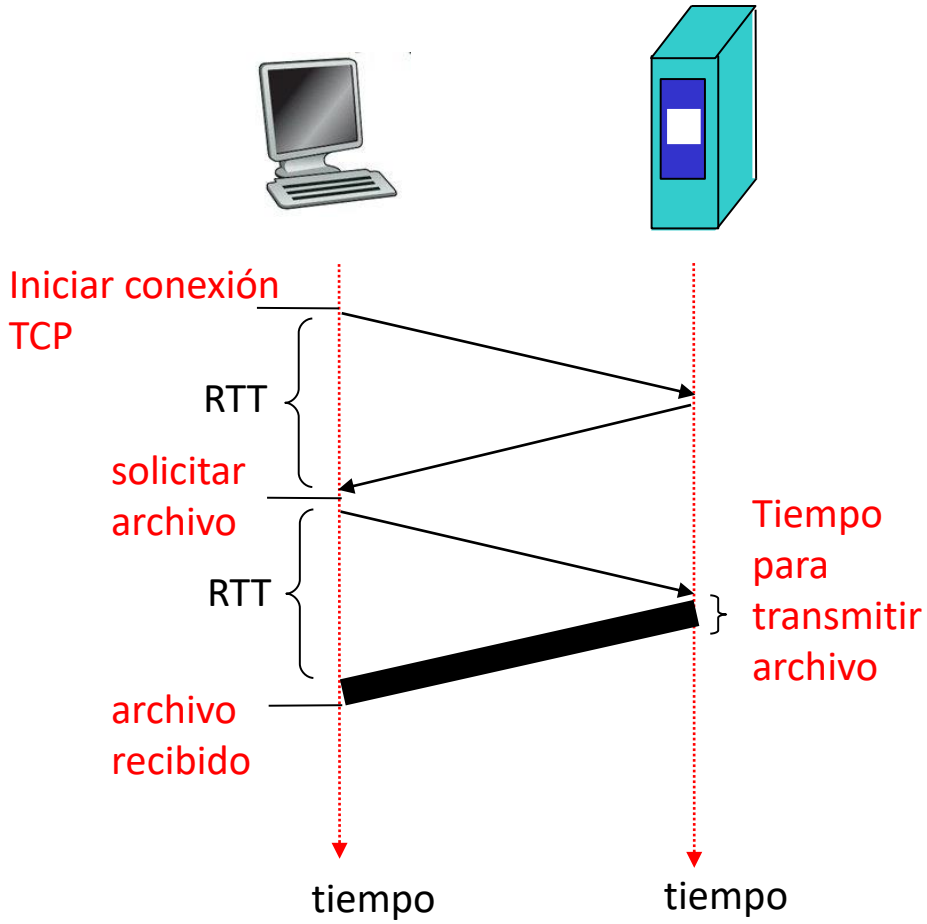
HTTP no persistente: tiempo de respuesta

Definición de RTT(round-trip time): tiempo ocupado en enviar un paquete pequeño desde el cliente al servidor y su regreso.

Tiempo de respuesta:

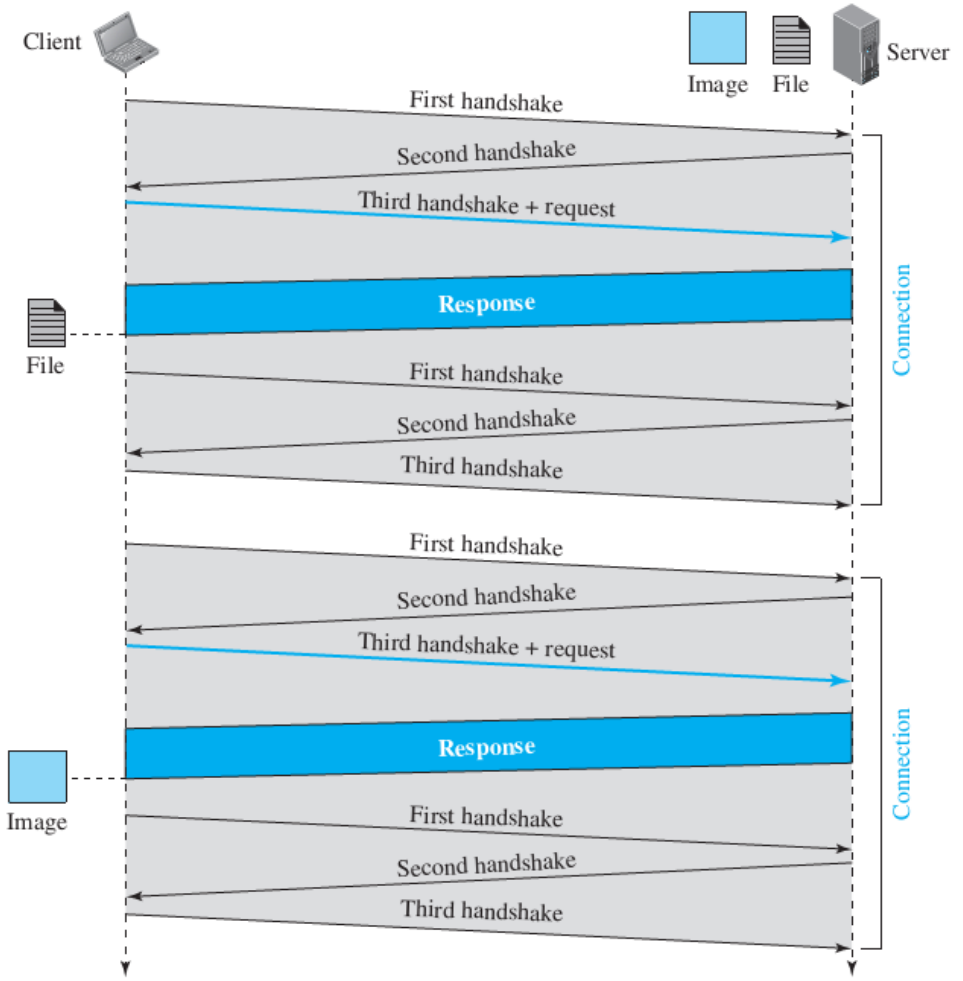
- Un RTT para iniciar la conexión TCP
- Un RTT por requerimiento HTTP y primeros bytes de la respuesta
- Tiempo de transmisión del archivo

total = 2RTT + tiempo de transmisión



HTTP no persistente

- En caso de tener que recuperar múltiples objetos, la penalidad puede ser alta



HTTP Persistente

Problemas de HTTP no-persistente:

- Requiere 2 RTT por objeto
- El navegador abre conexiones paralelas generalmente para traer objetos referenciados. => el OS debe trabajar y dedicar recursos para cada conexión TCP

HTTP Persistente

- Servidor deja las conexiones abiertas después de enviar la respuesta
- Mensajes HTTP subsecuentes entre los mismos cliente/servidor son enviados por la conexión abierta

Persistencia sin pipelining:

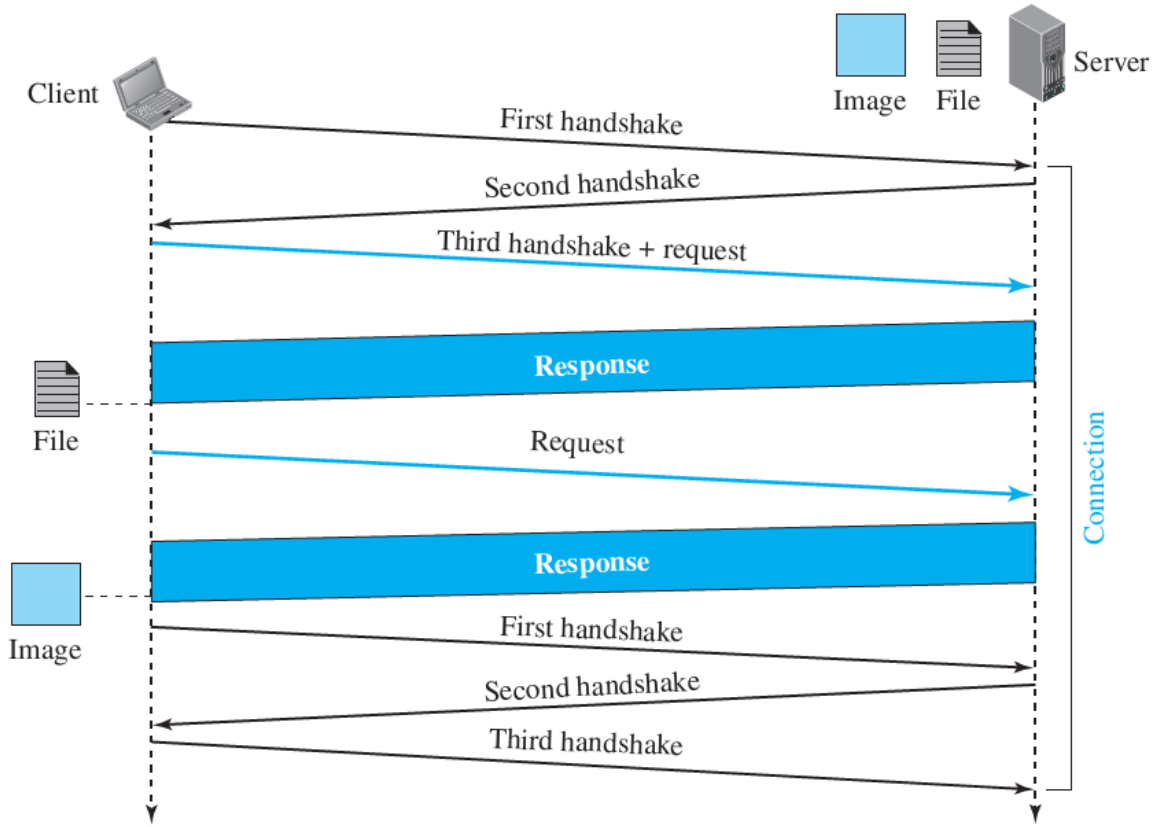
- Cliente envía nuevo requerimiento sólo cuando el previo ha sido recibido
- Un RTT por cada objeto referenciado

Persistencia con pipelining:

- Por defecto en HTTP/1.1
- Cliente envía requerimientos tan pronto éste encuentra un objeto referenciado
- Tan poco como un RTT para todas las referencias

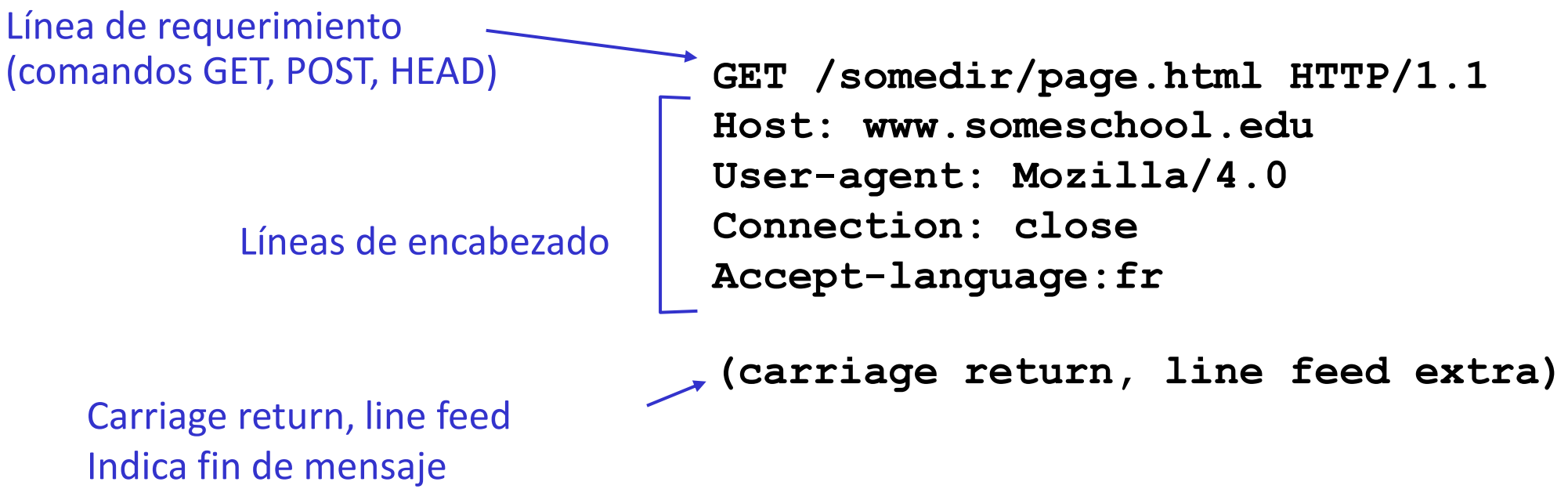
HTTP Persistente

- Mas eficiente, pues permite la transferencia de múltiples objetos a través de una sola conexión

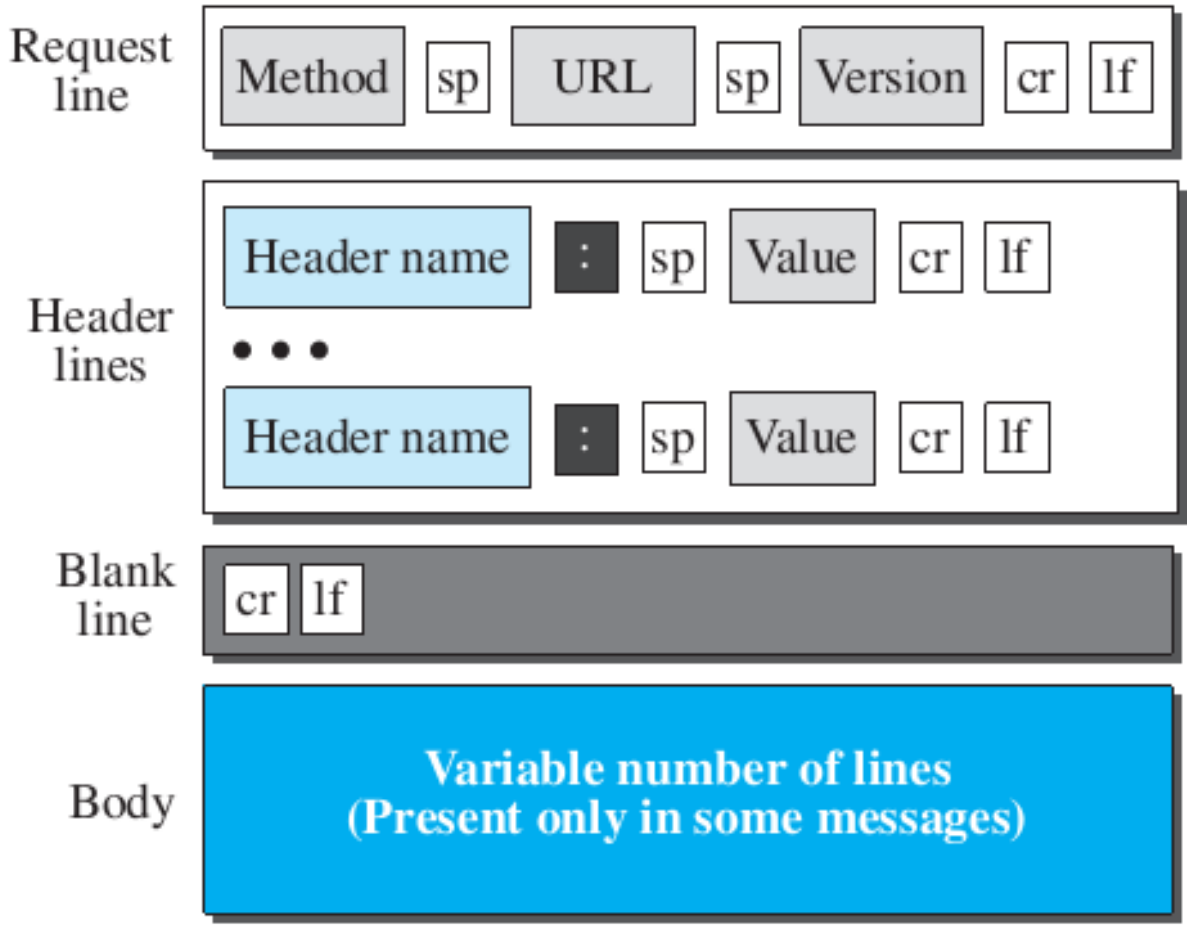


Mensaje HTTP de requerimiento

- Existe dos tipos de mensajes HTTP: *Requerimiento, Respuesta*
- Mensaje de requerimiento HTTP:
  - ASCII (es decir, formato legible)



Mensaje HTTP de requerimiento: formato general



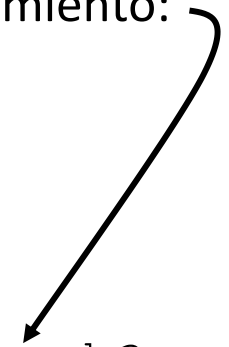
Subiendo input de formulario

Método POST:

- Páginas Web usualmente incluyen entradas de formularios
- Los datos son subidos al servidor en el cuerpo del mensaje

Métodos URL:

- Usa método GET
- Entrada es subida en campos URL de la línea de requerimiento:



```
www.somesite.com/animalsearch?monkeys&banana
```

Tipos de Métodos

HTTP/1.0

- GET
- POST
- HEAD: Pide al servidor que deje el objeto requerido afuera de la respuesta. Respuesta incluye sólo el encabezado.

HTTP/1.1

- GET, POST, HEAD
- PUT: Sube archivos en cuerpo del requerimiento en localización indicada por el campo URL
- DELETE: Borra archivo especificado en el campo URL

METODO	ACCION
GET	Solicitar un documento del servidor
HEAD	Solicitar información sobre un documento pero no el propio documento
PUT	Envía un documento del cliente al servidor
POST	Envia alguna información del cliente al servidor
TRACE	Repite y devuelve la solicitud entrante
DELETE	Elimina un objeto web
CONNECT	Reservado
OPTIONS	Consulta sobre opciones disponibles

Nombres de Headers de requerimiento

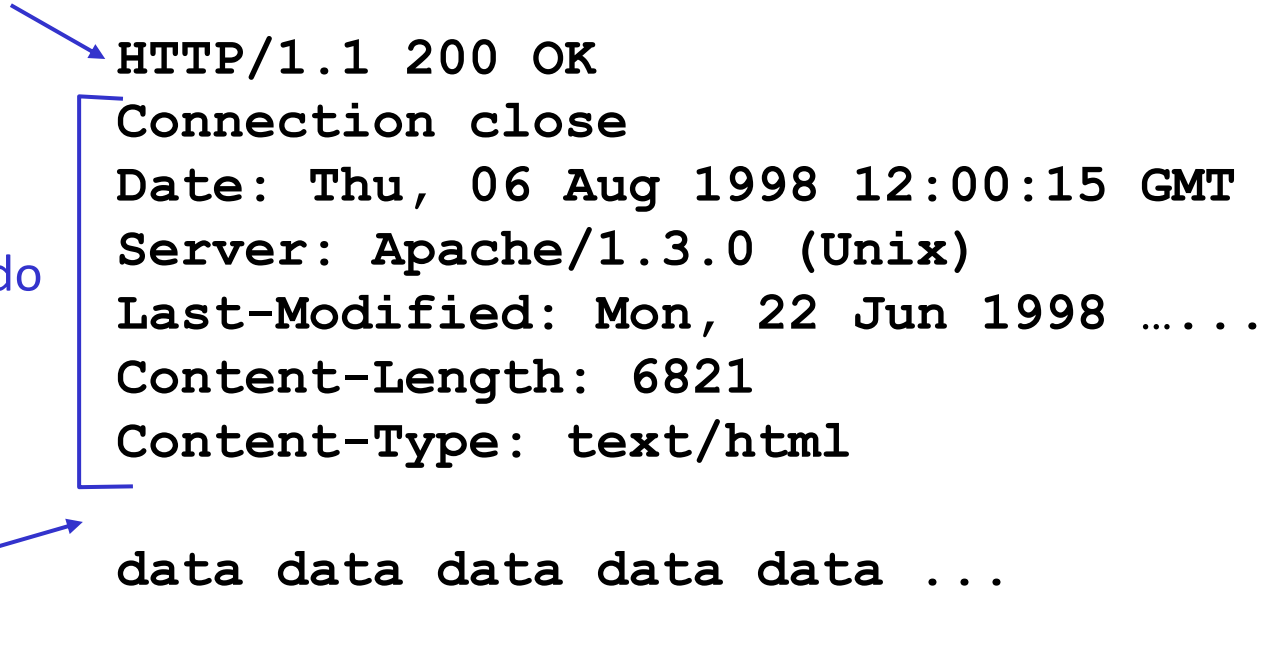
HEADER	DESCRIPCION
User-agent	Identifica al programa cliente
Accept	Muestra el formato del medio que el cliente puede aceptar
Accept-charset	Muestra el juego de caracteres que el cliente puede manejar
Accept-encoding	Muestra el esquema de codificación que el icliente puede manejar
Accept-language	Muestra el lenguaje que el cliente puede aceptar
Authorization	Muestra los permisos que tiene el cliente
Host	Muestra el número de puerto y host del cliente
Date	Muestra la fecha actual
Upgrade	Especifica el protocolo de comunicación preferido
Cookie	Devuelve el cookie al servidor
If-Modified-Since	Si el archivo se modificó desde una fecha específica

Mensajes HTTP de respuesta

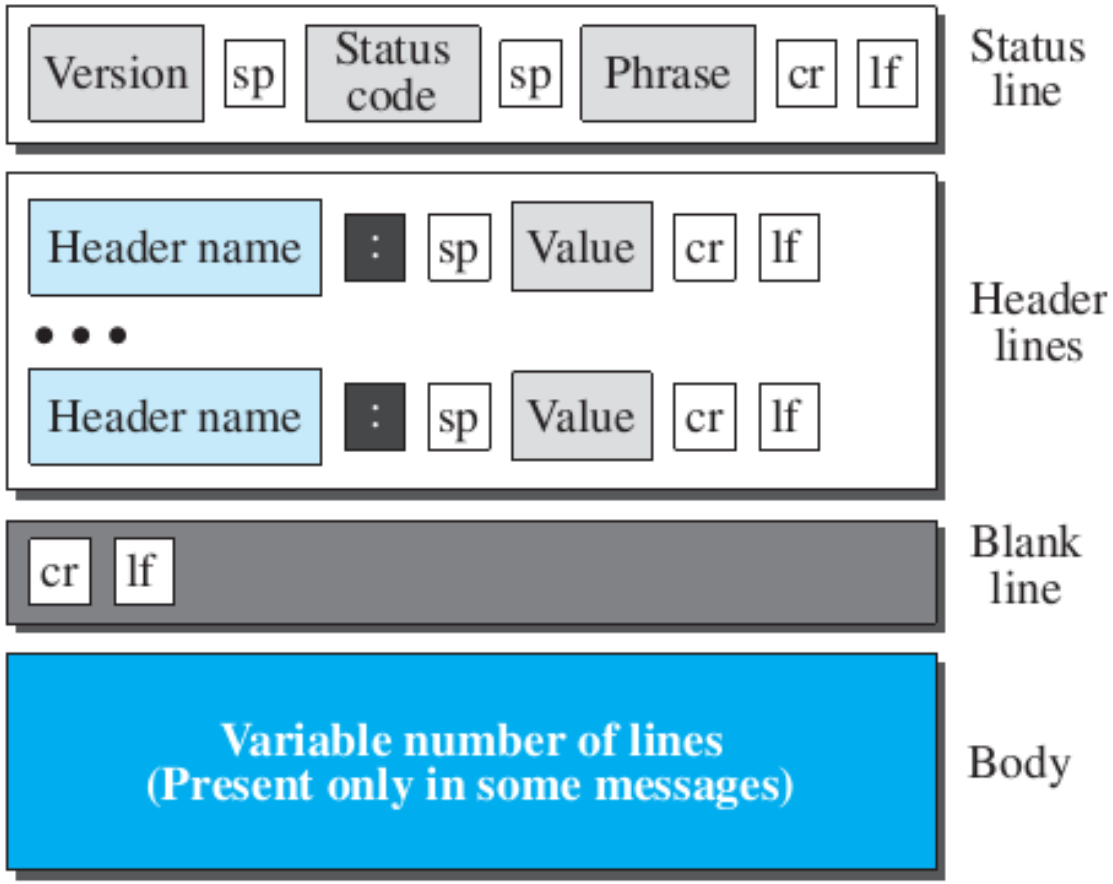
Línea de estados
(código de estado del protocolo
Frase de estado)

Líneas de encabezado

data, e.g.,
Archivo HTML solicitado



Mensajes HTTP de respuesta: formato general



Nombres de Headers de respuesta

HEADER	DESCRIPCION
Date	Muestra la fecha actual
Upgrade	Especifica el protocolo de comunicación preferido
Server	Brinda información sobre el servidor
Set-Cookie	El servidor solicita al cliente guardar una cookie
Content-Encoding	Especifica el esquema de codificación
Content-Language	Especifica el lenguaje
Content-Length	Muestra la longitud del documento
Content-Type	Especifica el tipo de medio
Location	Para pedir al cliente que envíe la petición a otro sitio
Last-Modified	Indica la fecha y hora del último cambio

Códigos HTTP de respuesta

Son devueltos en primera línea de respuesta del servidor al cliente.

Algunos códigos de muestra:

200 OK: Solicitud exitosa, objeto requerido es incluido luego en mensaje

204 No Content: Requerimiento tuvo existo, pero el cliente no necesita salir de su página actual

301 Moved Permanently: Se movió el objeto requerido, nueva ubicación es especificada luego en el mensaje (Location:)

400 Bad Request: Requerimiento no entendido por el servidor

401 Unauthorized: Falta de credenciales de autenticación válidas para el recurso requerimiento

403 Forbidden: Requerimiento entendido pero no autorizado por el servidor

404 Not Found: Documento no encontrado en servidor

409 Conflict: Requerimiento en conflicto con el estado actual del recurso solicitado

505 HTTP Version Not Supported: Version HTTP usada en la petición no es soportada por el servidor

Probando HTTP (lado cliente)

1. Telnet a nuestro servidor favorito:

```
telnet danbalthaser.net 80
```

Telnet abre una conexión TCP al puerto 80 (puerto servidor HTTP por omisión) en danbalthaser.net
Cualquier cosa ingresada es enviada al puerto 80 del servidor

2. Escribir un requerimiento GET HTTP:

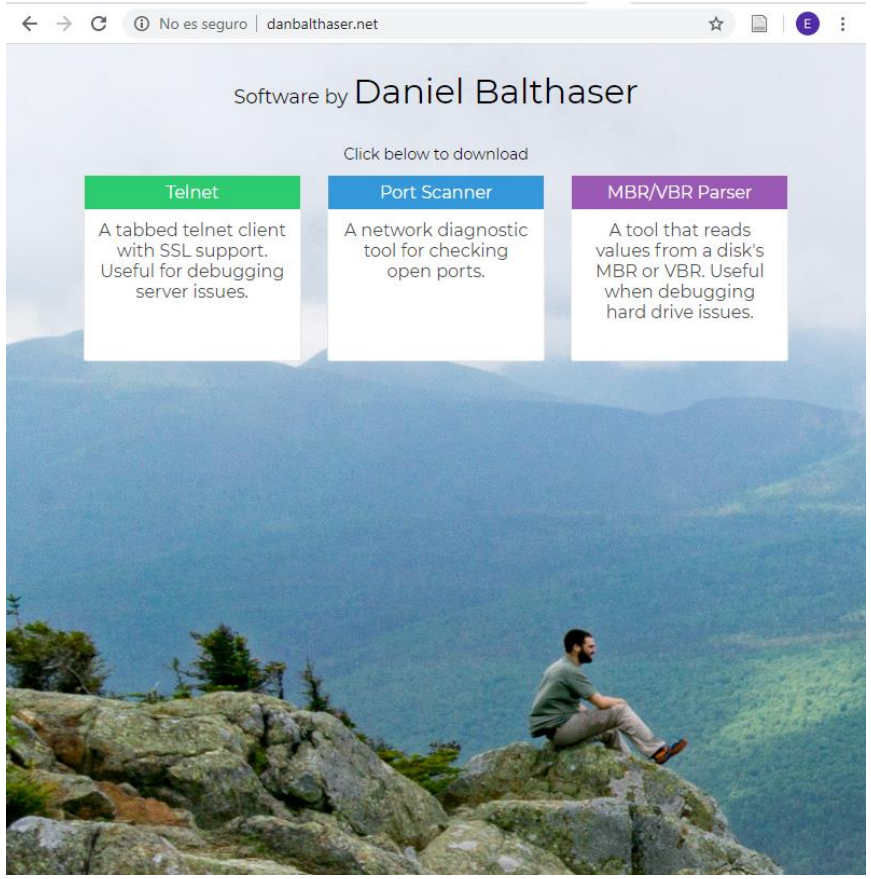
```
GET / HTTP/1.1
Host: danbalthaser.net

NOTA: Campo Host: obligatorio
En encabezado, requerido por
proxy
```

Tecleando esto, enviamos un requerimiento *GET* mínimo (pero completo) al servidor HTTP

3. Observar el mensaje de respuesta enviado por el servidor HTTP

Probando HTTP



Interacción usuario-servidor: cookies

Muchos sitios Web importantes usan cookies

- Las cookies fueron implementadas para permitir personalizar la información Web.
- Cookie es información generada por un servidor web y almacenada en el computador del usuario para **acceso** futuro.
- Las cookies son trasportadas entre el computador del usuario y el servidor.
- Por ejemplo, se usan cookies para almacenar ítems en un carro de compra mientras se recorre una tienda virtual.

Estado usuario-servidor: cookies

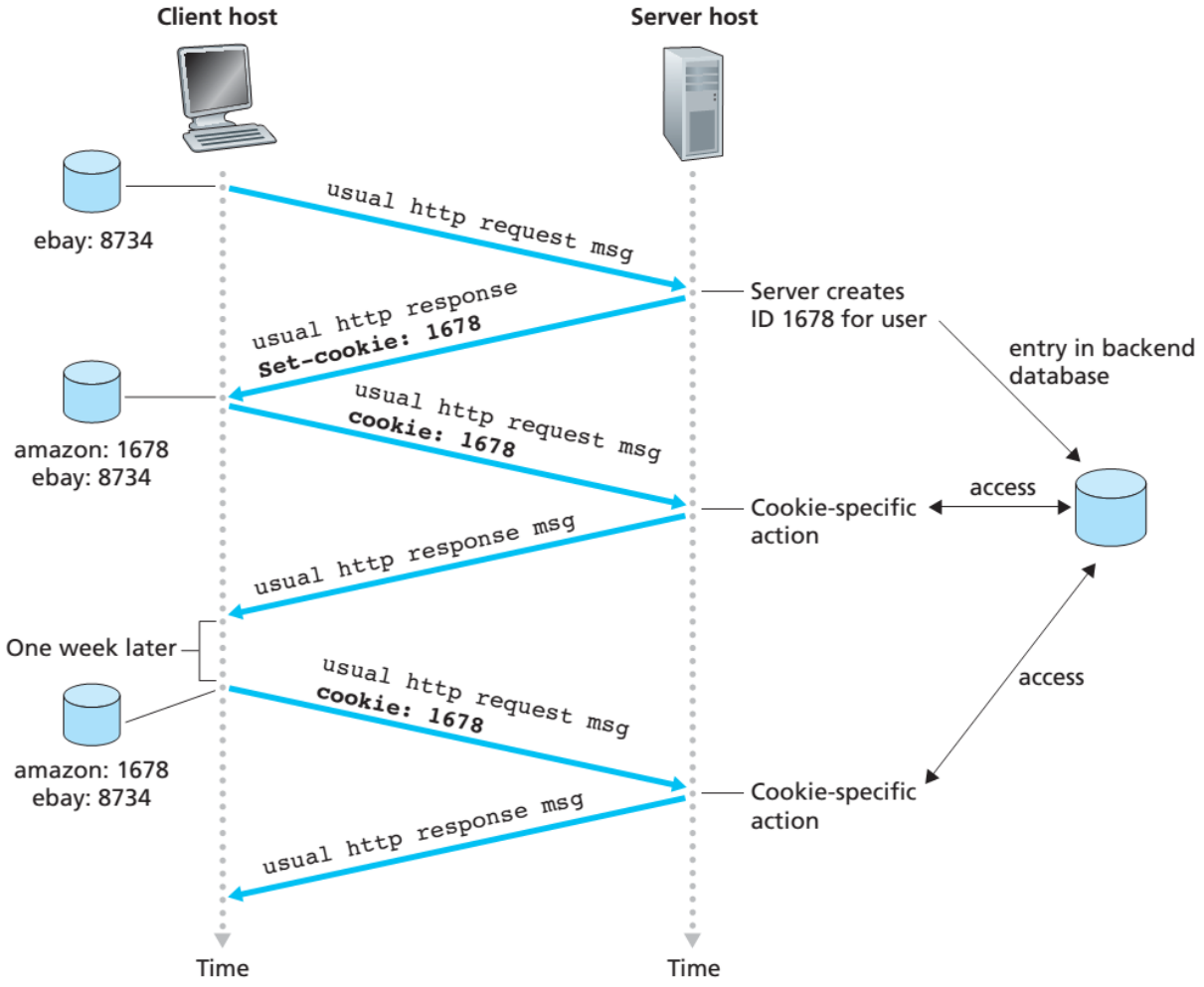
Cuatro Componentes:

1) Línea encabezado `cookie` en el mensaje respuesta HTTP
2) Línea de encabezado `cookie` en requerimiento HTTP
3) Archivo cookie es almacenado en la máquina del usuario y administrada por su navegador.
4) Base de datos en sitio Web

Ejemplo:

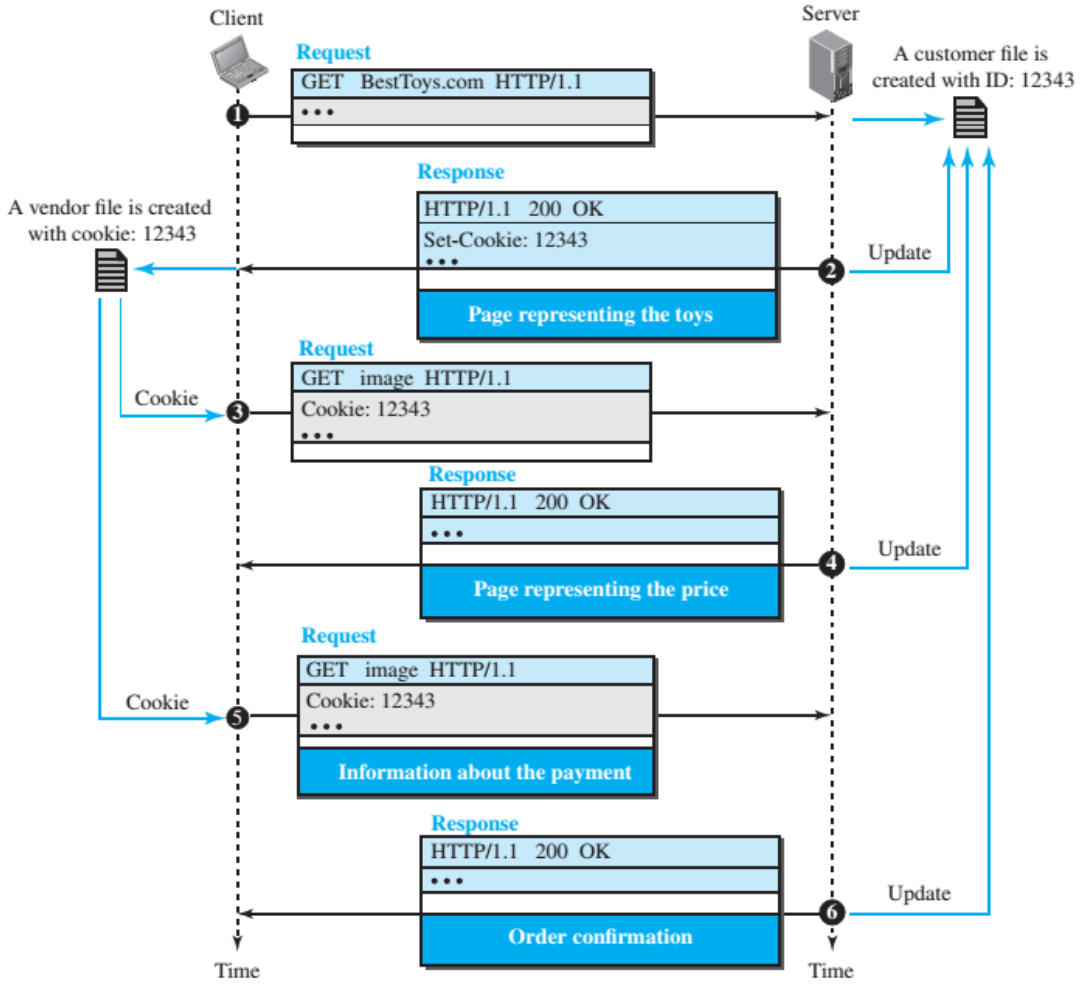
- Susana accede a Internet siempre desde el mismo PC
- Ella visita un sitio de e-commerce específico por primera vez.
- Cuando el requerimiento HTTP inicial llega al sitio, éste crea un ID único y crea una entrada en la base de datos para ese ID.

Cookies: conservando el “estado”



Cookies: conservando el “estado”

Ejemplo del uso de cookies para almacenar los datos necesarios de un usuario, para realizar una transacción comercial a través de Internet



Cookies

Qué pueden transportar las cookies:

- Autorización
- Carrito de compras
- Sugerencias
- Estado de la sesión del usuario (Web e-mail)

Al margen

Cookies y privacidad:

- Cookies permiten que el sitio aprenda mucho sobre uno.
- Podríamos proveer nombre y correo al sitio.
- Motores de búsqueda usan redirecciones y cookies para aprender aún más
- Compañías de avisos obtienen información de los sitios WEB

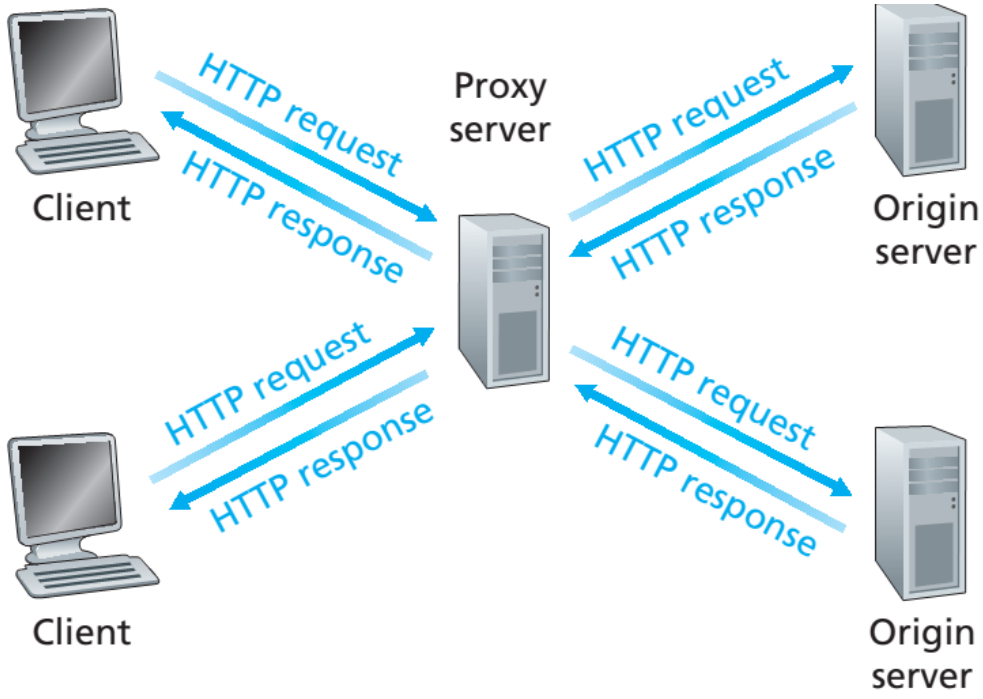
Cookies

```
function set_cookie(name, value, exp_y, exp_m, exp_d, path,
                                                  domain,
   secure) {
    var cookie_string = name + "=" + escape(value);
    if (exp_y) {
      var expires = new Date(exp_y, exp_m, exp_d);
      cookie_string += "; expires=" + expires.toGMTString();
    }
    if (path)    cookie_string += "; path=" + escape(path);
    if (domain) cookie_string += "; domain=" +
      escape(domain);
    if (secure) cookie_string += "; secure";
    document.cookie = cookie_string;
```

Web caches (también servidores proxy)

Objetivo: Satisfacer el requerimiento del cliente sin involucrar al servidor destino.

- Usuario configura el browser: Acceso Web vía cache
- Browser envía todos los requerimientos HTTP al cache
  - Si objeto está en cache: cache retorna objeto
  - Si no, cache requiere los objetos desde el servidor Web, y retorna el objeto al cliente



Caches v/s proxy

- La idea del cache es almacenar “localmente” datos ya solicitados y así poder acceder a los mismos datos más rápidamente en el futuro.
  - Un problema que debe atender el cache es la obsolescencia que puede tener los datos locales.
  - El cache puede usar tiempos de expiración, o consultar a la fuente por vigencia del dato local.

- Un proxy es un servicio que consiste en realizar una solicitud a pedido de otro.
- Por ejemplo podemos usar proxy para acceder a servicios externos de una intranet, para que desde fuera no se sepa que computadores hay dentro. El origen es siempre el mismo.

Más sobre Web cache

- Cache actúan como clientes y servidores
- Típicamente el cache está instalado por ISP (universidad, compañía ISP residencial)

Por qué Web caching?

- Reduce tiempo de respuesta de las peticiones del cliente.
- Reduce tráfico en el enlace de acceso al ISP.
- Internet densa con caches permite a proveedores de contenido “pobres” (no $$) entregar contenido en forma efectiva.

Ejemplo de Cache

Suposiciones

- Tamaño promedio de objetos: 1Mb
- Tasa de requerimientos promedio desde browsers de la institución al servidor WEB: 15 por segundo
- Retardo desde el enrutador institucional a cualquier servidor web y su retorno: 2s (retardo de Internet)

Consecuencias

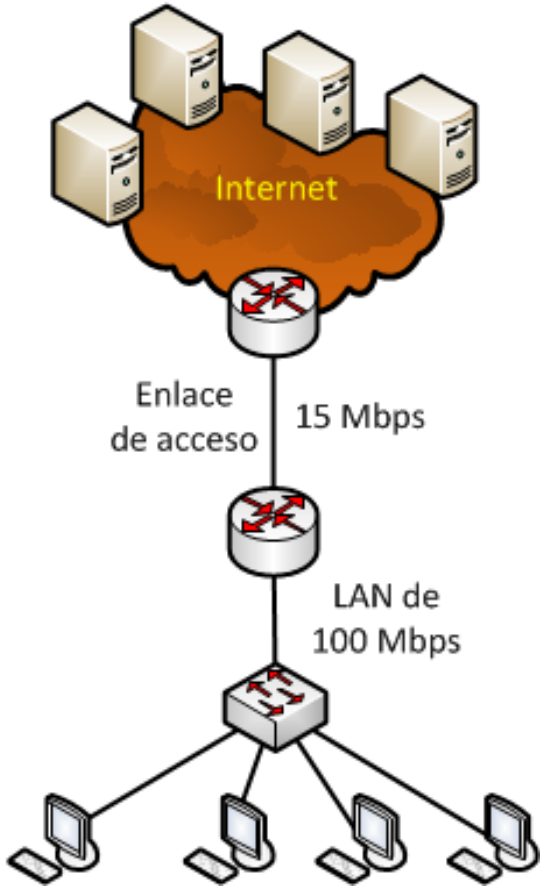
- Utilización de la LAN:

  (15 peticiones/s)*(1Mb/petición)/100Mbps = 0.15 (15%)
- Utilización del enlace de acceso(enrutador de Internet – enrutador institucional):

  (15 peticiones/s)*(1Mb/petición)/15Mbps = 1 (100%)

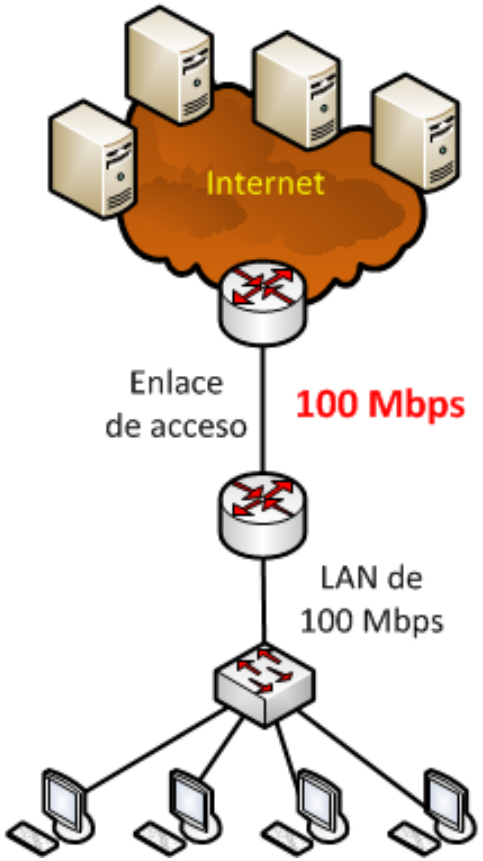
$$Retardo_{total} = Retardo_{LAN} + Retardo_{acceso} + Retardo_{Internet}$$

$$Retardo_{total} = milisegundos + minutos + 2s$$



Ejemplo de Cache

Sin cache institucional



Posible solución

- Aumentar ancho de banda del enlace de acceso a 100 Mbps

Consecuencias

- Utilización de la LAN = 15%
- Utilización del enlace de acceso = 15%

$$Retardo_{total} = Retardo_{LAN} + Retardo_{acceso} + Retardo_{Internet}$$

$$Retardo_{total} = milisegundos + milisegundos + 2\ segundos$$

A menudo un upgrade caro.

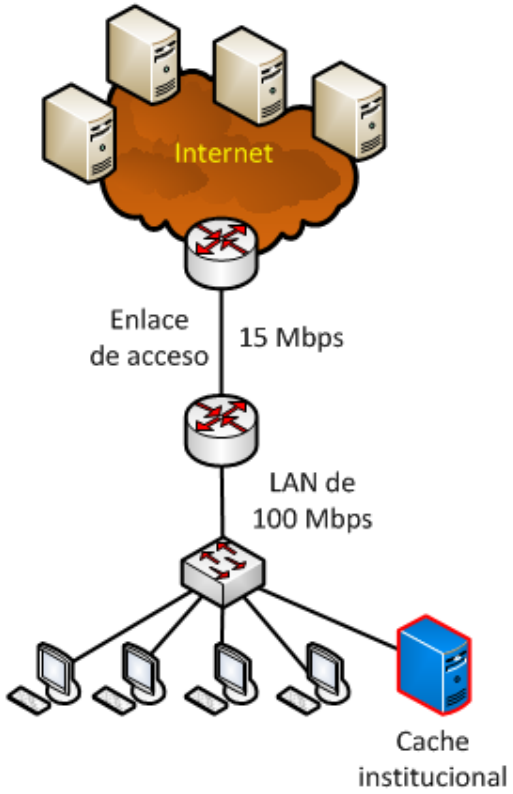
Ejemplo de cache

Instalar un web Cache

- Supongamos tasa de éxito (acierto) de 0.4

Consecuencias

- 40% de los requerimientos serán satisfechos en forma casi inmediata por el servidor cache (~10 ms)
- 60% de los requerimientos satisfechos por los servidores WEB de origen
- Utilización del enlace de acceso es reducido al 60%, resultando en retardo despreciable (digamos 10 ms)



$$Retardo_{total} = Retardo_{LAN} + Retardo_{acceso} + Retardo_{Internet}$$

$$Retardo_{total} = 0.4\ x\ 0.01\ s + 0\ s + 0.6\ x\ 2.01\ s = 1.21\ s$$

Esta solución es más económica que la alternativa previa y garantiza menor retardo !!!

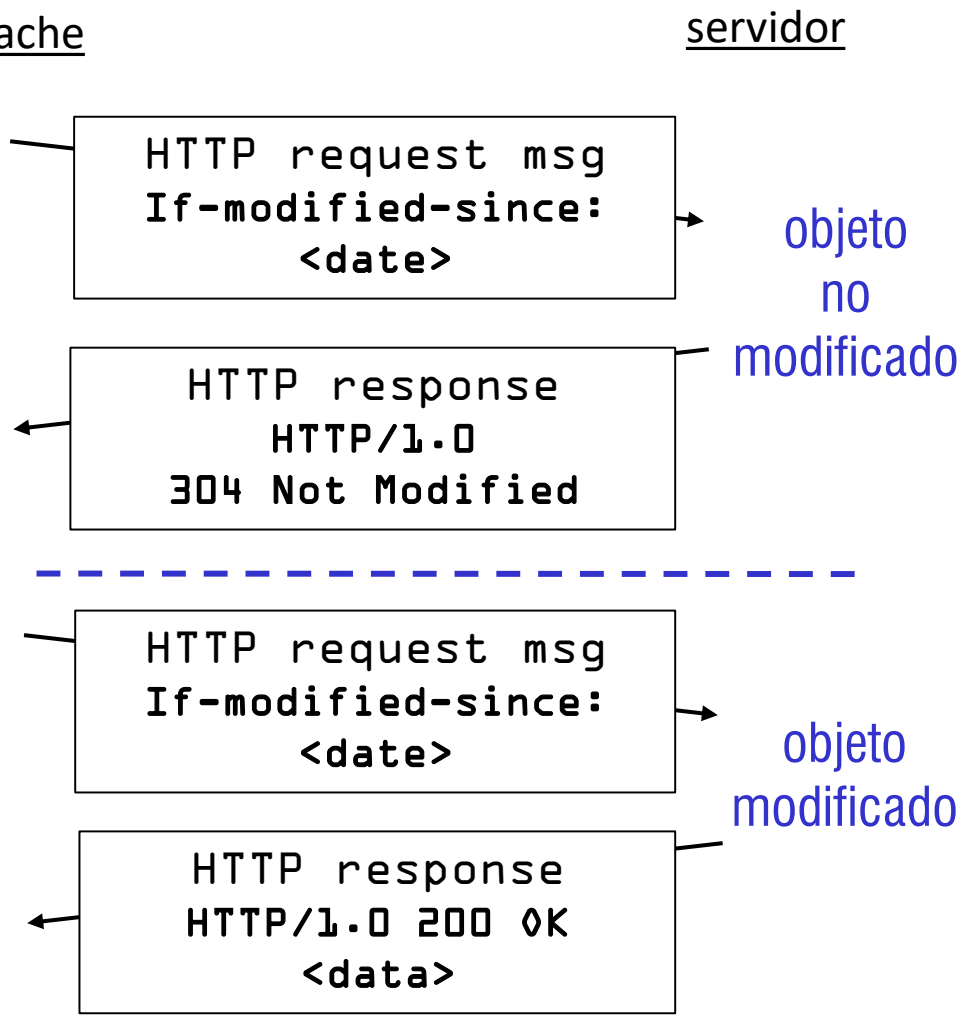
Get Condicional

- Objetivo: no enviar objetos si el cache tiene la versión actualizada
- Cache: especifica la fecha de la copia en el requerimiento HTTP

  If-modified-since: <date>

- Servidor: responde sin el objeto si la copia de la cache es la última. :

  HTTP/1.0 304 Not Modified



TRABAJO DE SESIÓN

- Prepare una presentación, mostrando las modificaciones incorporadas en HTTP/2 y HTTP/3

Referencias

- http://info.cern.ch/hypertext/WWW/TheProject.html
- https://www.w3.org/History/1989/proposal.html
- https://wpt.live/acid/acid3/test.html
- https://csvbase.com/blog/2