Capítulo 2: Capa Aplicación - II

Este material está basado en:

- material de apoyo al texto *Computer Networking: A Top Down Approach Featuring the Internet 3rd* edition. Jim Kurose, Keith Ross Addison-Wesley, 2004.

Capítulo 2: Capa Aplicación

- 2.1 Principios de la aplicaciones de red
- 2.2 Web y HTTP
- 2.3 FTP
- 2.4 Correo Electrónico
  - SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 P2P Compartición de archivos
- 2.7 Programación de Socket con TCP
- 2.8 Programación de socket con UDP
- 2.9 Construcción de un servidor WEB

Web y HTTP

Primero algo de jerga

- Una página Web consiste de objetos
- Objetos pueden ser archivos HTML, imágenes (JPEG, GIF,…), Java applets, archivos de audio, archivos de video,…
- Páginas Web consisten de un archivo HTML base el cual incluye referencias a objetos
- Cada objeto es direccionable por un Universal Resource Locator (URL)
- Ejemplo URL:

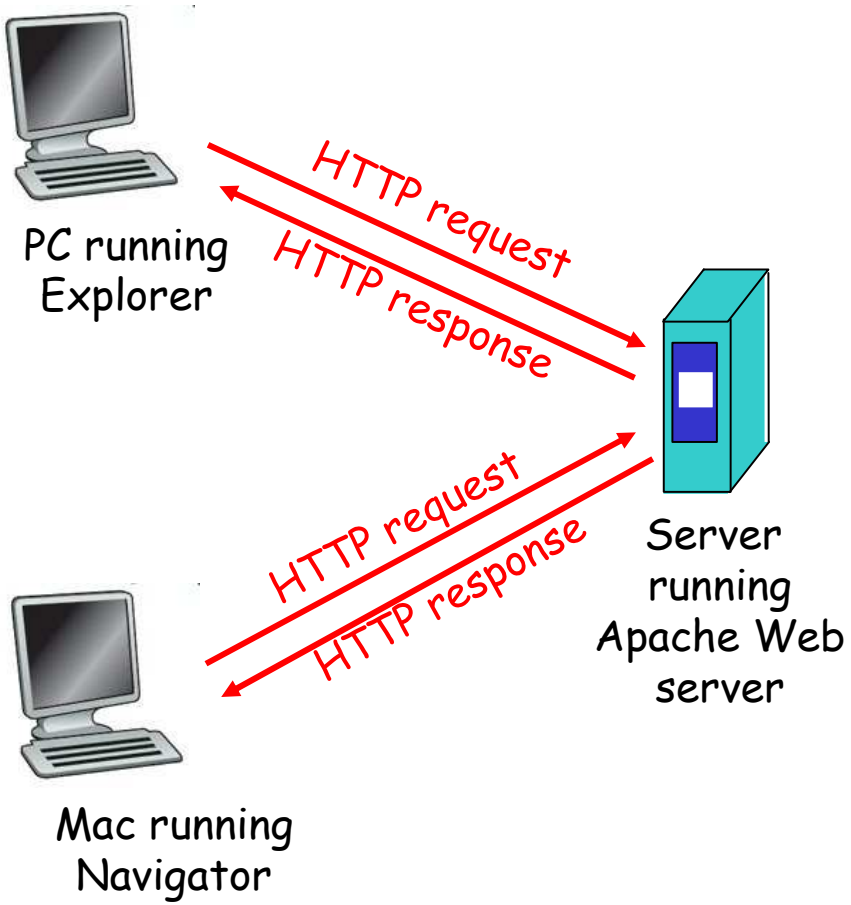
www.elo.utfsm.cl/images/logoelo.png

Nombre de la máquina Nombre de camino (path name)

HTTP Generalidades

HTTP: hypertext transfer protocol

- Protocolo de la capa aplicación de la Web
- Modelo cliente/servidor
  - *cliente:* browser que requiere, recibe, "despliega" objetos Web
  - *servidor:* Servidor Web envía objetos en respuesta a requerimientos
- HTTP 1.0: RFC 1945
- HTTP 1.1: RFC 2068



HTTP generalidades (cont.)

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 clientes

Protocolos que mantiene "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

Conexiones HTTP

HTTP No-persistente

- A lo más un objeto es enviado por una conexión TCP.
- HTTP/1.0 usa HTTP no-persistente

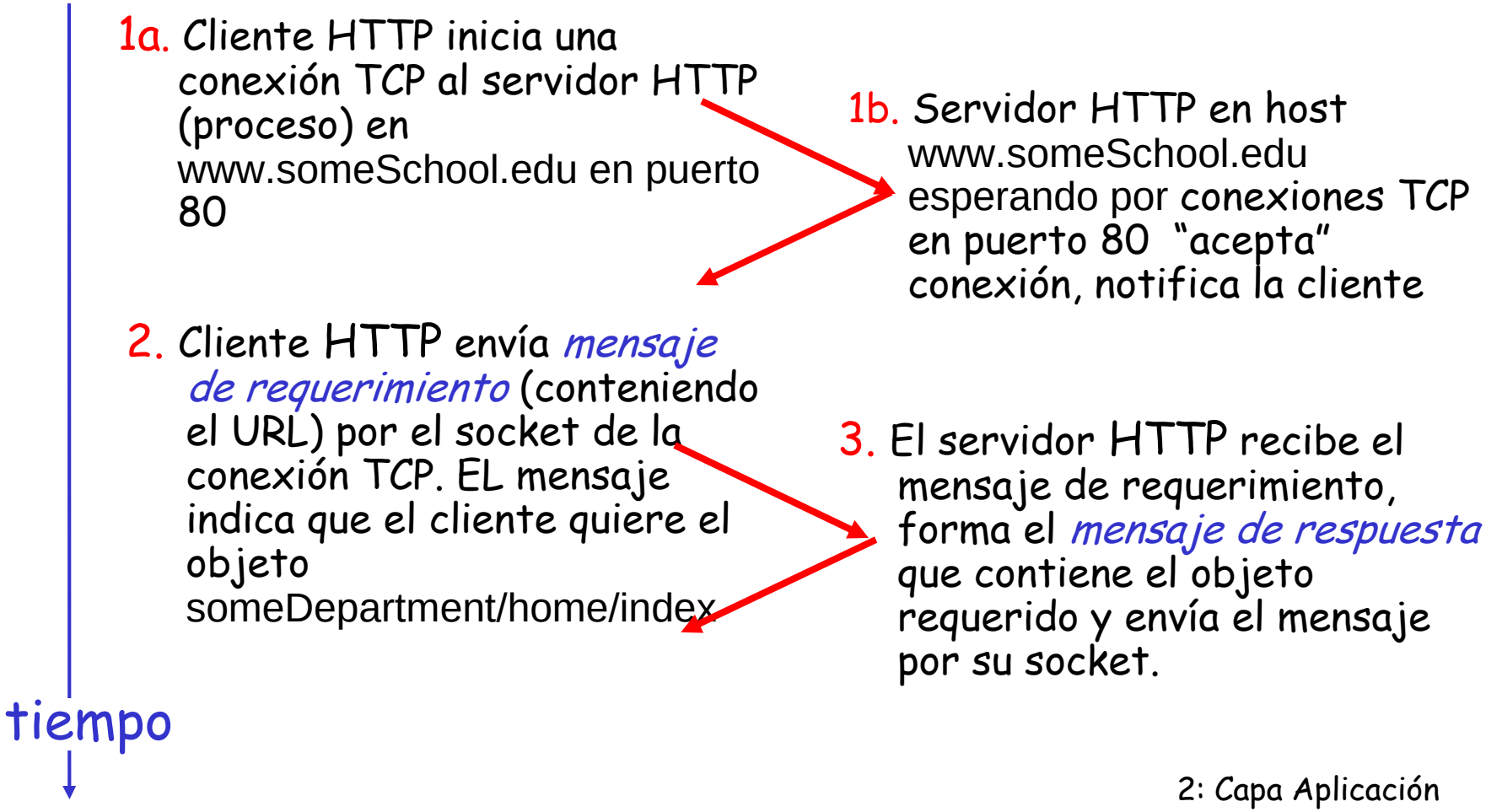
HTTP Persistente

- 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

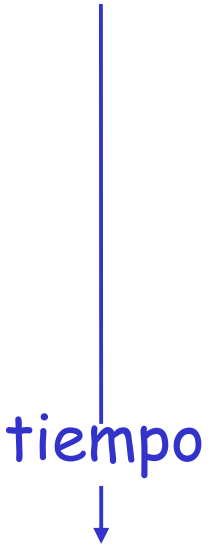
HTTP no-persistente

Supongamos que el usuario ingresa URL
`www.someSchool.edu/someDepartment/home/index`

(contiene texto, referencias a 10 imágenes jpeg)



HTTP no-persistente (cont.)



4. Servidor HTTP cierra la conexión.

5. Cliente HTTP recibe el mensaje respuesta que contiene el archivo html y despliega el html. Analizando el archivo html file, encuentra 10 referencias a objetos jpeg

6. Pasos 1-5 son repetidos para cada uno de los 10 objetos jpeg.

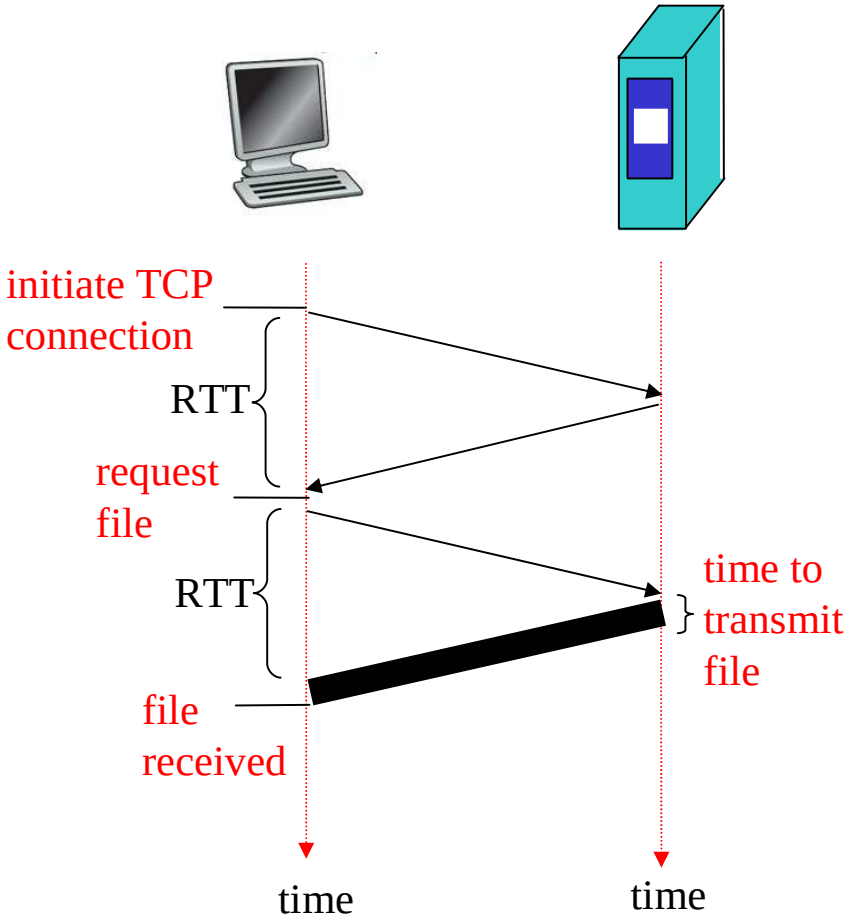
Modelo para tiempo de Respuesta

Definición de RTT: tiempo ocupado en enviar un paquete pequeño desde el cliente al servidor y su regreso.

Tiempo de respuesta:

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

total = 2RTT + tiempo de transmisión



HTTP Persistente

Problemas de HTTP no-persistente:

- requiere 2 RTTs por objeto
- OS debe trabajar y dedicar recursos para cada conexión TCP
- el navegador abre conexiones paralelas generalmente para traer objetos referenciados.

HTTP Persistente

- servidor deja las conexiones abiertas despues de enviar la respuesta
- mensajes HTTP subsecuentes entre los mimos cliente/servidor son enviados por la conexión

Persistencia sin pipelining:

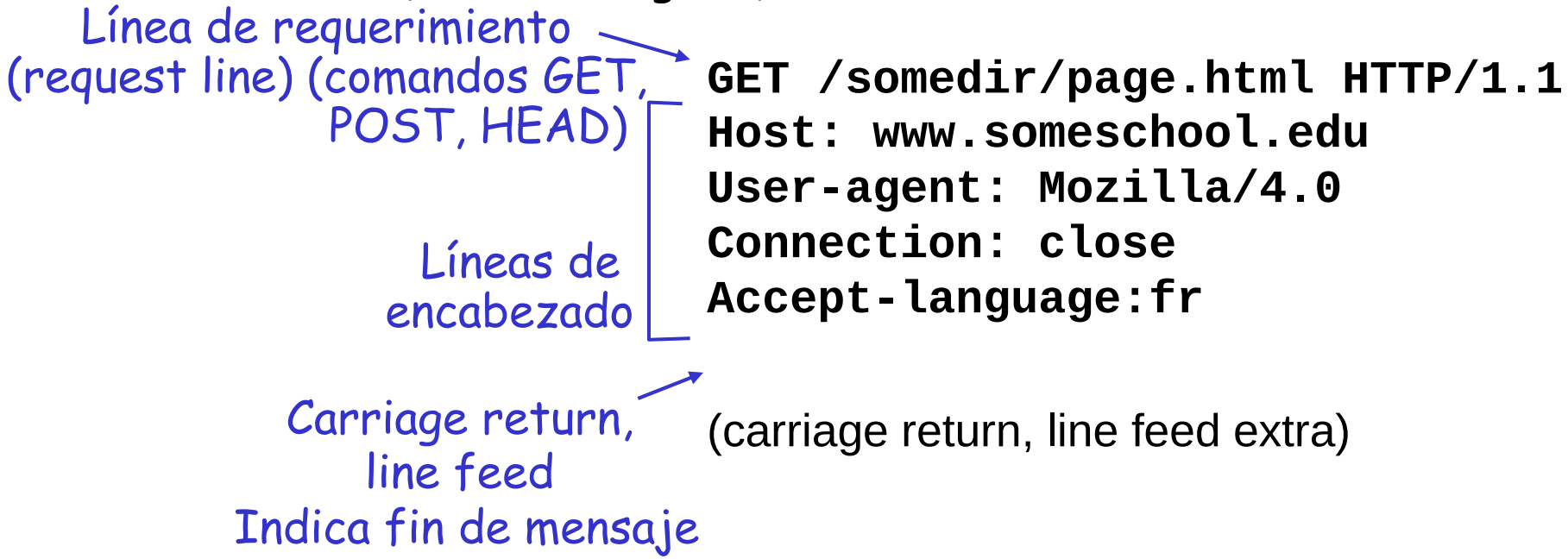
- cliente envía nuevo requerimiento sólo cuando el previo ha sido recibido
- un RTT por cada objeto referenciado

Persistencia con pipelining:

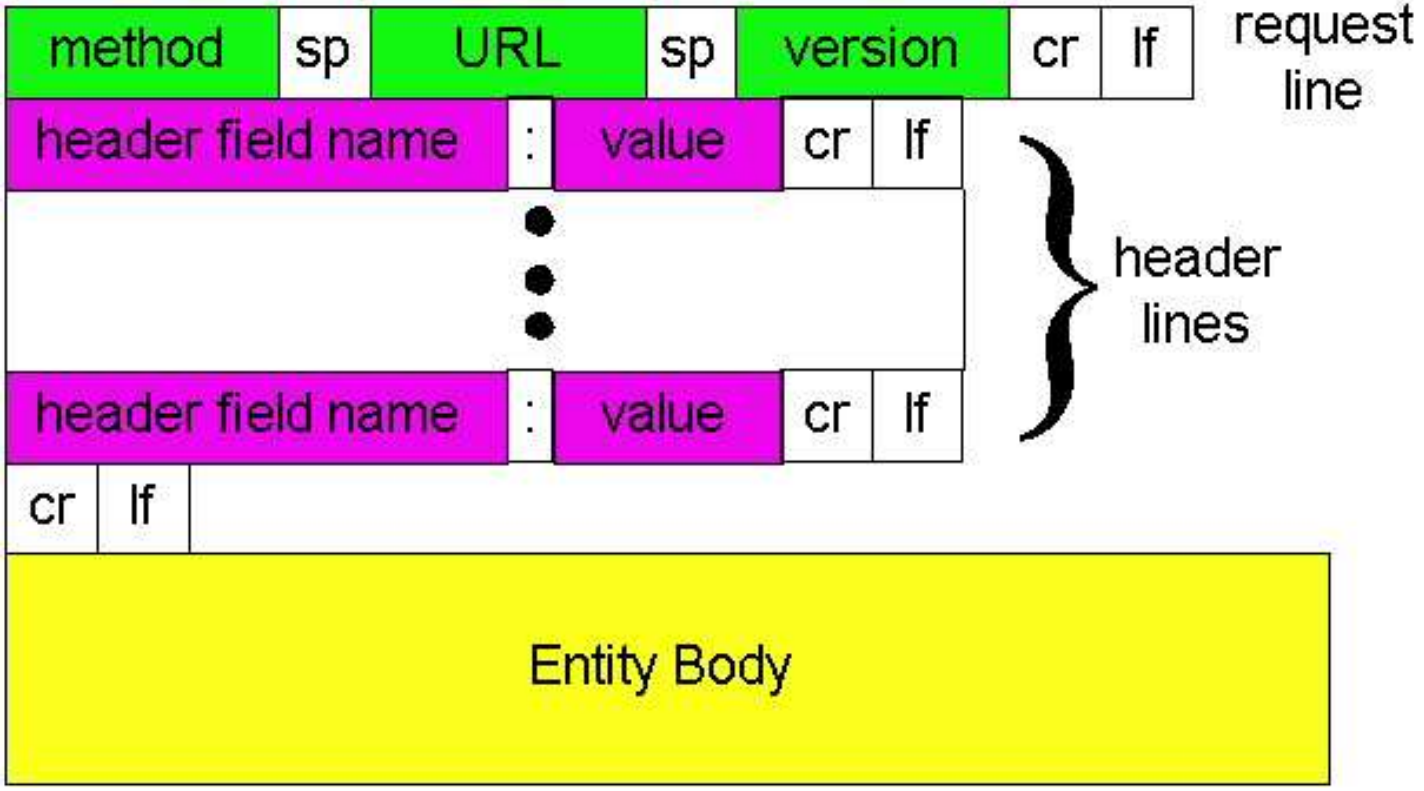
- default en HTTP/1.1
- cliente envia requerimientos tan pronto éste encuentra un objeto referenciado
- tan poco como un RTT para todas las referencias

Mensaje HTTP de requerimiento

- Dos tipos de mensajes HTTP: *requerimiento, respuesta*
- Mensaje de requerimiento HTTP:
  - ASCII (formarlo legible)



Mensaje HTTP de requerimiento: formato general



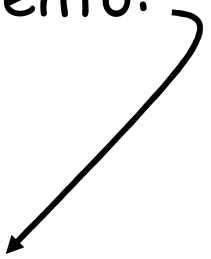
Subiendo input de formulario

Método Post:

- Páginas Webs usualmente incluyen input de formulario
- Input es subido 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

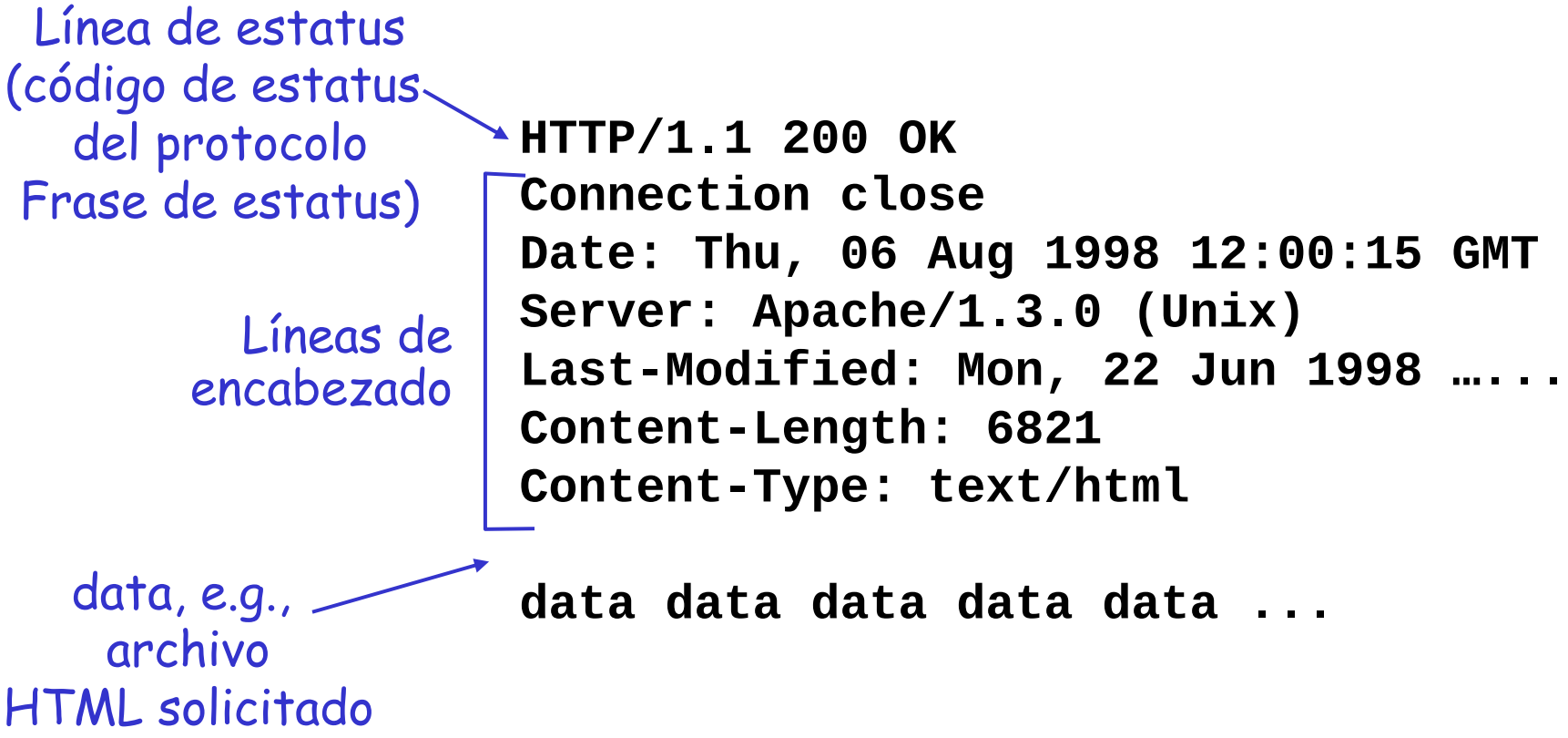
HTTP/1.0

- GET
- POST
- HEAD
  - Pide al servidor que deje el objeto requerido afuera de la respuesta.

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

Mensajes HTTP de respuesta



Códigos HTTP de respuesta

En primera línea de respuesta del servidor-> cliente.

Algunos códigos de muestra:

200 OK

- request exitoso, objeto requerido es incluido luego en mensaje

301 Moved Permanently

- Se movió el objeto requerido, nueva hubicación es especificada luego en el mensaje (Location:)

400 Bad Request

- Requerimiento no entendido por el servidor

404 Not Found

- Documento no encontrado en servidor

505 HTTP Version Not Supported

Probando HTTP (lado cliente)

1. Telnet a tu servidor favorito:

```
telnet mateo.elo.utfsm.cl 80
```

abre una conexión TCP al puerto 80 (puerto servidor HTTP por omisión) en mateo.elo.utfsm.cl. Cualquier cosa ingresada en enviada A puerto 80 de mateo

2. Escribir un requerimiento GET HTTP:

```
GET /~tarredondo/info/networks/prueba/hola.html HTTP/1.1
Host: mateo.elo.utfsm.cl
```

Tipeando esto (doble carriage return), enviamos un GET request mínimo (pero completo) al servido HTTP

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

Probando HTTP (lado cliente): Ejemplo

```
tarredondo@aragorn:~$ telnet mateo.elo.utfsm.cl 80
Trying 200.1.17.4...
Connected to mateo.elo.utfsm.cl.
Escape character is '^]'.
GET /~tarredondo/info/networks/prueba/hola.html HTTP/1.1
Host: mateo.elo.utfsm.cl

HTTP/1.1 200 OK
MIME-Version: 1.0
Server: Roxen·Challenger/1.3.111
Content-type: text/html
Last-Modified: Mon, 01 Aug 2005 16:23:25 GMT
Date: Mon, 01 Aug 2005 16:06:06 GMT
Content-length: 614

<!DOCTYPE html PUBLIC "-//w3c//dtd html 4.0 transitional//en">
<html>
```

Probando HTTP (lado cliente): Ejemplo (cont)

```
<head>
  <meta http-equiv="Content-Type"
 content="text/html; charset=iso-8859-1">
  <meta name="Author" content="Tomas Arredondo Vidal">
  <meta name="GENERATOR"
 content="Mozilla/4.77 [en] (X11; U; Linux 2.4.3 i686) [Netscape]">
  <title>Redes de Computadores I</title>
  <meta content="Tomas Arredondo" name="author">
  <meta content="Redes de Cmputadores, Internet, WEB, IT"
 name="description">
</head>
<body>
<big style="font-style: italic;">Pagina de prueba para Redes de
Computadores I</big><br>
<br>
</body>
</html>
Connection closed by foreign host.
```

Estado usuario-servidor: cookies

Muchos sitios Web importantes usan cookies

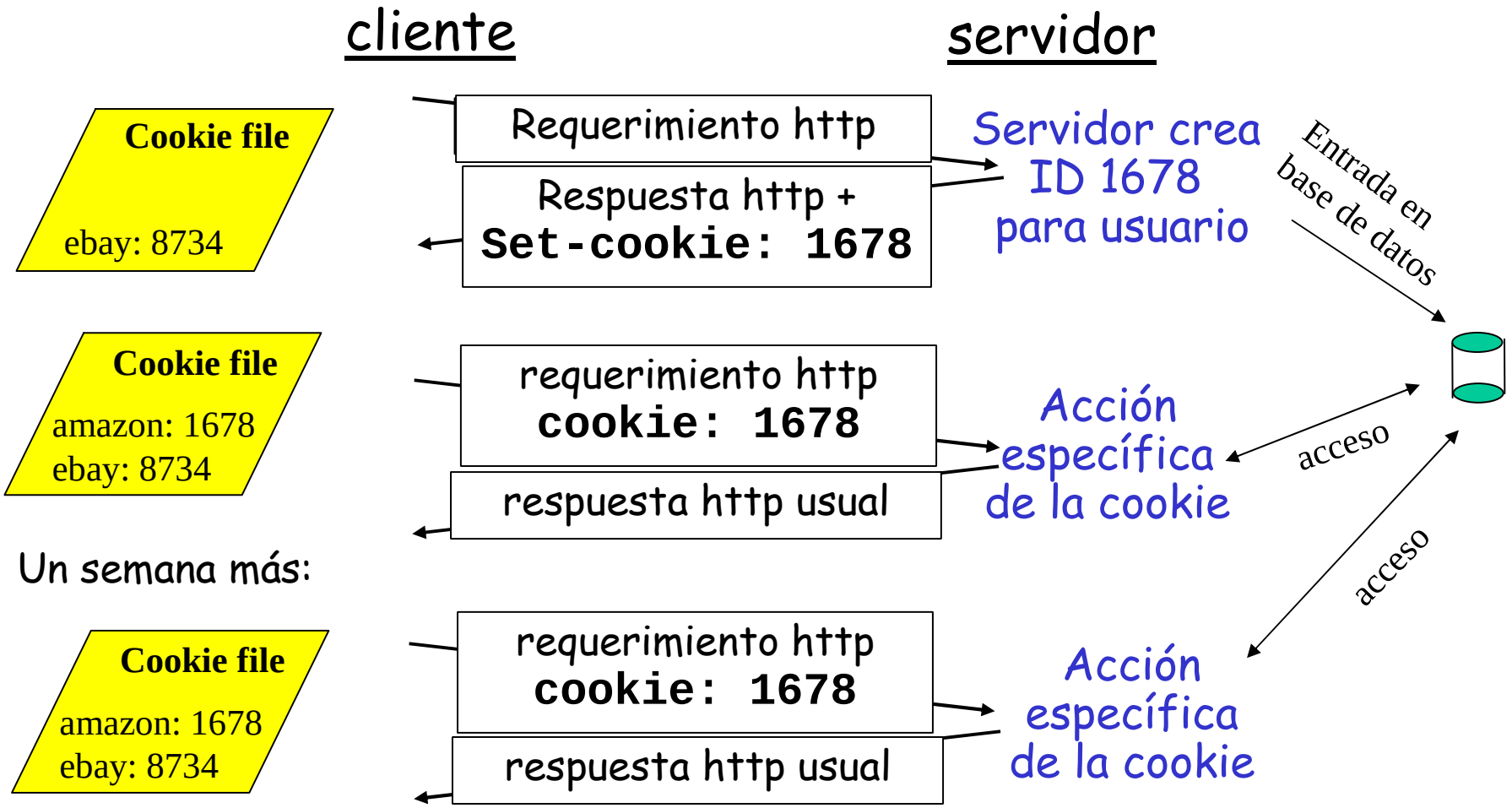
Cuatro componentes:

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:

- Susan accede Internet siempre desde el mismo PC
- Ella visita un sitio 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” (cont.)



Cookies (cont.)

Qué pueden transportar las cookies:

- autorización
- shopping carts
- sugerencias
- Estado de la sesión del usuario (Web e-mail)

Al margen

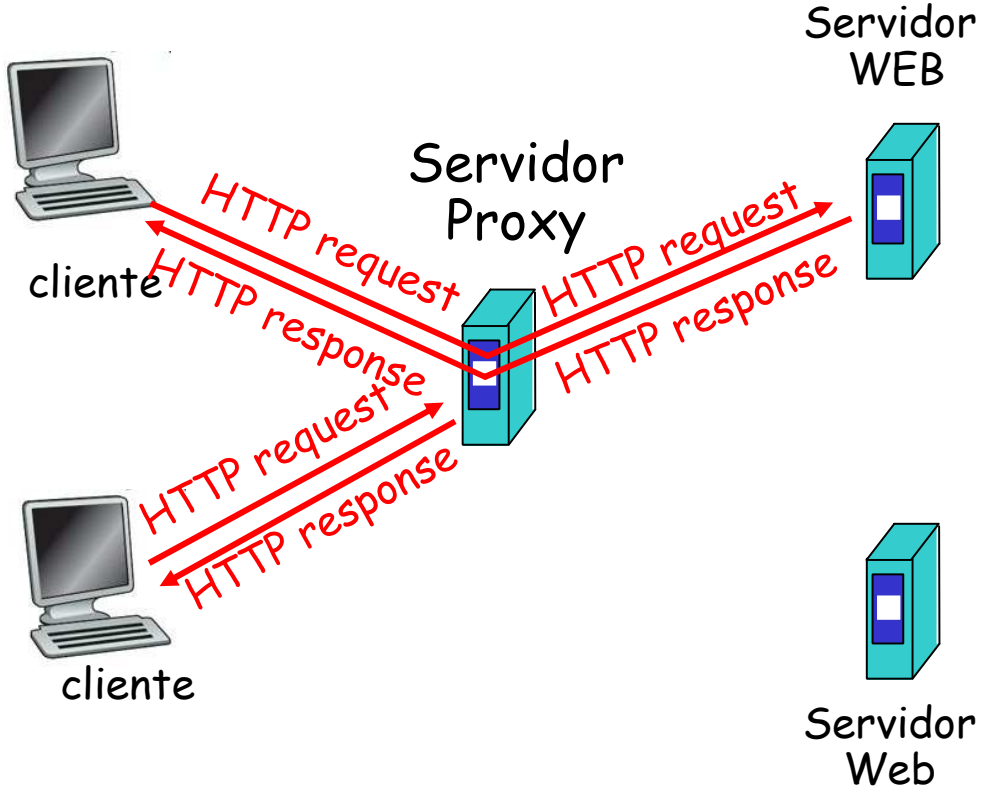
Cookies y privacidad:

- 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

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
  - Sino cache requiere los objetos desde el servidor Web, y retorna el objeto al cliente



Más sobre Web caching

- 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 trafico 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

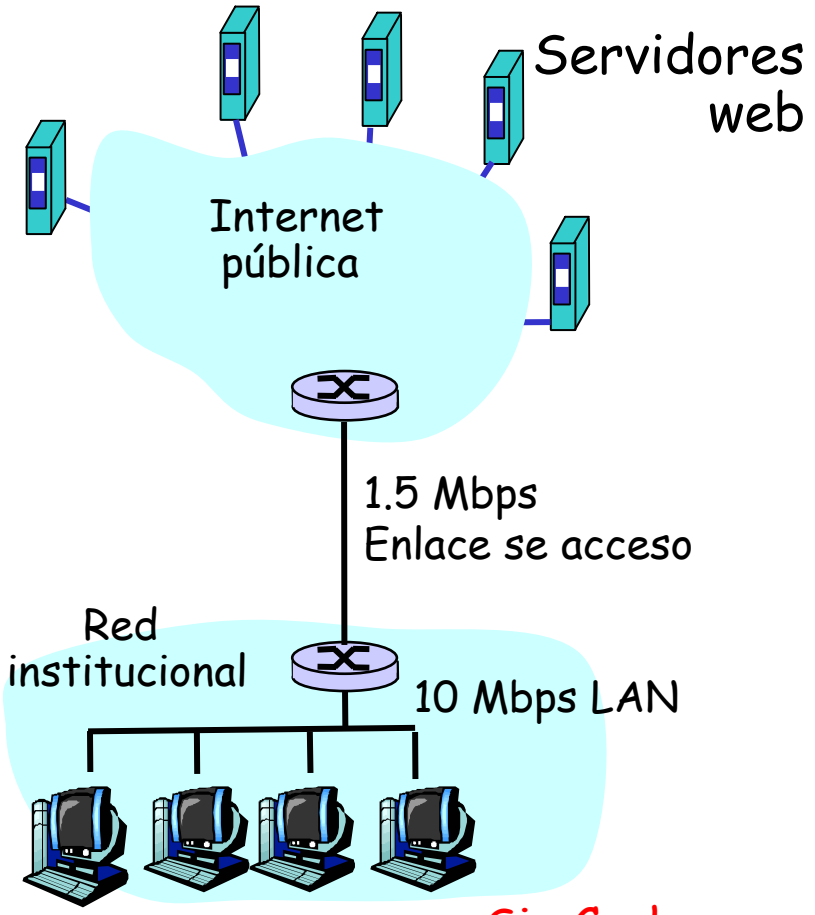
Suposiciones

- Tamaño promedio de objetos = 100,000 bits
- Tasa de requerimientos promedio desde browsers de la institución al servidor WEB = 15/sec
- Retardo desde el router institucional a cualquier servidor web y su retorno = 2 sec

Consecuencias

- utilización de la LAN = 15%
- utilización del enlace de acceso = 100%
- Retardo total = retardo Internet + retardo de acceso + retardo LAN

  = 2 sec + minutos + millisegundos



Sin Cache institucional

Ejemplo de Cache (cont)

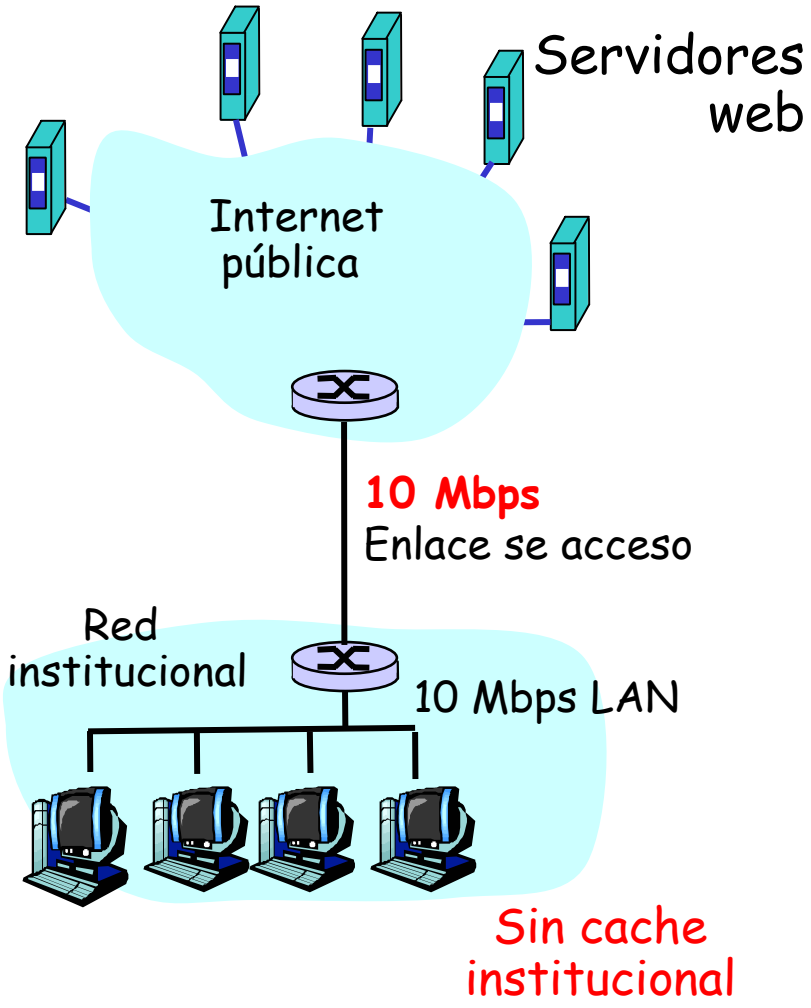
Posible solución

- Aumentar ancho de banda del enlace a, por ejemplo, 10 Mbps

Consecuencias

- Utilización de la LAN = 15%
- Utilización del enlace de acceso = 15%
- Retardo Total = Retardo Internet + retardo de acceso + retardo LAN

  = 2 sec + msecs + msecs
- A menudo un upgrade caro.



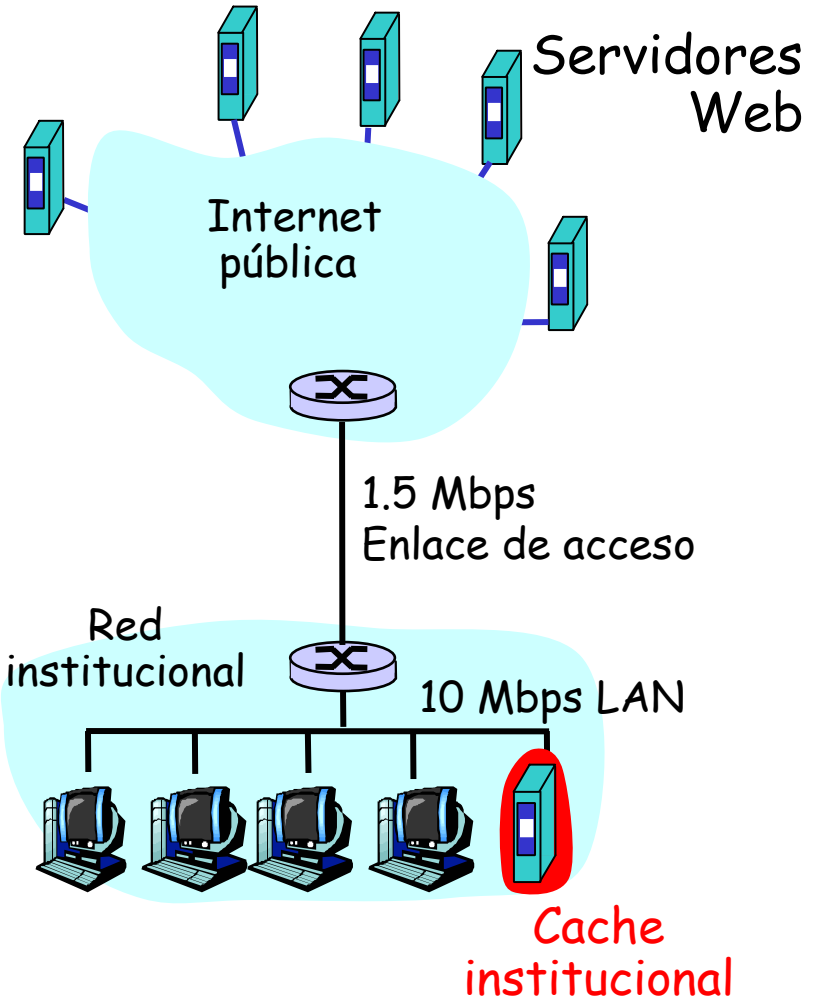
Ejemplo de cache (cont)

Instalar un web Cache

- Supongamos tasa de éxito[1] (acierto) de 0.4

Consecuencias

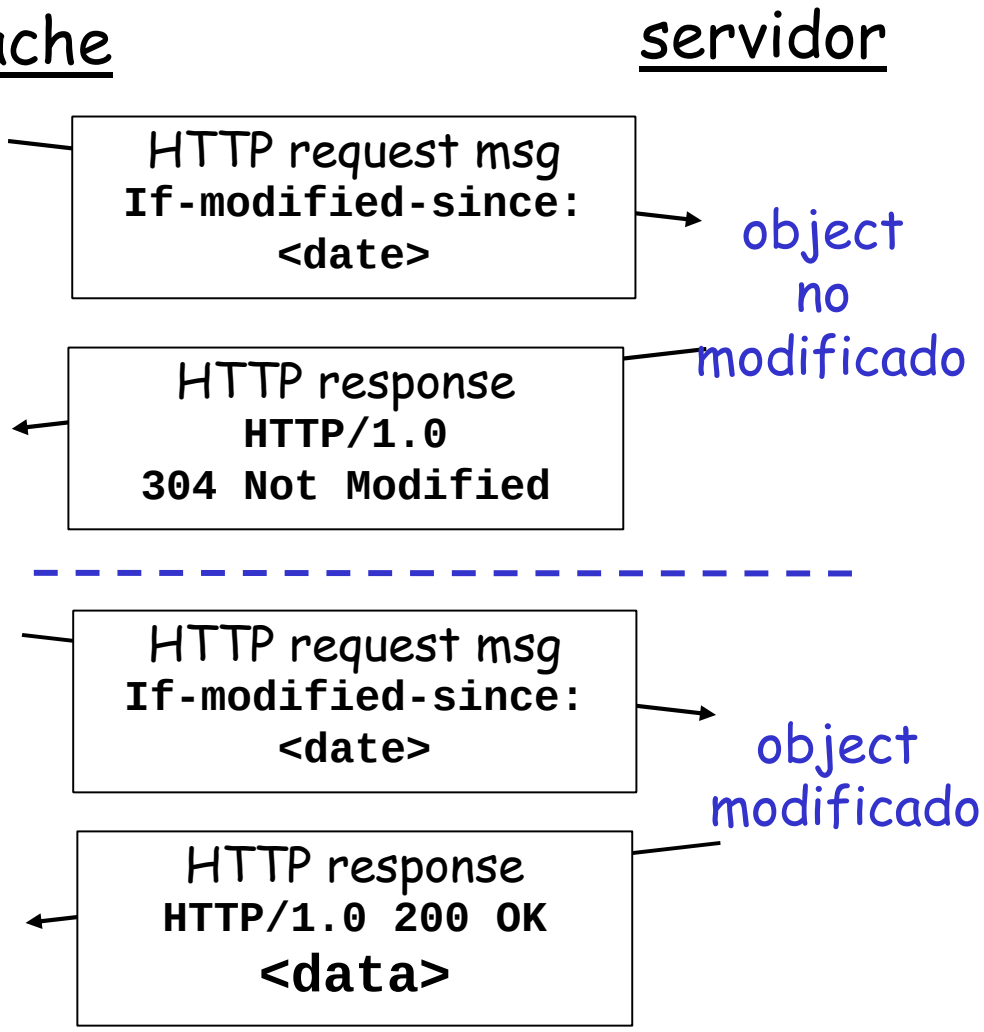
- 40% de los requerimientos serán satisfechos en forma casi inmediata (~10 msec)
- 60% de los requerimientos satisfechos por el servidor WEB
- Utilización del enlace de acceso es reducido al 60%, resultando en retardo despreciable (digamos 10 msec)
- Retardo total = Retardo Internet + retardo acceso + retardo LAN = 0.6*(2.01) sec + 0.4*0.01 < 1.3 sec



[1]Tasa de éxito: Fracción de los requerimientos satisfechos por la cache.

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`



Capítulo 2: Capa Aplicación

- 2.1 Principios de la aplicaciones de red
- 2.2 Web y HTTP
- 2.3 FTP
- 2.4 Correo Electrónico
  - SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 P2P Compartición de archivos
- 2.7 Programación de Socket con TCP
- 2.8 Programación de socket con UDP
- 2.9 Construcción de un servidor WEB