Tecnología Digital IV: Redes de Computadoras

Clase 4: Nivel de Aplicación - Parte 2

Lucio Santi & Emmanuel Iarussi

Licenciatura en Tecnología Digital
Universidad Torcuato Di Tella

18 de marzo 2025

HTTP: experimento interactivo

1. Utilizamos la herramienta `netcat` para conectarnos a un servidor web

```
$ nc -c -v www.httpforever.com 80
```

- Establece una conexión TCP al puerto 80 (HTTP) en `www.httpforever.com`
- Cualquier cosa que escribamos será enviado allí

2. Escribimos un request HTTP

```
GET / HTTP/1.1
Host: www.httpforever.com
```

- Solicitamos el HTML en `www.httpforever.com` (recordar apretar enter dos veces)

3. Inspeccionamos la respuesta enviada por el servidor

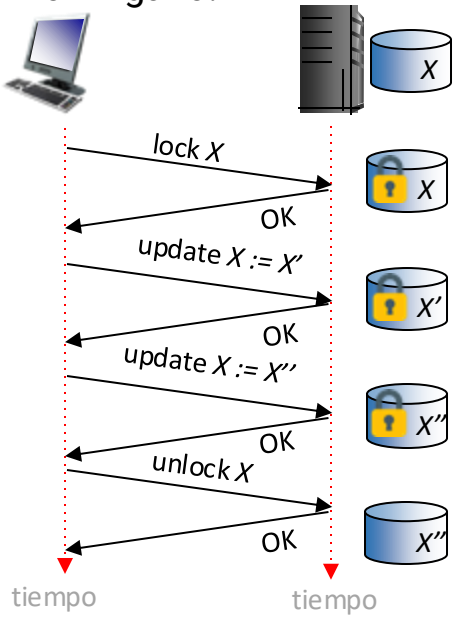
(también podemos utilizar Wireshark para observar el intercambio de paquetes)

Manteniendo estado en HTTP: *cookies*

Las interacciones HTTP son **stateless**

- No existe una noción de intercambios en varios pasos para completar una “transacción” web
  - No es necesario mantener estado en cliente/servidor durante sucesivos intercambios.
  - Cada request es independiente de los demás
  - No es necesario que el cliente o el servidor se “recuperen” de transacciones parcialmente completas.

En un protocolo ***stateful***, el cliente realiza dos cambios a X o ninguno:



Manteniendo estado en HTTP: *cookies*

Los sitios web y los navegadores utilizan cookies para mantener estado entre requests

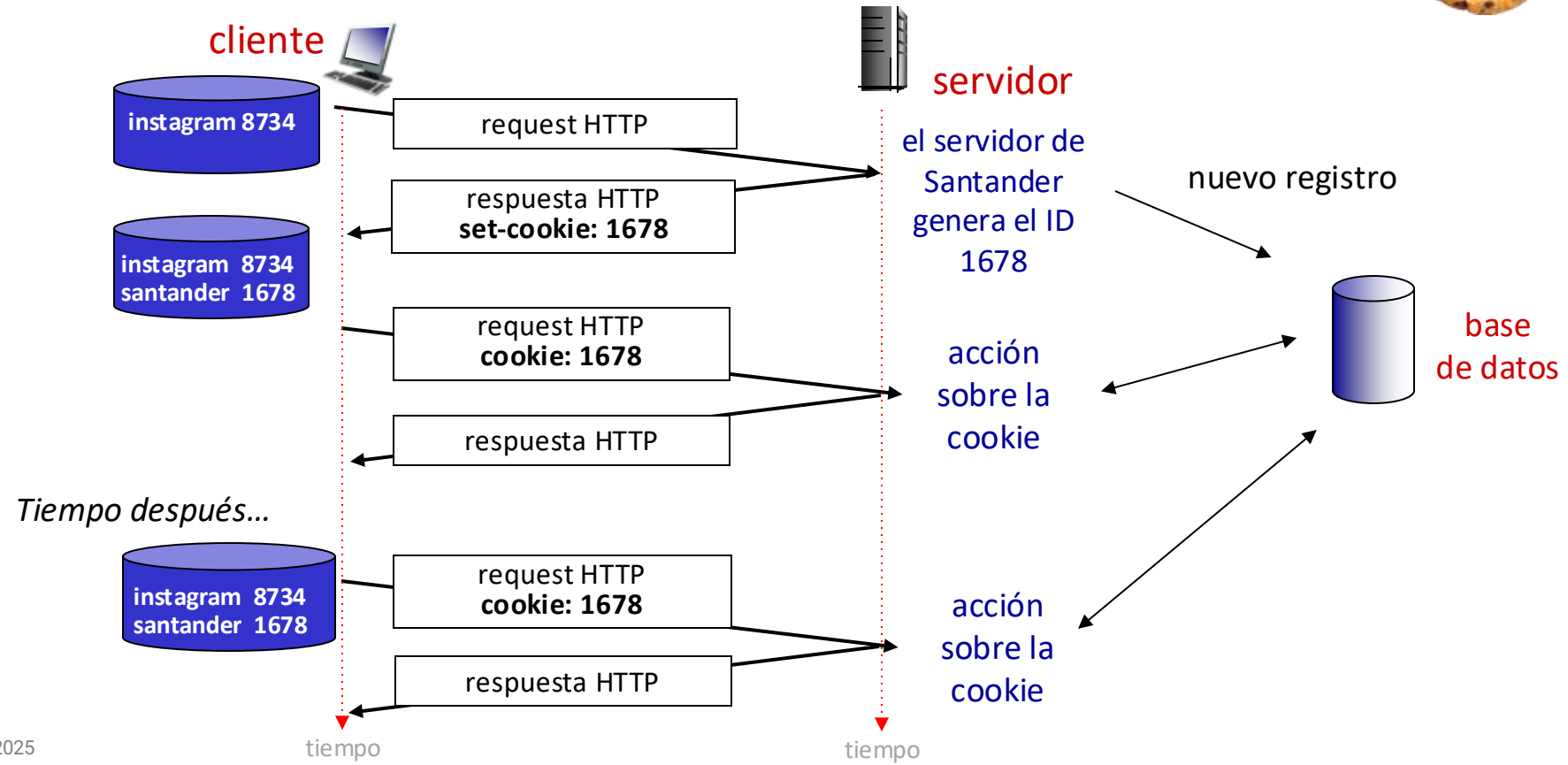
Cuatro componentes:

1) Header `Set-Cookie` en las respuestas HTTP
2) Header `Cookie` en el request subsiguiente
3) Archivo con la cookie almacenado en el host y administrado por el navegador
4) Base de datos en el sitio web

Ejemplo:

- Cierta persona usa una nueva notebook para visitar su Home Banking por primera vez
- Al recibir el request HTTP inicial, en el servidor genera:
  - Un identificador unívoco (la *cookie*)
  - Un registro en la base de datos para este identificador
- Los requests subsiguientes de la notebook contendrán el identificador de la cookie, lo cual permite que el sitio *identifique* al usuario

Manteniendo estado en HTTP: *cookies*



Cookies HTTP

Pueden utilizarse para:

- Autorización
- Carritos de compra
- Recomendaciones
- Estado de sesión

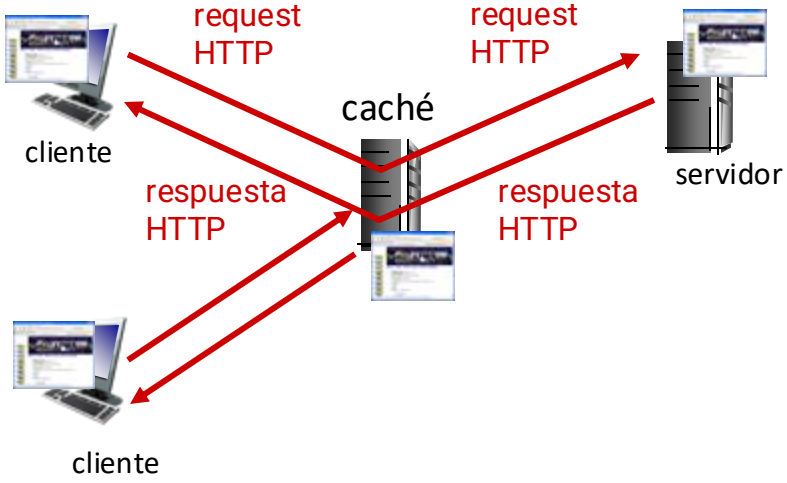
Privacidad

- Las cookies pueden tener algunas implicancias en la privacidad de los usuarios
- En particular, las *tracking cookies* las administra un tercero (distinto al sitio al que accedemos) y pueden revelar por ejemplo hábitos de navegación (qué sitios se visitaron y en qué orden)

Cachés web

Permiten servir solicitudes sin necesidad del servidor de origen

- El usuario configura al navegador apuntándolo a una caché web (*proxy*)
- El navegador envía todos los requests HTTP a la caché
  - *Si* el objeto está en la caché: se le devuelve al cliente
  - *Si no,* la caché solicita el objeto del servidor original, lo cachea y lo devuelve al cliente



Cachés web

- Actuán como cliente y como servidor
  - Servidor para el cliente que solicita recursos
  - Cliente para el servidor original

- El servidor indica parámetros en los headers de su respuesta a la caché:

```
Cache-Control: max-age=<seconds>
```

```
Cache-Control: no-cache
```

Ventajas:

- Reducen tiempo de respuesta al cliente (por cercanía)
- Reducen tráfico en enlaces de acceso a servidores (y en Internet en general)

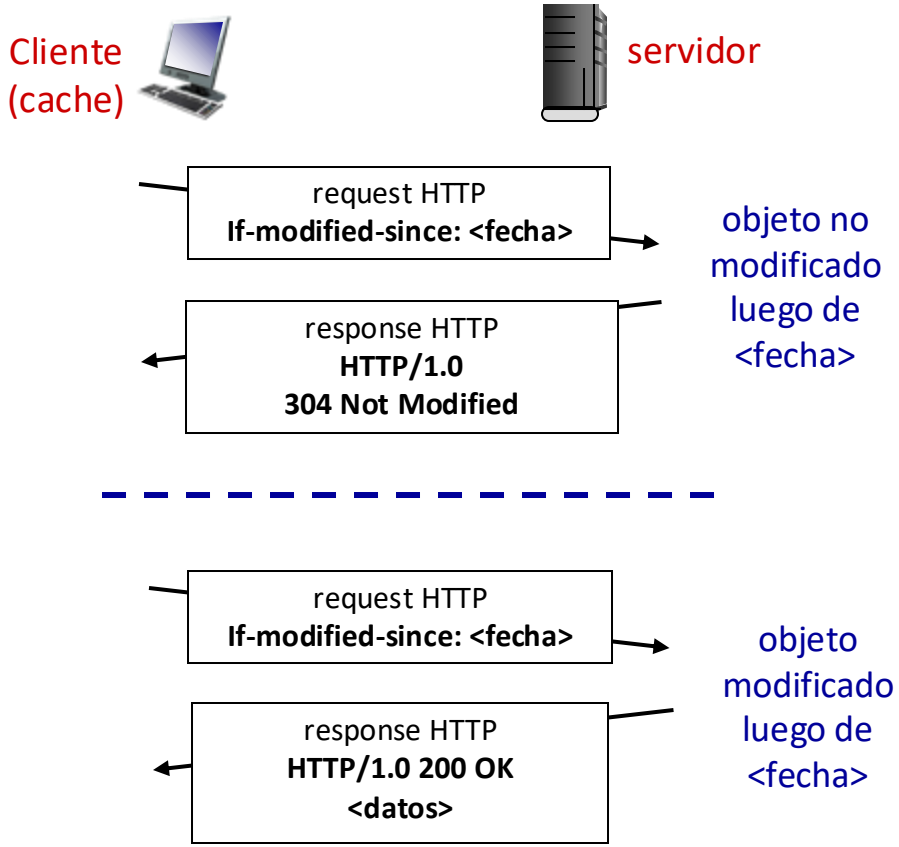
GET condicional

Objetivo: no enviar objetos si la caché posee una versión actualizada

- Evita delay de transmisión y uso de recursos de red

- *cliente:* indica la fecha de la copia cacheada en un header del request (`If-modified-since`)
- *servidor:* la respuesta no contiene el objeto si la copia está actualizada:

`HTTP/1.0 304 Not Modified`



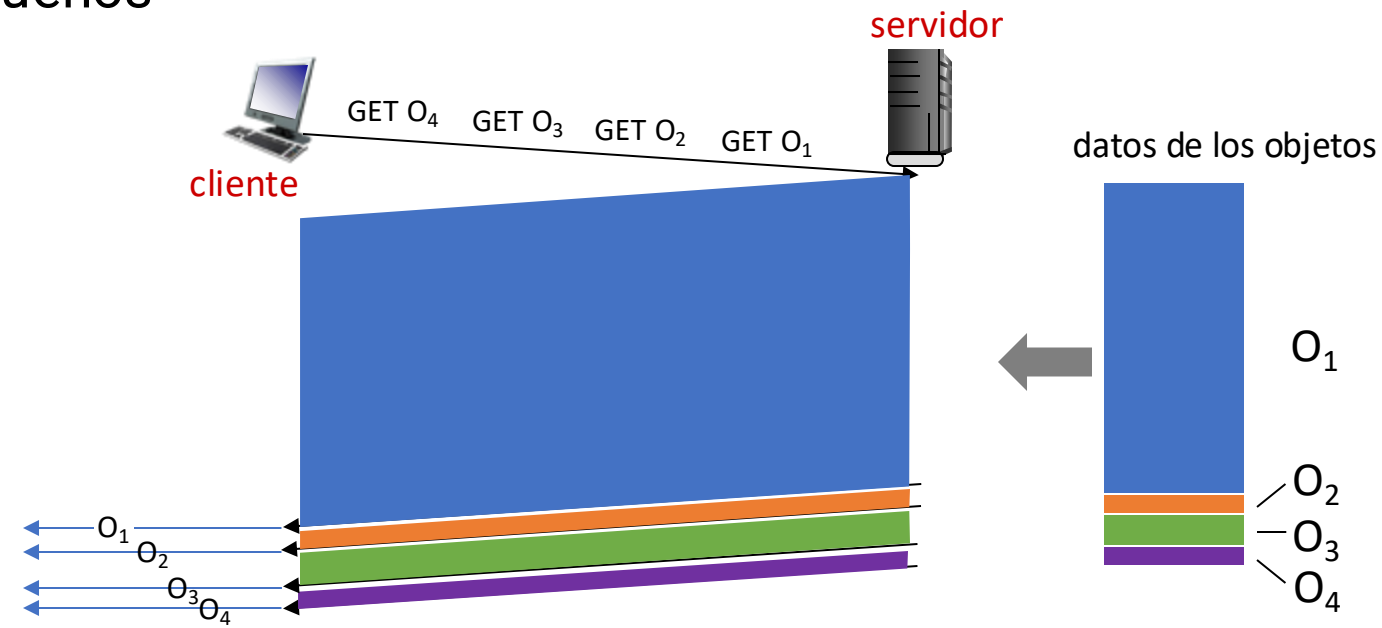
HTTP/1.1: *Head-Of-Line Blocking* (HOL)

HTTP/1.1 introdujo múltiples GETs en *pipeline* sobre una misma conexión TCP (con el fin de reducir delays en requests HTTP de múltiples objetos)

- El servidor responde en orden a los GETs (*first come, first served*)
- De este modo, los objetos pequeños podrían quedar esperando hasta ser transmitidos si hay objetos grandes antes (*head-of-line blocking*, HOL)
- Las retransmisiones ante pérdidas de paquetes frenan la transmisión de objetos

HTTP/1.1: *Head-Of-Line Blocking* (HOL)

HTTP/1.1: el cliente solicita un objeto grande (e.g., un video) y tres pequeños



Los objetos se entregan en el orden solicitado: O_2 , O_3, O_4 esperan detrás de O_1

HTTP/2

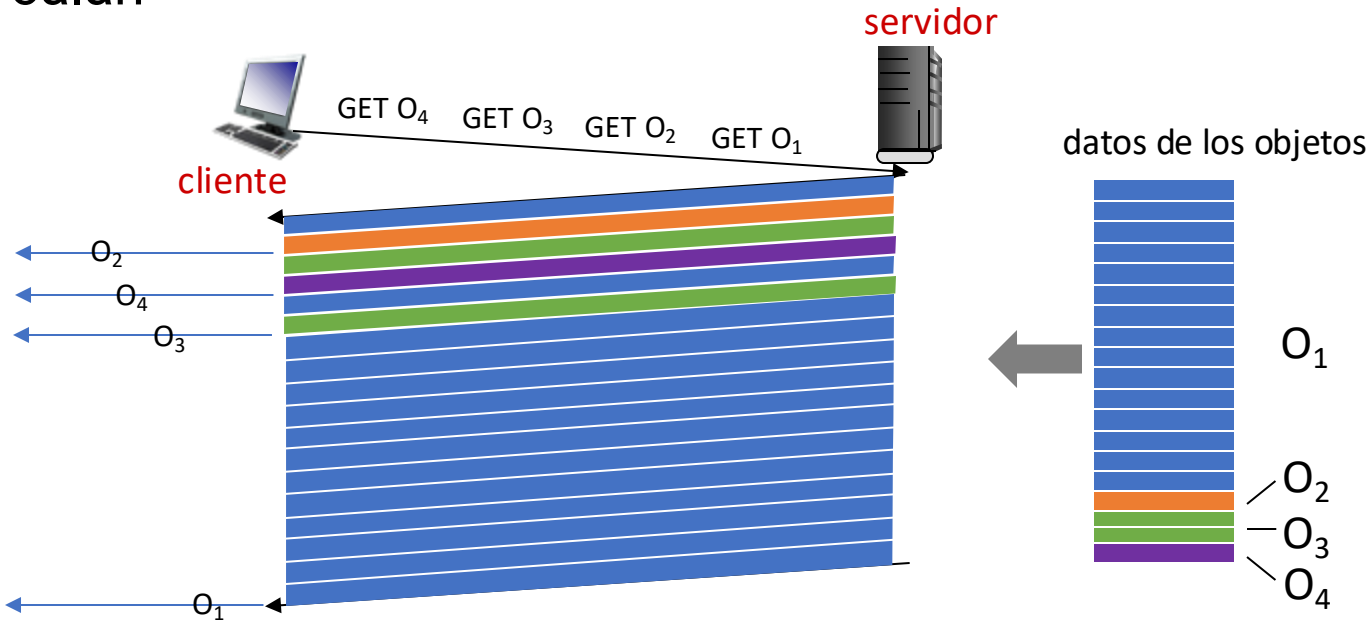
Objetivo: reducir delay en requests HTTP de múltiples objetos

HTTP/2 [RFC 7540, 2015] aumentó la flexibilidad en los servidores al enviar objetos a los clientes:

- Métodos, códigos de status y la mayor parte de los headers permanecen sin cambios respecto de HTTP/1.1
- Se dividen los objetos en frames; estos se acomodan para mitigar el problema de HOL blocking
- El orden de transmisión de objetos se basa en prioridades definidas por el cliente
- Envío de objetos no solicitados al cliente (***server push***)

HTTP/2: mitigación de HOL blocking

HTTP/2: los objetos se dividen en frames, cuyas transmisiones se intercalan



O_2, O_3, O_4 se envían rápidamente; O_1 apenas demorado

De HTTP/2 a HTTP/3

HTTP/2 sobre una misma conexión TCP implica:

- Retransmisiones por pérdida de paquetes pueden frenar transmisiones de objetos
  - Como en HTTP/1.1, los navegadores pueden abrir múltiples conexiones en paralelo para reducir estas demoras e incrementar el *throughput*
- Además, no ofrece seguridad sobre la conexión TCP subyacente
- HTTP/3 agrega seguridad, control de errores y control de congestión por objeto; se monta sobre UDP
  - Se implementa un nuevo protocolo de transporte en nivel de aplicación: **QUIC**

Ejercicio!

Consideren el siguiente string capturado por Wireshark cuando un cliente envió un HTTP GET:

```
GET /td4/index.html HTTP/1.1<cr><lf>Host:
utdt.edu<cr><lf>User-Agent: Mozilla/5.0 (Windows;U;
Windows NT 5.1; en-US; rv:1.7.2) Gec ko/20040804/7.2
(ax) <cr><lf>Accept:ext/xml,
application/xml,application/xhtml+xml,text/html;q=0.9,te
xt/plain;q=0.8,image/png,*/*;q=0.5<cr><lf>Accept-
Language:en-us,en;q=0.5<cr><lf>Accept-
Encoding:zip,deflate<cr><lf>Accept-Charset:ISO-8859-
1,utf-8;q=0.7,*;q=0.7<cr><lf>Keep-
Alive:300<cr><lf>Connection:keep-alive<cr><lf><cr><lf>
```

- ¿Cuál es la URL del documento requerido?
- ¿Qué versión de HTTP está ejecutando el navegador?
- ¿La conexión es persistente o no persistente?
- ¿Cuál es la dirección IP del host sobre el que está ejecutando el navegador?
- ¿Qué tipo de navegador envía el mensaje?