



Introducción a HTTP/2

Teleinformática y Redes 2016

<http://www.labredes.unlu.edu.ar/tyr>

Lic. Marcelo Fidel Fernández

<http://www.marcelofernandez.info>

mail@marcelofernandez.info

[@fidelfernandez](#)

Agenda

- Características de la web antes y ahora
- HTTP y la Web actual, inconvenientes
- Introducción a HTTP/2, características
- Ejemplos
- Estado actual y futuro del protocolo
- Conclusiones Generales



En los orígenes de la Web...

- **1991**: El servicio de WWW nace y **HTTP/0.9** fue “definido”. Sólo permitía un único método: GET.
- **1996**: **HTTP/1.0**. Se estandarizó la base mínima de lo que usamos a diario.
- **1997-1999**: **HTTP/1.1**. Se completó el protocolo. Escalabilidad, proxies, Keep-Alive y Pipelining.

¿Y cómo era la Web en ese entonces?




Universidad Nacional de Luján - Mozilla Firefox

Universidad Nacional d... x

https://web.archive.org/web/19990422222844/http://www.unlu.edu.ar/ Buscar

Universidad Nacional de Luján



Sede Central
Rutas Nac. 5 y 7 (6700) LUJAN
Buenos Aires - República Argentina
Tels.:(02323) 423171 (9 líneas rotativas)
Fax: (02323) 425795

Sede Capital
Ecuador 871 (1214) CAPITAL FEDERAL
Tel.:(011) 4962-7045 / 7026

Centro Regional Chivilcoy
Balcarce 120 (6620) CHIVILCOY
Buenos Aires - República Argentina
Tel. y Fax: (02346) 424593

Centro Regional General Sarmiento
Farias 1590 esquina Mitre (1663) SAN MIGUEL
Buenos Aires - República Argentina
Tel. y Fax: (011) 4664-7843

Centro Regional Campana
Bertolini 183
(2804) CAMPANA
Buenos Aires - República Argentina
Tel.y Fax : (03489) 428342 / 425934/ 438069

Contenidos :

- 1- Inicio
- 2- Institucional
- 3- Autoridades
- 4- Unidades Académicas
- 5- Carreras
- 6- Graduados
- 7- Novedades
- 8- Investigación
- 9- Biblioteca
- 10- Direcciones
- 11- Area de Cultura
- 12- Servidor FTP
- 13- Eventos
- 14- Búsqueda en Nuestro sitio
- 15- Calendario Académico 2000
- 16- Galería de Arte
- 17- Libro de Visitas
- 18- Asuntos Estudiantiles
- 19- E-mails Unlu
- 20- Programa Emprendedor

Radio Uni
88.9 MHz

"En el año del XV Aniver:

Universidad Nacional de Luján
Int. Ruta 5 y 7
6700 Luján
República Argentina.

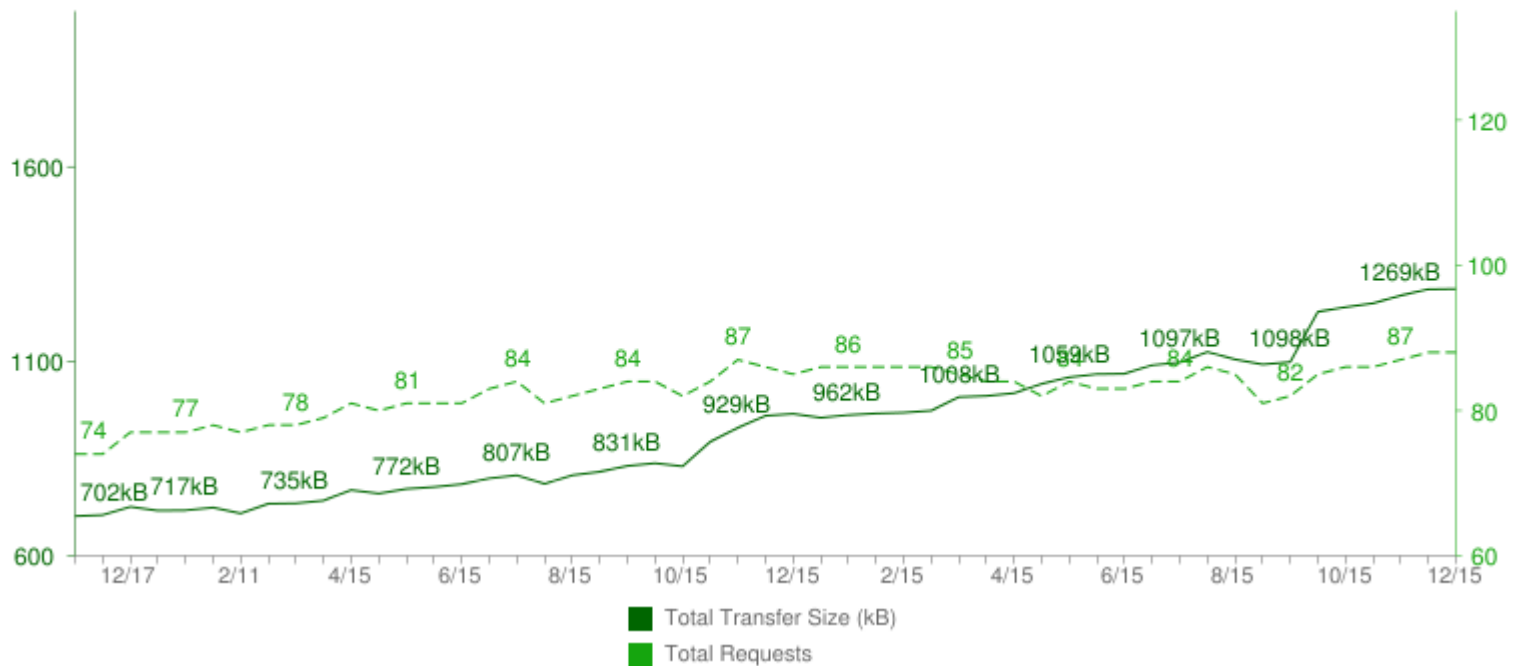
*Esta página es actualizada por [webadmin](#).
Ultima modificacion: 17 de marzo de 1999*

- Prácticamente de texto, pocas imágenes, nada de interactividad.
- 60 KB de tamaño promedio [ref]
[<http://www.pantos.org/atw/35654.html>]

¿Cómo es la Web de Hoy?

Tamaño de página y de peticiones promedio (2010-2012)

Total Transfer Size & Total Requests

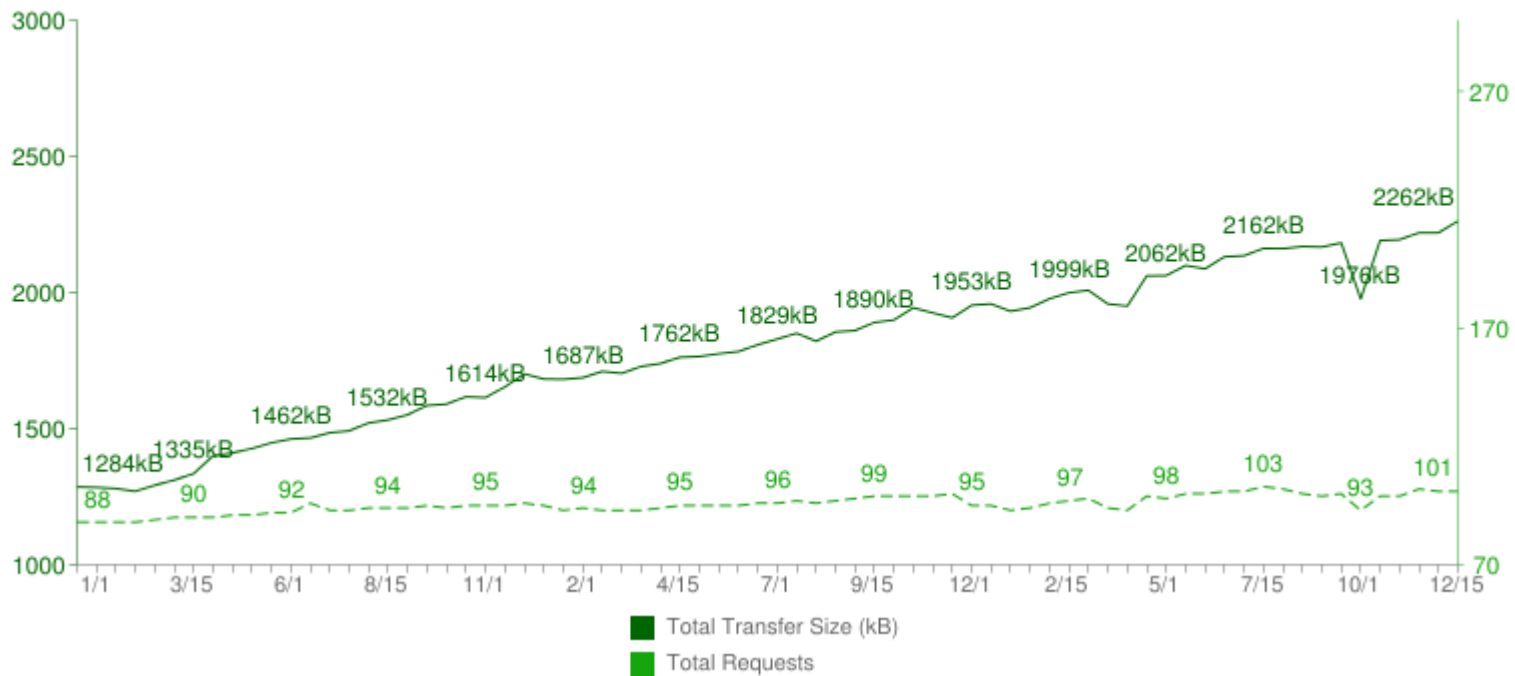


2010: **74** peticiones HTTP → casi **90** en 2012
2010: **705 KB** → **1269 KB** en 2012

¿Cómo es la Web de Hoy?

Tamaño de página y de peticiones promedio (2012-2015)

Total Transfer Size & Total Requests

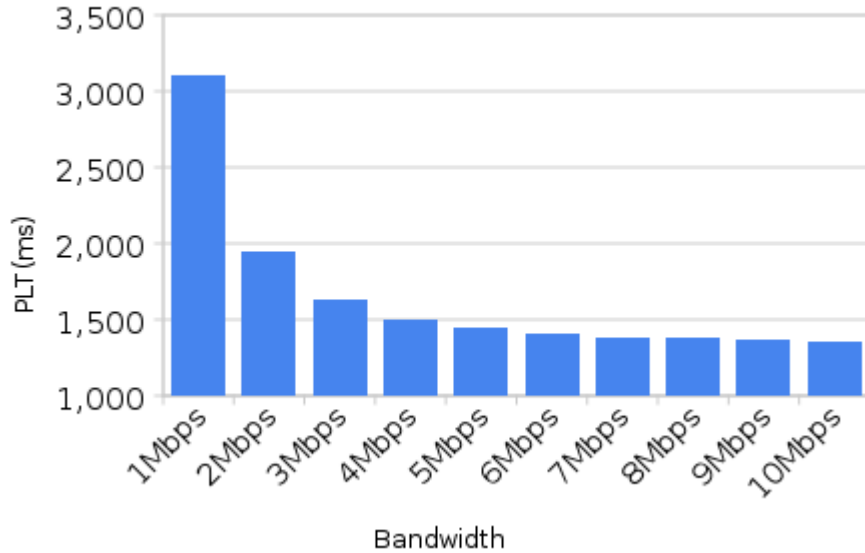


2012: casi **90** peticiones HTTP → **+100** en 2015
2012: **1269 KB** → **2262 KB** en 2015

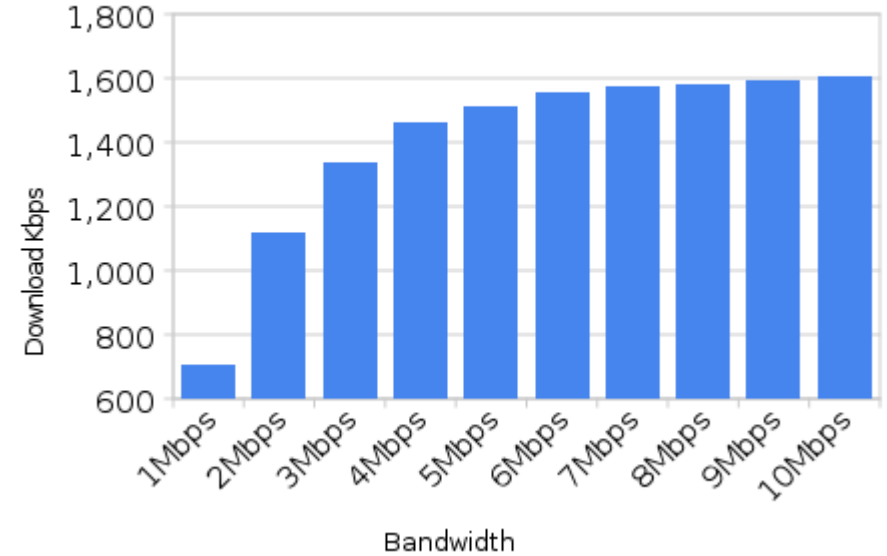
¿Cómo es la Web de Hoy?

Ancho de Banda y Latencia (RTT)

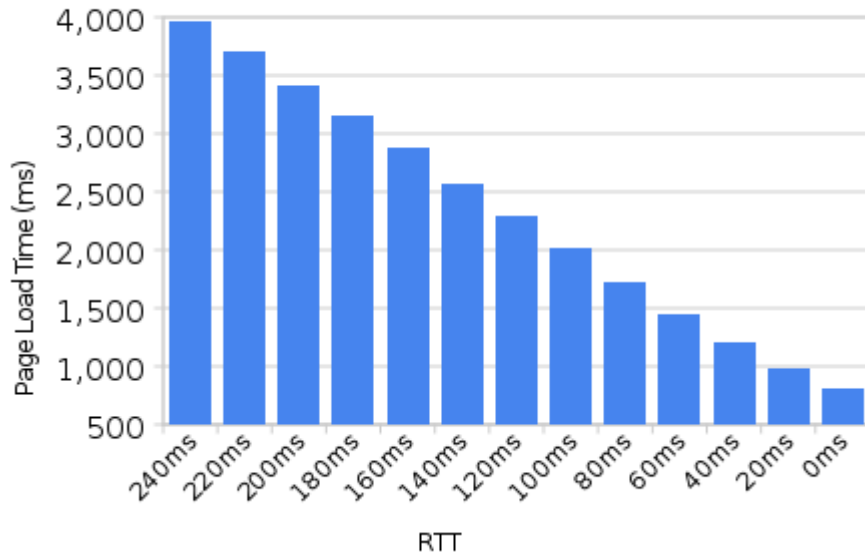
PLT Latency per Bandwidth @ 60ms RTT



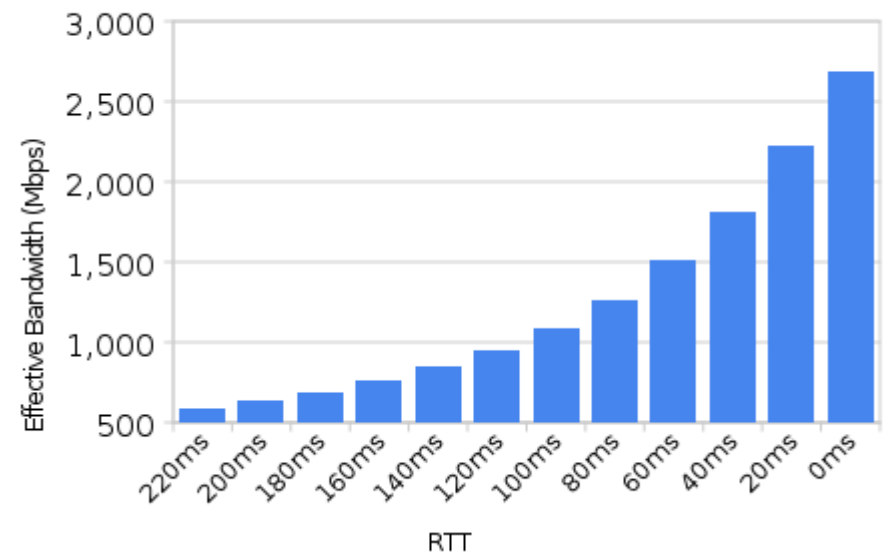
Effective Bandwidth of HTTP @ 60ms RTT



Page Load Time As RTT Decreases @ 5 Mbps

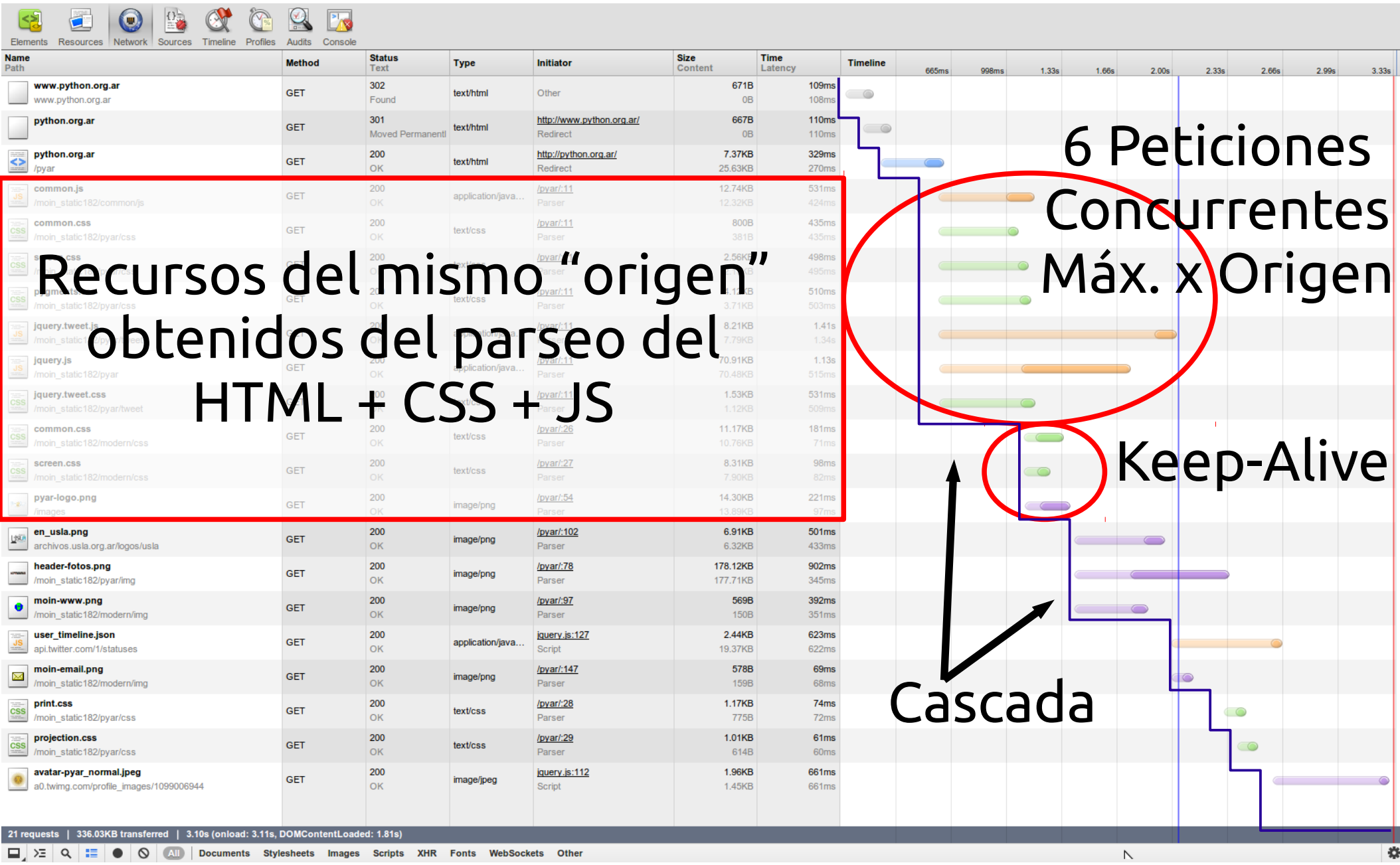


Effective Bandwidth as RTT decreases @ 5 Mbps



“Cascada” HTTP/1.1 - www.python.org.ar

(18 + 3 requests, ~336 KB, 3.1 seg = ~100 KB/seg)



Recursos del mismo “origen”
obtenidos del parseo del
HTML + CSS + JS

6 Peticiones
Concurrentes
Máx. x Origen

Keep-Alive

Cascada

HTTP/1.1 y la Web actual

- El **RTT es determinante** en el tiempo de carga de la página en HTTP/1.1.
- HTTP/1.1 es un protocolo que obliga a serializar las peticiones.
- Mucha heurística de optimización de tráfico y recursos en el browser.



HTTP/1.1 y la Web actual (cont.)

- *Hacks* para evitar limitaciones de HTTP/1.1
 - *Domain Sharding*
 - Recursos *inline*, *minificados*, *image maps*, *CSS sprites*
 - Ordenamiento, dependencias...
- *Headers* cada vez más grandes
- La realidad es que TCP fue hecho para conexiones con un tiempo de vida largo.
- En cambio, los browsers usan HTTP sobre TCP con ráfagas de conexiones.



HTTP/2 – RFC 7540

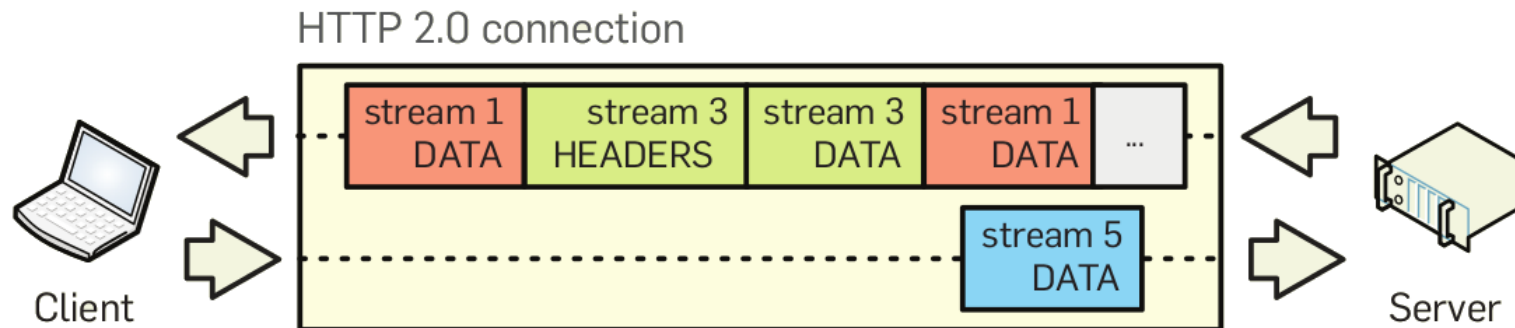
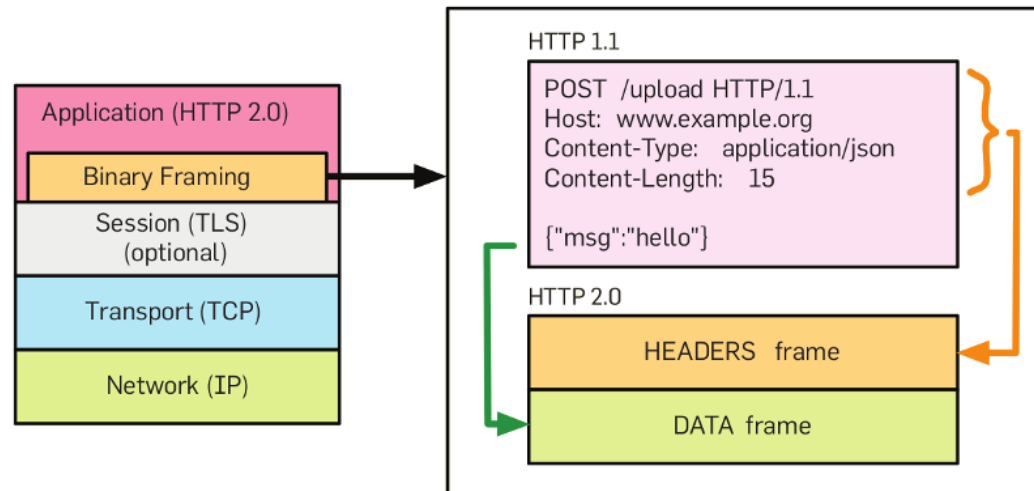
- Basado en SPDY, un protocolo desarrollado por Google desde 2009.
- Modifica cómo se lee/escribe el tráfico HTTP en el socket TCP (“sintaxis”).
- Toda la semántica de HTTP se mantiene.
- El objetivo es reducir el tiempo de carga de las páginas web en forma global.
- Lo que hace no es nada novedoso.



HTTP/2 – Características elementales

- Multiplexación del tráfico por una única conexión TCP persistente.
- Binario.
- Compresión de encabezados.
- Nuevas posibilidades: Server-Push, Priorización, Dependencias, Control de Flujo.
- En la práctica, se utiliza sobre TLS: Cifrado.
- En el camino, se definen varios RFC más:
 - **TLS ALPN**: Application-Layer Protocol Negotiation Extension.
 - **HPACK**: HTTP Header Compression.

HTTP/2 – Framing y Streams



HTTP/2 – Upgrade desde HTTP/1.1

1. Sabiendo de antemano que el server lo soporta.
2. Puerto 80 'http://' URIs – HTTP Upgrade

```
GET /page HTTP/1.1
Host: server.example.com
Connection: Upgrade, HTTP2-Settings
Upgrade: HTTP/2.0
HTTP2-Settings: (SETTINGS payload)
```

```
HTTP/1.1 200 OK
Content-length: 243
Content-type: text/html
```

(... HTTP 1.1 response ...)

(or)

```
HTTP/1.1 101 Switching Protocols
Connection: Upgrade
Upgrade: HTTP/2.0
```

(... HTTP 2.0 response ...)

3. Puerto 443 'https://' URIs – HTTPS → TLS + ALPN

HTTP/2 – Delta headers

Request #1

:method	GET
:scheme	https
:host	example.com
:path	/resource
accept	image/jpeg
user-agent	Mozilla/5.0 ...

implicit

implicit

implicit

implicit

implicit

Request #2

:method	GET
:scheme	https
:host	example.com
:path	/new_resource
accept	image/jpeg
user-agent	Mozilla/5.0 ...

HEADERS frame (Stream 1)

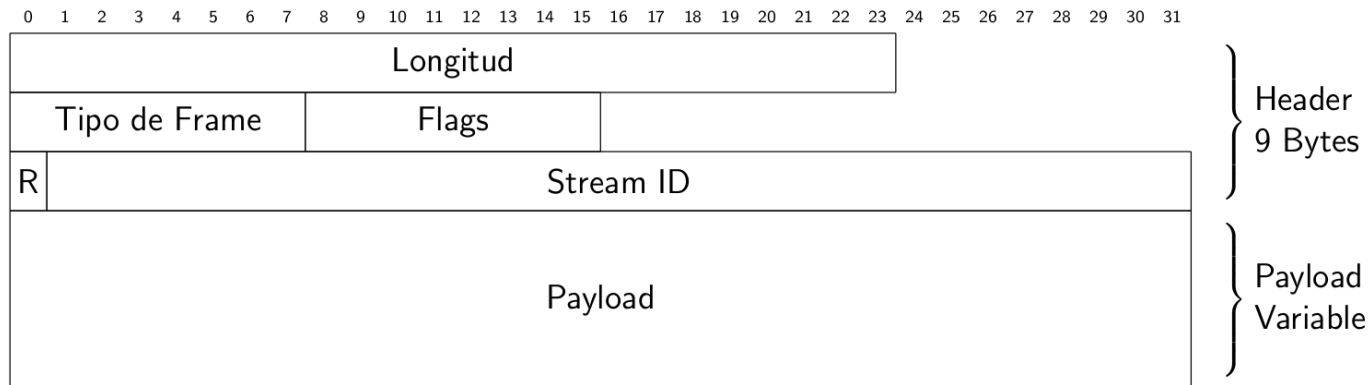
:method:	GET
:scheme:	https
:host:	example.com
:path:	/resource
accept:	image/jpeg
user-agent:	Mozilla/5.0 ...

HEADERS frame (Stream 3)

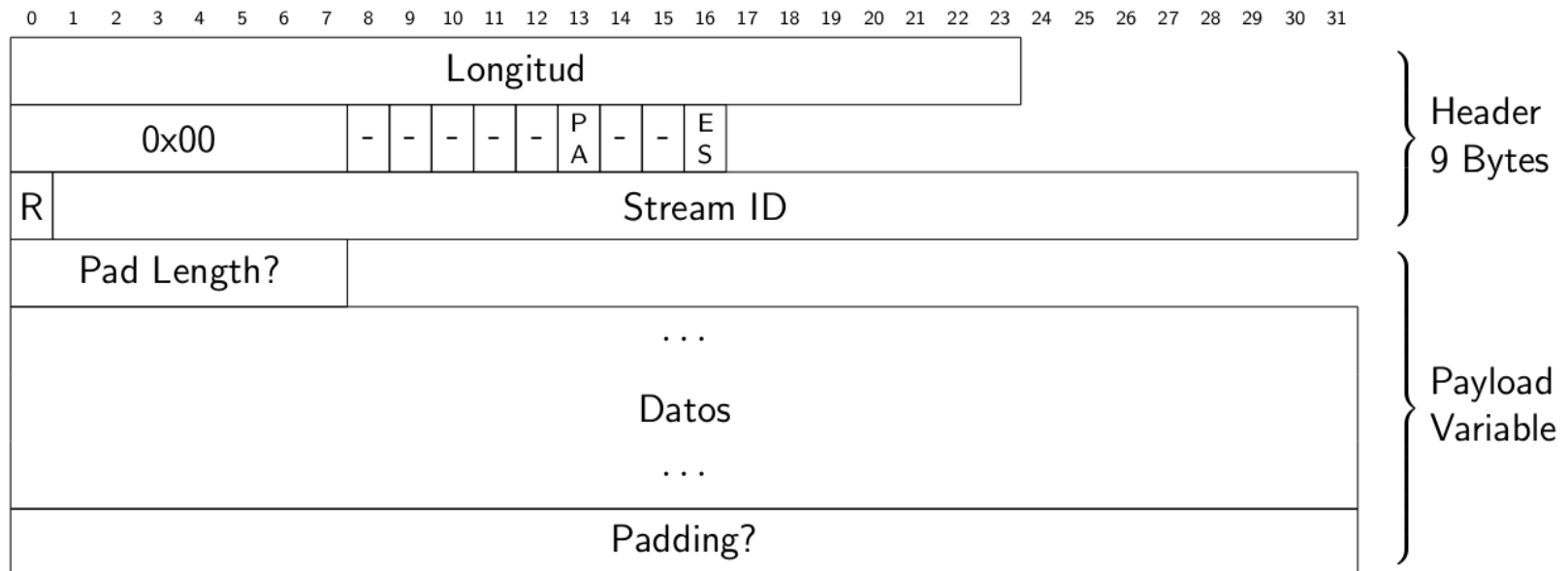
:path:	/new_resource
--------	---------------

HTTP/2 – Frames / Data Frame

Estructura común de todos los frames

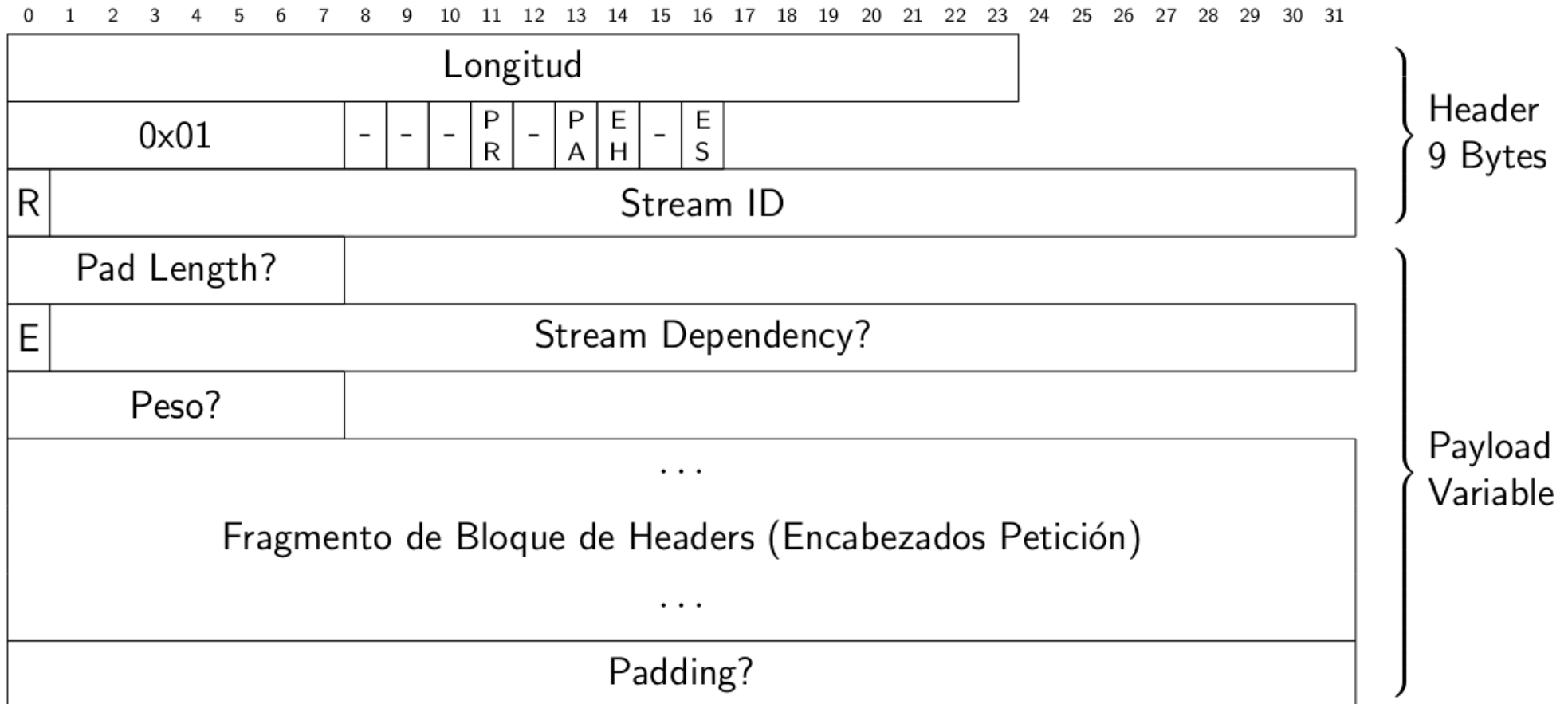


DATA Frame (0x00)



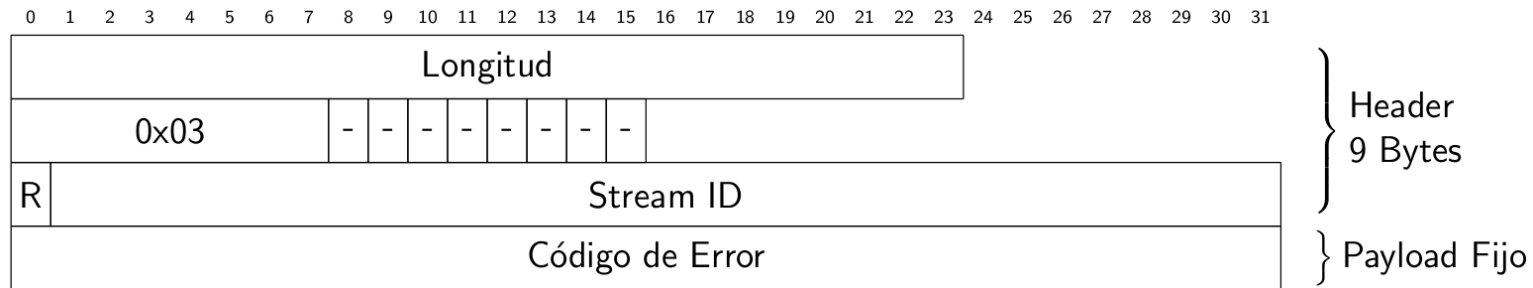
HTTP/2 – Headers Frame

HEADERS Frame (0x01)

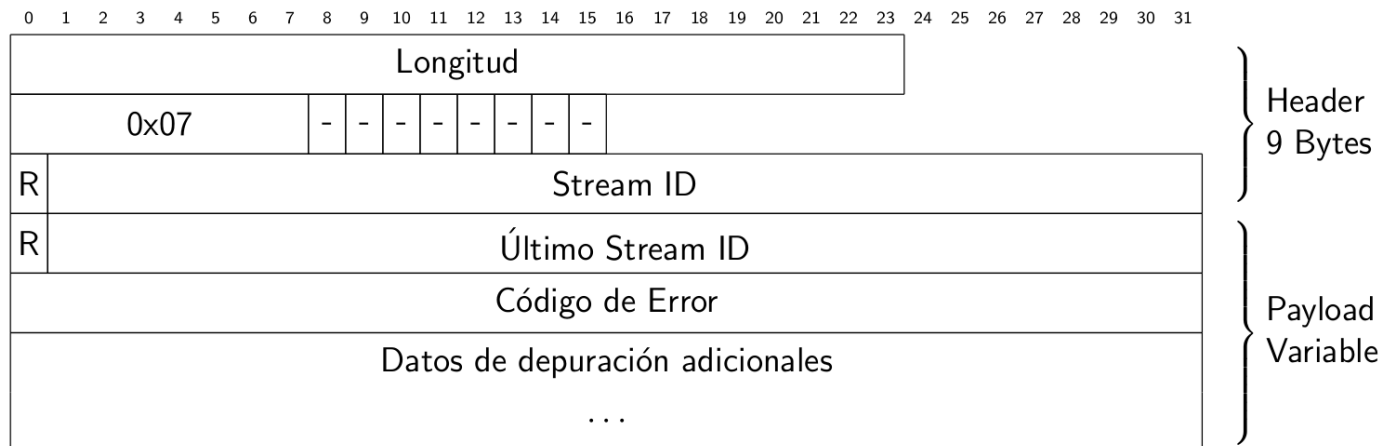


HTTP/2 – Reset & Goaway Frames

RST_STREAM Frame (0x03)



GOAWAY Frame (0x07)

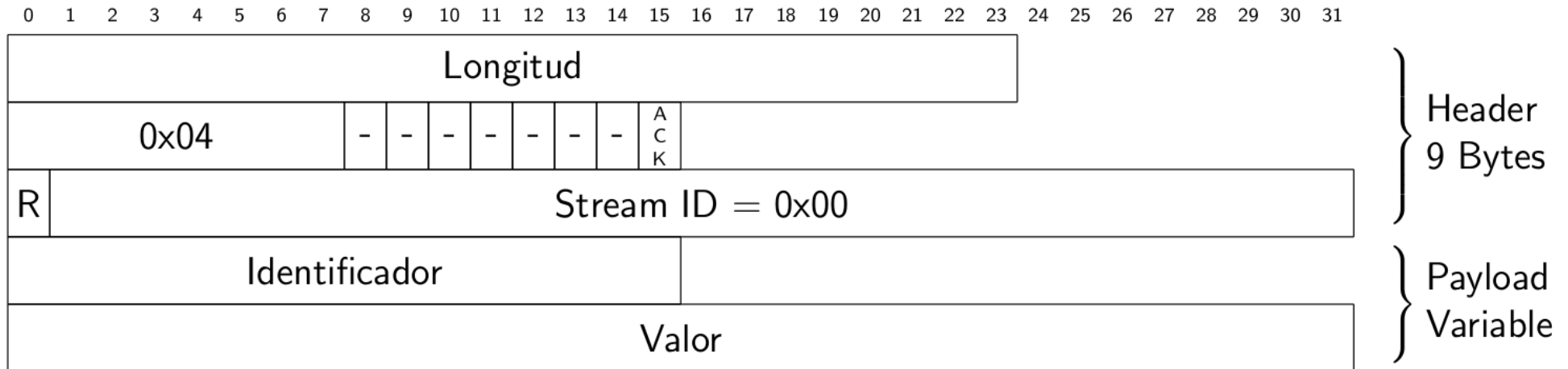


Algunos códigos de error

- 0x0: NO_ERROR
- 0x1: PROTOCOL_ERROR
- 0x2: INTERNAL_ERROR
- 0x3: FLOW_CONTROL_ERROR
- 0x4: SETTINGS_TIMEOUT
- 0x5: STREAM_CLOSED
- 0x6: FRAME_SIZE_ERROR
- 0x7: REFUSED_STREAM
- 0x8: CANCEL
- 0x9: COMPRESSION_ERROR

HTTP/2 – SETTINGS Frame

SETTINGS Frame (0x04)



Algunos Settings

- 0x2: ENABLE_PUSH
- 0x3: MAX_CONCURRENT_STREAMS
- 0x4: INITIAL_WINDOW_SIZE
- 0x5: MAX_FRAME_SIZE

Otros Frames:

- **0x02 – PRIORITY:** Prioridad de un Stream
- **0x06 – PING:** Medir RTT al destino
- **0x08 – WINDOW_UPDATE:** Control de Flujo
- **0x09 – CONTINUATION:** headers extra



Ejemplos

<chrome://net-internals/#http2>

<http://www.http2demo.io/>

<https://http2.akamai.com/demo>

<https://www.cloudflare.com/http2/>

Estado Actual y Futuro de HTTP/2

- Es un estándar del IETF desde Mayo de 2015.
- Implementaciones:
 - Clientes: prácticamente todos los navegadores y herramientas como Curl, Wireshark, etc.
 - Servidores: Nginx, Apache (beta), IIS, F5, Jetty, HAProxy, etc.
 - Infraestructura: Google (GAE), Twitter, Wordpress, Akamai, Cloudflare, Strangeloop, Facebook....
 - Lenguajes de programación, algo pendiente aún.
- Más: <https://github.com/http2/http2-spec/wiki/Implementations>

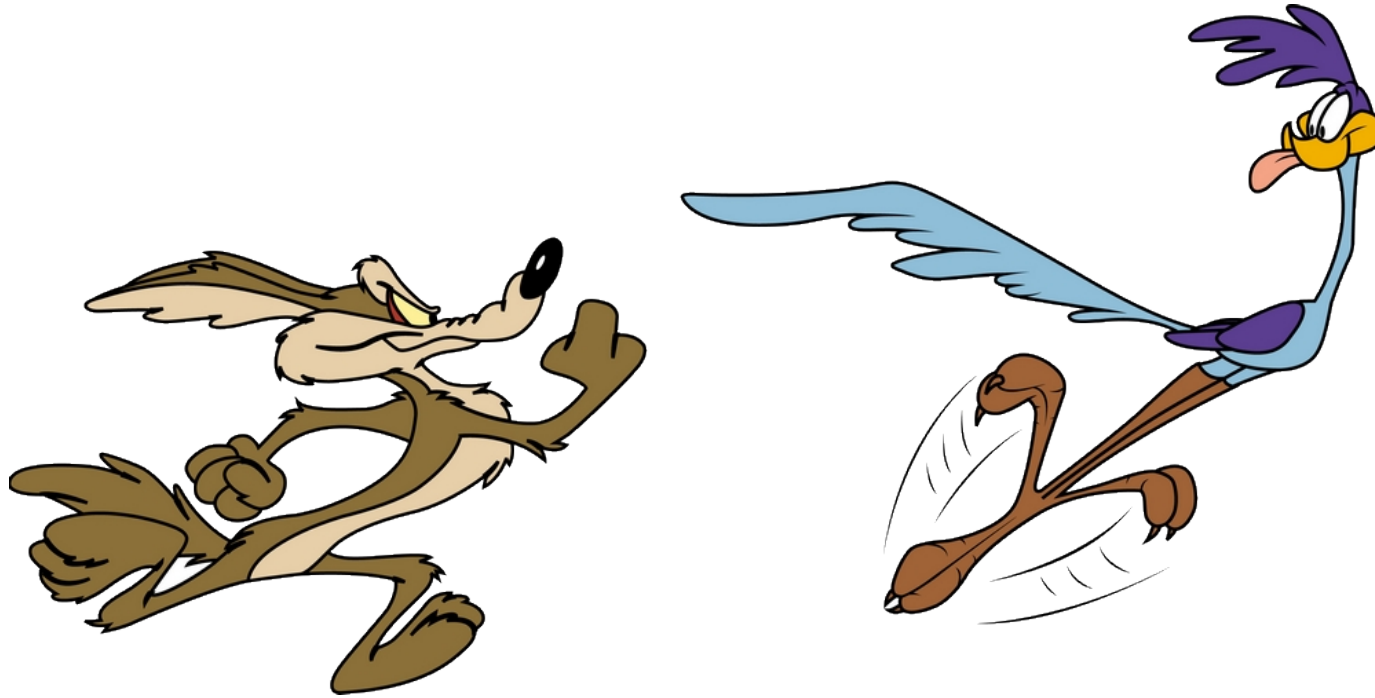
Estado Actual y Futuro de HTTP/2

Quedan cosas por desarrollar e investigar:

- Mecanismo de descubrimiento y negociación de HTTP/2, por ejemplo registros SRV de DNS.
- “Encriptación oportunística” para HTTP/2.
- Integración con aplicaciones del lado del servidor.
- Interrelación con Websockets.
- **QUIC** (HTTP sobre UDP).
- Aprovechamiento de características nuevas, por ejemplo, en proxys intermediarios.
- Desarrollo de herramientas y guías para optimizar la web actual escrita para HTTP/1.1.

Conclusiones Generales

- HTTP/1.1 está mostrando sus años con las características de los sitios y conexiones actuales.
- Los *hacks* no escalan y aumentan la complejidad.
- Prácticamente todos los browsers actuales soportan HTTP/2 [ref].
- HTTP/2 mejora mucho el rendimiento, pero para implementarlo bien™ hay que *deshackear* lo hecho.
- La migración no es *painless* (aunque podría ser peor).
- Resta software dentro de la arquitectura Web por construir y estabilizar (Proxys, Load Balancers, Servers, Firewalls...)



Teleinformática y Redes 2016

