

Protocolo HTTP

(capítulo 1)

Laboratorio III - clase 1

Lic. Mauro Gullino maurogullino@gmail.com

UTN FRH

Intro a HTTP

- HyperText Transfer Protocol
- inventado por Tim Berners-Lee (CERN Francia)
en la década del 90
- para transportar hipertexto = páginas web
- es el fundamento de toda la web

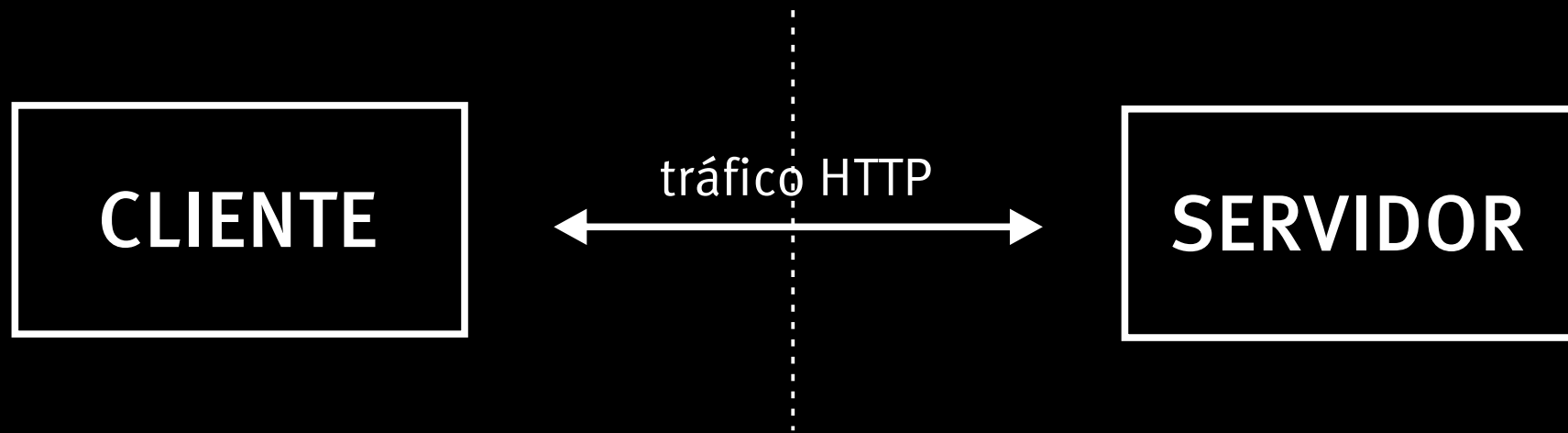
¿Web o internet?

- internet: grupo interconectado de routers
- web: tráfico HTTP en el puerto 80 (páginas) a través de la red internet

¿Qué cosas de internet no son *web*?

Skype, FTP, voz sobre IP, DNS, torrent...

HTTP es cliente-servidor



- **clientes HTTP:** navegadores, browsers, user agent
- **servidores HTTP:** llamados simplemente así

Tráfico web

- está compuesto por mensajes / paquetes
- hay de dos tipos:

PETICIONES / REQUESTS

RESPUESTAS / RESPONSES

Estructura de los paquetes

- peticiones y respuestas tienen mismo “esqueleto”

qué quiero / qué pasó

1 línea

cabeceras = headers
info sobre este paquete

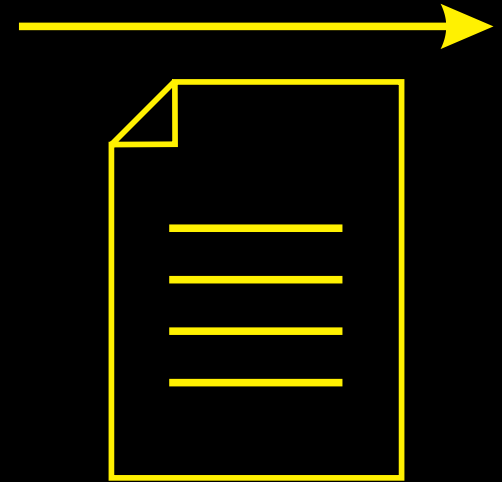
varias líneas

cuerpo del mensaje / body

entre 0 y muchísimas
líneas

Petición real

- sitio web de informática



```
GET /modems.html HTTP/1.1
```

```
Host: www.todocompu.net
```

```
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; es-ES; rv:1.9...
```

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*...
```

```
Accept-Language: es-es,es;q=0.8,en-us;q=0.5,en;q=0.3
```

```
Accept-Encoding: gzip,deflate
```

```
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
```

```
Cache-Control: max-age=0
```

Respuesta real

HTTP/1.1 200 OK

Date: Fri, 01 Aug 2016 17:11:55 GMT

Server: Apache

X-Powered-By: PHP/5.6.9

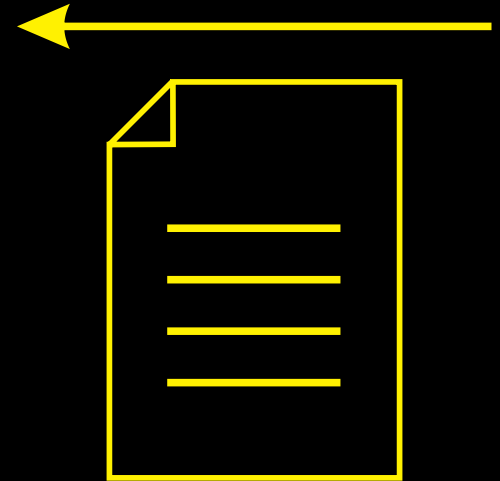
Keep-Alive: timeout=1, max=64

Connection: Keep-Alive

Transfer-Encoding: chunked

Content-Type: text/html

(en el cuerpo está la página que veo en el browser, bytes...)



Lo importante: peticiones

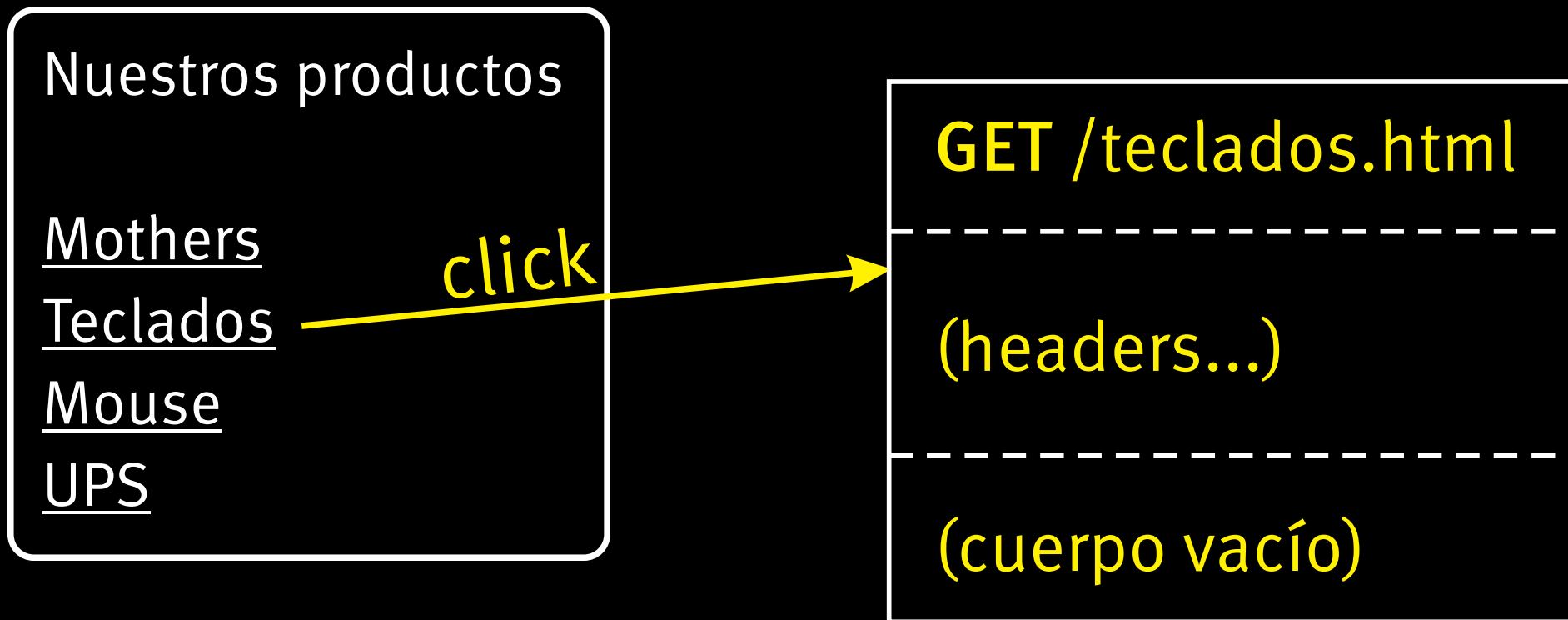
- peticiones:

HTTP tiene dos métodos principales: GET y POST

GET: cliente *pide* un recurso al servidor

POST: cliente *envía* datos para procesar en serv.

Ejemplo método GET



Ejemplo método POST

Contáctenos

Nombre:

Mensaje:

POST /contacto.php

(headers...)

nombre=Roberto&
mensaje=Hola+mundo!

Otra definición

GET = idempotente = "safe"

POST = no idempotente = se solicita confirmación
en los clientes

Otros métodos: PUT, DELETE, HEAD...

Lo importante: respuestas

- respuestas:

hay distintos códigos de estado / status code

2xx OK (200)

3xx redirecciones (301, 302)

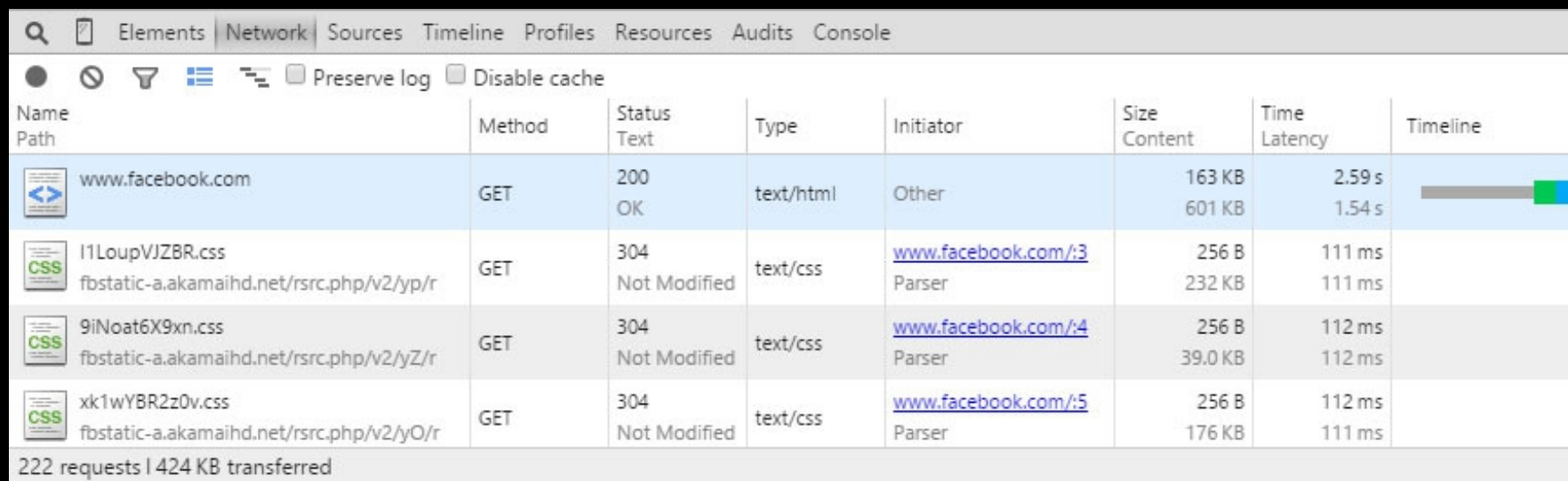
4xx error del cliente (404)






5xx error del servidor (500)

Para verlo...

en Chrome:

Herramientas de desarrollador -> Network (se accede con F12)



Name Path	Method	Status Text	Type	Initiator	Size Content	Time Latency	Timeline
 www.facebook.com	GET	200 OK	text/html	Other	163 KB 601 KB	2.59 s 1.54 s	
 l1LoupVJZBR.css fbstatic-a.akamaihd.net/rsrc.php/v2/yp/r	GET	304 Not Modified	text/css	www.facebook.com/:3 Parser	256 B 232 KB	111 ms 111 ms	
 9iNoat6X9xn.css fbstatic-a.akamaihd.net/rsrc.php/v2/yZ/r	GET	304 Not Modified	text/css	www.facebook.com/:4 Parser	256 B 39.0 KB	112 ms 112 ms	
 xk1wYBR2z0v.css fbstatic-a.akamaihd.net/rsrc.php/v2/yO/r	GET	304 Not Modified	text/css	www.facebook.com/:5 Parser	256 B 176 KB	112 ms 111 ms	

222 requests | 424 KB transferred

HTTP es protocolo de texto

en línea de comandos:

```
telnet www.google.com.ar 80
```

```
GET /
```

Entonces: responsabilidades

clientes HTTP:

- armar y enviar las peticiones
- entender las respuestas
- atender al usuario con la interfaz

servidores HTTP:

- esperar peticiones en el puerto 80
- buscar el recurso solicitado
- armar las respuestas y enviar el recurso

Los más usados

clientes:

Firefox, Internet Explorer, Chrome, Safari, móviles...

servidores:

Apache, Microsoft IIS, lighttpd, nginx...

Deberían todos respetar el protocolo

¿Y cómo son los servidores?

- 2 significados: servidor software / hardware
- son computadoras funcionando 24hs.

Desde	24GB de RAM DDR3	Desde	24GB de RAM DDR3
US\$725.00/mes	Intel Xeon E5645, 2.4Ghz 1x 6 núcleos 2x 146GB 15K SAS 4x Puertos NIC, 1GB Firewall Cisco ASA dedicado	US\$775.00/mes	Intel Xeon L5520 2.26GHz 2x 4 núcleos 2x 300GB 15K SAS 4x Puertos NIC, 1GB Firewall Cisco ASA dedicado
			

fuelle: rackspace.com

Otras opciones de hosting

- hosting compartido
- VPS: servidores privados virtuales



- PaaS: Platform as a Service

*Google App Engine, Heroku, Amazon CloudFront,
Microsoft Azure* Servicios en ¿la nube?

There is no cloud
It's just someone else's computer



Ubicación de recursos

Las URL

Uniform Resource Locator (RFC 1738)

protocolo://máquina:puerto/directorio/archivo

<http://es.wikipedia.org>

<ftp://ftp.servidor.com/imagenes/hola.jpg>

<https://seguro.afip.gob.ar/monotributo>

<http://secreto.com:81/prueba.php>

Los dominios

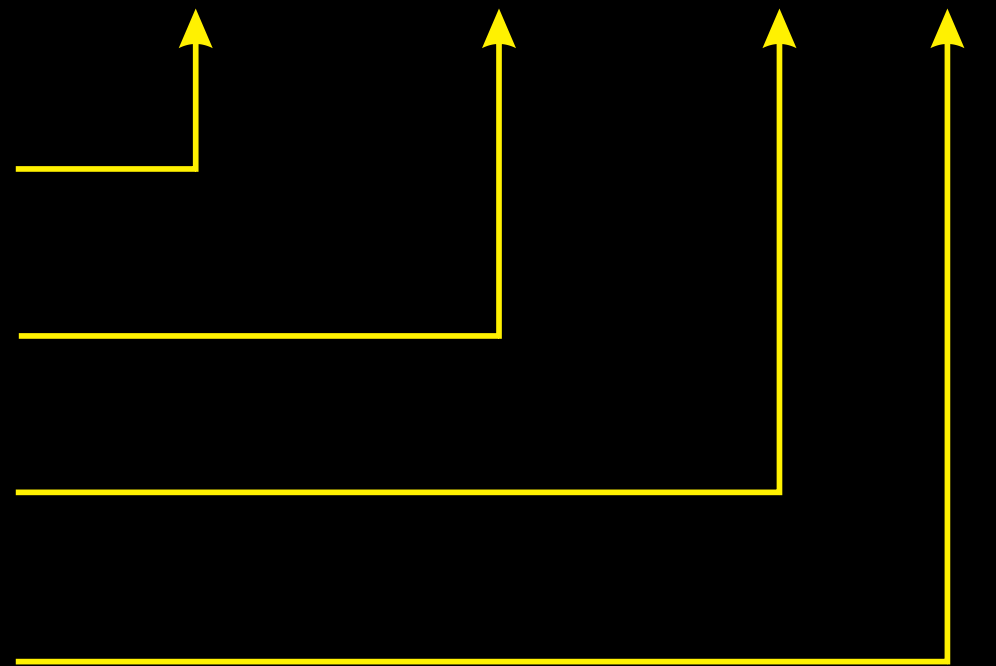
www.google.com.ar

subdominio (opcional)

nombre de dominio

top-level domain (TLD)

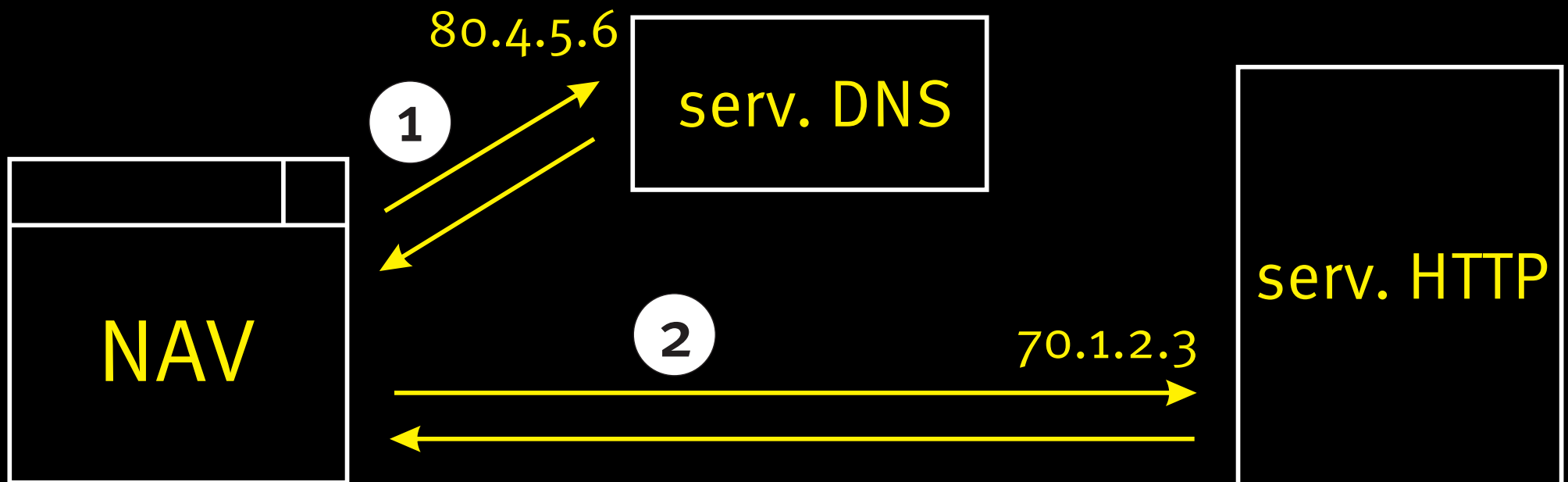
country code TLD (ccTLD)



Internet Corporation for Assigned Names and Numbers (ICANN)

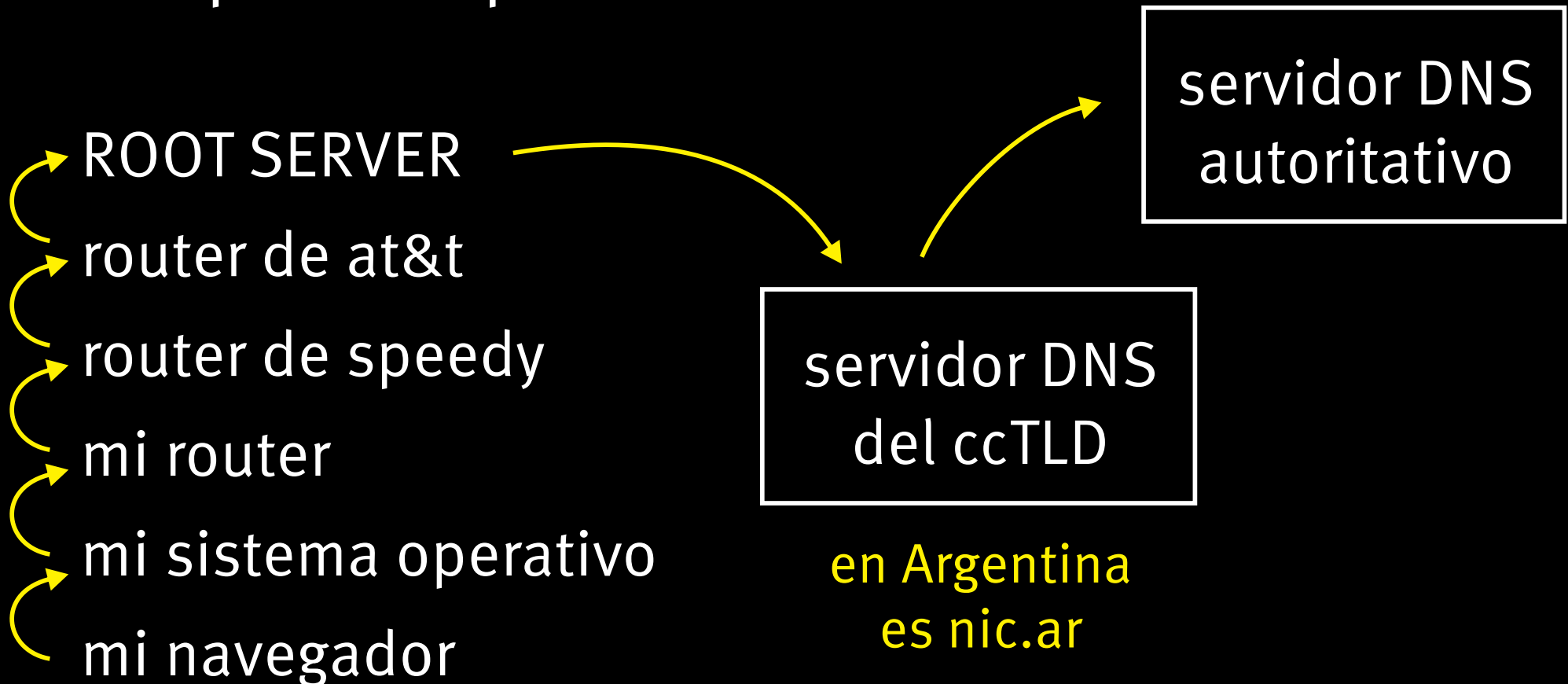
Traducción a IP

- Los servidores se identifican por su IP
- Los DNS traducen dominios en IPs (*resolución*)



Estructura de DNS

- ¿a qué servidor de DNS le pregunto?
- siempre al superior



Ejemplo de resolución

<http://www.google.com.ar/search?q=utn>

es en realidad

<http://74.125.73.103:80/search?q=utn>

para hacer “dns lookup”:

network-tools.com

Otro ejemplo

```
c:\> nslookup www.google.com.ar
```

Servidor: dns1.iplanisp.com

Address: 200.69.193.1

Respuesta no autoritativa:

Nombre: www.l.google.com

Address: 209.85.195.104

Aliases: www.google.com.ar, www.google.com

!= 74.125.73.103



Usar otros DNS

```
nslookup www.google.com.ar dns1.iplanisp.com
```

```
nslookup www.google.com.ar 8.8.8.8
```

Para hacer un sitio web...

- 1- Necesito registrar un **dominio** (nic.ar)
- 2- Alquilar espacio en servidor HTTP (**hosting**)
- 3- Tener un servidor **DNS** autoritativo que conteste en qué IP está mi sitio
- 4- Hacer el sitio y subirlo...

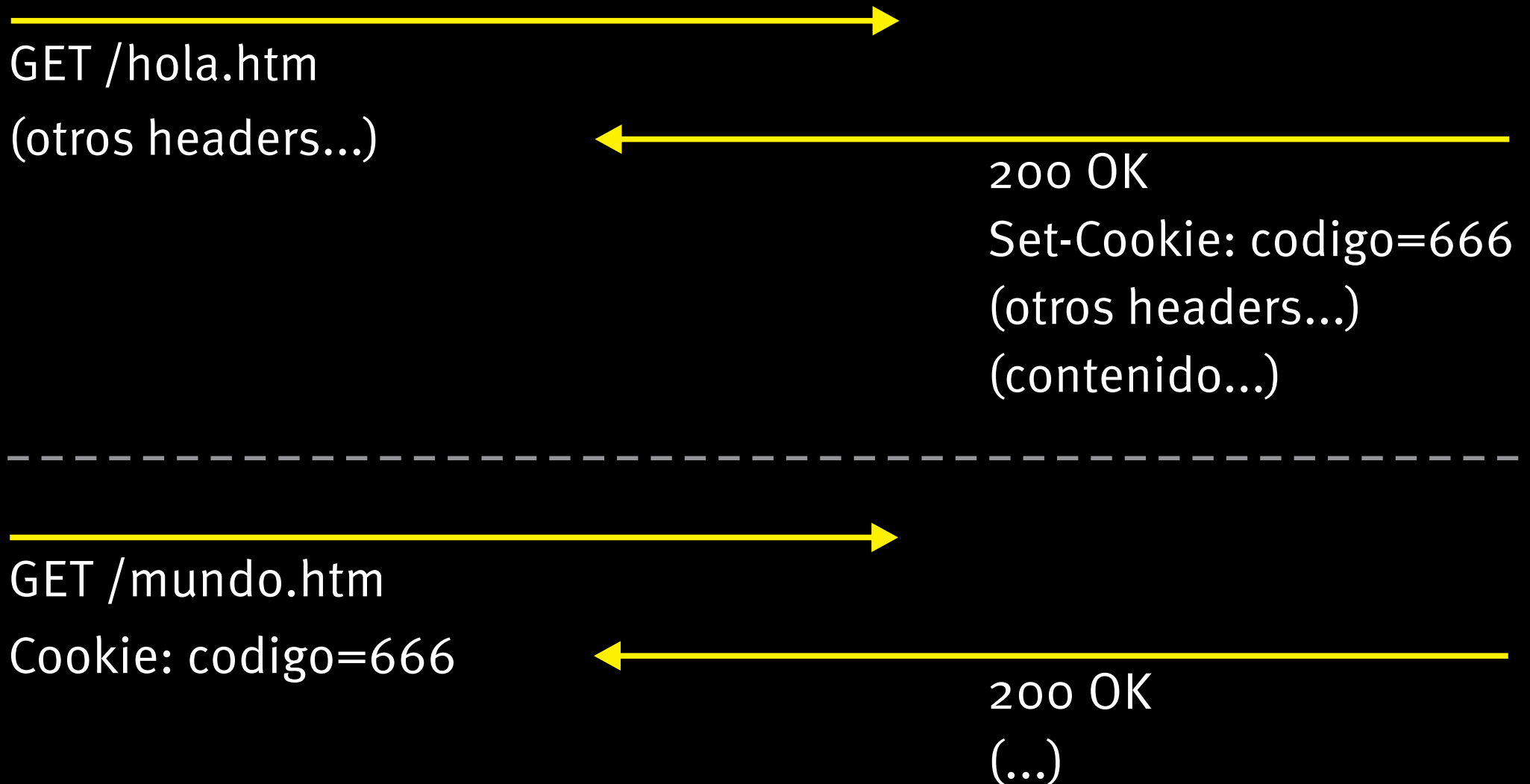
Conservación de estado

¿Qué son las cookies?

¿Qué son las cookies?

- HTTP es un protocolo sin estado (*state-less*)
- las cookies son **variables**, enviadas en las peticiones y devueltas en las respuestas
- con esto el servidor puede identificar pedidos sucesivos del mismo cliente

Proceso de seteo de cookie

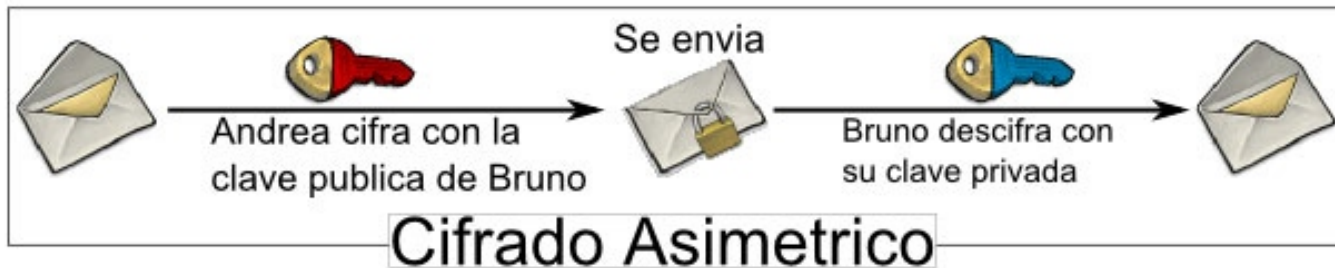


Mitos de las cookies

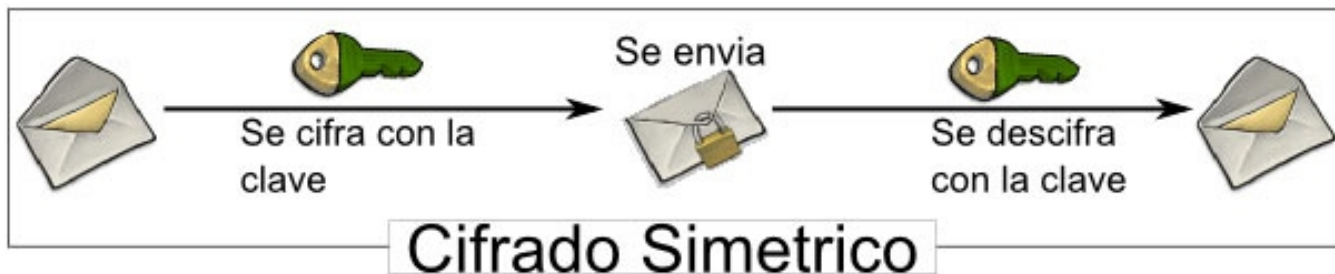
- *Las cookies son como virus por que pueden borrar datos de los discos de los usuarios*
- *Las cookies son un tipo de spyware porque pueden leer información personal almacenada en la PC*
- *Las cookies generan popups*
- *Las cookies se utilizan para generar spam*
- *Las cookies sólo se utilizan con fines publicitarios*

Seguridad del tráfico

Primero...



RSA, ECC

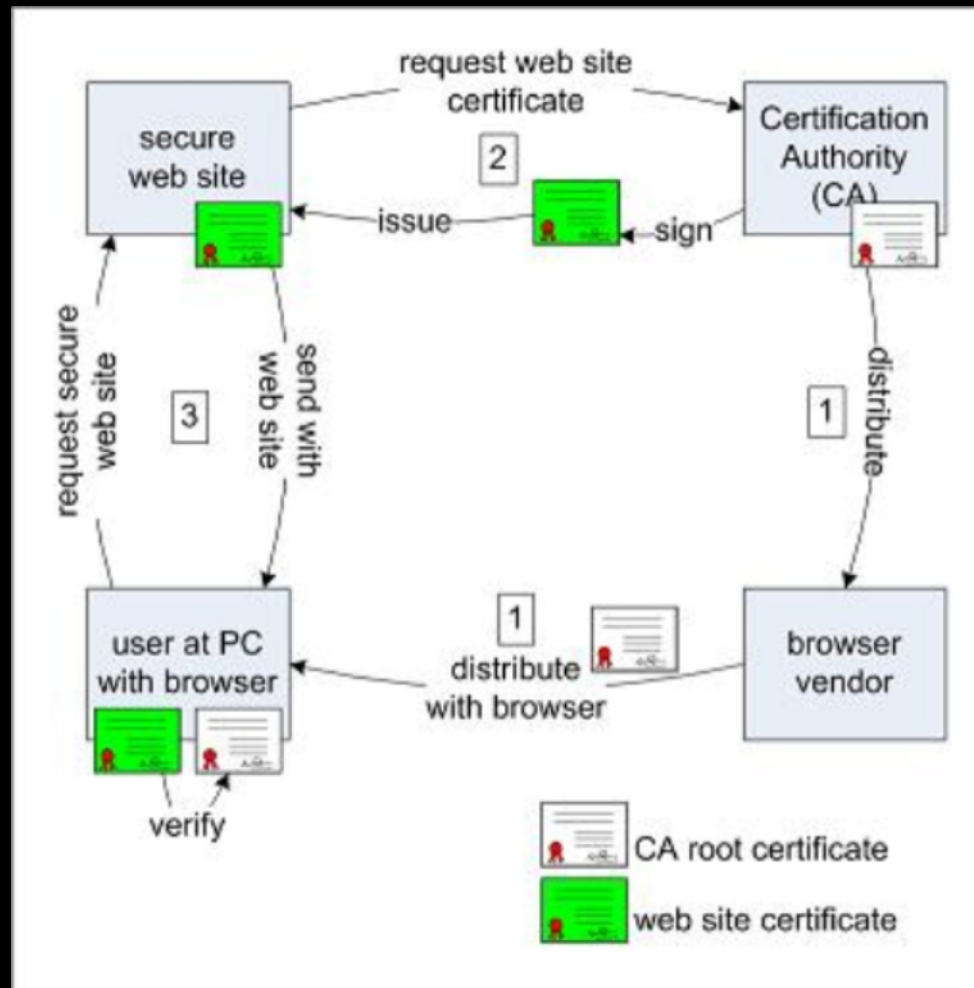


DES, 3DES, AES

https://

- puerto 443
- "túnel" que evita que el tráfico HTTP sea escuchado o modificado (*man-in-the-middle*)
- se basa en certificados que se "negocian" al inicial la conexión

Certificados



Diálogo TLS

CLIENTE

SERVIDOR

"hola"

envía certificado y clave pública

verifica certificado,
inventa una "clave de sesión" y la encripta

todo el tráfico siguiente
se cifra simétricamente con la "clave de sesión"

En el navegador



¡Bienvenidos!

```
void nsHttpResponseHead::ParseVersion(const char *str) {
    // Parse HTTP-Version: "HTTP" "/" 1*DIGIT "." 1*DIGIT

    LOG(("nsHttpResponseHead::ParseVersion [version=%s]\n", str));

    // make sure we have HTTP at the beginning
    if (PL_strncasecmp(str, "HTTP", 4) != 0) {
        LOG(("looks like a HTTP/0.9 response\n"));
        mVersion = NS_HTTP_VERSION_0_9;    return;
    }
    str += 4;

    if (*str != '/') {
        LOG(("server did not send a version number; assuming HTTP/1.0\n"));
        mVersion = NS_HTTP_VERSION_1_0;    return;
    }

    char *p = PL_strchr(str, '.');
    if (p == nullptr) {
        LOG(("mal-formed server version; assuming HTTP/1.0\n"));
        mVersion = NS_HTTP_VERSION_1_0;    return;
    }

    ++p; // let b point to the minor version
    int major = atoi(str + 1);
    int minor = atoi(p);
```

(Firefox source code)