Protocolo HTTP y WEB-Cache

Andres Barbieri

Historia

- WWW (red de sistemas de hypertexto inter-linkeados accesibles vía Internet).
- Desarrollada en 1990 por Tim Berners-Lee en el CERN.
  - Desarrolla el protocolo HTTP y el lenguaje HTML.
- 1993: el primer cliente/browser GUI: Mosaic.
- 1994: Netscape Navigator 1.0.
- Desarrollo de servidores, por ejemplo 1995: Apache Server.
- Buscadores, Indexadores son parte importante del servicio.
- Modelo anterior de interacción: servidor produce, cliente consume.
- Nuevos Modelos: Web 2.0, Tim O'Reilly en 2004, modelo de interacción entre usuarios, consumen y producen mediante servicios especiales: blogs, redes sociales, wikis, multimedia, etc.

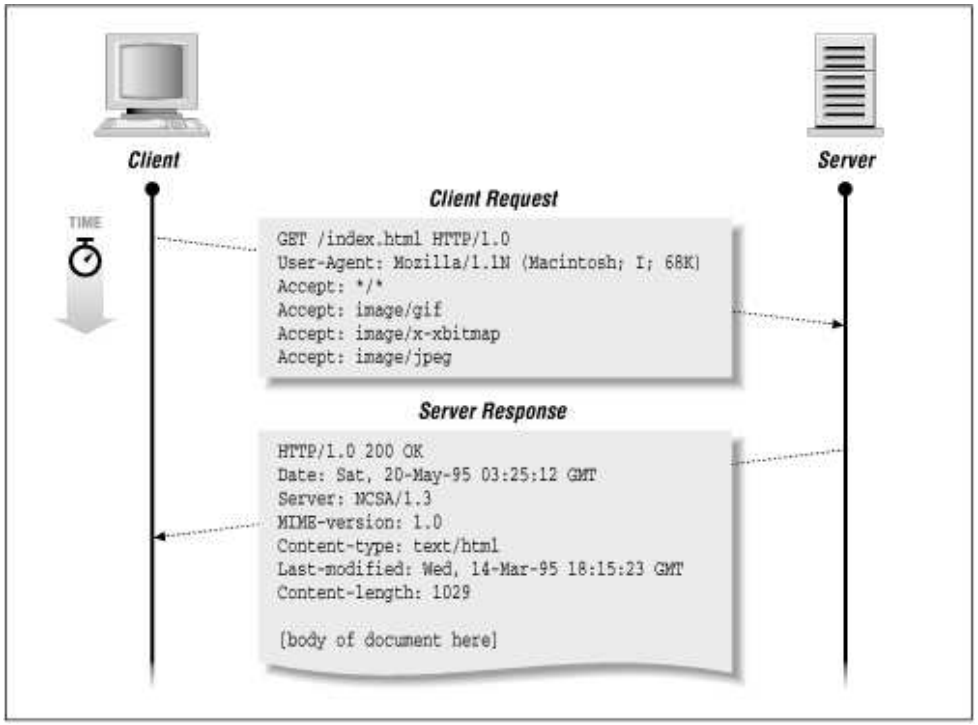
Elementos WEB

- Recurso u Objeto HTTP: ej. web page.
- Referenciado por una URI (Uniform Resourse Id): URL (Uniform Resource Location) o URN (Name).
- Formato de URL:
  `protocol://[user:pass@]host:[port]/[path]`.
- Ejemplo:
  `http://www.NN.unlp.edu.ar:8080/dir/index.html`.
- URN: no indica ubicación, solo identifican, categorías, poco impl. ej. urn:isbn:0132856204, urn:ietf:rfc:2616.
- Objetos pueden ser página web (web page), imágenes JPEG, PNG, GIF, Java Applet, archivos de multimedia: MP3, AVI, etc.
- Páginas web: archivo HTML que incluye vínculos o directamente otros objetos.

Funcionamiento de HTTP

- Modelo cliente/servidor, Request/Response (sin estados - stateless).
- Protocolo corre sobre TCP (requiere protocolo de transporte confiable), puerto 80 el default.
- El cliente escoge cualquier puerto no privilegiado.
- Trabaja sobre texto ASCII, permite enviar información binaria con encabezados MIME.
- Clientes (llamados browsers o navegadores): Firefox, IE, Opera, Safari, Chrome.
- Servidores: Apache Server, MS IIS, NGINX , Google GWS, Tomcat.

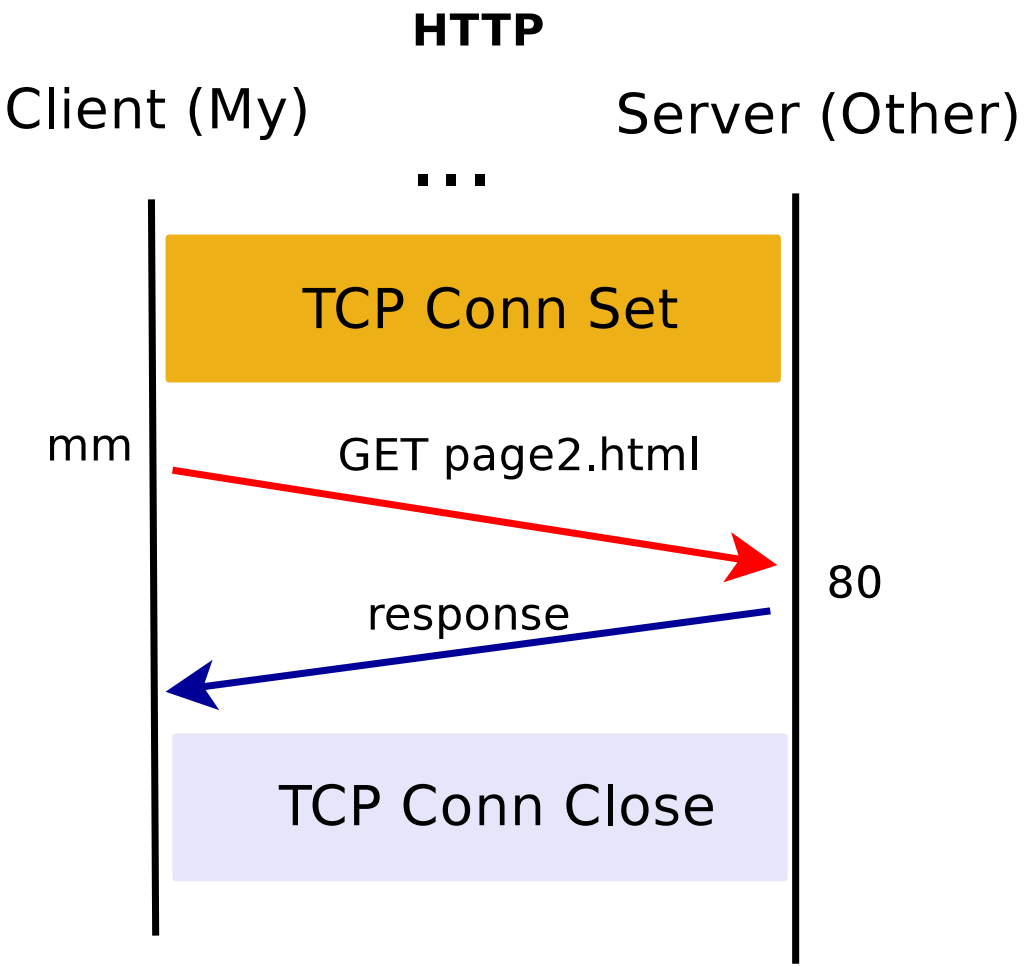
Esquema de HTTP



Versión HTTP 0.9

- Primera versión de HTTP fue la 0.9 nunca se estandarizó.
- Pasos para obtener un documento:
  1. Establecer la conexión TCP
  2. HTTP Request vía comando `GET`.
  3. HTTP Response enviando la página requerida.
  4. Cerrar la conexión TCP por parte del servidor.
  5. Si no existe el documento o hay un error directamente se cierra la conexión.

Diagrama de Pasos HTTP



Versión HTTP 0.9 (cont'd)

- Solo una forma de Requerimiento.
- Solo una forma de Respuesta.
- Request/Response sin estado.

```
Request ::== GET <document-path> <CR><LF>

Response ::== ASCII chars HTML Document.

GET /hello.html <CR><LF>

GET / <CR><LF>
```

Versión HTTP 1.0

- Versión de HTTP 1.0 estándar [RFC-1945].
- Define formato, proceso basado en HTTP 0.9:
  - Se debe especificar la versión en el requerimiento del cliente.
  - Para los Request, define diferentes métodos HTTP.
  - Define códigos de respuesta.
  - Admite repertorio de caracteres, además del ASCII, como: ISO-8859-1, UTF-8, etc.
  - Admite MIME (No solo sirve para descargar HTML e imágenes).
  - Por default NO utiliza conexiones persistentes.

Request HTTP 1.0

```
<Method> <URI> <Version>
[<Headers Opcionales>]
<Blank>
[<Entity Body Opcional>]
<Blank>

<Method HTTP 1.0> ::== GET, POST, HEAD, PUT,
                       DELETE, LINK, UNLINK
```

Response HTTP 1.0

```
<HTTP Version> <Status Code> <Reason Phrase>
[<Headers Opcionales>]
<Blank>
[<Entity Body Opcional>]
```

Ejemplos de respuestas HTTP/1.0

```
HTTP/<version> 200 OK
HTTP/<version> 301 Moved Permanently
HTTP/<version> 400 Bad Request
HTTP/<version> 403 Access Forbidden
HTTP/<version> 404 Not Found
HTTP/<version> 405 Method Not Allowed
HTTP/<version> 500 Internal Server Error (CGI Error)
HTTP/<version> 501 Method Not Implemented
```

Métodos HTTP/1.0

GET: obtener el documento requerido. Puede enviar información, pero no demasiada. Es enviada en la URL. Formato `?var1=val1&var2=val2...` Limitación de tamaño de URL por parte de las implementaciones.

HEAD: idéntico a GET, pero sólo requiere la meta información del documento, por ejemplo, su tamaño. Usado por clientes con caché.

POST: hace un requerimiento de un documento, pero también envía información en el Body. Generalmente, usado en el fill-in de un formulario HTML(FORM). Puede enviar mucha más información que un GET.

Métodos HTTP/1.0 (Cont'd)

PUT: usado para reemplazar un documento en el servidor. En general, deshabilitada. Utilizado, por ejemplo, por protocolos montados sobre HTTP, como WebDAV [WDV].

DELETE: usado para borrar un documento en el servidor. En general, deshabilitada. También, puede ser utilizada por WebDAV.

LINK, UNLINK: establecen/des-establecen relaciones entre documentos.

GET/Response HTTP 1.0

```
GET /index2.html HTTP/1.0
User-Agent: telnet/andres (GNU/Linux)
Host: estehost.com
Accept: */*

HTTP/1.1 200 OK
Date: Mon, 21 Apr 2008 00:28:51 GMT
Server: Apache/2.2.4 (Ubuntu)
Last-Modified: Mon, 21 Apr 2008 00:18:14 GMT
ETag: "a3b36-1f-91d5d80"
Accept-Ranges: bytes
Content-Length: 31
Connection: close
Content-Type: text/plain
```

GET/Response HTTP 1.0 (Cont'd)

```
<HTML>
<H1> HOLA </H1>
...
</HTML>
```

HEAD/Response HTTP 1.0

```
HEAD /index.html HTTP/1.0
User-Agent: telnet/andres (GNU/Linux)
Host: estehost.com
Accept: */*

HTTP/1.1 200 OK
Date: Mon, 21 Apr 2008 00:40:25 GMT
Server: Apache/2.2.4 (Ubuntu)
Last-Modified: Mon, 21 Apr 2008 00:18:14 GMT
ETag: "a3b36-1f-91d5d80"
Accept-Ranges: bytes
Content-Length: 31
Connection: close
Content-Type: text/plain
```

POST/Response HTTP 1.0

```
POST /index.html HTTP/1.0
User-Agent: telnet/andres (GNU/Linux)
Host: estehost.com
Accept: */*
Content-Type: text/plain
Content-Length: 10

1234567890

HTTP/1.1 200 OK
Date: Mon, 21 Apr 2008 00:37:22 GMT
Server: Apache/2.2.4 (Ubuntu)
Last-Modified: Mon, 21 Apr 2008 00:18:14 GMT
ETag: "a3b36-1f-91d5d80"
```

POST/Response HTTP 1.0 (Cont'd)

```
Content-Length: 31
Connection: close
Content-Type: text/plain

<HTML>
<H1> HOLA </H1>
...
</HTML>
```

GET HTTP/1.0 con Host Virtuales

- Mediante el parámetro `Host` se pueden multiplexar varios servicios sobre un mismo host.

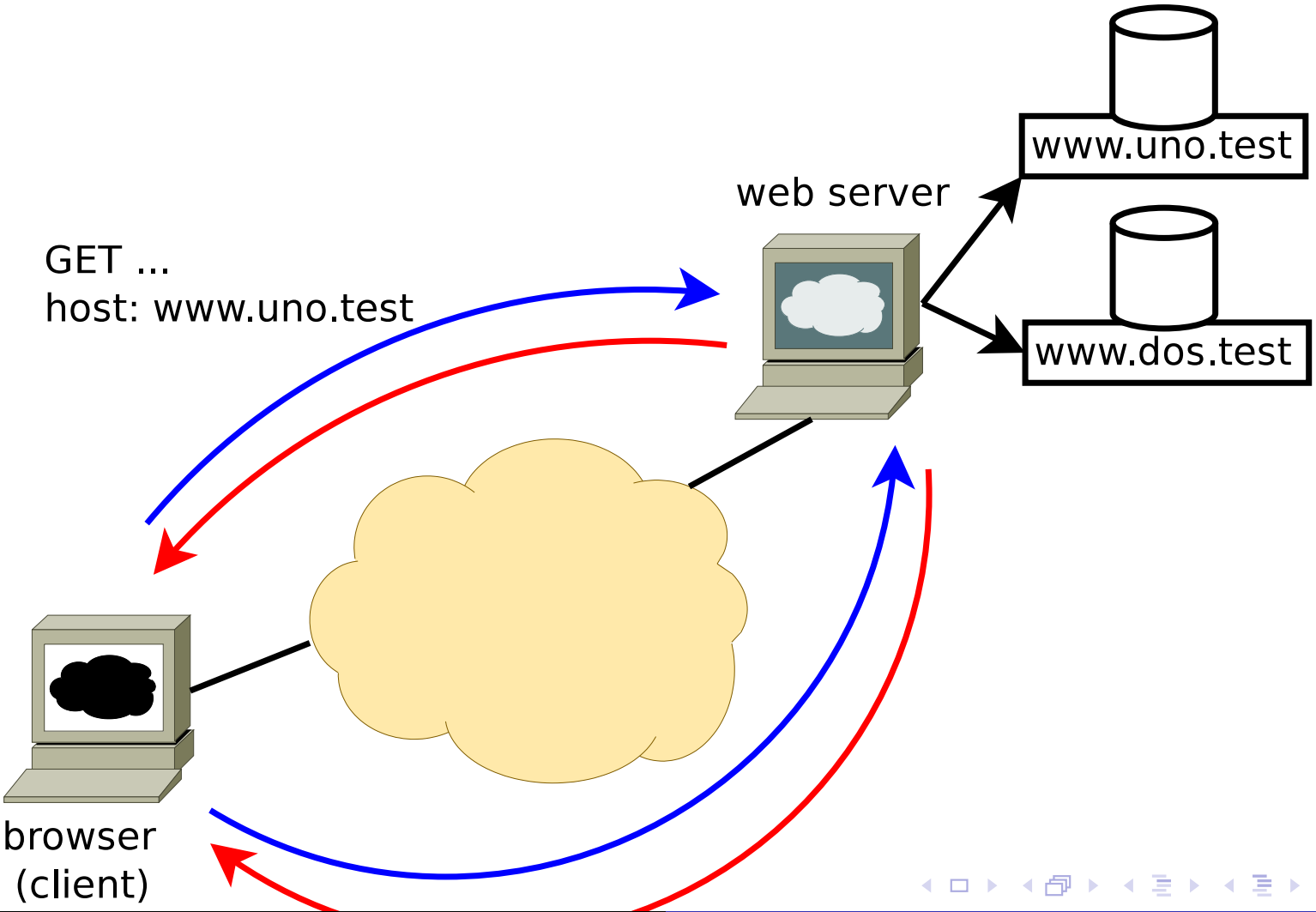
```
? ./mozilla http://www.uno.test

? ./mozilla http://www.dos.test

? host www.uno.test
www.uno.test has address 192.168.0.2

? host www.dos.test
www.dos.test has address 192.168.0.2
```

Virtual Hosts



PUT, DELETE HTTP 1.0

- Los método `PUT/DELETE` ya existían en HTTP/1.0.
- No se implementan directamente en el servidor.
- Se deben agregar como una extensión CGI para manejarlos.
- Debe leer el campo de longitud y leer esa cantidad de información de la entrada, luego generando el archivo.
- Los utilizan protocolos como WebDAV, MS File sharing/OpenCloud.
- Otra forma de subir o borrar archivos es usando el método `POST` y programando una CGI adecuada.
- La diferencia con el POST que el archivo que se indica puede no existir y el script de procesamiento es el mismo.

Ejemplo de PUT

```
LoadModule actions_module .../mod_actions.so
...
Script PUT /cgi-bin/put.pl
```

```
PUT /otro.txt HTTP/1.0
Host: otrohost.com
Accept-Encoding: gzip, deflate
Content-Type: text/xml
User-Agent: telnet/andres (GNU/Linux)
Content-Length: 10

0123456789
```

Autenticación HTTP

- HTTP/1.0 contempla autenticación con `WWW-Authenticate Headers`.
- Encabezados, el cliente y el servidor intercambiar información auth.
- El servidor, ante un requerimiento de un documento que requiere autenticación, enviará un mensaje 401 indicando la necesidad de autenticación y un Dominio/Realm.
- El navegador solicitará al usuario los datos de user/password (si es que no los tiene cachedos) y los enviará en texto claro al servidor.
- El servidor dará o no acceso en base a esos valores.
- Para los siguientes requerimientos, el navegador usará los valores que tiene almacenados para el Realm solicitado.

Autenticación HTTP (Cont'd)

Authorization Required

This server could not verify that you are authorized to access the document requested. Either you supplied the wrong credentials (e.g., bad password), or your browser doesn't understand how to supply the credentials required.

Apache/2.2.9 (Debian) DAV/2 SVN/1.5.1 PHP/5.2.6-1+lenny3 with Suhosin-Patch mod_ssl/2.2.9 OpenSSL/0.9.8g Server at Port 443

HTTP/1.0 con conexiones persistentes

- En HTTP/1.0 no se contemplaron las conexiones persistentes por default.
- A partir de HTTP/1.1 [RFC-2068], si.
- En HTTP/1.0 se pueden solicitar de forma explícita.

```
GET /index.html HTTP/1.0
Connection: Keep-Alive
User-Agent: telnet/andres (GNU/Linux)
Host: estehost.com
Accept: */*

HTTP/1.1 200 OK
Date: Tue, 22 Apr 2008 01:31:56 GMT
Server: Apache/2.2.4 (Ubuntu)
Last-Modified: Mon, 21 Apr 2008 00:18:14 GMT
```

HTTP/1.0 con conexiones persistentes (Cont'd)

```
ETag: "a3b36-1f-91d5d80"
Accept-Ranges: bytes
Content-Length: 31
Keep-Alive: timeout=15, max=10
Connection: Keep-Alive
Content-Type: text/html

<HTML>
<H1> HOLA </H1>
</HTML>
GET /index.html HTTP/1.0
Connection: Keep-Alive
...
```

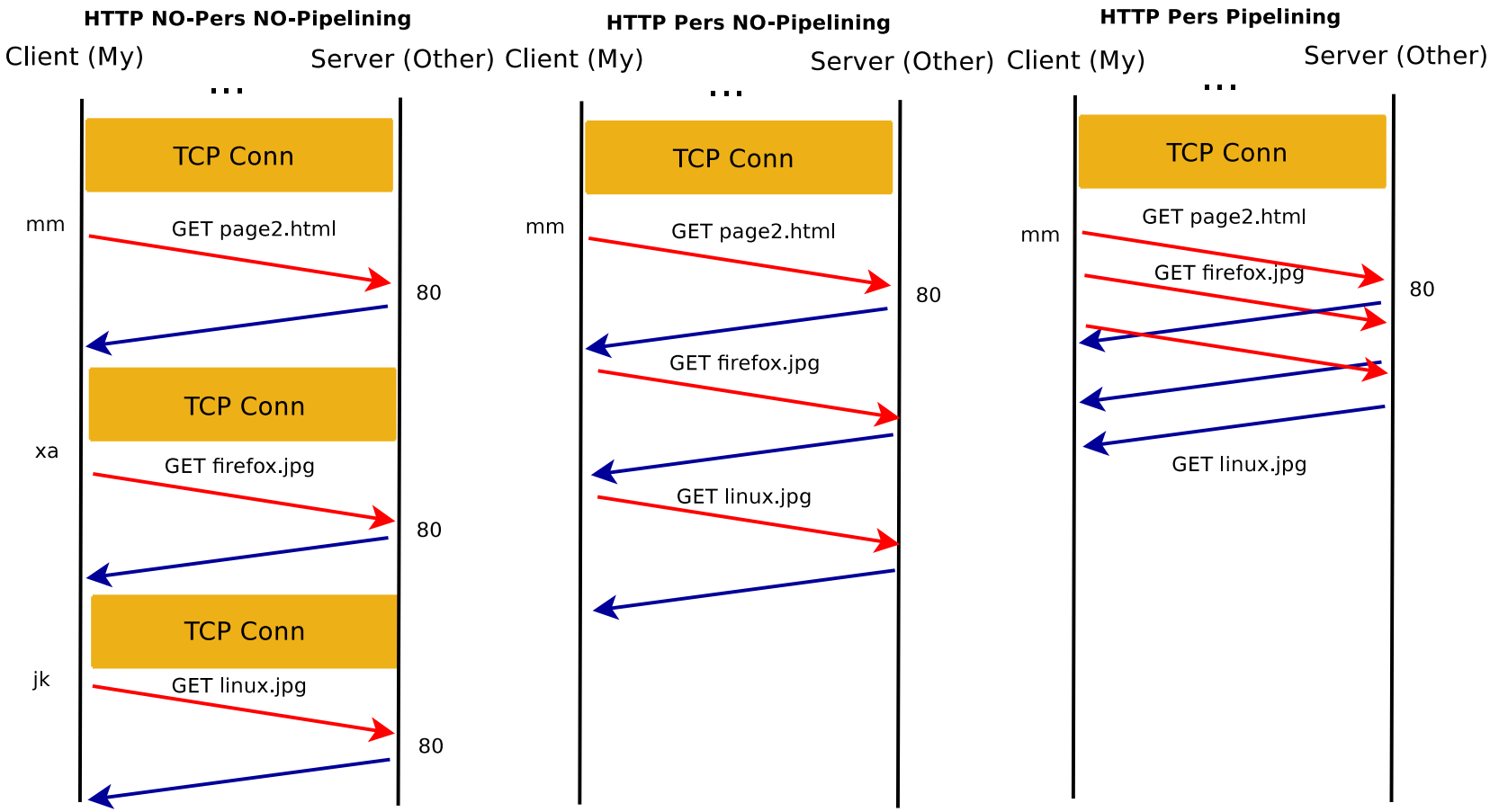
Versión HTTP 1.1

- HTTP/1.1, 1997 con la [RFC-2068] se actualiza con [RFC-2616].
- Nuevos mensajes HTTP 1.1: `OPTIONS`, `TRACE`, `CONNECT`.
- Conexiones persistentes por omisión.
- Pipelining, mejora tiempo de respuestas.

Pipelining HTTP 1.1

- No necesita esperar la respuesta para pedir otro objeto HTTP.
- Solo se utiliza con conexiones persistentes.
- Mejora los tiempos de respuestas.
- Sobre la misma conexión de debe mantener el orden de los objetos que se devuelven.
- Se pueden utilizar varios threads para cada conexión.
  - Sin pipelining: $1RTT + FT$ por cada objeto, n objetos $nRTT + nFT$.
  - Con pipelining óptimo: n objetos: $1RTT + nFT$.

Pipelining HTTP 1.1 (Cont'd)



TRACE y CONNECT HTTP 1.1

- `TRACE` utilizada para debugging.
- El servidor debe copiar el requerimiento tal cual.
- El cliente puede comparar lo que envía con lo que recibe.
- `CONNECT` utilizada para generar conexiones a otros servicios montadas sobre HTTP.
- Proxy-Agent genérico.

Ejemplo de CONNECT

```
CONNECT 127.0.0.1:25
HTTP/1.0 200 Connection Established
Proxy-agent: Apache/2.2.4 (Ubuntu)

220 khartum ESMTP Postfix (Ubuntu)
HELO otrohost.com
250 khartum
QUIT
221 2.0.0 Bye
...
```

Redirects HTTP

- Redirect temporal 302, indicando la nueva URL/URI.
- El user-agent no debería re-direccionarlo salvo que el usuario confirme.
- Moved Permanently 301, se indica que cualquier acceso futuro debe realizarse sobre la nueva ubicación (mejora Indexadores).
- Se pueden generar problemas con Cookies.

CGIs Scripts y JavaScript

- Necesidad de dinamismo para generar aplicaciones.
- Server-Side Script:
  - Ejecuta del lado del servidor.
  - CGI (Common Gateway Interface): aplicación que interactúa con un servidor web.
  - CGIs leen de STDIN (POST) o de variables de entorno (GET): `QUERY_STRING` datos de usuario.
  - Escriben en la STDOUT response.
  - Deben anteponer el content-type en el header.
  - POST permite enviar más datos.
  - Lenguajes de scripting más flexibles y seguros: PHP, ASP, JSP.
  - Implementados como CGIs, dentro o módulos propios del web-server.

CGIs Scripts y JavaScript (Cont'd)

- Client-Side Script:
  - Ejecuta del lado del cliente, en el browser.
  - JavaScript estándar W3C.
  - Usan modelo de objetos DOM (Document Object Model).
  - Otros lenguajes JScript, VBScript.
  - Permiten extensiones como AJAX (Asynchronous JavaScript And XML).
  - AJAX hace requerimientos particulares y no necesita recargar toda la página.
  - Parseo XML para comunicarse.
  - Existen numerosos frameworks que encapsulan esta funcionalidad brindando una interfaz de programación, API fácil de utilizar.

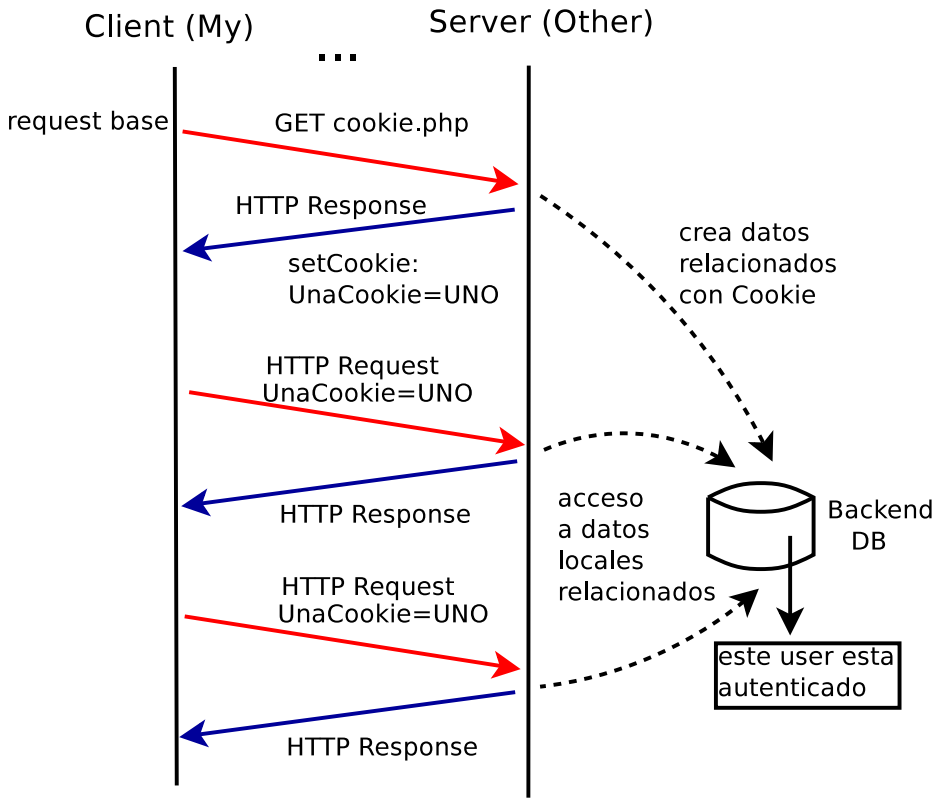
CGIs Scripts y JavaScript (Cont'd)

- Server-Side to Server-Side Script:
  - Permiten comunicación entre servidores.
  - Modelo de "objetos" y servicios distribuidos.
  - Conjunto de convenciones para implementar RMI (Remote Method Invocation) sobre HTTP (u otro protocolo de texto).
  - Previo XML-RPC.
  - SOAP (Simple Object Access Protocol).
  - Web-Services.
  - REST.

Cookies

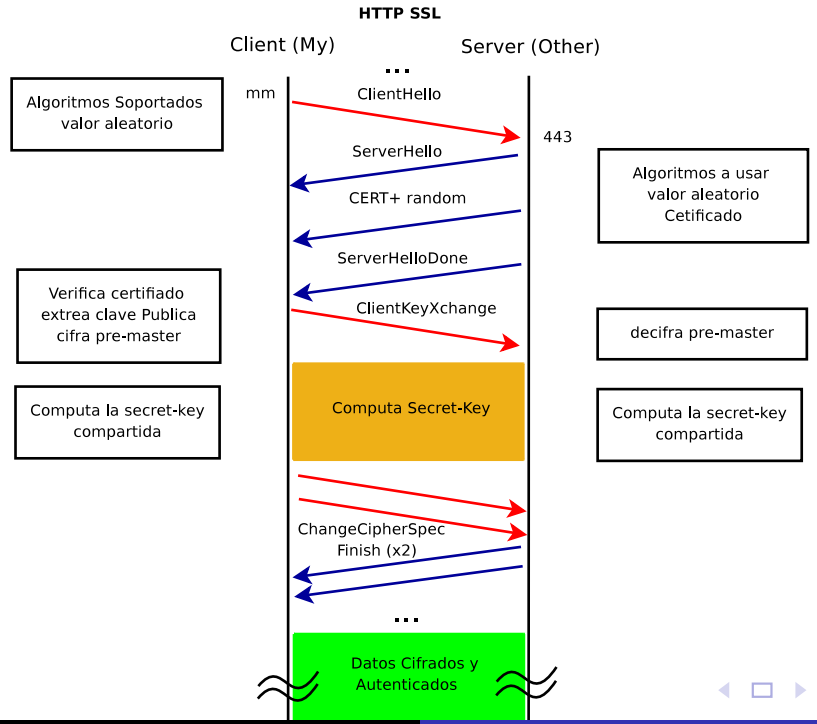
- Mecanismo que permite a las aplicaciones web del servidor "manejar estados".
- El cliente hace un request.
- El servidor retorna un recurso (un objeto HTTP, como una página HTML) indicando al cliente que almacene determinados valores por un tiempo.
- La Cookie es introducida al cliente mediante el mensaje en el header `Set-Cookie:` mensaje que indica un par (nombre,valor).
- El cliente en cada requerimiento luego de haber almacenado la Cookie se la enviará al servidor con el header `Cookie:`.
- El servidor puede utilizarlo o no.
- El servidor puede borrarlo.
- Esta información puede ser usada por client-side scripts.

Ejemplo de Cookie



HTTPS (HTTP sobre TLS/SSL)

- Utiliza el port 443 por default.
- Etapa de negociación previa.
- Luego se cifra y autentica todo el mensaje HTTP (incluso el header).



WEB-Cache

- "Proxiar" y Chachear recursos HTTP.
- Objetivos:
  - Mejorar tiempo de respuesta (reducir retardo en descarga).
  - Ahorro de BW (recursos de la red).
  - Balance de carga, atender a todos los clientes.
- Se solicita el objeto, si esta en cache y esta "fresco" se retorna desde allí (HIT).
- Si el objeto no esta o es viejo se solicita al destino y se cachea (MISS).
- Se puede realizar control de acceso.

WEB-Cache (Cont'd)

- Cache del lado del cliente.
- Los web browser tienen sus propias cache locales.
- Los servidores agregan headers:
  - `Last-Modified:` date
  - `ETag:` (entity tag) hash
- Requerimientos condicionales desde los clientes:
  - `If-Modified-Since:` date
  - `If-None-Match:` hash
- Respuestas de los servidores:
  - 304 Not Modified.
  - 200 OK.

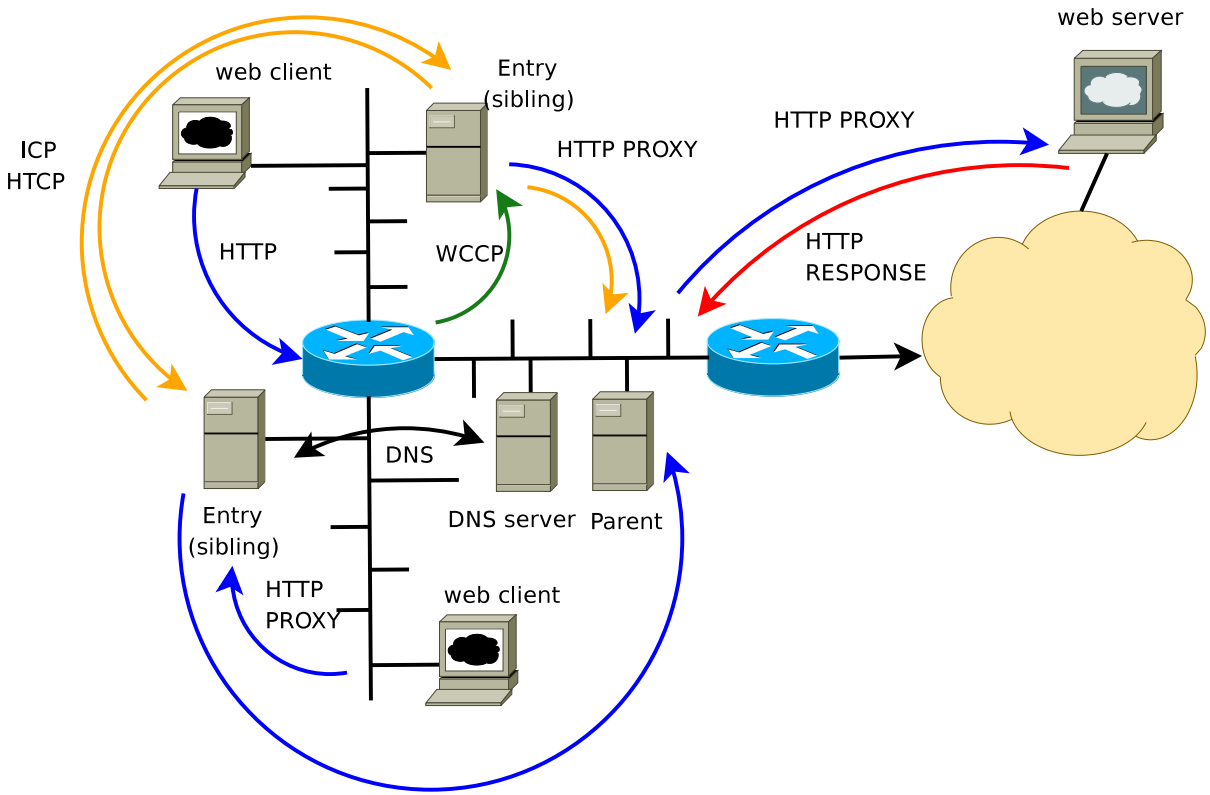
WEB-Cache (Cont'd)

- Los cache como servers funcionan como Proxy.
- Son servidores a los clientes y clientes a los servidores web.
- Los instalan ISP o redes grandes que desean optimizar el uso de los recursos.
- Existen:
  - Proxy no-transparente.
  - Proxy transparentes.
  - Proxy en jerarquía o mesh (ICP y HTCP).
  - CDN (Content Delivery Network), funcionan por DNS.

WEB-Cache (Cont'd)

- Protocolos de comunicación entre web-cache servers.
  - ICP (Internet Cache Protocol).
  - (HTCP) Hyper Text Caching Protocol.
- Diferentes relaciones: parent, siblings
- Protocolo de comunicación entre router y web-cache servers.
  - WCCP (Web Cache Control Protocol).
- En general corren sobre UDP.

WEB-Cache (Cont'd)




TCP/IP Illustrated, Volume 1: The Protocols, Addison-Wesley, (2nd. Ed). 2011. Kevin R. Fall, W. Richard Stevens.


Computer Networking: A Top-Down Approach, Addison-Wesley, (6th Edition). 2012. Kurose/Ross.


The Linux Home Page: http://www.linux.org/.


Linux in a Nutshell, Fourth Edition June, 2003. O'Reilly. Ellen Siever, Stephen Figgins, Aaron Weber.


http://www.rfc-editor.org/rfc/rfc793.txt. TCP Transmission Control Protocol (Jon Postel 1981 USC-ISI IANA).


The Original HTTP as defined in 1991.
http://www.w3.org/Protocols/HTTP/AsImplemented.html.


http://www.faqs.org/rfcs/rfc1738.html. Uniform Resource Locators (URL). (Berners-Lee, T., Masinter, L., and M. McCahill, CERN, Xerox PARC, University of Minnesota, 1994).


http://www.faqs.org/rfcs/rfc1866.html. Hypertext Markup Language - 2.0. (Berners-Lee (MIT/W3C) , D. Connolly, 1995).


http://www.w3.org/MarkUp/html3/CoverPage. Raggett, D., "HyperText Markup Language Specification Version 3.0", September 1995. (Available at


http://www.w3.org/TR/html401 Raggett, D., et al., "HTML 4.01 Specification", W3C Recommendation, December 1999.


http://www.w3.org/TR/xhtml1. "XHTML 1.0: The Extensible HyperText Markup Language: A Reformulation of HTML 4 in XML 1.0", W3C Recommendation, January 2000.


http://www.faqs.org/rfcs/rfc1945.html. Hypertext Transfer Protocol – HTTP/1.0. (T. Berners-Lee (MIT/LCS) , R. Fielding (UC Irvine) , H. Frystyk (MIT/LCS) , 1996).


HTTP/1.1. Fielding, R., Gettys, J., Mogul, J., Frystyk, H., and T. Berners-Lee, "Hypertext Transfer Protocol HTTP/1.1", RFC 2068, 1997.


http://www.faqs.org/rfcs/rfc2616.html. HTTP/1.1. (R. Fielding (UC Irvine) , J. Gettys (Compaq/W3C) , J. Mogul (Compaq) , H. Frystyk (W3C/MIT) , L. Masinter (Xerox) , P. Leach (Microsoft) , T. Berners-Lee (W3C/MIT), 1999)


http://www.ietf.org/rfc/rfc2246.txt. The TLS Protocol Version 1.0. (Dierks, C. Allen (Certicom), 1999).


http://www.w3.org/CGI/. The WWW Common Gateway Interface Version 1.1.


Referencias al método HTTP/1.0 PUT. http://www.apacheweek.com/features/put. http://www.w3.org/Amaya/User/Put.html. http://www.w3.org/Library/Examples/.


http://httpd.apache.org/docs/2.0/server-wide.html


Mozilla Firefox. http://www.mozilla.com.


http://www.oreilly.com/openbook/webclient/ Web Client Programming with Perl. Clinton Wong, 1997.


OpenSSL project: http://www.openssl.org/.


WebDAV: http://www.webdav.org/. RFC-4918.


http://marketshare.hitslink.com.

http://www.nationmaster.com.
http://www.w3counter.com/globalstats.php.
http://en.wikipedia.org/wiki/Usage_share_of_web_browsers.


http://news.netcraft.com.


The nature of the beast: recent traffic measurements from an Internet backbone. K Claffy, caida, kc@caida.org. Greg Miller and Kevin Thompson, MCI/vBNS, gmiller,kthomp@mci.net. 1998.


http://www.youtube.com/.


Ethereal, Wireshark. Autor original Gerald Combs, 2005.
http://www.ethereal.com/.
http://www.wireshark.org/.


W3C Recommendation (27 April 2007). SOAP Version 1.2 Part 0: Primer (Second Edition). W3C.
http://www.w3.org/TR/2007/REC-soap12-part0-20070427/.


Cookies Spec. http://curl.haxx.se/rfc/cookie_spec.html.