UNCUYO
UNIVERSIDAD NACIONAL DE CUYO
FACULTAD DE INGENIERÍA
Licenciatura en Ciencias de la Computación

Redes de Computadoras

Unidad 5

Capa de Aplicación

Dr. Ing. Pablo Daniel Godoy (pablo.godoy@ingenieria.uncuyo.edu.ar)

Modelo cliente servidor

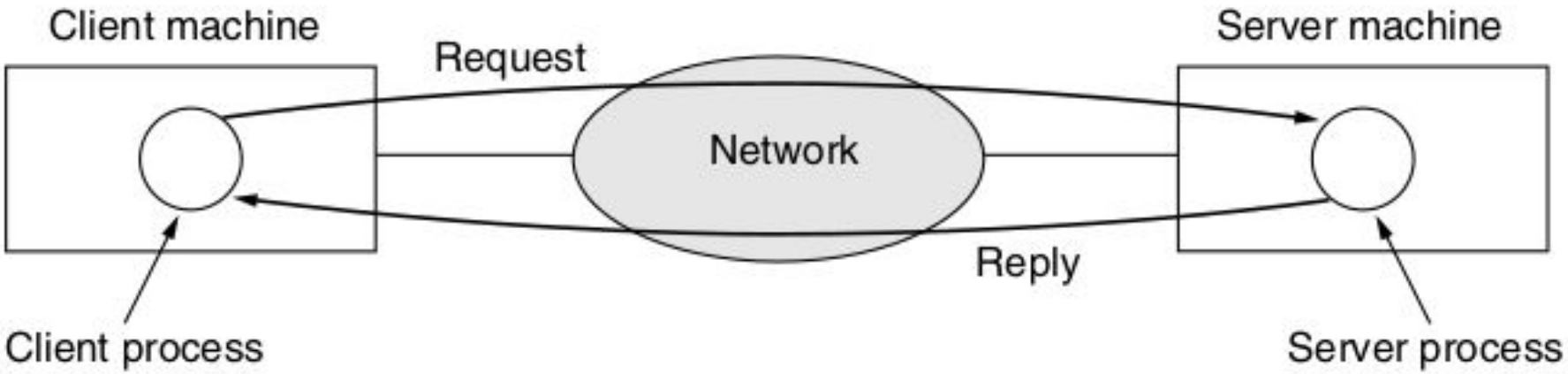
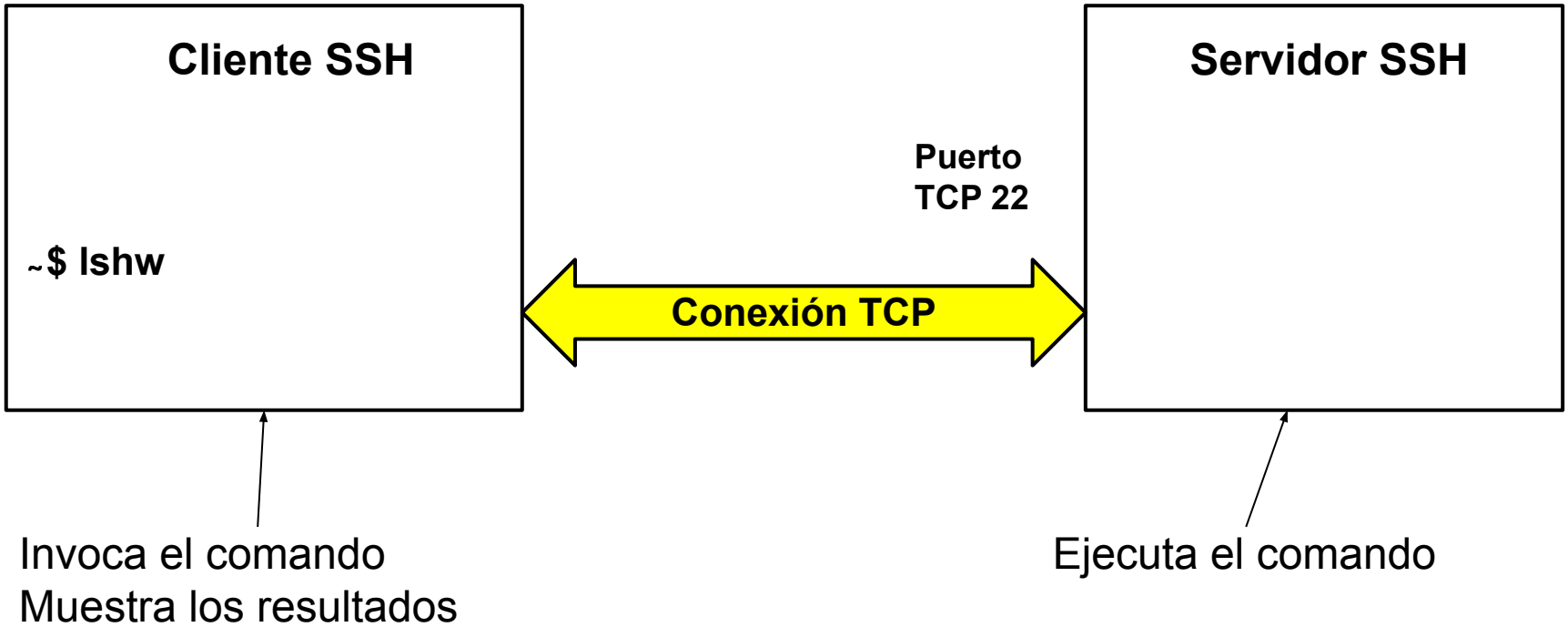


Figura obtenida de: A. Tanenbaum, D. Wetherall, "Redes de computadoras", Quinta edición (2012), pag. 4

SSH (Secure Shell)

- Aplicación cliente servidor que permite **ejecutar comandos** en una **máquina remota** a través de una consola de comandos.
- Ubuntu:
  - openssh-server y openssh-client (usualmente instalado por defecto).
  - Servicio: ssh.
- El servidor escucha en el puerto TCP puerto 22.
- Manual Linux: “man ssh”
- Requiere encriptación y autentificación.
- Petición:
  - **ssh usuario@ip** (establece una sesión con la máquina remota)
  - **ssh usuario@ip comando** (ejecuta el comando en la máquina remota)

SSH (Secure Shell)



SSH

Algunas opciones interesantes:

- Forwarding X11 (servidor gráfico o sistema de ventanas de Linux).
  - ssh **-X** usuario@IP
  - Previamente debe modificarse el archivo “/etc/ssh/ssh_config” configurando: **X11Forwarding yes**.

SCP (secure copy)

- Permite copiar archivos desde una máquina a otra.
- Trabaja sobre SSH.
- **scp origen destino**
  - **scp usuario@IP_origen:/ruta/archivo /ruta_destino**
  - **scp /ruta/archivo usuario@IP_destino:/ruta_destino**

FTP (File Transfer Protocol)

- Aplicación cliente-servidor para transferencia de archivos.
- Optimiza velocidad.
- Variantes:
  - FTP clásico. Puertos TCP 20 y 21.
  - TFTP (Trivial FTP), versión simple de FTP. Puerto UDP 69.
  - FTPS (FTP sobre SSL). Puertos TCP 989 y 990. Viene instalado con el paquete SSH.
- Ejemplo:
  - Servidor: **vsftpd** (Very Secure File Transfer Protocol Daemon)
  - Cliente:
    - **ftp** (línea de comandos Ubuntu)
    - **filezilla** (Ubuntu con interfaz gráfica)

DNS (Domain Name System)

- Desventajas de usar direcciones IP para identificar máquinas servidores:
  - Recordar direcciones IP es complejo.
  - Si el servidor se mueve a otra máquina con otra IP, hay que avisar a todos los clientes.
- Se necesita un mecanismo para convertir **nombres a direcciones IP**.
  - Las máquinas no comprenden nombres.
- Primer antecedente: archivo **hosts.txt** (RFC 606).
  - Archivo disponible online por el Network Information Center (NIC) en Stanford.
  - **Lista** con **nombres de computadoras** (servidores) e **IP** correspondientes.
  - Los que deseaban ofrecer servicios debían anotarse.
  - Los usuarios debían descargar el archivo a sus computadoras.
  - Problemas:
    - **Tamaño** del archivo hosts.txt que todos debían descargar.
    - **Conflictos** por los nombres.
    - **Cuello de botella** y **punto de falla**.

DNS (Domain Name System)

- RFC 1034 (inicial) + muchas otras.
- **Base de datos distribuida**, **jerárquico** basado en "**dominios**".
- Aplicación cliente-servidor sobre **UDP**.
  - Cliente (resolvedor) envía una solicitud de DNS al servidor local indicando un nombre de dominio.
  - El servidor busca la IP asociada al nombre de dominio y la informa al cliente.
    - Procedimiento usualmente de biblioteca incluido en todos los SO o lenguajes. Primitivas como "gethostbyname".
    - Puerto DNS: 53 (asignado por IANA).

DNS (Domain Name System)

- Administrador del **nivel superior** de la jerarquía de DNS: **ICANN**[1]
- Nivel superior: 250 dominios, divididos en dos grupos:
  - **Por paises** (ISO 3166)
  - **Genéricos**
    - Anteriores a 1980
    - Solicitados directamente a la ICANN.
- Los dominios pueden estar divididos en **subdominios de diversas jerarquías** (2, 3, 4, etc. jerarquías).
  - Nomenclatura: desde el más pequeño al más genérico.
    - **ingenieria.uncuyo.edu.ar**
  - Cada dominio controla los dominios debajo de él.

[1] Internet Corporation for Assigned Names and Numbers

DNS (Domain Name System)

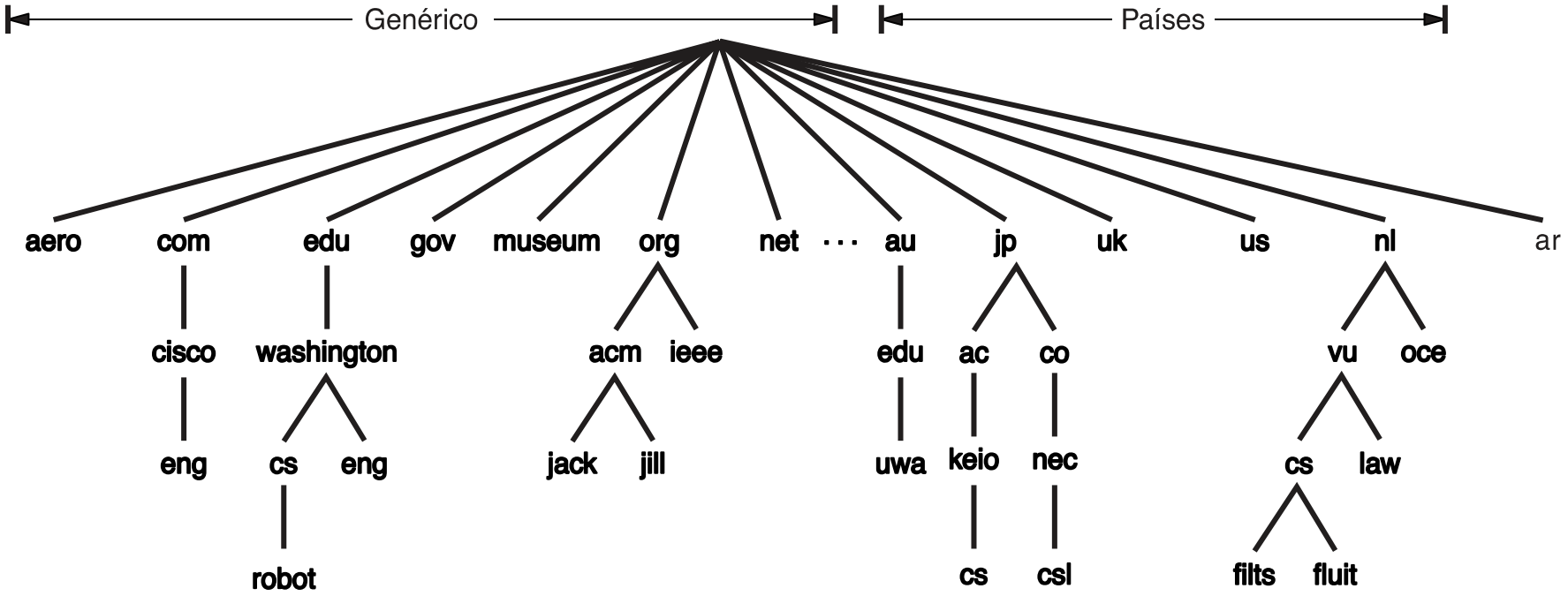


Figura obtenida de: A. Tanenbaum, D. Wetherall, "Redes de computadoras", Quinta edición (2012), pag. 526

DNS (Domain Name System)

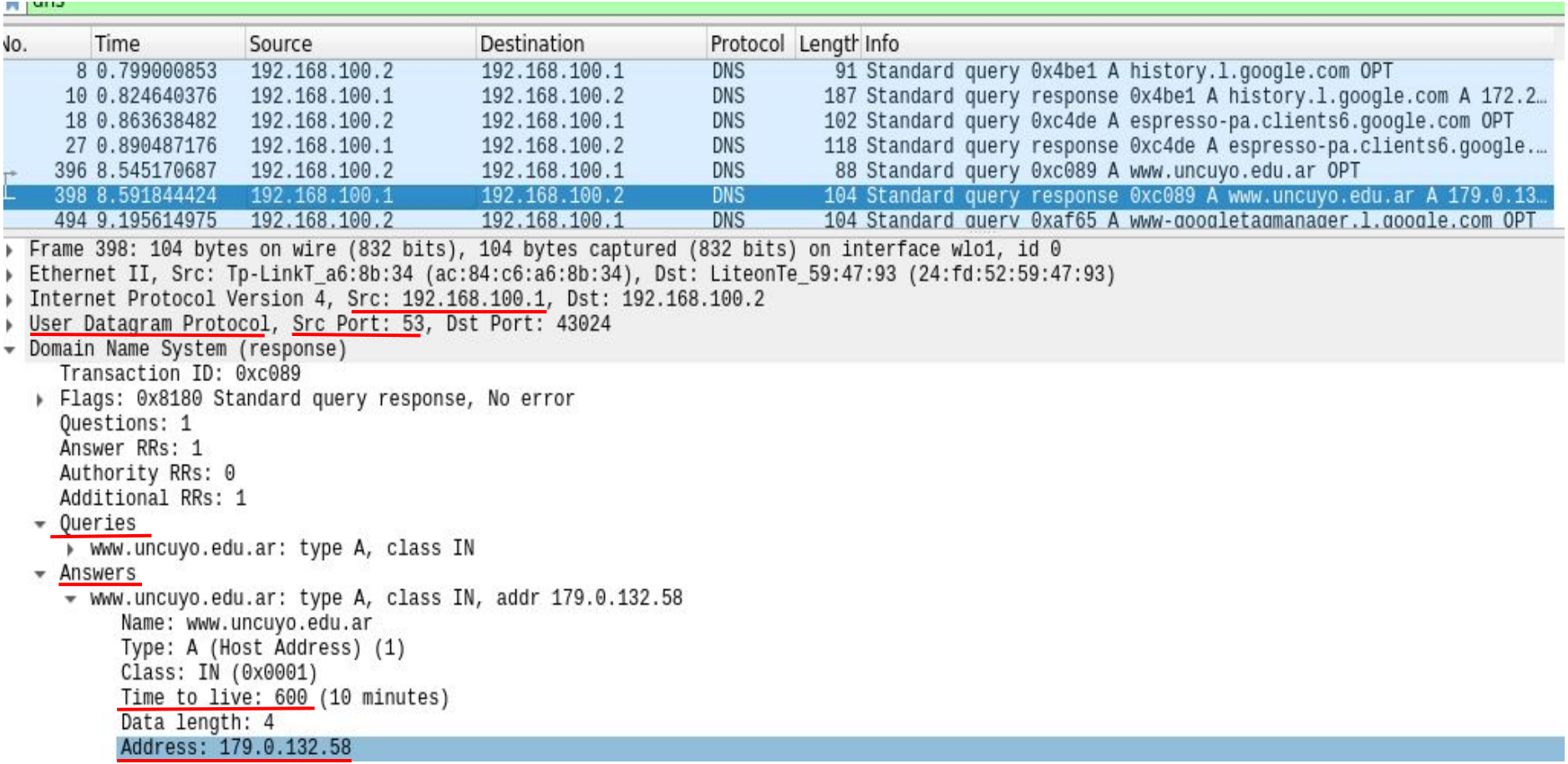
- Solo pueden utilizarse nombres disponibles y que no sean marcas comerciales registradas.
- Algunos dominios tienen restricciones (edu, mil, gov), otros no (com, net).
- Algunos problemas: Ciberocupación.
  - Ejemplos de dns en venta en 2022: www.dinerodesdecasa.com
  - Ejemplos de nombres de dominios ciberocupados y vendidos:
    - pizza.com (registrado por 20 USD, vendido por 2.6 millones de USD).
    - business.com (vendido en 2007 por 345 millones de USD. El más caro).
    - vacationrentals.com (vendido en 2007 por 35 millones de USD, para impedir que la competencia lo comprara).
    - fb.com (comprado por Facebook en 2010 por 8.5 millones de USD)
    - tesla.com (comprado por 11 millones de USD)
    - BuyDomains (colección con cientos de miles de DNS vendido por 80 millones de USD. “trolls de dominios”).

DNS (Domain Name System)

	Uso	Restricciones	Ejemplo
.com	Comercial	No	google.com
.edu	Instituciones educativas	Si	uncuyo.edu.ar
.mil	Militar	Si	colegiomilitar.mil.ar
.gov	Gobierno	Si	mendoza.gov.ar
.aero	Transporte aereo	Si	apcon.aero
.net	Proveedores de red, comercial	No	coches.net
.biz	Negocios	No	educacion.biz
.org	Organizaciones sin fines de lucro	Si	change.org
.xxx	adultos	No	

DNS (Domain Name System)
Registros de recursos de dominios

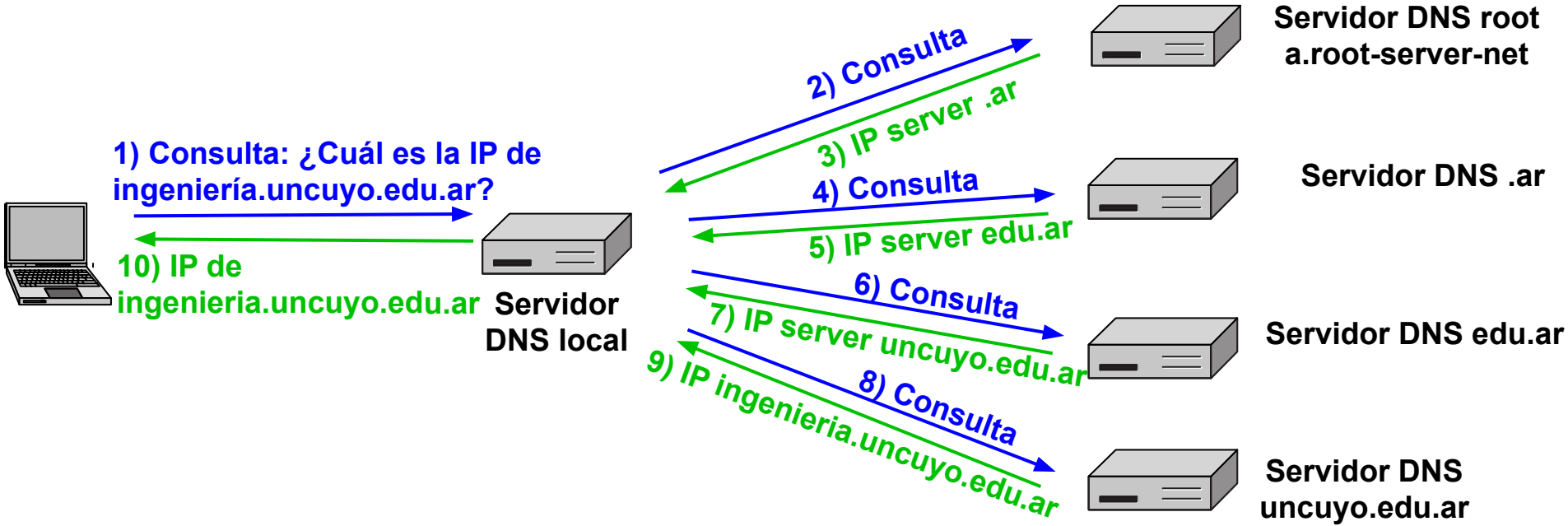
- Cada dominio posee un conjunto de **registros de recursos**.
- Los registros poseen la siguiente información:
  - Nombre de dominio
  - Tiempo de vida: En segundos. Indica la estabilidad del registro. Mientras más estable la información, mayor el valor.
  - Clase: Para Internet vale IN. Otros valores no se utilizan.
  - Tipo:
    - A: Dirección IPv4
    - AAAA: Dirección IPv6
    - NS: Nombre de un servidor para este dominio.
  - Cname: canonical name. El verdadero dns de la IP. DNS permite asociar varios “alias” a un mismo DNS, para ofrecer diferentes servicios.
  - Valor: Valor del registro.


No. Time Source Destination Protocol Length Info
8 0.799000853 192.168.100.2 192.168.100.1 DNS 91 Standard query 0x4be1 A history.l.google.com OPT
10 0.824640376 192.168.100.1 192.168.100.2 DNS 187 Standard query response 0x4be1 A history.l.google.com A 172.2…
18 0.863638482 192.168.100.2 192.168.100.1 DNS 102 Standard query 0xc4de A espresso-pa.clients6.google.com OPT
27 0.890487176 192.168.100.1 192.168.100.2 DNS 118 Standard query response 0xc4de A espresso-pa.clients6.google.…
396 8.545170687 192.168.100.2 192.168.100.1 DNS 88 Standard query 0xc089 A www.uncuyo.edu.ar OPT
398 8.591844424 192.168.100.1 192.168.100.2 DNS 104 Standard query response 0xc089 A www.uncuyo.edu.ar A 179.0.13…
494 9.195614975 192.168.100.2 192.168.100.1 DNS 104 Standard query 0xaf65 A www-googletagmanager.l.google.com OPT
Frame 398: 104 bytes on wire (832 bits), 104 bytes captured (832 bits) on interface wlo1, id 0
Ethernet II, Src: Tp-LinkT_a6:8b:34 (ac:84:c6:a6:8b:34), Dst: LiteonTe_59:47:93 (24:fd:52:59:47:93)
Internet Protocol Version 4, Src: 192.168.100.1, Dst: 192.168.100.2
User Datagram Protocol, Src Port: 53, Dst Port: 43024
Domain Name System (response)
Transaction ID: 0xc089
Flags: 0x8180 Standard query response, No error
Questions: 1
Answer RRs: 1
Authority RRs: 0
Additional RRs: 1
Queries
www.uncuyo.edu.ar: type A, class IN
Answers
www.uncuyo.edu.ar: type A, class IN, addr 179.0.132.58
Name: www.uncuyo.edu.ar
Type: A (Host Address) (1)
Class: IN (0x0001)
Time to live: 600 (10 minutes)
Data length: 4
Address: 179.0.132.58

DNS (Domain Name System)
Servidores de nombres

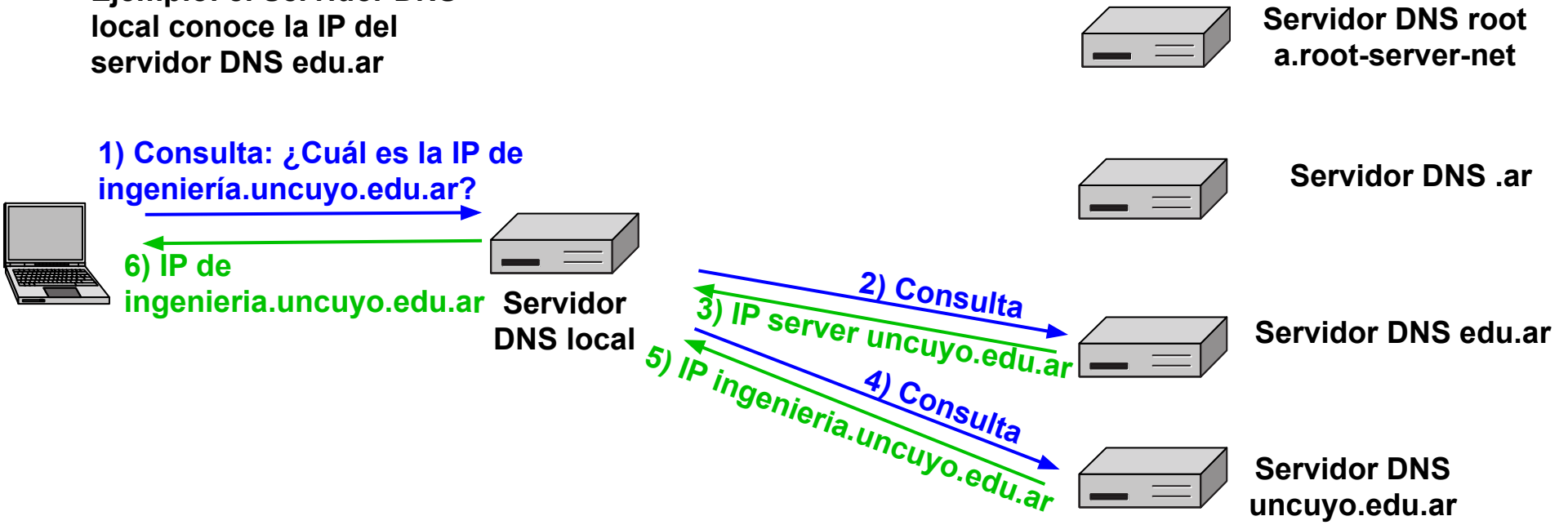
- Sistema distribuido y jerárquico.
- Nivel más alto: **13 servidores raiz** (https://root-servers.org/).
  - Nombres: **letra-root-servers.net**, donde letra va de a a m. (ejemplo: a-root-servers.net, b-root-servers.net, ….).
  - Cada servidor raíz es un sistema distribuido con servidores replicados en muchos países.
  - Direccionamiento anycast.
- Servidores de los dominios de mayor jerarquía:
  - Dominios genéricos
  - Dominios por paises
- Servidores de menor jerarquía.
- El servidor DNS local es el responsable de encontrar a IP correspondiente a un nombre de dominio

DNS (Domain Name System)
Proceso de resolución de nombres

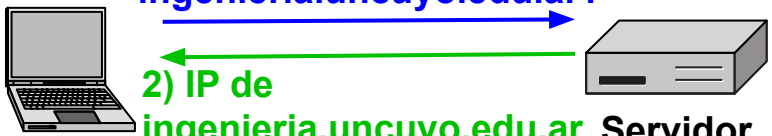


DNS (Domain Name System)
Proceso de resolución de nombres + Caché

Ejemplo: el Servidor DNS local conoce la IP del servidor DNS edu.ar



DNS (Domain Name System)
Proceso de resolución de nombres + Caché




Servidor DNS root
a.root-server-net


Servidor DNS .ar

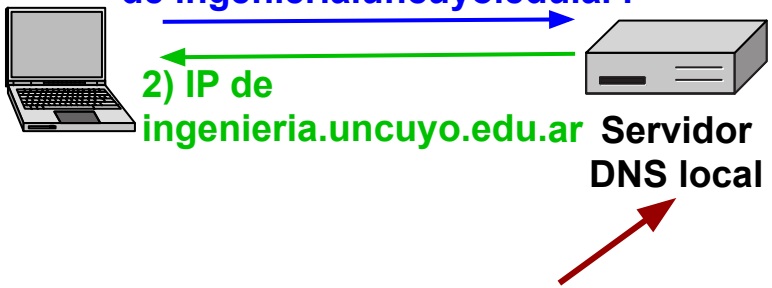

Servidor DNS edu.ar


Servidor DNS
uncuyo.edu.ar

Problema: ¿Que pasa si una asignación DNS-IP cambia?
Solución: Registros autorizados vs Registros en caché

DNS (Domain Name System)
Proceso de resolución de nombres + Caché

Registros autoriados: Los brindados por la autoridad inmediata superior. Ejemplo: El registro autorizado de ingenieria.uncuyo.edu.ar es el que brinda el servidor uncuyo.edu.ar




Servidor DNS root a.root-server-net


Servidor DNS .ar


Servidor DNS edu.ar

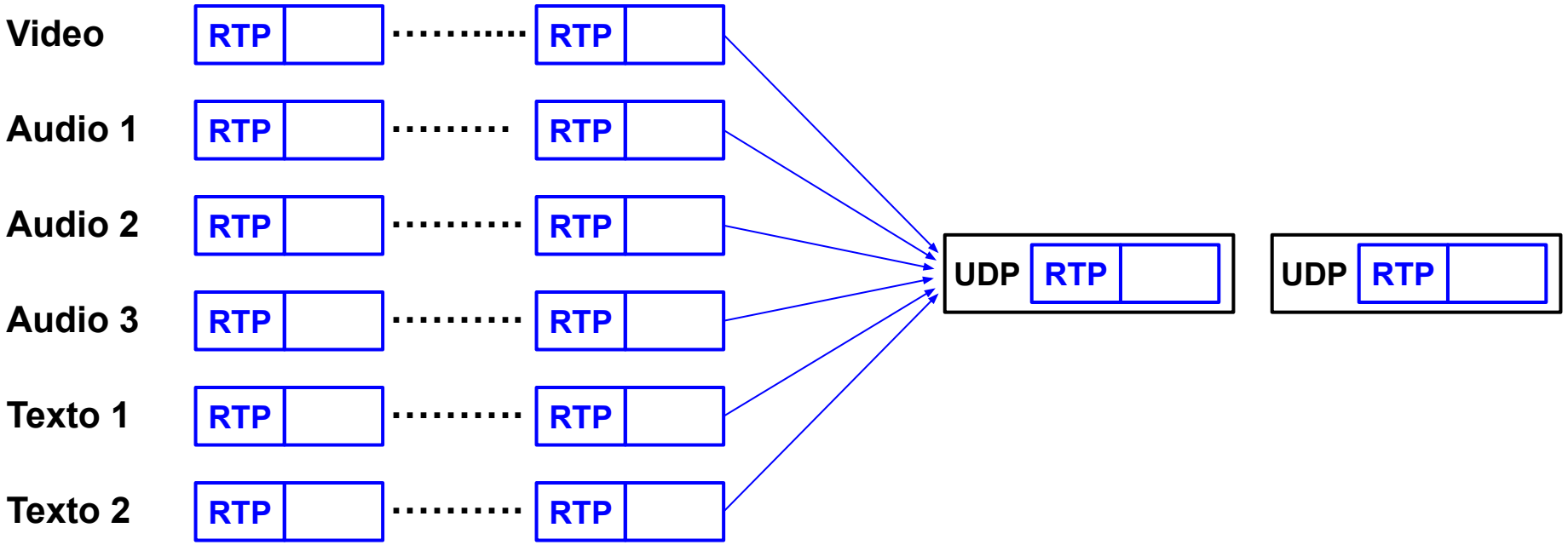

Servidor DNS uncu.edu.ar

No es una autoridad de ingeniería.uncuyo.edu.ar, su registro será válido por el tiempo indicado en el campo tiempo de vida. Luego, el registro expirará, y una consulta a ingenieria.uncuyo.edu.ar comenzará nuevamente el proceso

RTP (Real-time Transport Protocol) y RTCP (Real Transport Control Protocol)

- **Multimedia en tiempo real** (RFC 3550, año 2003).
  - Telefonía y videoconferencias en tiempo real.
  - Radio sobre Internet.
  - Música y vídeo bajo demanda.
- Transporta datos de audio y video en paquetes.
- Mayoritariamente UDP.
- Una aplicación multimedia consiste en **varios flujos** de datos:
  - Uno de video
  - Varios de sonido:
    - Dos o más para sonido estéreo o envolvente.
    - Sonido en varios idiomas.
  - Texto (subtítulos)

RTP (Real-time Transport Protocol) y RTCP (Real Transport Control Protocol)



RTP y RTCP

- Cada paquete posee un **número de secuencia** y una **estampa de tiempo**.
  - Número de secuencia: permite ordenarlos.
    - Si un paquete se pierde, puede ignorarse o deducirse su valor desde otros paquetes (para audio), **nunca retransmitirse**.
  - Estampa de tiempo:
    - Permite que cada paquete se **reproduzca** en el **momento justo**.
    - Permite **sincronizar** diferentes flujos.
    - Son **relativas al primer paquete**.
- Cada flujo puede codificarse en diferentes formas (MP3, MP4, PCM, etc.), y el código puede cambiar dinámicamente según el ancho de banda disponible en la red (por eso los paquetes indican el código utilizado en el campo carga útil).

RTP y RTCP

- **Jitter**: Variación en el retardo. Parámetro crítico en multimédica.
- Solución: **buffer en el receptor** que almacena los paquetes y permiten retrasar la reproducción.
- Tiempo en el buffer: parámetro crítico que depende de cada aplicación.

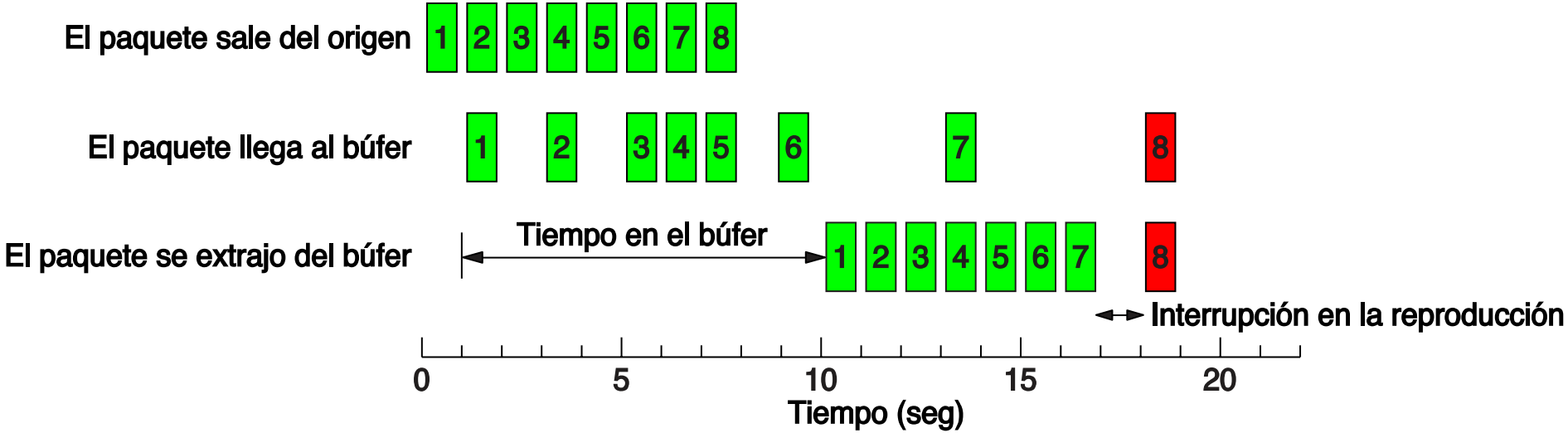
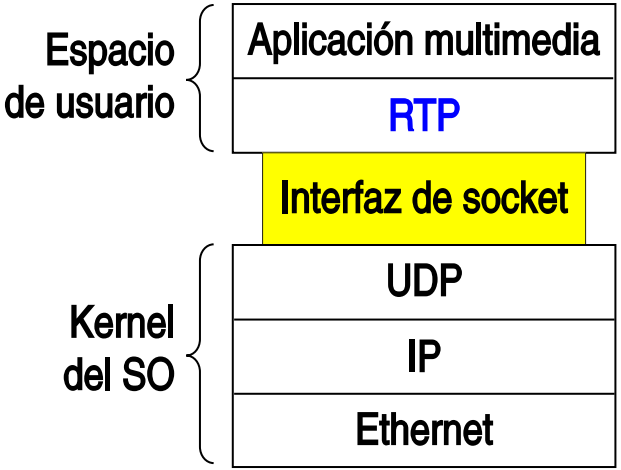


Figura obtenida de: A. Tanenbaum, D. Wetherall, "Redes de computadoras", Quinta edición (2012), pag. 472



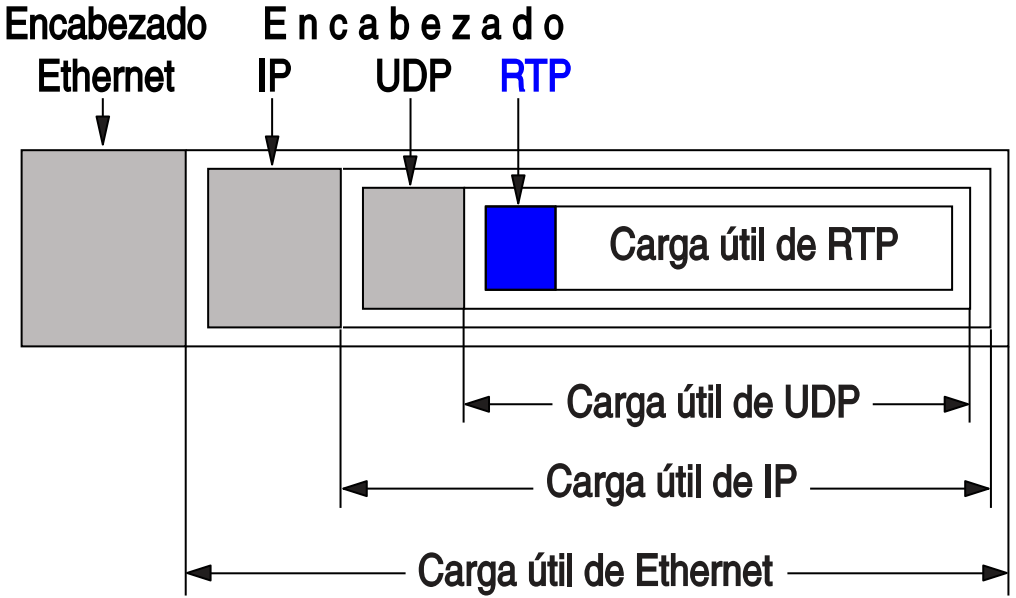


Figura obtenida de: A. Tanenbaum, D. Wetherall, "Redes de computadoras", Quinta edición (2012), pag. 469

RTP (Real-time Transport Protocol) y RTCP (Real Transport Control Protocol)

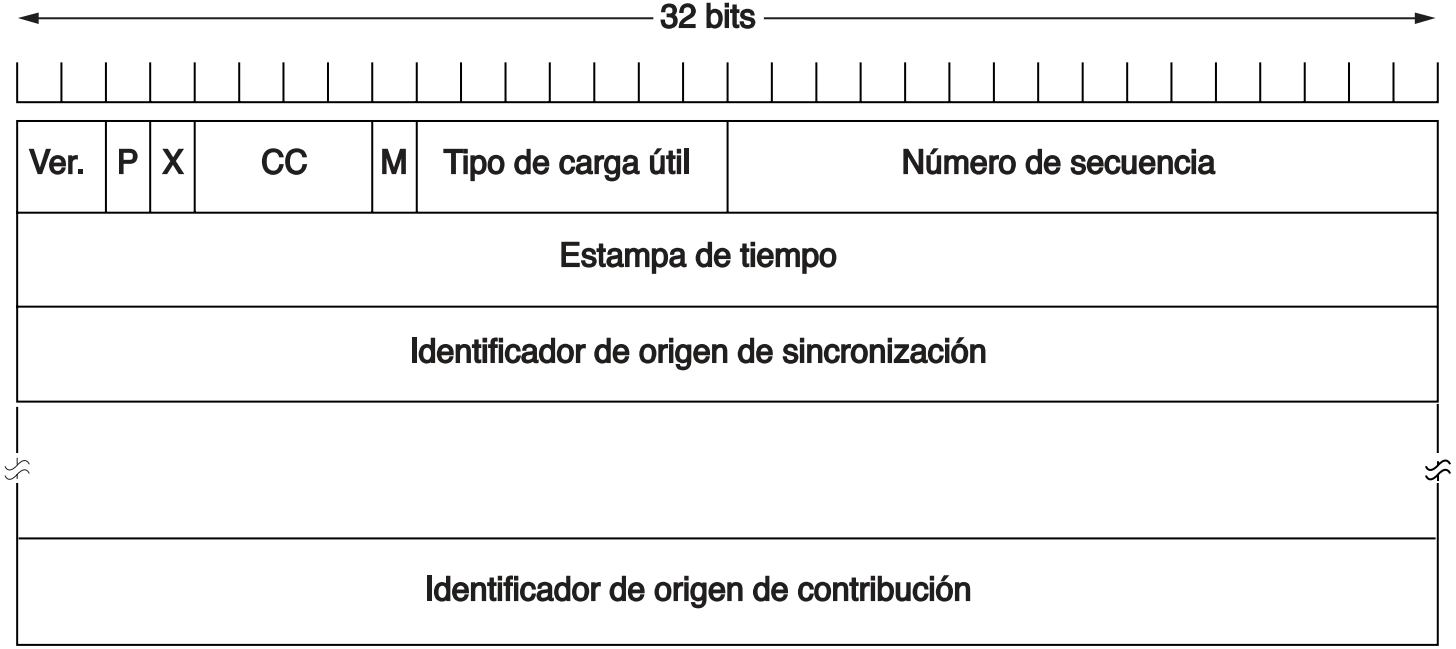


Figura obtenida de: A. Tanenbaum, D. Wetherall, "Redes de computadoras", Quinta edición (2012), pag. 471

RTP y RTCP

- Ver.: Versión
- P: Indica si se han agregado bytes de relleno (utilizado para cifrado).
- X: Hay encabezado de extensión.
- CC: Número de fuentes de contribución.
- M: Bit que se deja para que la aplicación lo utilice.
- Tipo de carga útil: Algoritmo de codificación utilizado para los datos de la carga útil (RFC 1890).
- Origen de sincronización: Puede haber un origen diferente para la sincronización.
- Origen de contenidos: Lista de orígenes del contenido.
- RTCP: Envía mensajes de retroalimentación al emisor indicando que:
  - Puede aumentar o debe disminuir la velocidad de transmisión
  - El emisor puede cambiar el algoritmo de codificación.

SMTP

- SMTP (Simple Mail Transfer Protocol, Protocolo Simple de Transferencia de Correo). RFC 821 y RFC 5321.

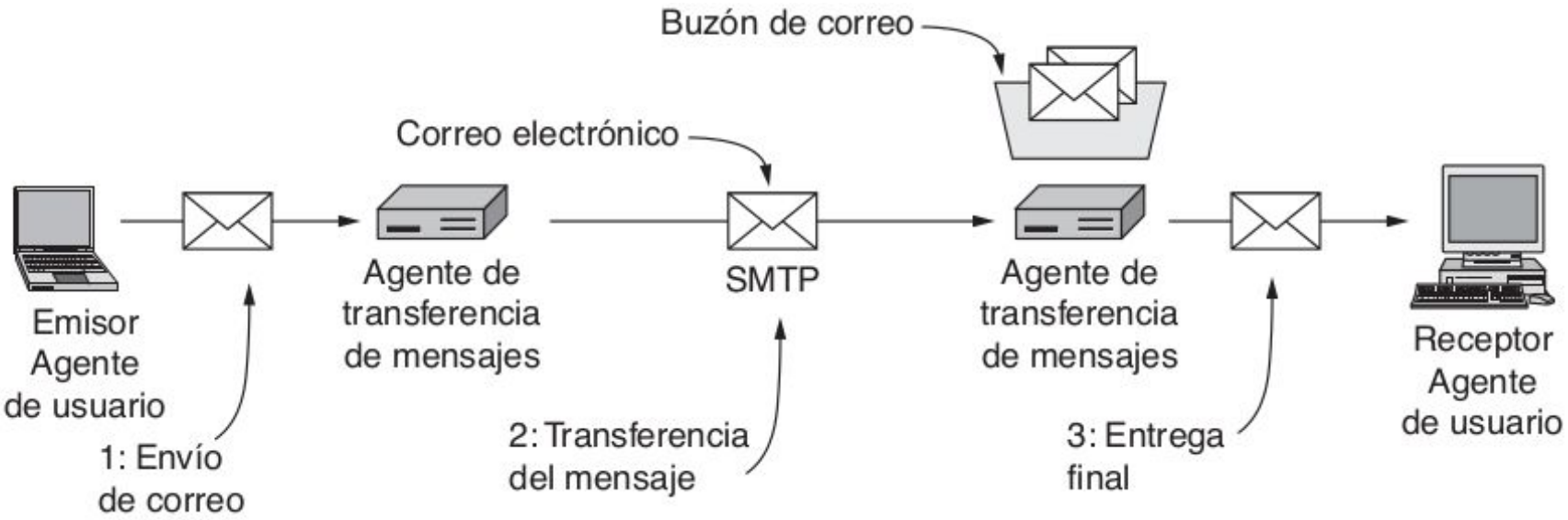


Figura obtenida de: A. Tanenbaum, D. Wetherall, "Redes de computadoras", Quinta edición (2012), pag. 536

SMTP

- Se encarga de:
  - Mover el e-mail desde el origen al destino.
  - Notificación de estado de la entrega y errores.
- Basado en el sistema DNS (registros MX): usuario@dirección-dns.
- Trabaja sobre **TCP**. No realiza cifrado ni autentificación.
- Mejoras.
  - Versión original: Solo texto ASCII de 7 bits.
  - **MIME** (Extensiones Multipropósito de Correo Internet). Indica el **contenido del e-mail**, para que la **aplicación cliente sepa si podrá mostrar el contenido**, o necesitará instalar **plugins** o **ejecutar aplicaciones externas**.
  - ESMTP: Extensiones a SMTP (RFC 1869).
- Otros protocolos que intervienen:
  - IMAP (actual) o Pop3 (anterior): Protocolo para entrega de correos a la aplicación cliente final (aplicación de usuario).

WWW (World Wide Web)

- Enorme **colección de contenido** en forma de documentos (archivos) llamadas páginas web. Estos archivos contienen:
  - Vínculos a otras páginas web del mundo
  - Instrucciones de como presentar el contenido (**html**).
  - Instrucciones de lenguajes encriptados a ejecutar por el cliente (**Javascript**).
  - Instrucciones a ejecutar por el servidor (**PHP**, **ASP**).
- **Es un gran sistema distribuido** en millones de máquinas.
- Comenzó en el CERN (Centro europeo para la investigación nuclear) en 1989.
  - Tim Berners-Lee (Desarrolló HTML, HTTP y URL. Fundó el W3C)
- Consorcio W3C (World Wide Web Consortium):
  - Crear protocolos.
  - Fomentar la interoperabilidad.
  - www.w3.org

WWW (World Wide Web)

- Los documentos se escriben usando **lenguaje HTML** (HyperText Markup Language o lenguaje de marcas de hipertexto).
- Sigue el modelo **cliente-servidor**:
  - El cliente (navegador) solicita al servidor una página web (archivo html).
  - El servidor envía la página web.
  - El cliente la interpreta la página web.
- El protocolo que controla el proceso de intercambio de documentos web se denomina **HTTP (HyperText Transfer Protocol)**.
  - Los servidores HTTP trabajan sobre **TCP puerto 80 (http) o 443 (https)**.

WWW (World Wide Web)

- **Navegador web**: Cliente HTML Intérprete HTML y lenguajes encapsulados (scripts).
  - Generar texto, imágenes, formatos, etc., de acuerdo a lo indicado en la página mediante HTML.
  - Pedir contenido a otros sitios.
  - Ejecutar secuencias de comandos (lenguajes incrustados).
- Páginas web pueden ser
  - **Estáticas**: son siempre iguales.
  - **Dinámicas**: No son siempre iguales.
    - Un programa en el servidor las genera cuando el cliente las solicita.
    - Ejecuta instrucciones en el servidor o el cliente que modifican la página.

Primer navegador gráfico: Mosaic, luego mejorado y llamado Netscape Navigator.
Google Chrome y Firefox: botón derecho y luego “ver código fuente”

WWW (World Wide Web)

- **URL (Uniform Resource Locator)**: **identifica** de manera unívoca **como llegar** a un recurso (página web, imagen, video, etc.)
- Tres partes:
  - Protocolo (http, https, ftp, mailto, file, etc.)
  - IP o DNS: www.ingenieria.uncuyo.edu.ar (el www es opcional)
  - Ruta al recurso en la computadora donde se encuentra el recurso.
    - Ejemplos:
      - https://ingenieria.uncuyo.edu.ar/estudios/carrera/licenciatura-en-ciencias-de-la-computacion/index.php
      - http://ingenieria.uncuyo.edu.ar/index.php
    - Los servidores HTTP indican una ruta de trabajo (usualmente **/var/www/html/**).
    - Usualmente la ruta indicada en la URL es relativa a la ruta de trabajo.
    - Los archivos index.html, index.php, etc. son por defecto. No necesitan especificarse.

WWW (World Wide Web)

- **URI (Uniform Resource Identifiers)**:
  - URI que indican cómo llegar al recurso: URL
  - URI que no indican cómo llegar al recurso: **URN** (Uniform Resource Names)
    - Desventaja de los URL: Si cambia la ubicación de los recursos, **cambia el URL**, pero **no cambia el URN**.
    - Utilizados cuando los recursos pueden haber copias de los recursos distribuidos en diferentes ubicaciones.
    - Necesidad de un “resolvedor de nombres”.
    - Ejemplo URN: Doi (Digital Object Identifier) https://www.doi.org/
    - 10.1016/S0169-7552(98)00110-X
  - Reglas para escribir URI: RFC 3986.

WWW (World Wide Web)
Extensiones

- Los tipos de archivos que los proveedores desean proveer a los clientes son cada vez más variados (video, imágenes, etc.).
- Muchos tipos de documentos pueden ser manejados por los navegadores, pero algunos no. Soluciones:
  - Complementos (plug-ins, extensiones, etc.).
  - Llamar a aplicaciones auxiliares.
- Cuando un servidor devuelve una página web, también devuelve información sobre el tipo de contenido (llamados tipos MIME[1]).

[1] Multipurpose Internet Mail Extensions (Extensiones Multipropósito de Correo Internet). Fueron creadas para el correo electrónico, luego utilizadas para páginas web. Ejemplos: text/html, image/jpeg, image/png, audio/mpeg, video/mp4, application/xml, application/pdf

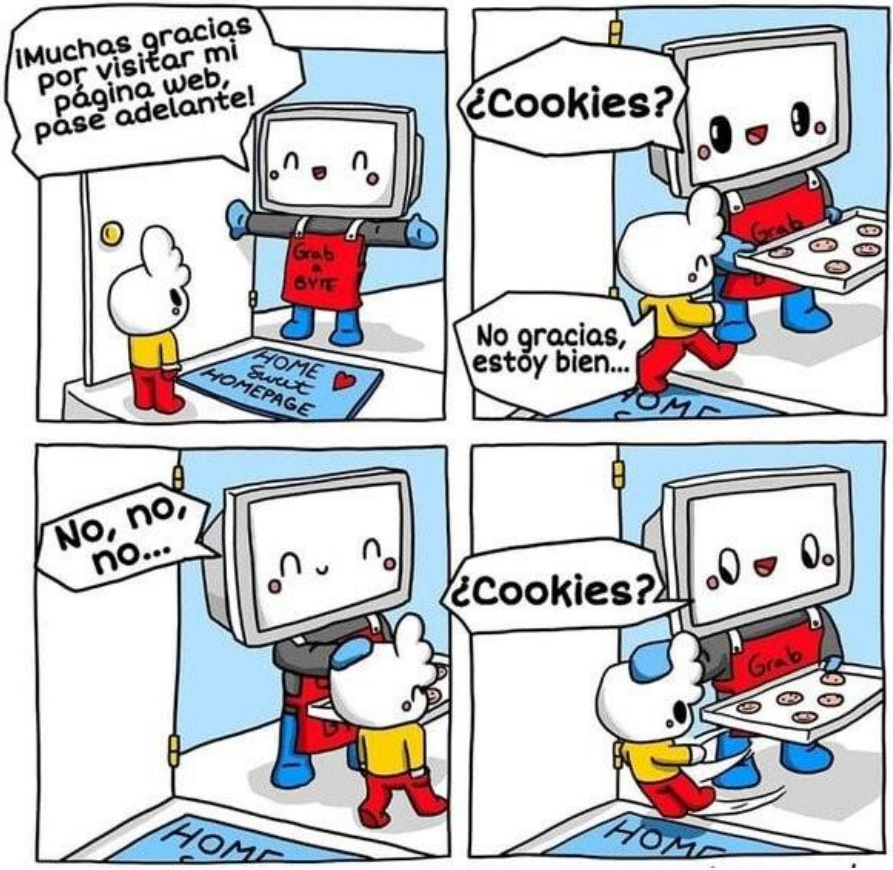
WWW (World Wide Web)

¿Qué hace el **cliente** (navegador) cuando escribimos la URL?

1. Verifica si hay **cookies** asociados con el DNS solicitado.
2. Obtiene la **IP** del servidor mediante **DNS**.
3. Solicita una **conexión TCP** al puerto 80 (HTTP) o 443 (HTTPS) de la máquina indicada.
4. Realiza una **petición HTTP** incluyendo la **ruta** a la página web indicada.
   a. Anexa **información adicional** (navegador, versión, idioma, cookies, etc.).
5. Recibe el **archivo con código html** y lo interpreta.
6. Obtiene las **URLs de recursos incluidos en la página** (imágenes, videos, banners de publicidad, etc.) y realiza las correspondientes peticiones HTTP.
   a. Obtiene estos recursos realizando los pasos anteriores para cada uno.
7. Si la conexión TCP permanece inactiva, la libera.
8. **Ejecuta secuencias de instrucciones** en algún lenguaje incrustado (**Javascript**).
   a. Llamadas por eventos (presionar un botón, mover el mouse, cargar la página).
9. Permite ejecutar **otros protocolos** sobre el navegador (ftp, video, etc.)

Cookies

- Archivos que almacenan información enviada por los servidores web.
- Cada cookie está asociada a un DNS.
- Información que almacenan:
  - Cantidad de visitas.
  - Contenido carrito de compras.
  - Preferencias de navegación (idioma, autocompletar, etc.)
  - Información para mostrar publicidad “de interés para el usuario”.
- Desarrolladas por Netscape en 1994.
- Reglamento General de Protección de Datos (2018) de la UE: los usuarios deben dar su consentimiento explícito para permitir cookies en su computadora.



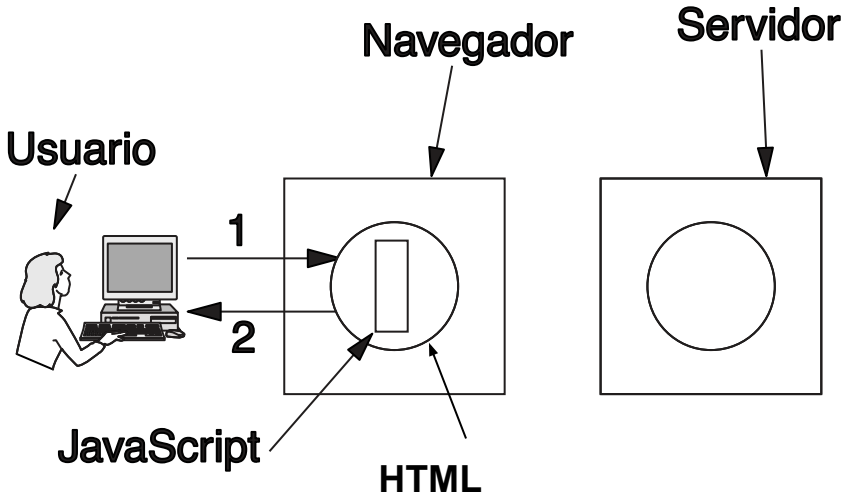
WWW (World Wide Web)

¿Qué hace el **servidor** web cuando recibe una petición?

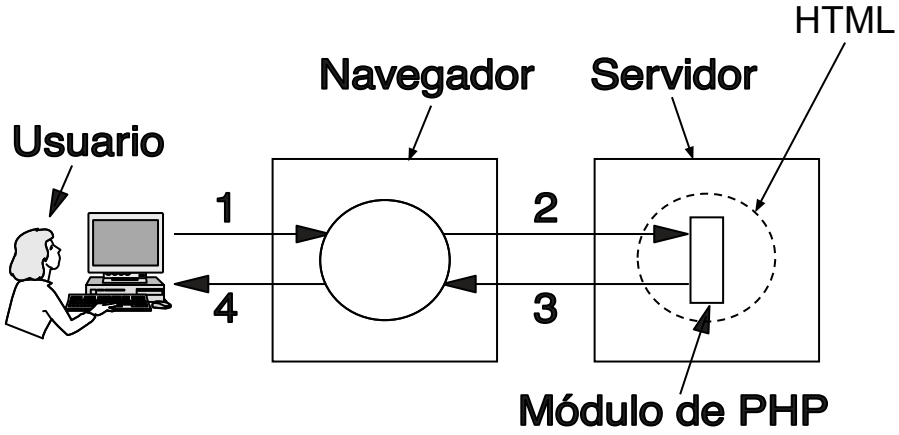
- **Espera y acepta conexiones TCP** al puerto 80 (http) o 443 (http).
- **Recibe y acepta peticiones HTTP**.
- Cuando recibe la petición HTTP:
  - **Obtiene la página** solicitada (archivo escrito con código HTML) e información adicional provista en la petición (navegador del usuario, idioma, cookies, etc).
    - Página por defecto: index.html o index.php
  - **Ejecuta** secuencias de instrucciones de **lenguajes del lado del servidor** (PHP, ASP, CGI) para generar o modificar la página web.
  - Determina **información adicional** a incluir en la respuesta (tipos **MIME**).
- **Envía la respuesta** al cliente empleando http o https.
  - Puede haber diferentes respuestas según el navegador del usuario, idioma, dispositivo utilizado (PC, teléfono móvil, etc.)
- Servidores web populares: **Apache**.

Páginas Web dinámicas

Ejecución de instrucciones del lado del cliente:
JavaScript, VBScript, applets

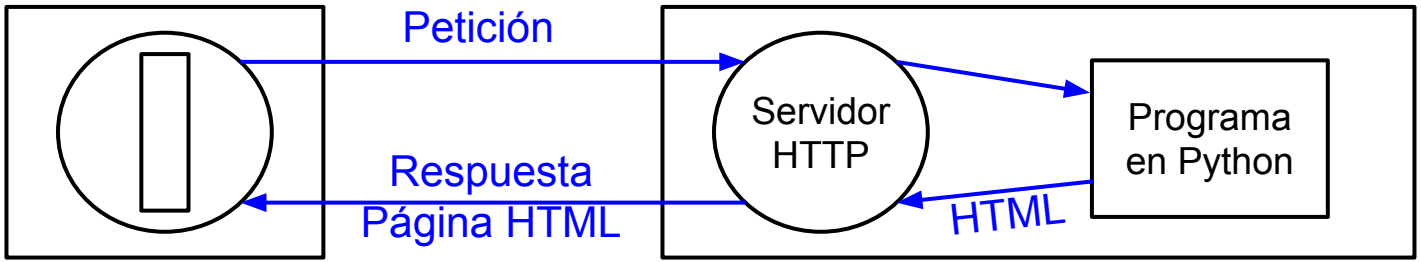


Ejecución de instrucciones del lado del servidor
PHP, ASP, CGI



Ejecución del lado del Servidor: CGI

- CGI (Common Gateway Interface o Interfaz de Puerta de Enlace Común).
  - RFC 3875.
  - Interfaz que permite a un servidor web comunicarse con programas externos que pueden aceptar entradas y generar páginas HTML en respuesta.
    - Pueden estar escritos en cualquier lenguaje (Python, Ruby, Perl, C++, etc.)



Ejecución del lado del Servidor: PHP

- **PHP** (Hypertext Preprocessor - Preprocesador de Hipertexto).
- Las instrucciones **PHP se incrustan dentro de los documentos HTML**, que el servidor ejecuta.
  - El servidor debe interpretar PHP.
- Acciones que permite PHP (Algunas):
  - Operaciones matemáticas o lógicas complejas.
  - Crear, abrir y modificar archivos en el servidor.
  - Elegir entre varias secuencias de código HTML.
  - Imprimir datos en el documento HTML, etc.
- Otras opciones a PHP o CGI: JSP (JavaServer Pages), ASP (versión Microsoft).

WWW (World Wide Web)
HTML (HyperText Markup Language)

- Basado en marcas o etiquetas.
- Dos partes:
  - Encabezado: Metadatos (información general).
  - Cuerpo: Contenido de la página.

```
<html>
  <head></head>
  <body>
    <p>texto en un párrafo</p>
    <img src="url a la imagen">
    <form ACTION="url programa" method=POST>
      <input type=text name="cliente">
      <input type=submit value="nombre">
    </form>
  </body>
</html>
```

Ejecución en el cliente: Ejemplo programa JavaScript

```
<html>
 <head>
  <script language="JavaScript" type="text/javascript">
   function respuesta() {
    var mensaje=document.getElementById("nombre_id").value;
    alert("usted escribió: " + mensaje);
   }
  </script>
 </head>
 <body>
  <img src="http://www.uncuyo.edu.ar/images/logo_color.png">
  <form name="mi_form" id="form_id" onsubmit="return respuesta();">
   <input type="text" name="nombre" value="" id="nombre_id">
   <input type="submit" value="Enviar">
  </form>
 </body>
</html>
```

Función Javascript

Llamada a la función

Formularios

- Permiten recolectar información ingresada por el usuario.
- La información puede enviarse al servidor.
- Permiten incluir botones, cuadros de texto, botones de opciones, realizar dibujos, etc.

Función Javascript que será llamada al presionar “enviar”

Método HTTP a utilizar

```
<form name="mi_formulario" method="POST" id="formulario_id"
onsubmit="return validar_datos();" action="verificar.php">
  <label for="nombre">Ingrese su nombre de usuario: </label>
  <input type="text" name="usuario" id="usuario_id"><br><br>
  <input type="submit" value="Enviar">
</form>
```

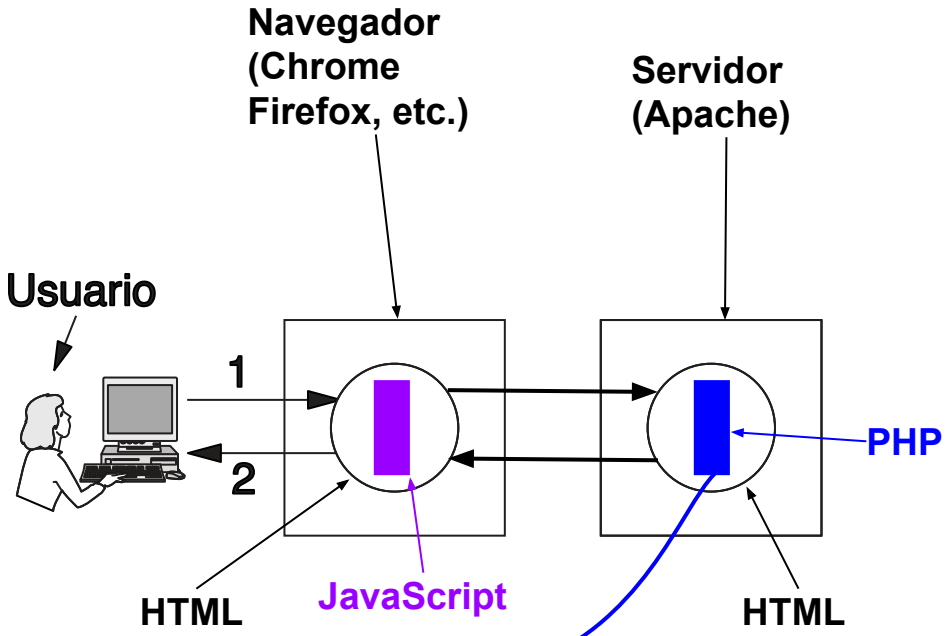
Código PHP que procesará los datos en el servidor

Ejemplo de código PHP

Variable

Recuperación de datos de un formulario

```
<body>
    <?php
        $clave_correcta="abcde";
        $usuario = $_POST["usuario"];
        $clave = $_POST["clave"];
        if($clave==$clave_correcta){
            ……..
        }
        else{
            ……..
        }
    ?>
```

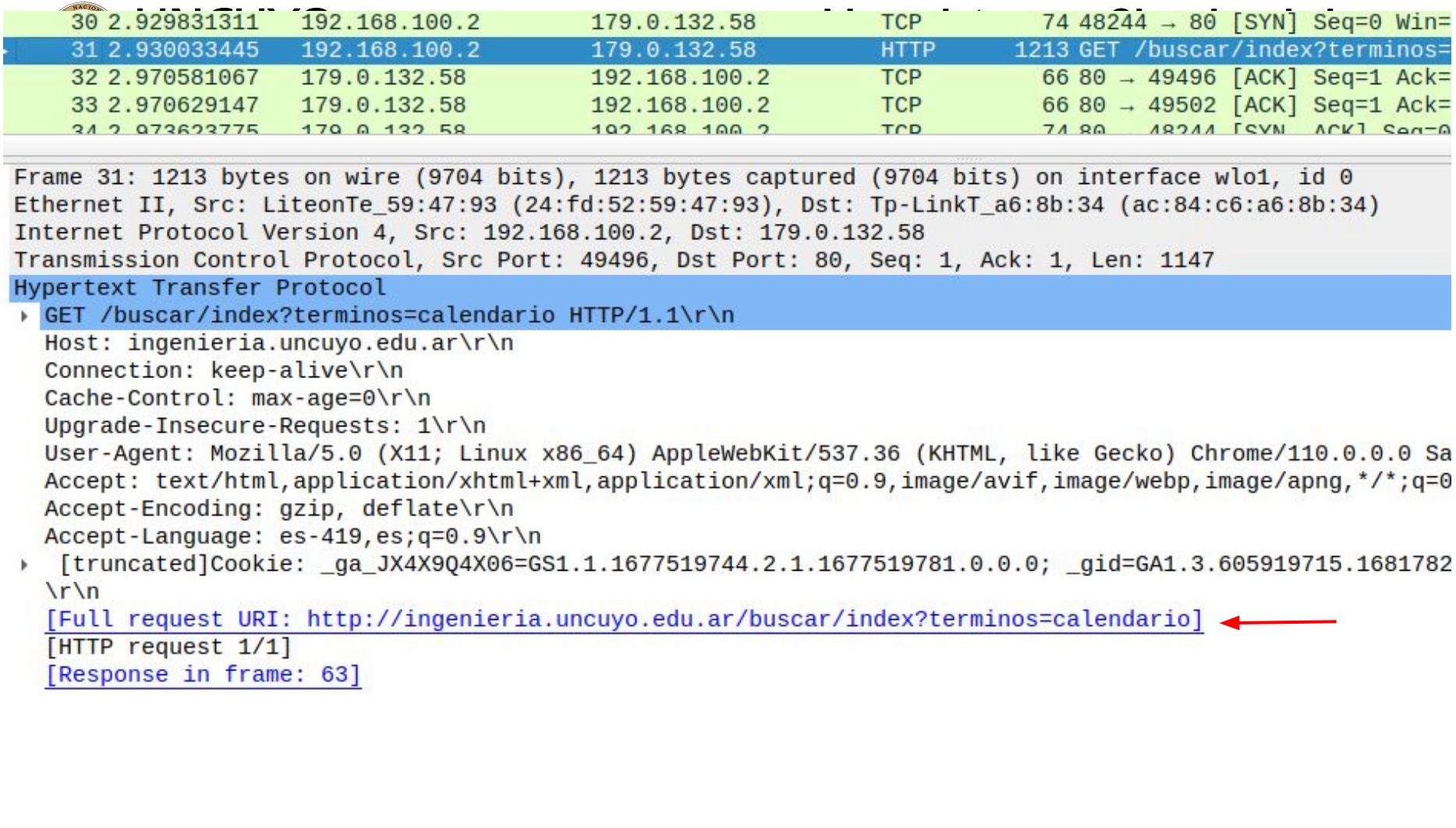


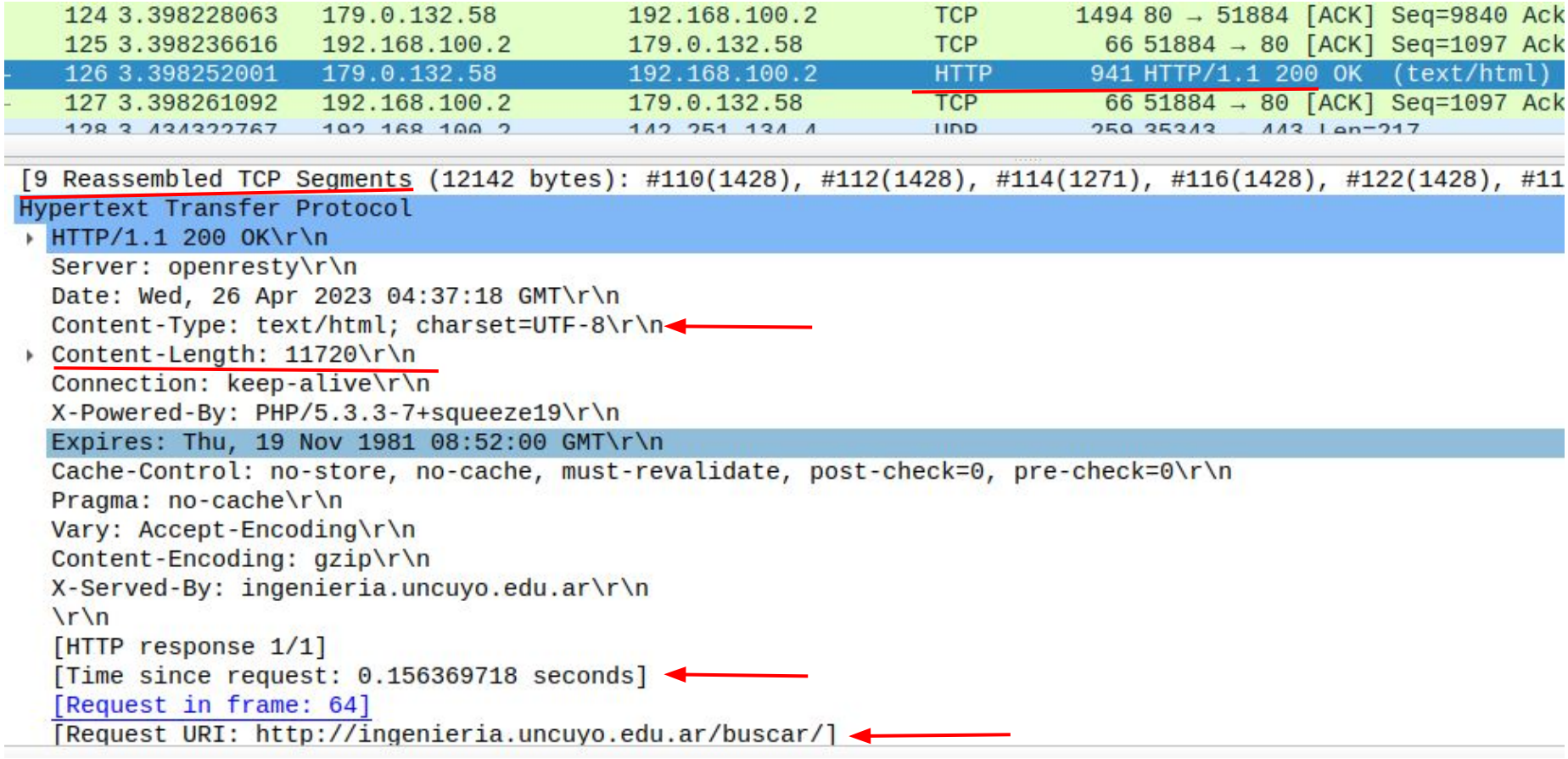
HTTP

- Controla el intercambio de información entre clientes y servidores.
- Paquetes: Consisten en una o más líneas de texto ASCII.
- GET: Solicita un recurso (por ejemplo, una página web, una imagen, etc.). Puede incluir datos de entrada a ser procesados por el servidor en la URL.
  - GET nombredearchivo HTTP/1.1
  - https://www.google.com/search?q=palabra
  - http://ingenieria.uncuyo.edu.ar/buscar/index?terminos=calendario
- POST: Envía datos al servidor indicando el archivo que almacena el código que debe procesar esos datos. Los datos se envían en el cuerpo de la solicitud (usualmente los almacena, realiza algún cálculo, etc.).
- HEAD: Solicita solo el encabezado de la respuesta (para conocer de antemano los datos a recibir y características de los mismos).

```
    62 3.241487341   179.0.132.58     192.168.100.2    TCP      74 80 → 51884 [SYN, ACK] Seq=0
    63 3.241534403   192.168.100.2    179.0.132.58     TCP      66 51884 → 80 [ACK] Seq=1 Ack=1
    64 3.241882283   192.168.100.2    179.0.132.58     HTTP   1162 GET /buscar/ HTTP/1.1
    65 3.243821686   179.0.132.58     192.168.100.2    TCP      74 443 → 59552 [SYN, ACK] Seq=0
    66 3.243854072   192.168.100.2    179.0.132.58     TCP      66 59552 → 443 [ACK] Seq=1 Ack=
```

```
▸ Frame 64: 1162 bytes on wire (9296 bits), 1162 bytes captured (9296 bits) on interface wlo1, id 0
▸ Ethernet II, Src: LiteonTe_59:47:93 (24:fd:52:59:47:93), Dst: Tp-LinkT_a6:8b:34 (ac:84:c6:a6:8b:34)
▸ Internet Protocol Version 4, Src: 192.168.100.2, Dst: 179.0.132.58
▸ Transmission Control Protocol, Src Port: 51884, Dst Port: 80, Seq: 1, Ack: 1, Len: 1096
▾ Hypertext Transfer Protocol
  ▸ GET /buscar/ HTTP/1.1\r\n
    Host: ingenieria.uncuyo.edu.ar\r\n
    Connection: keep-alive\r\n
    Upgrade-Insecure-Requests: 1\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.0.0 Saf
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.
    Accept-Encoding: gzip, deflate\r\n
    Accept-Language: es-419,es;q=0.9\r\n
  ▸ [truncated]Cookie: _ga_JX4X9Q4X06=GS1.1.1677519744.2.1.1677519781.0.0.0; _gid=GA1.3.605919715.16817823
    \r\n
    [Full request URI: http://ingenieria.uncuyo.edu.ar/buscar/]
    [HTTP request 1/1]
    [Response in frame: 126]
```


30 2.929831311 192.168.100.2 179.0.132.58 TCP 74 48244 → 80 [SYN] Seq=0 Win=
31 2.930033445 192.168.100.2 179.0.132.58 HTTP 1213 GET /buscar/index?terminos=
32 2.970581067 179.0.132.58 192.168.100.2 TCP 66 80 → 49496 [ACK] Seq=1 Ack=
33 2.970629147 179.0.132.58 192.168.100.2 TCP 66 80 → 49502 [ACK] Seq=1 Ack=
Frame 31: 1213 bytes on wire (9704 bits), 1213 bytes captured (9704 bits) on interface wlo1, id 0
Ethernet II, Src: LiteonTe_59:47:93 (24:fd:52:59:47:93), Dst: Tp-LinkT_a6:8b:34 (ac:84:c6:a6:8b:34)
Internet Protocol Version 4, Src: 192.168.100.2, Dst: 179.0.132.58
Transmission Control Protocol, Src Port: 49496, Dst Port: 80, Seq: 1, Ack: 1, Len: 1147
Hypertext Transfer Protocol
GET /buscar/index?terminos=calendario HTTP/1.1\r\n
Host: ingenieria.uncuyo.edu.ar\r\n
Connection: keep-alive\r\n
Cache-Control: max-age=0\r\n
Upgrade-Insecure-Requests: 1\r\n
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.0.0 Sa
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0
Accept-Encoding: gzip, deflate\r\n
Accept-Language: es-419,es;q=0.9\r\n
[truncated]Cookie: _ga_JX4X9Q4X06=GS1.1.1677519744.2.1.1677519781.0.0.0; _gid=GA1.3.605919715.1681782
\r\n
[Full request URI: http://ingenieria.uncuyo.edu.ar/buscar/index?terminos=calendario]
[HTTP request 1/1]
[Response in frame: 63]


124 3.398228063 179.0.132.58 192.168.100.2 TCP 1494 80 → 51884 [ACK] Seq=9840 Ack
125 3.398236616 192.168.100.2 179.0.132.58 TCP 66 51884 → 80 [ACK] Seq=1097 Ack
126 3.398252001 179.0.132.58 192.168.100.2 HTTP 941 HTTP/1.1 200 OK (text/html)
127 3.398261092 192.168.100.2 179.0.132.58 TCP 66 51884 → 80 [ACK] Seq=1097 Ack
[9 Reassembled TCP Segments (12142 bytes): #110(1428), #112(1428), #114(1271), #116(1428), #122(1428), #11
Hypertext Transfer Protocol
HTTP/1.1 200 OK\r\n
Server: openresty\r\n
Date: Wed, 26 Apr 2023 04:37:18 GMT\r\n
Content-Type: text/html; charset=UTF-8\r\n
Content-Length: 11720\r\n
Connection: keep-alive\r\n
X-Powered-By: PHP/5.3.3-7+squeeze19\r\n
Expires: Thu, 19 Nov 1981 08:52:00 GMT\r\n
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0\r\n
Pragma: no-cache\r\n
Vary: Accept-Encoding\r\n
Content-Encoding: gzip\r\n
X-Served-By: ingenieria.uncuyo.edu.ar\r\n
\r\n
[HTTP response 1/1]
[Time since request: 0.156369718 seconds]
[Request in frame: 64]
[Request URI: http://ingenieria.uncuyo.edu.ar/buscar/]

Necesidad de gran cantidad de memoria RAM que actuará como caché para minimizar accesos a discos.

WWW (World Wide Web)
Servidores multihilos

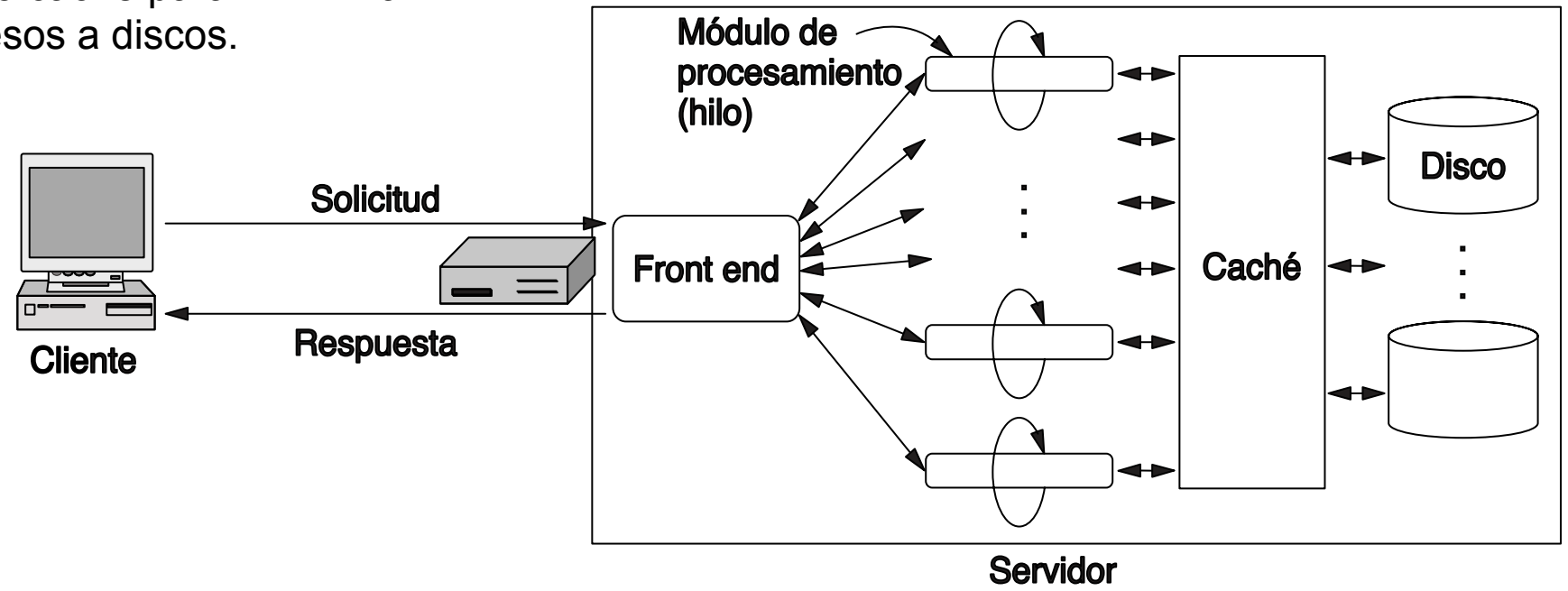


Figura obtenida de: A. Tanenbaum, D. Wetherall, "Redes de computadoras", Quinta edición (2012), pag. 564

Hojas de estilo en cascada (CSS: Cascading Style Sheets)

- Inicialmente HTML tenía el objetivo de mostrar información, sin permitir controlar la apariencia.
- Cuando se hizo importante la apariencia, se agregaron más etiquetas.
  - Ejemplos <b> (negrita), <font face="helvética" size="24" color="red"> (tipo de letra), <i> (cursiva o itálica), etc.
  - El código se hizo **tedioso** de escribir o leer y **no portable**[1].
- Hojas de estilo: Asocian "**texto** - **estilo**". Los estilos se definen por separado y son opcionales (el navegador puede utilizarlas o no).
  - Los estilos pueden agregarse dentro del mismo documento html (etiqueta **<style>**) o en archivos diferentes de tipo **css**.

[1] Una página diseñada para un navegador puede hacerse no apta para otro navegador o pantalla.

Hojas de estilo en cascada (CSS: Cascading Style Sheets)

- Ejemplos de definiciones css:
  body {background-color:linen; color:navy; font-family:Arial;}
  h1 {font-size:200%}
  p {font-size:150%}

- Ejemplo de inclusión:
  <head>
  <title>WIDGETS AMALGAMADOS, S.A. </title>
  <link rel=”stylesheet” type=”text/css” href= “estilo.css” />
  </head>

Evolución de HTML

Elemento	HTML 1.0	HTML 2.0	HTML 3.0	HTML 4.0	HTML 5.0
Hipervínculos	x	x	x	x	x
Imágenes	x	x	x	x	x
Listas	x	x	x	x	x
Mapas e imágenes activas		x	x	x	x
Formularios		x	x	x	x
Ecuaciones			x	x	x
Barras de herramientas			x	x	x
Tablas			x	x	x
Características de accesibilidad				x	x
Incrustación de objetos				x	x
Hojas de estilo				x	x
Secuencias de comandos				x	x
Video y audio					x
Gráficos vectoriales en línea					x
Representación de XML					x
Hilos en segundo plano					x
Almacenamiento del navegador					x
Lienzo de dibujo					x

XML y JSON

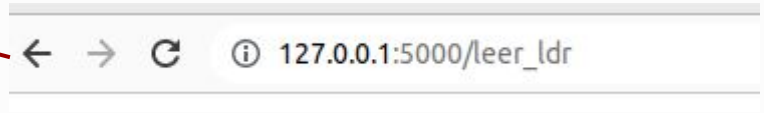
- Especifican **contenido** estructurado (HTML especifica formato+contenido).
- No hay etiquetas definidas. Cada usuario puede definir sus propias etiquetas.
- Interpretados de manera estricta.
- **XML** (eXtensible Markup Language)
  - Utiliza etiquetas (similar a HTML).
  - Pensado como estándar para comunicación entre aplicaciones (dentro y fuera de WWW).
- **JSON** (JavaScript Object Notation)
  - Utiliza notación similar a Javascript ({};"" etc.)
  - Creado para intercambio de datos JavaScript, pero aceptado por muchos lenguajes.
- **AJAX** (Asynchronous JavaScript And XML): XML+JavaScript (o JSON + JavaScript). Permite intercambiar información en segundo plano entre el servidor y el cliente en formato XML y mostrarlo dinámicamente mediante JavaScript (sin actualizar la página completa).

Frameworks de diseño web

- Node js:
  - Entorno de desarrollo y ejecución que permite escribir aplicaciones completas con **JavaScript**.
  - Requiere máquina virtual V8 (MV creada por Google, utilizada por Chrome) en el servidor.
- Flask, Django:
  - Framework que permite escribir aplicaciones web en **Python**.

```python
@app.route('/leer_ldr', methods=['GET'])
def leer_ldr():
    @after_this_request
    def add_header(response):
        response.headers['Access-Control-Allow-Origin'] = '*'
        return response
    global lectura_ldr
    jsonResp = []
    jsonResp.append(lectura_ldr)
    print(lectura_ldr)
    return jsonify(jsonResp)

@app.route('/encender_apagar', methods=['POST'])
def encender_apagar():
    enviar_serial("onoff\n");
    url_for('static', filename='estilo.css')
    global lectura_ldr
    return render_template('pagina.html',lectura_ldr=lectura_ldr)
```



Ejemplo código en Flask

Retorna un JSON

Se invoca desde un formulario

Retorna una página web

Caché Web

- Objetivo: Almacenar información de URL visitados (páginas web, imágenes, hojas de estilo, etc.) de manera que posibles futuros accesos sean más veloces.
  - Concepto similar a memoria caché del procesador.
- La información puede almacenarse en:
  - La computadora del usuario (caché del navegador).
  - Computadoras intermedias (caché proxy).
    - Usado por ISPs o empresas.
- Problema de **actualización de los datos** (¿El dato en la caché fue o no actualizado en el servidor?). Dos estrategias:
  - El servidor provee como dato la **fecha y hora de expiración**. Una segunda petición revisa si la URL existe en la caché y si sigue siendo válida. Si lo es, la recupera de la caché.
    - Problema: No todas las páginas incluyen datos sobre expiración.

Caché Web

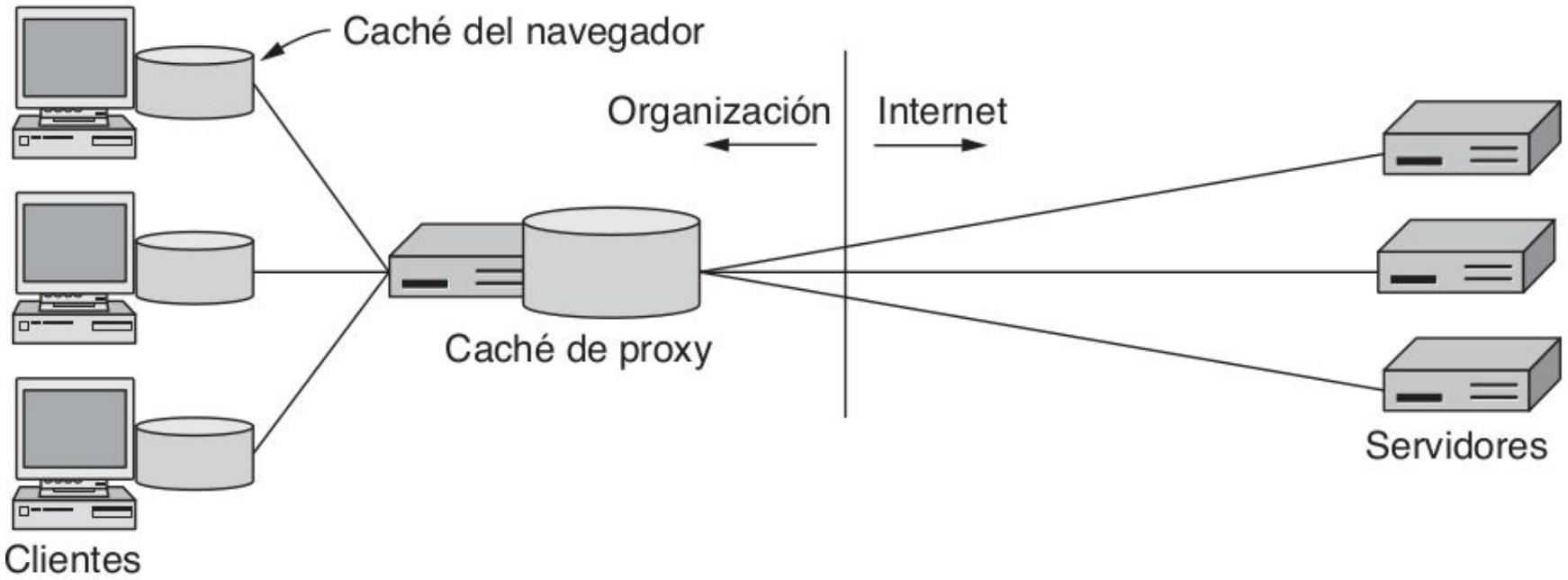


Figura obtenida de: A. Tanenbaum, D. Wetherall, "Redes de computadoras", Quinta edición (2012), pag. 638

Caché Web

- **GET condicional**: Dos formas:
  - El cliente envía un GET indicando fecha y hora de última actualización.
  - Con cada envío de un recurso, el servidor envía una **etiqueta de validez** (hash llamados ETag). Luego, en el GET condicional, el cliente envía las etiquetas que posee, con las cuales el servidor puede determinar si el cliente posee una copia actualizada.
    - Si no hubo actualizaciones, el servidor lo indica, y el navegador toma la URL desde la caché.
    - Si hubo actualizaciones, el servidor envía el recurso actualizado archivo.
- El servidor puede enviar directivas adicionales (ejemplo: no-caché).
- En un proxy web, puede haber varios niveles de caché (en una empresa, luego en el ISP).

El cliente **no** posee una copia del recurso en su caché

Primer GET

Recurso + HASH

El cliente **si** posee una copia del recurso en su caché

GET condicional + lista de HASHs

Posibles respuestas

No modificado (Si el cliente tiene un HASH actualizado)

Recurso + HASH (Si el cliente no tiene un HASH actualizado)

Cliente **Servidor**

Caché Web

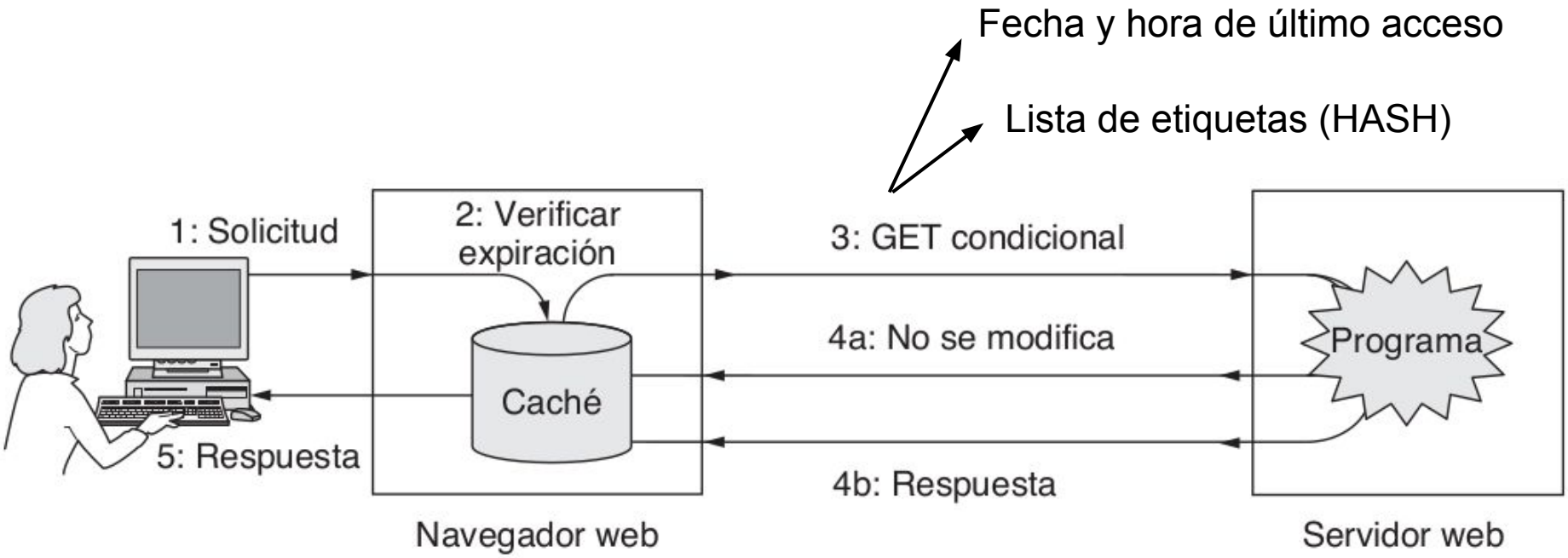
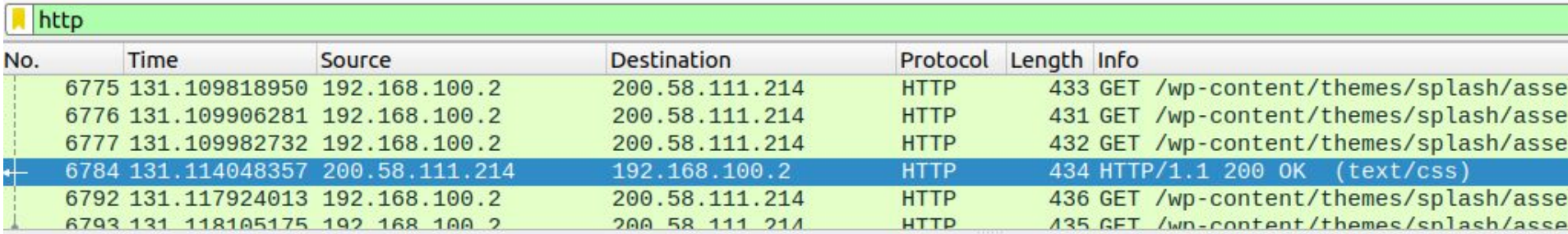


Figura obtenida de: A. Tanenbaum, D. Wetherall, "Redes de computadoras", Quinta edición (2012), pag. 594


http
No. Time Source Destination Protocol Length Info
6775 131.109818950 192.168.100.2 200.58.111.214 HTTP 433 GET /wp-content/themes/splash/asse
6776 131.109906281 192.168.100.2 200.58.111.214 HTTP 431 GET /wp-content/themes/splash/asse
6777 131.109982732 192.168.100.2 200.58.111.214 HTTP 432 GET /wp-content/themes/splash/asse
6784 131.114048357 200.58.111.214 192.168.100.2 HTTP 434 HTTP/1.1 200 OK (text/css)
6792 131.117924013 192.168.100.2 200.58.111.214 HTTP 436 GET /wp-content/themes/splash/asse
Frame 6784: 434 bytes on wire (3472 bits), 434 bytes captured (3472 bits) on interface wlo1, id 0
Ethernet II, Src: Tp-LinkT_a6:8b:34 (ac:84:c6:a6:8b:34), Dst: LiteonTe_59:47:93 (24:fd:52:59:47:93)
Internet Protocol Version 4, Src: 200.58.111.214, Dst: 192.168.100.2
Transmission Control Protocol, Src Port: 80, Dst Port: 46648, Seq: 42814, Ack: 843, Len: 380
[7 Reassembled TCP Segments (9020 bytes): #6778(1440), #6779(1440), #6780(1440), #6781(1440), #6782(1440), #678
Hypertext Transfer Protocol
HTTP/1.1 200 OK\r\n
Date: Sun, 07 May 2023 01:17:45 GMT\r\n
Server: Apache\r\n
Last-Modified: Mon, 03 Apr 2023 14:43:38 GMT\r\n
ETag: "c9d6-5f86f95923a80-gzip"\r\n
Accept-Ranges: bytes\r\n
Cache-Control: max-age=2592000\r\n
Expires: Tue, 06 Jun 2023 01:17:45 GMT\r\n
Vary: Accept-Encoding,User-Agent\r\n
Content-Encoding: gzip\r\n

CDN (Content Delivery Networks)

- El proveedor de un servicio coloca **copias del contenido** (página web) **en nodos en distintas ubicaciones**, e indica a los clientes usar esos nodos.
  - Concepto inverso a los proxy web.
- Estructura árbol.
- ¿Cómo obtiene el cliente la IP del nodo al que debe acceder?
  - Espejos: El servidor réplica su contenido en los nodos, a los que se les llama espejos. La página web principal posee vínculos a los diferentes nodos.
    - Utilizado para distribuir grandes paquetes de software o contenido de gran tamaño.
  - Que el dueño de la CDN administre el servidor DNS. Devuelve la IP del nodo en función de:
    - La IP origen de la solicitud.
    - La carga de cada nodo.

CDN (Content Delivery Networks)

- Existen empresas que proveen CDN como servicio. El cliente provee el contenido, el propietario de la CDN lo distribuye entre sus nodos.
  - Ejemplo: https://www.akamai.com/ (posee más de 200mil servidores en más de 120 países).
- Ventajas adicionales:
  - Menor vulnerabilidad a picos de tráfico elevado.
  - No hay un servidor o red cuello de botella.

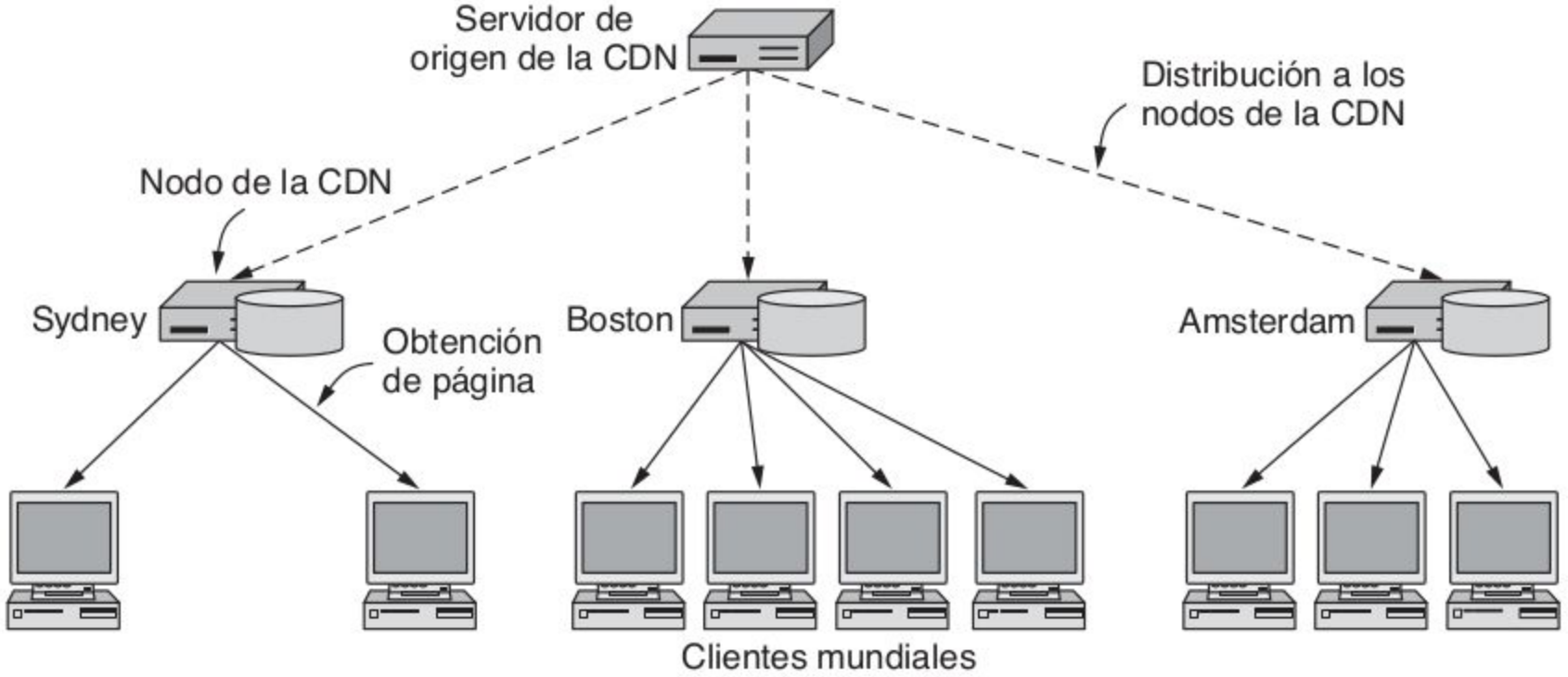


Figura obtenida de: A. Tanenbaum, D. Wetherall, "Redes de computadoras", Quinta edición (2012), pag. 639

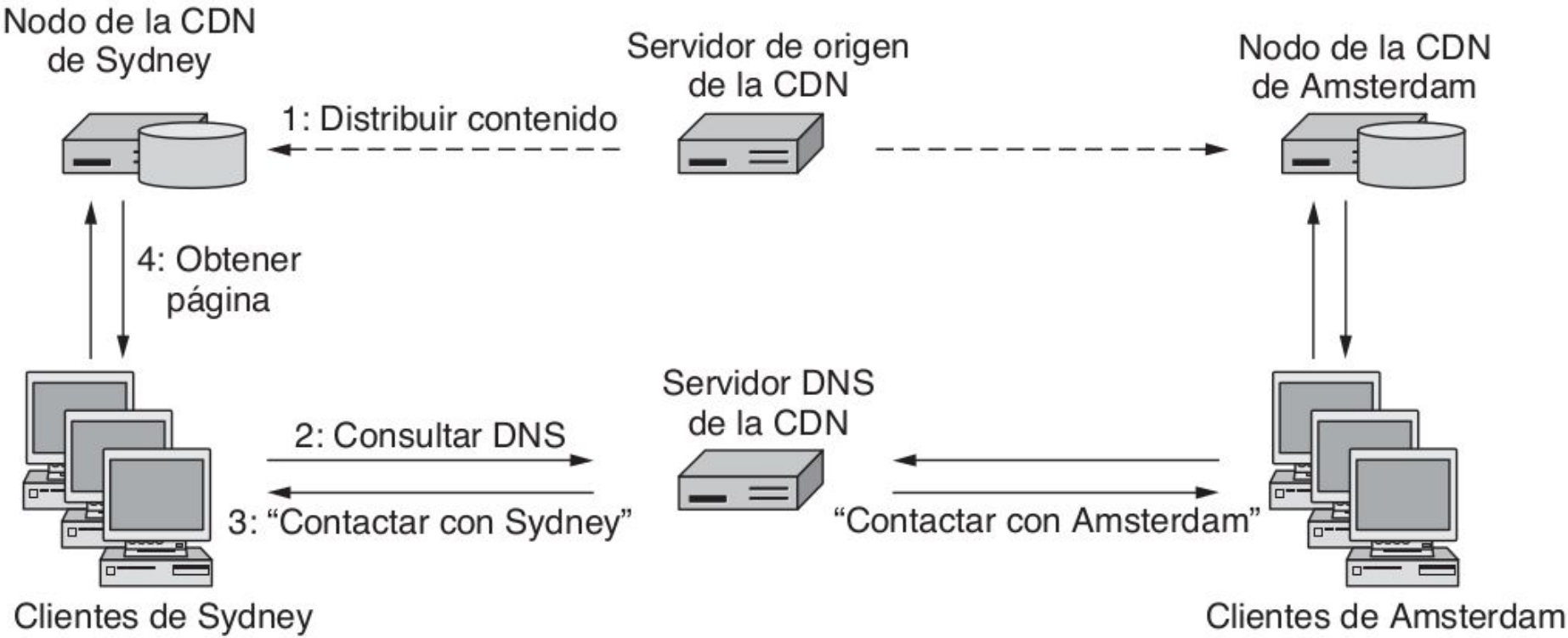


Figura obtenida de: A. Tanenbaum, D. Wetherall, "Redes de computadoras", Quinta edición (2012), pag. 641

Redes P2P
(Peer-to-Peer o Igual a Igual)

- Objetivo: Distribución de contenido.
- Primer antecedente: Napster (1999).
- Formadas por nodos (llamados pares o iguales) que actúan como cliente (para obtener contenido) y como servidores (para proveer contenido).
  - Todos los nodos actúan de igual manera para distribuir el contenido.
  - No hay una estructura centralizada.

Protocolo BitTorrent

- Los archivos a compartir se dividen en trozos de igual tamaño. El nombre de cada trozo se identifica con un hash (SHA-1 de 160 bits del trozo).
- Trackers (rastreadores): Para cada contenido, contiene una lista de nodos con trozos del contenido (hashes).
  - A medida que los clientes descargan trozos del contenido, informan al tracker que poseen el trozo, y pueden actuar como servidores para otros clientes.

Redes P2P

- Cada proveedor de contenido crea un archivo de descripción de contenido llamado “torrent” que contiene:
  - El nombre de un rastreador.
  - La lista de trozos del contenido (lista de hashes).
  - Se contactan con el rastreador periódicamente para enviarle esta información.
  - Sembradores: nodos que contienen todos los trozos de un contenido.
- Pasos para la descarga de un archivo:
  1. Obtener el torrent (de una página web o buscador del cliente).
  2. Localizar el rastreador y la lista de nodos con trozos del contenido.
  3. Descargar trozos y convertirse en proveedor de los trozos descargados.
  4. Cada nodo descarga diferentes trozos para que no haya nodos con pocas copias en la red.
- La velocidad de descarga será función de los trozos provistos a otros clientes.

Redes P2P

Torrent
- Rastreador
- Listra de trozos
- Sembradores

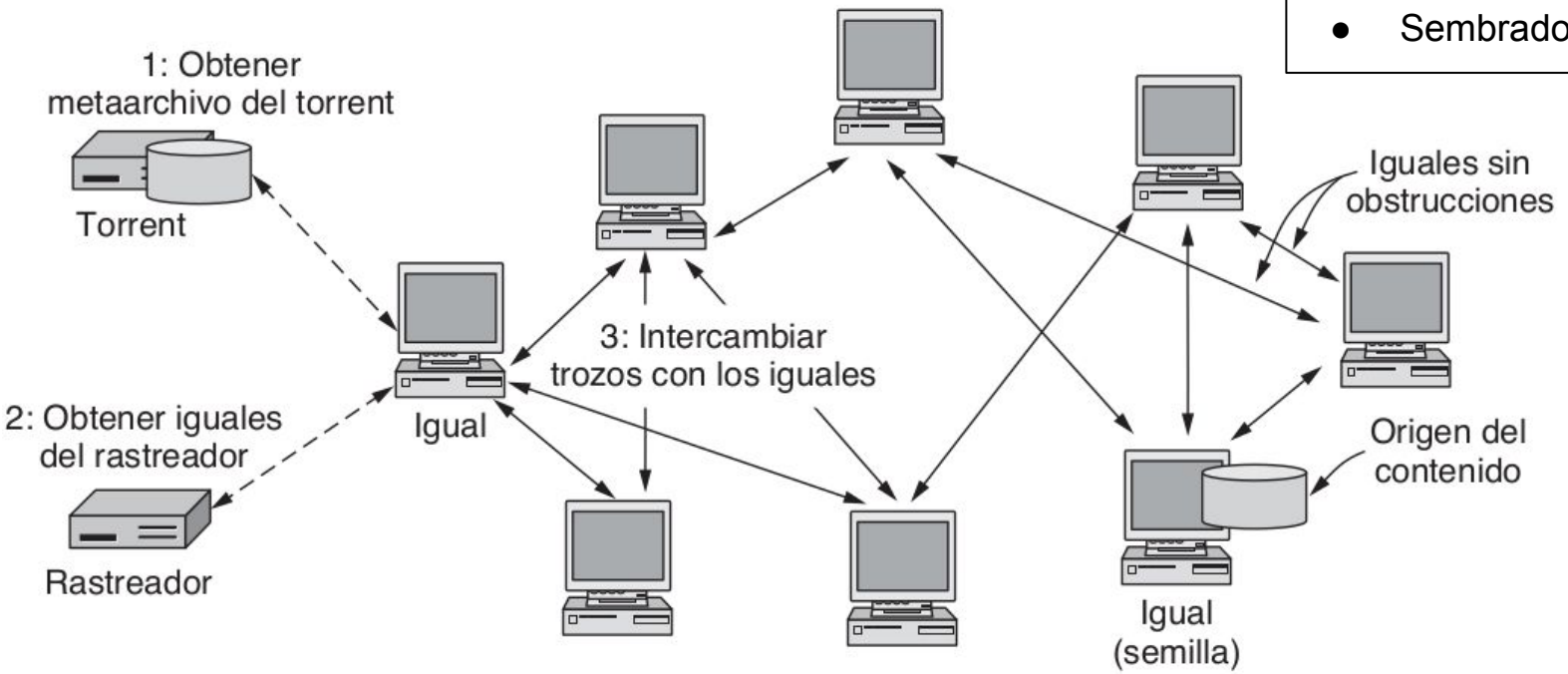


Figura obtenida de: A. Tanenbaum, D. Wetherall, "Redes de computadoras", Quinta edición (2012), pag. 646

UNCUYO UNIVERSIDAD NACIONAL DE CUYO

FACULTAD DE INGENIERÍA

Licenciatura en Ciencias de la Computación

Intercambio dinámico de información entre cliente y servidores: Ajax (Asynchronous JavaScript And XML)

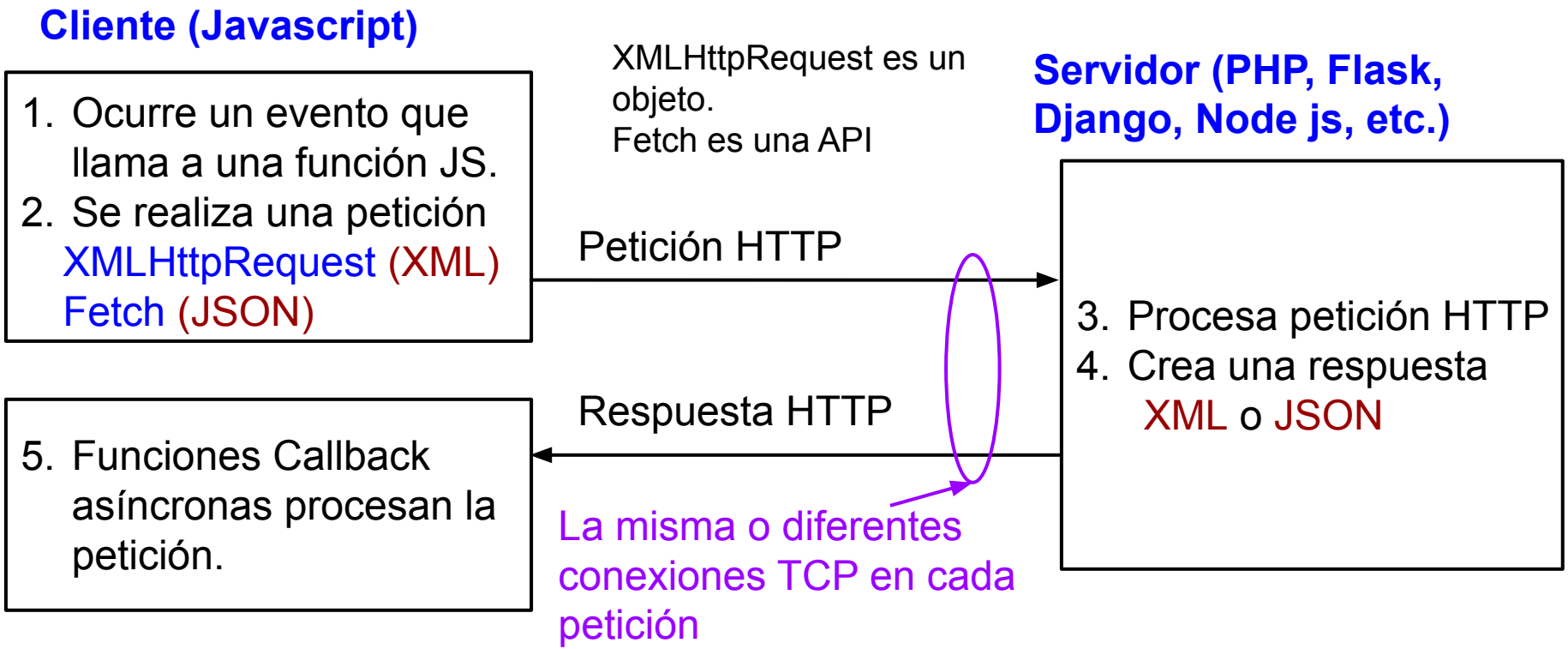


Figura basada en: https://www.w3schools.com/php/php_ajax_intro.asp

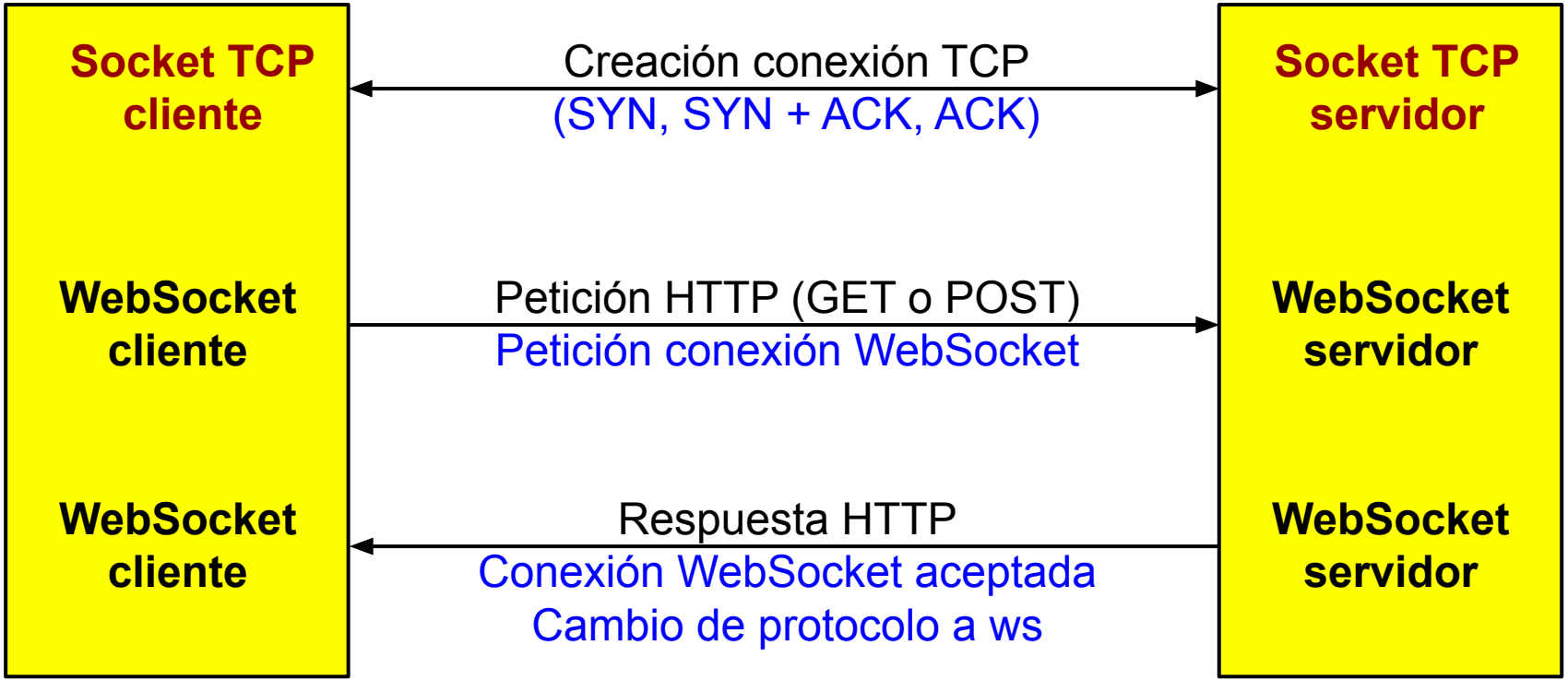
Source	Destination	Protocol	Length	Source port	Dest Port	Information
192.168.100.2	192.168.100.7	TCP	66	5000	44122	5000 → 44122 [ACK] Seq=206 Ack=508 Win=64768
192.168.100.7	192.168.100.2	TCP	74	44124	5000	44124 → 5000 [SYN] Seq=0 Win=65535 Len=0 MSS=
192.168.100.2	192.168.100.7	TCP	74	5000	44124	5000 → 44124 [SYN, ACK] Seq=0 Ack=1 Win=65160
192.168.100.7	192.168.100.2	TCP	66	44124	5000	44124 → 5000 [ACK] Seq=1 Ack=1 Win=87808 Len=
192.168.100.7	192.168.100.2	HTTP	572	44124	5000	GET /leer_ldr HTTP/1.1
192.168.100.2	192.168.100.7	TCP	66	5000	44124	5000 → 44124 [ACK] Seq=1 Ack=507 Win=64768 Le
192.168.100.2	192.168.100.7	TCP	262	5000	44124	5000 → 44124 [PSH, ACK] Seq=1 Ack=507 Win=647
192.168.100.2	192.168.100.7	HTTP/J…	74	5000	44124	HTTP/1.1 200 OK , JavaScript Object Notation
192.168.100.7	192.168.100.2	TCP	66	44124	5000	44124 → 5000 [ACK] Seq=507 Ack=197 Win=87808
192.168.100.7	192.168.100.2	TCP	66	44124	5000	44124 → 5000 [FIN, ACK] Seq=507 Ack=206 Win=8
192.168.100.2	192.168.100.7	TCP	66	5000	44124	5000 → 44124 [ACK] Seq=206 Ack=508 Win=64768
192.168.100.7	192.168.100.2	TCP	74	44126	5000	44126 → 5000 [SYN] Seq=0 Win=65535 Len=0 MSS=
192.168.100.2	192.168.100.7	TCP	74	5000	44126	5000 → 44126 [SYN, ACK] Seq=0 Ack=1 Win=65160
192.168.100.7	192.168.100.2	TCP	66	44126	5000	44126 → 5000 [ACK] Seq=1 Ack=1 Win=87808 Len=
192.168.100.7	192.168.100.2	HTTP	572	44126	5000	GET /leer_ldr HTTP/1.1
192.168.100.2	192.168.100.7	TCP	66	5000	44126	5000 → 44126 [ACK] Seq=1 Ack=507 Win=64768 Le
192.168.100.2	192.168.100.7	TCP	262	5000	44126	5000 → 44126 [PSH, ACK] Seq=1 Ack=507 Win=647
192.168.100.2	192.168.100.7	HTTP/J…	73	5000	44126	HTTP/1.1 200 OK , JavaScript Object Notation
192.168.100.7	192.168.100.2	TCP	66	44126	5000	44126 → 5000 [ACK] Seq=507 Ack=197 Win=87808

s on wire (592 bits), 74 bytes captured (592 bits) on interface wlo1, id 0

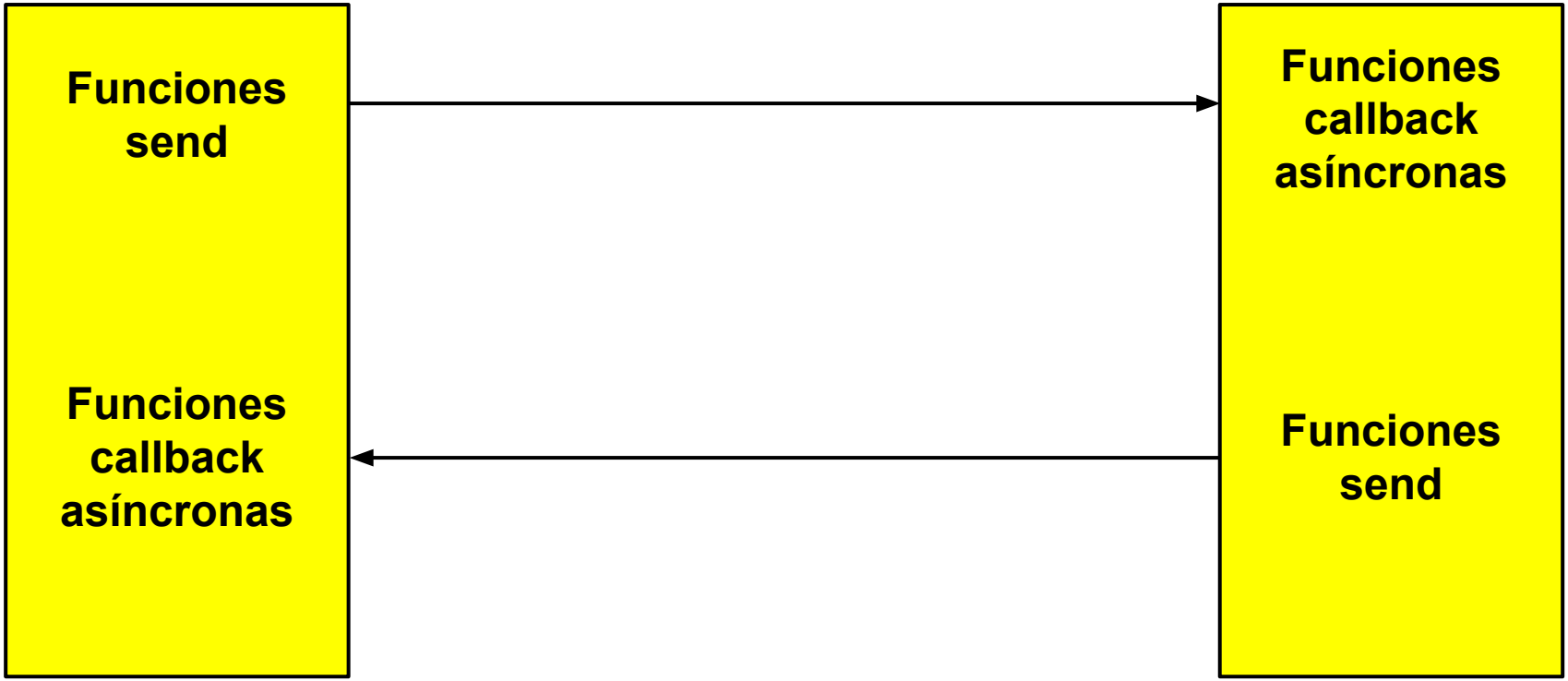
Intercambio dinámico de información entre cliente y servidores: Websocket

- Protocolo que permite intercambio de información bidireccional entre aplicaciones a nivel de capa de aplicación.
  - Similar a un socket TCP pero a nivel de capa de aplicación.
- Trabaja sobre sockets TCP estables (se mantienen durante largo tiempo).
- Utiliza HTTP para establecer la conexión. Luego cambia a protocolo WebSocket (ws).
- Implementado en muchos lenguajes y frameworks:
  - Nativo en PHP y Javascript.
  - Librerías para Python, C++, Java, etc.
    - Integrarse a proyectos Flask o Django.
  - Frameworks Nodejs, Django etc.

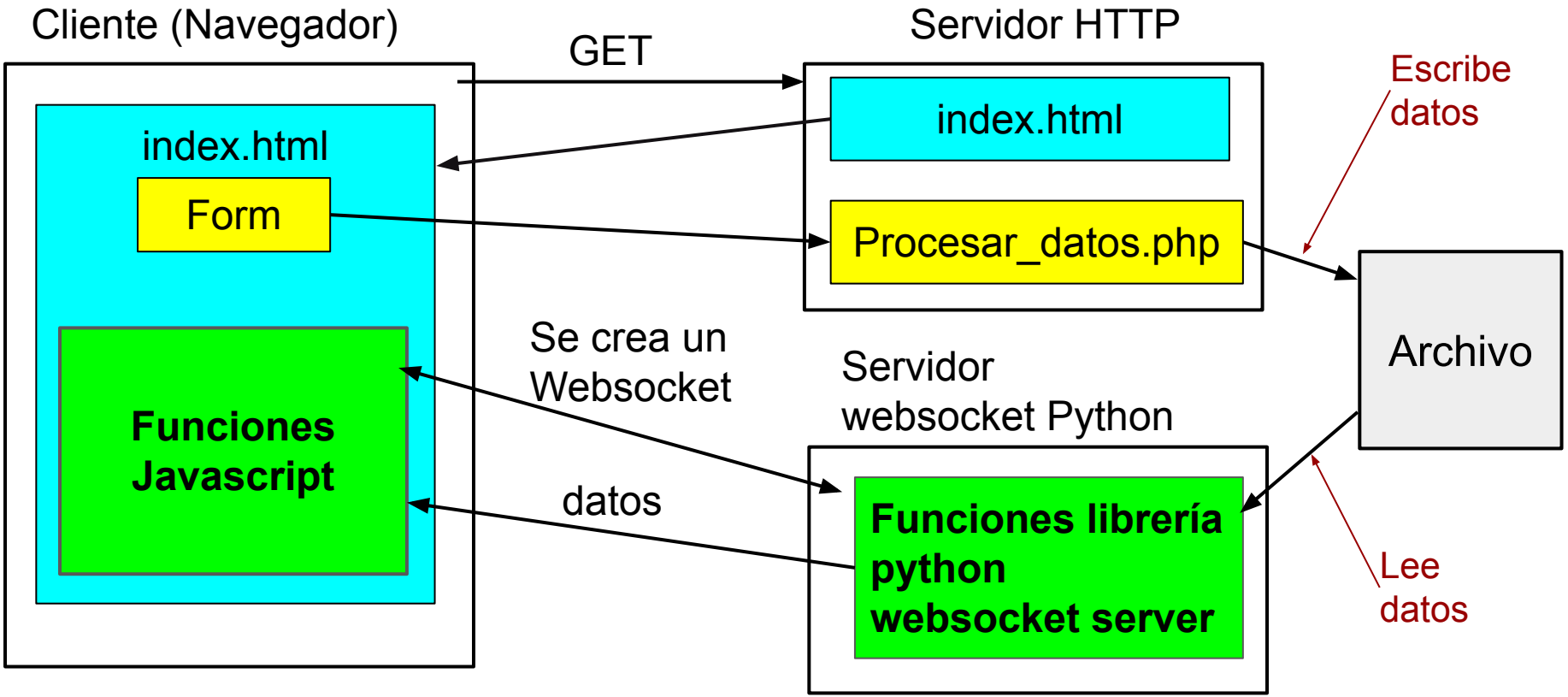
Intercambio dinámico de información entre cliente y servidores: Websocket



Intercambio dinámico de información entre cliente y servidores: Websocket



Websocket: Ejemplo con Javascritp y Python



Ejemplo Websocket servidor con Python (módulo websocket-server)
https://github.com/Pithikos/python-websocket-server

```python
import logging
import threading
import time
from websocket_server import WebsocketServer

def new_client(client, server):
    print("Se conectó: " + str(client["address"]))

def cliente_se_desconecta(client, server):
    print("Se desconectó el cliente: " + str(client["address"]))

def enviar(websocket_server):
    while 1:
        print("Cantidad clientes: " + str(len(websocket_server.clients)))
        time.sleep(1)
        archivo_votos=open("votos.txt","r")
        cadena=archivo_votos.readline().rstrip()
        #....Se arma una cadena de caracteres a enviar al cliente.....
        archivo_votos.close()
        for cliente in websocket_server.clients:
            websocket_server.send_message(cliente,cadena)
            print("Enviando datos a " + str(cliente["address"]))

websocket_server = WebsocketServer(host='192.168.100.2', port=60000, loglevel=logging.INFO)
websocket_server.set_fn_new_client(new_client)
websocket_server.set_fn_client_left(cliente_se_desconecta)
hilo_enviar = threading.Thread(target=enviar, args=(websocket_server,))
hilo_enviar.start()
websocket_server.run_forever()
```

Crea y configura un objeto websocket

Configurando funciones callback asíncronas

Inicializando el websocket

Ejemplo Websocket cliente con Javascript

```
<script language="JavaScript">
    function abrir_websocket()
    {
        var url_websocket = "ws://192.168.100.2:60000/servidor_TCP.py";
        websocket = new WebSocket(url_websocket);
        websocket.onmessage = function(ev)
        {
            var response = ev.data;
            response_list=response.split(",");
            document.getElementById("boca_juniors_votos_id").innerHTML= /***********/
            /*Se modifica la página web en función de los datos recibidos*/
        };
    }

    function cerrar_socket()
    {
        websocket.close();
    }
</script>
</head>
<body onload="return abrir_websocket();">
```

Crea y configura un objeto websocket

Define una función asíncrona (funciones asociadas a eventos open, onmessage, close y error)

Cierra el websocket

o.	Time	Source	Destination	Protocol	Length	Source port	Dest Port	Information
90	18.322020052	192.168.100.7	192.168.100.2	TCP	74	35236	80	35236 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=
91	18.322089808	192.168.100.2	192.168.100.7	TCP	74	80	35236	80 → 35236 [SYN, ACK] Seq=0 Ack=1 Win=65160
94	18.327244398	192.168.100.7	192.168.100.2	TCP	66	35236	80	35236 → 80 [ACK] Seq=1 Ack=1 Win=87808 Len=
96	18.329653222	192.168.100.7	192.168.100.2	HTTP	693	35236	80	GET /index.php HTTP/1.1
97	18.329741374	192.168.100.2	192.168.100.7	TCP	66	80	35236	80 → 35236 [ACK] Seq=1 Ack=628 Win=64640 Le
98	18.330691638	192.168.100.2	192.168.100.7	HTTP	1135	80	35236	HTTP/1.1 200 OK (text/html)
99	18.332248396	192.168.100.7	192.168.100.2	TCP	66	35236	80	35236 → 80 [ACK] Seq=628 Ack=1070 Win=89856
100	18.461408673	192.168.100.7	192.168.100.2	TCP	74	54488	60000	54488 → 60000 [SYN] Seq=0 Win=65535 Len=0 M
101	18.461465239	192.168.100.2	192.168.100.7	TCP	74	60000	54488	60000 → 54488 [SYN, ACK] Seq=0 Ack=1 Win=65
102	18.463275742	192.168.100.7	192.168.100.2	TCP	66	54488	60000	54488 → 60000 [ACK] Seq=1 Ack=1 Win=87808 L
103	18.464037935	192.168.100.7	192.168.100.2	HTTP	762	54488	60000	GET /servidor_TCP.py HTTP/1.1
104	18.464091304	192.168.100.2	192.168.100.7	TCP	66	60000	54488	60000 → 54488 [ACK] Seq=1 Ack=697 Win=64512
105	18.464652591	192.168.100.2	192.168.100.7	HTTP	195	60000	54488	HTTP/1.1 101 Switching Protocols
106	18.465852801	192.168.100.7	192.168.100.2	TCP	66	54488	60000	54488 → 60000 [ACK] Seq=697 Ack=130 Win=878
107	18.904846141	192.168.100.2	192.168.100.7	WebSo…	82	60000	54488	WebSocket Text [FIN]
108	18.928629393	192.168.100.7	192.168.100.2	TCP	66	54488	60000	54488 → 60000 [ACK] Seq=697 Ack=146 Win=878
112	19.906416841	192.168.100.2	192.168.100.7	WebSo…	82	60000	54488	WebSocket Text [FIN]
113	20.116545280	192.168.100.2	192.168.100.7	TCP	82	60000	54488	[TCP Retransmission] 60000 → 54488 [PSH, AC
114	20.328523313	192.168.100.2	192.168.100.7	TCP	82	60000	54488	[TCP Retransmission] 60000 → 54488 [PSH, AC

```
Transmission Control Protocol, Src Port: 54488, Dst Port: 60000, Seq: 1, Ack: 1, Len: 696
Hypertext Transfer Protocol
  GET /servidor_TCP.py HTTP/1.1\r\n
  Host: 192.168.100.2:60000\r\n
  Connection: Upgrade\r\n
  Pragma: no-cache\r\n
  Cache-Control: no-cache\r\n
  User-Agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.0.0 Mobile Safari/537.3
  Upgrade: websocket\r\n
  Origin: http://192.168.100.2\r\n
```

RPC (Remote Procedure Call)

- Un proceso **llamado cliente** puede llamar procedimientos que **están en otra máquina como si fueran locales**.
- El proceso que contienen los procedimientos remotos se llama **servidor**.
- La llamada puede incluir el paso de datos como **argumentos** y el procedimiento remoto puede **devolver resultados** (como una llamada a un procedimiento local).
- La llamada, los parámetros y los resultados deben **encapsularse en paquetes** que deben enviarse al servidor. Este proceso debe ser transparente.
- Emplea mayormente **UDP**, ya que la llamada, argumento y respuestas implican pocos datos, **pero pueden perderse paquetes y fallar la llamada**.
- También puede emplear **TCP**, en casos donde la llamada no puede fallar:
  - No todas las funciones pueden llamarse dos veces (cada llamada puede modificar dato, como un contador).

RPC (Remote Procedure Call)

- Necesidad de un procedimiento **stub de cliente** que representa o simula al procedimiento que está el servidor:
  - Permite llamar al procedimiento remoto como si fuera local.
  - Empaqueta la llamada y los datos pasados como parámetros en un paquete que envía a la máquina remota.
  - Implementa los mecanismos para transmitir el paquete.
- Necesidad de un procedimiento **stub de servidor**.
  - Realiza la llamada al procedimiento como si fuera local (aunque la llamada fue en otra máquina).
  - Empaqueta los resultados de la ejecución del procedimiento y los envía a al proceso cliente.
- **Desventajas:**
  - **No se pueden utilizar punteros (apuntadores).**

RPC

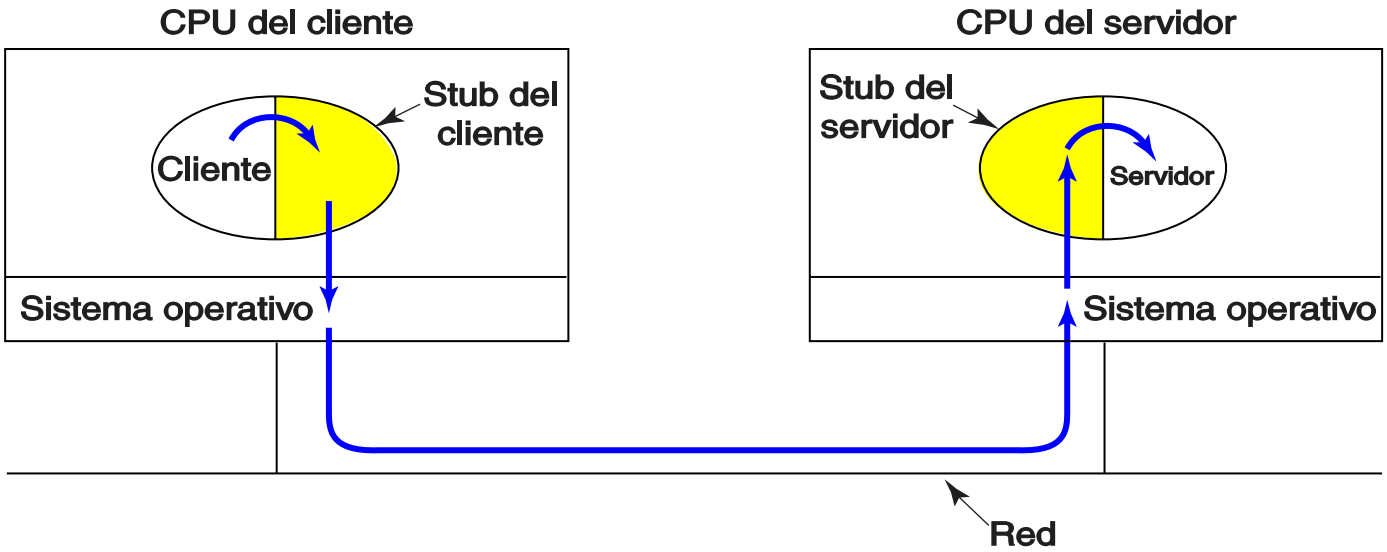
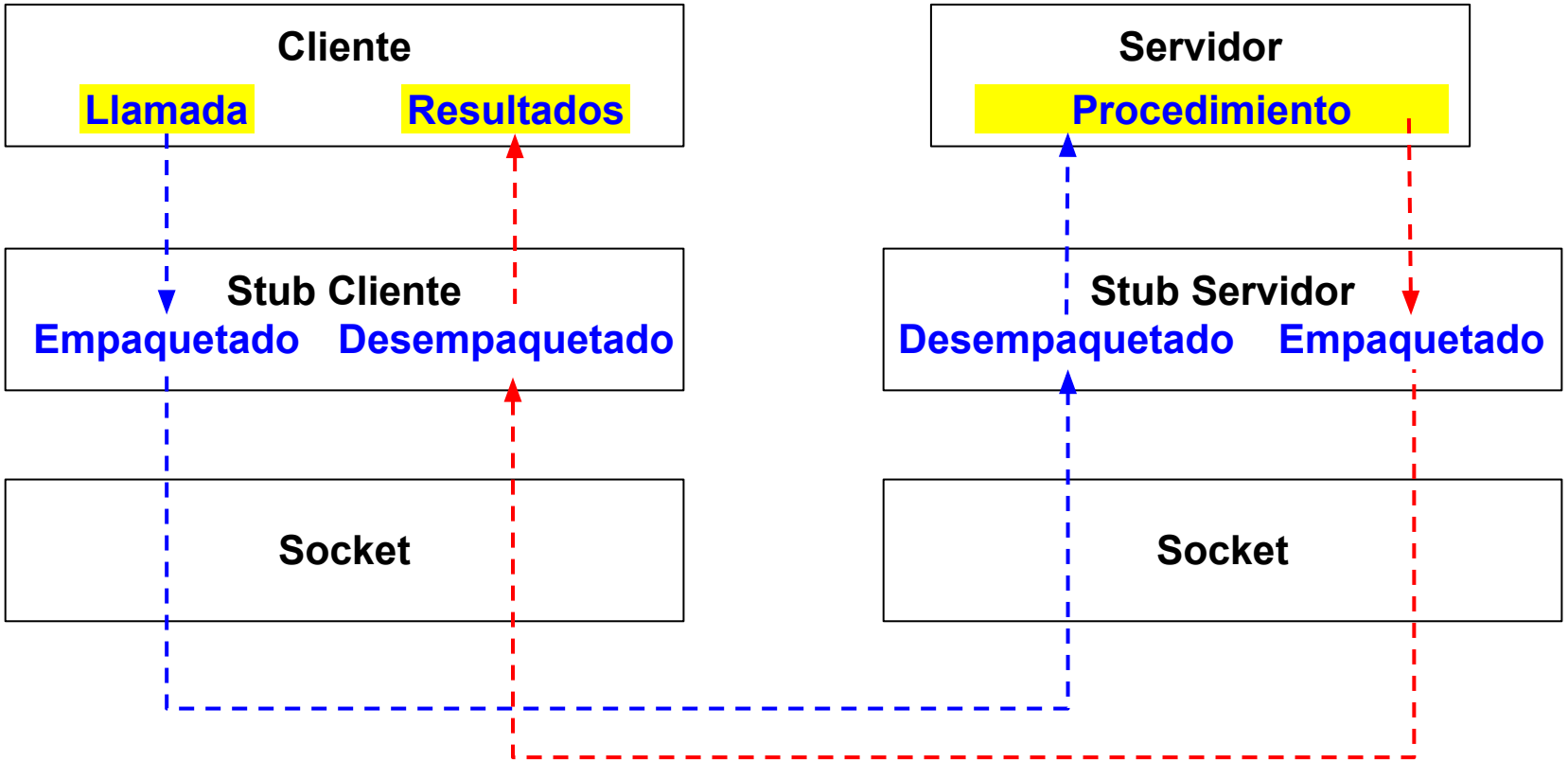


Figura obtenida de: A. Tanenbaum, D. Wetherall, "Redes de computadoras", Quinta edición (2012), pag. 467


Cliente
Llamada
Resultados
Servidor
Procedimiento
Stub Cliente
Empaquetado
Desempaquetado
Stub Servidor
Desempaquetado
Empaquetado
Socket
Socket

RPC Ejemplo Cliente usando Google Cloud Platform

```
registro_actividad.recuperar_registro(usuario, new AsyncCallback<String>(){
        @Override
        public void onFailure(Throwable caught) {
                actividad_TextArea.setText("ERROR recuperando datos");
        }
        @Override
        public void onSuccess(String result) {
                actividad_TextArea.setText("\nActividad realizada por el usuario " + usuario + ":\n" +
                result);
        }
}
```

RPC Ejemplo Servidor

(el procedimiento recuperar_registro se encuentra dentro de la clase registro_actividad)

```
public String recuperar_registro(String usuario) {
        String answer = "";
        ……...
        final Query query = new Query(Registro_actividad1.Registro_actividad1_ENTITY);
        query.addSort(Registro_actividad1.DATE, SortDirection.ASCENDING);
        query.addFilter(Registro_actividad1.USER, Query.FilterOperator.EQUAL, usuario);
        ……...
        for (Entity entity : datastoreService.prepare(query).asIterable()) {
                ……...
        for (Registro_actividad1 registro : registros_de_actividad_List) {
                        answer = answer + "\n" + registro.get_date() + " " + registro.get_user() ……...
        }
        return result;
}
```