Nuevos protocolos

Area de Ingeniería Telemática
http://www.tlm.unavarra.es

Factores en el rendimiento

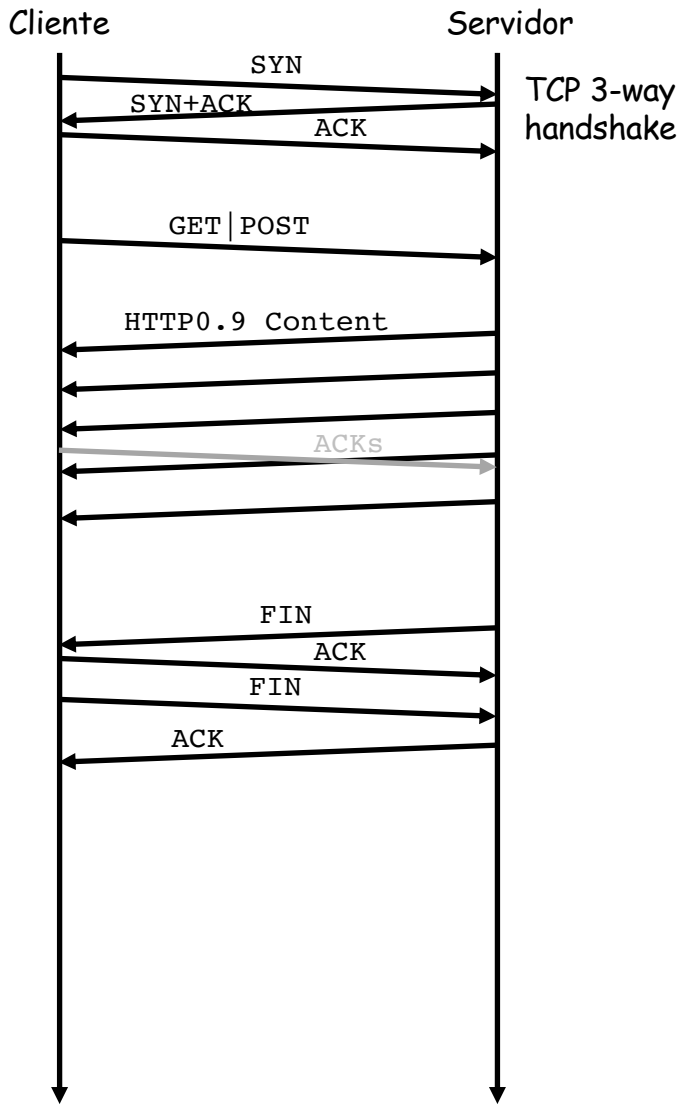
- La infraestructura de red
  - Capacidad en el camino
  - Probabilidad de pérdidas
  - Desórdenes
  - Retardos
- El protocolo de transporte
  - Mecanismos de control de conexión
  - Reacción ante pérdidas
  - Adaptación ante congestión (y evitarla)
- El protocolo de aplicación

upna
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa
Redes de Nueva Generación
Área de Ingeniería Telemática

HTTP 1.1 – Conexiones persistentes y pipelining

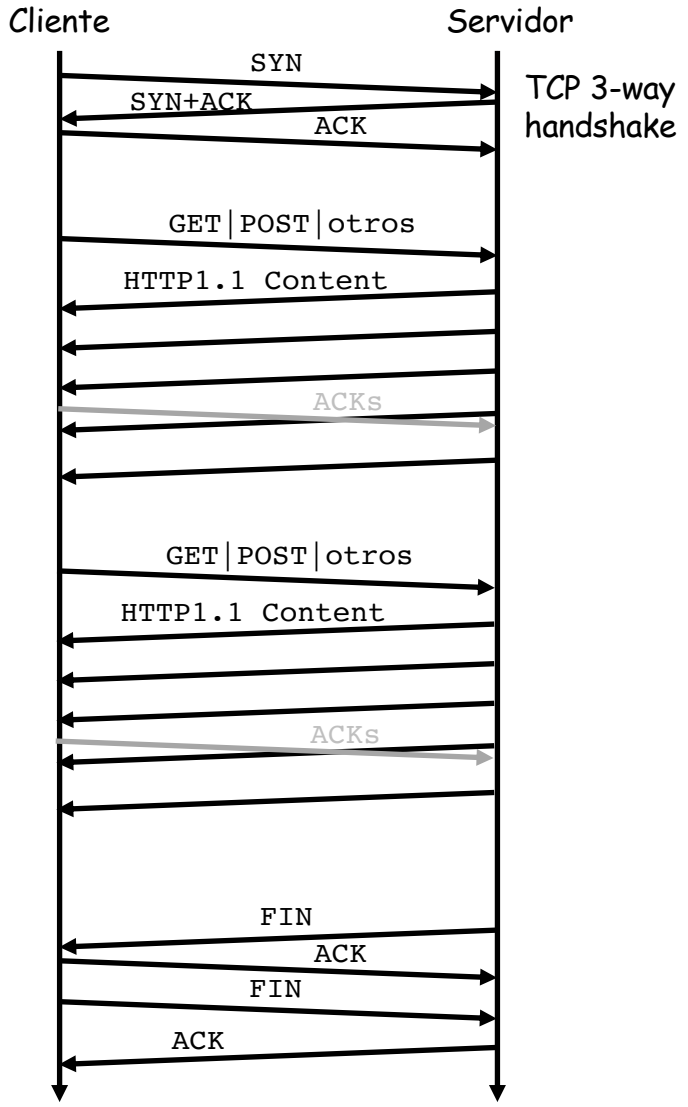
HTTP < 1.1

- Una petición por conexión
- Cierre marca el final de la respuesta



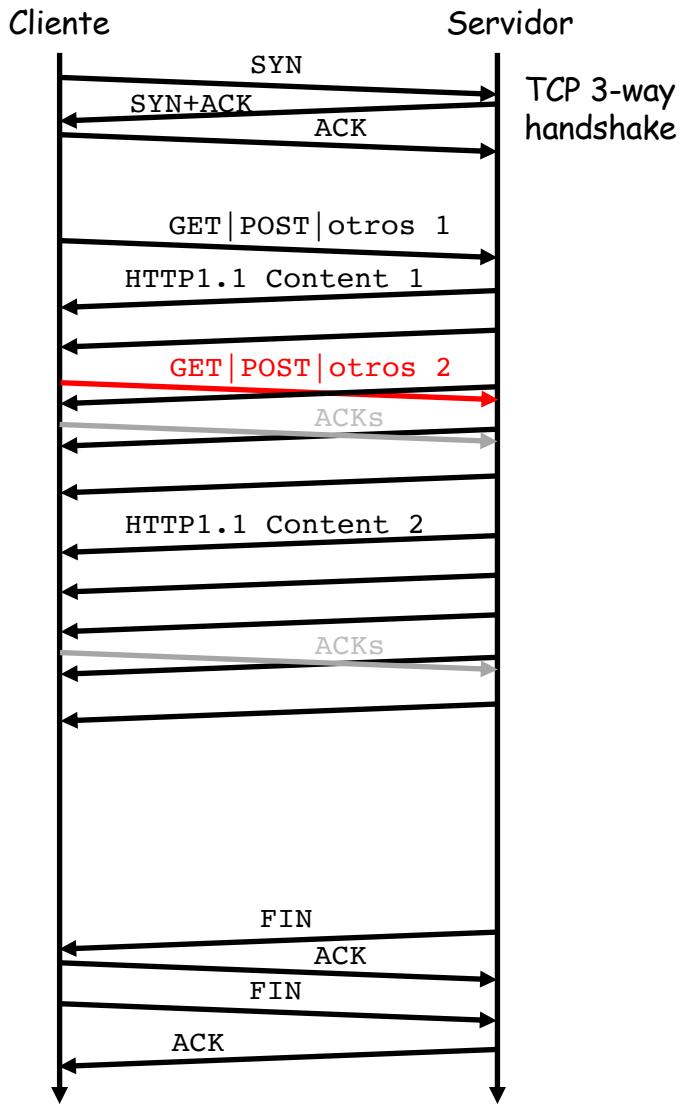
HTTP 1.1

- Conexiones persistentes



HTTP 1.1

- Conexiones persistentes
- Pipelining: respuestas en el mismo orden que las peticiones
- Posible entrega de rangos de bytes



HTTP1.1

- Algunas mejoras en HTTP1.1
  - Conexiones persistentes
  - Pipelining
  - Entrega de rangos de bytes

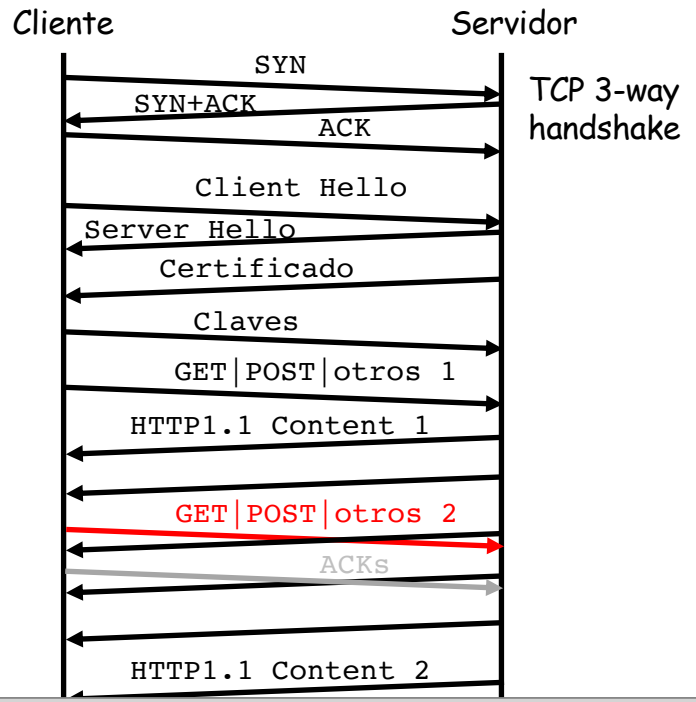
- Y algunos “hacks” de navegadores y desarrolladores
  - Conexiones en paralelo (limitadas a 5-6 para un servidor)
  - *Domain sharding* (recursos en diferentes dominios)
  - Unión de ficheros pequeños (CSS, Javascript, imágenes)
  - Inline de ficheros con el HTML

HTTP1.1 - Problemas

- Conexiones en paralelo y domain sharding crean mayor número de sockets, más conexiones en NATs, más DNS
- Concatenación e inlining rompe la modularidad y entorpece las caches
- Optimizado para grandes transferencias pero una web contiene muchos recursos pequeños
- Pipelining no suele emplearse (problemas con proxies y algunos servidores)
- Aún con pipelining tiene HOL blocking (hasta completar respuesta grande no se atiende a otra del pipeline)
- Para mejorar el tiempo de carga hay que reducir el RTT (CDNs), pero esto tiene un límite (“c”)
- Reducir RTT reduciendo *chattiness*
- *TLS* añade aún más RTTs

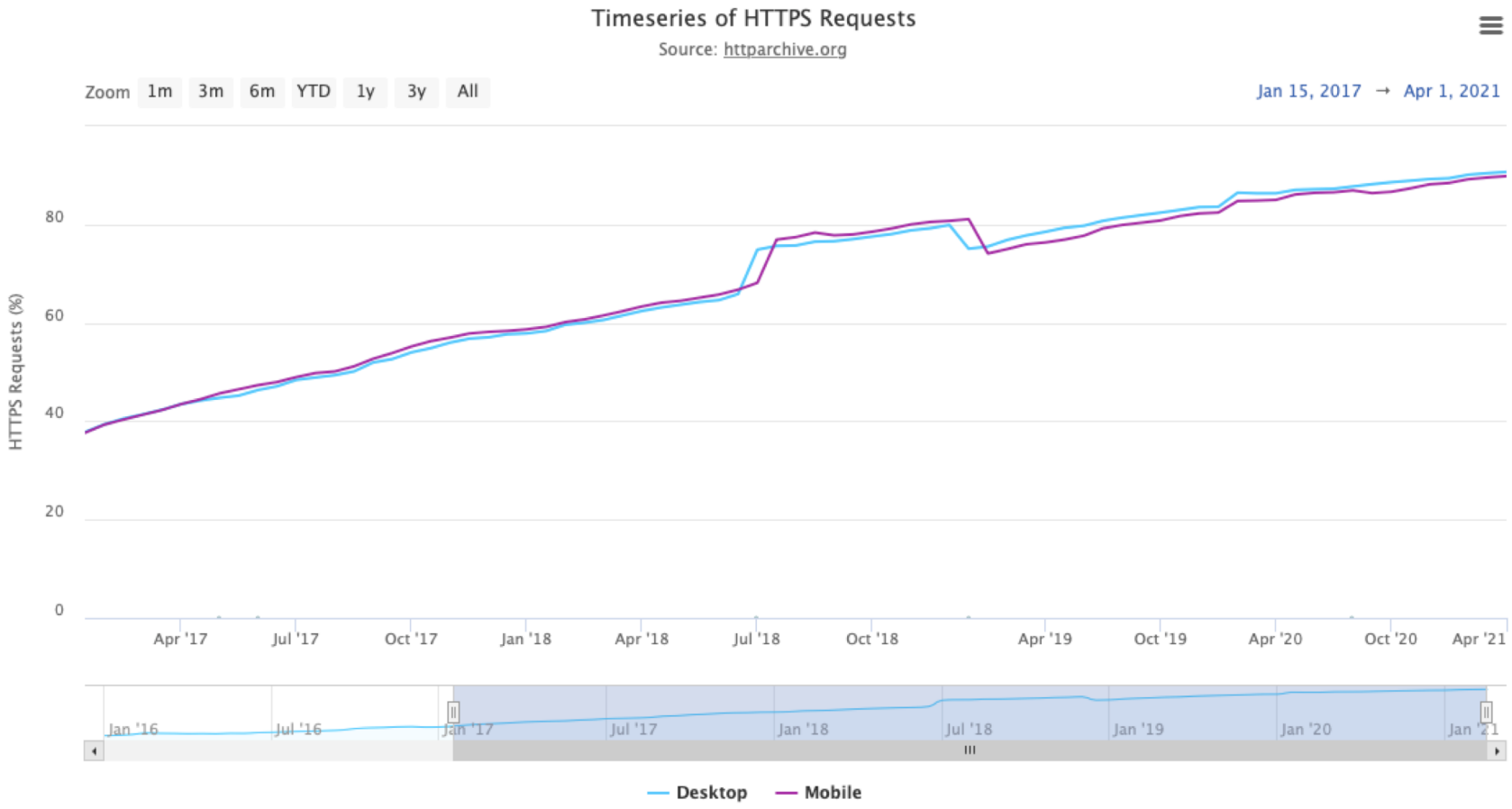
HTTP 1.1 + TLS

- Identificación de capacidades (ciphers, versiones)
- Intercambio de certificados
- Intercambio de información para la construcción de claves
- Al menos 1 RTT



	Time	Source	Destination	Total Length	Info
1	0.000000	10.6.4.40	108.174.11.37	60	45782 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2004512222 TSecr=0 WS=128
2	0.185023	108.174.11.37	10.6.4.40	60	443 → 45782 [SYN, ACK] Seq=0 Ack=1 Win=43440 Len=0 MSS=1460 SACK_PERM=1 TSval=1697854058 TSe
3	0.185106	10.6.4.40	108.174.11.37	52	45782 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2004512407 TSecr=1697854058
4	0.201494	10.6.4.40	108.174.11.37	569	Client Hello
5	0.386559	108.174.11.37	10.6.4.40	52	443 → 45782 [ACK] Seq=1 Ack=518 Win=45056 Len=0 TSval=1697854260 TSecr=2004512423
6	0.388269	108.174.11.37	10.6.4.40	1500	Server Hello
7	0.388341	10.6.4.40	108.174.11.37	52	45782 → 443 [ACK] Seq=518 Ack=1449 Win=64128 Len=0 TSval=2004512610 TSecr=1697854261
8	0.388394	108.174.11.37	10.6.4.40	1500	443 → 45782 [ACK] Seq=1449 Ack=518 Win=45056 Len=1448 TSval=1697854261 TSecr=2004512423 [TCP
9	0.388394	108.174.11.37	10.6.4.40	476	Certificate, Server Key Exchange, Server Hello Done
10	0.388452	10.6.4.40	108.174.11.37	52	45782 → 443 [ACK] Seq=518 Ack=3321 Win=63488 Len=0 TSval=2004512610 TSecr=1697854261
11	0.392660	10.6.4.40	108.174.11.37	145	Client Key Exchange, Change Cipher Spec, Finished

HTTP 1.1 + TLS


Timeseries of HTTPS Requests
Source: httparchive.org
Zoom 1m 3m 6m YTD 1y 3y All
Jan 15, 2017 → Apr 1, 2021
HTTPS Requests (%)
80
60
40
20
0
Apr '17
Jul '17
Oct '17
Jan '18
Apr '18
Jul '18
Oct '18
Apr '19
Oct '19
Apr '20
Oct '20
Apr '21
Jan '16
Jul '16
Jan '17
Jul '17
Jan '18
Jul '18
Jan '19
Jan '20
Jan '21
Desktop
Mobile

upna
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Redes de Nueva Generación
Área de Ingeniería Telemática

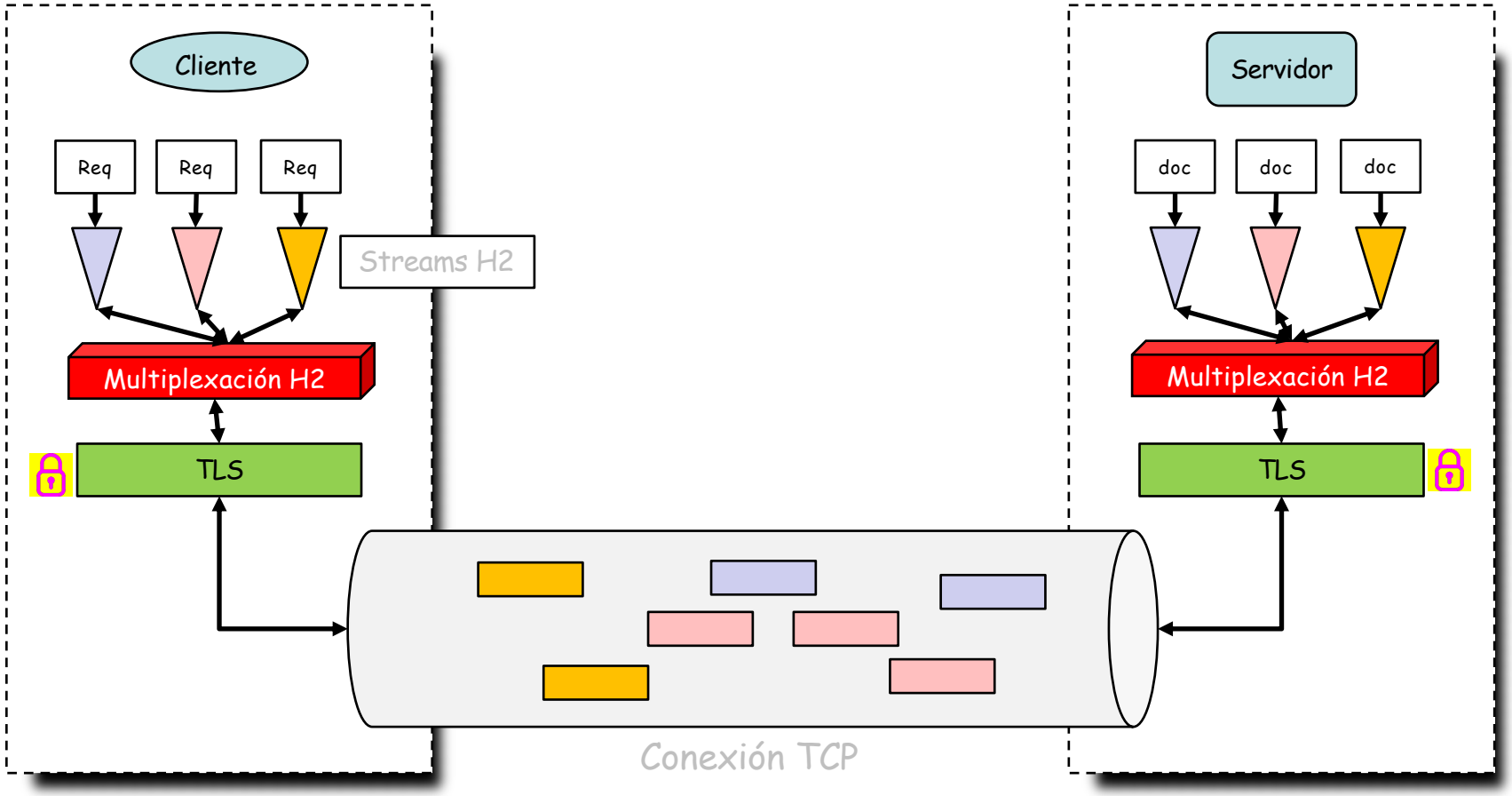
HTTP/2

HTTP/2 (H2)

- RFC 7540 “Hypertext Transfer Protocol Version 2 (HTTP/2)” (BitGo, Google, Mozilla, 2015)
- (Antes SPDY, de Google)
- Binario
  - Ya no es texto, más compacto, más eficiente al procesarlo
  - Se pueden comprimir las cabeceras (evita tener que enviarlas en cada petición o respuesta)
  - Permite multiplexación
- Multiplexación
  - Peticiones y respuestas pueden estar multiplexadas
  - Eso quiere decir por ejemplo que no es necesario completar una respuesta para empezar a enviar otra
  - Trabaja con *streams* dentro de una sola conexión
  - El navegador puede asignarles prioridades en las peticiones
  - Esto elimina el HOL blocking y la necesidad de conexiones en paralelo
- Server Push
  - El servidor puede entregar recursos al cliente aunque no los haya pedido
  - Ahorra la petición (esto se estaba haciendo con el *inlining*)

Multiplexación en H2

- Frames de los diferentes streams HTTP/2 en mensajes *Application Data* de TLS enviados por la conexión TCP
- Frame HEADERS contienen la cabecera HTTP
- Frame DATA con el contenido de la respuesta (o del POST)



Ejemplo

No.	Time	Source	Destination	Info
61	0.411754	10.6.4.40	72.247.210.11	HEADERS[9]: GET /wp-content/themes/nobelprize/dist/css/style.0.4.2.min.css?ver=0.4.2
62	0.411803	10.6.4.40	72.247.210.11	HEADERS[11]: GET /wp-content/plugins/jetpack/css/jetpack.css?ver=9.2.1
63	0.411803	10.6.4.40	72.247.210.11	HEADERS[13]: GET /wp-content/themes/nobelprize/assets/js/frontend/lib/loadjs.js?ver=0.4.2
64	0.422646	72.247.210.11	10.6.4.40	443 → 60836 [ACK] Seq=29891 Ack=1635 Win=32256 Len=0 TSval=2231141594 TSecr=154066703
65	0.423735	10.6.4.40	72.247.210.11	HEADERS[15]: GET /wp-content/themes/nobelprize/assets/js/frontend/lib/clamp/clamp.min.js?ver=0.4.2
66	0.423785	10.6.4.40	72.247.210.11	HEADERS[17]: GET /wp-content/plugins/nobelprize/assets/js/admin/lib/flickity/flickity.pkgd.js?ver=0.2.0-3
67	0.423834	10.6.4.40	72.247.210.11	HEADERS[19]: GET /wp-content/themes/nobelprize/dist/js/frontend.0.4.2.min.js?ver=0.4.2
68	0.425261	72.247.210.11	10.6.4.40	443 → 60836 [ACK] Seq=29891 Ack=1746 Win=32256 Len=1448 TSval=2231141596 TSecr=154066703 [TCP segment of
69	0.425357	72.247.210.11	10.6.4.40	HEADERS[3]: 200 OK, DATA[3], DATA[3] (text/css)
70	0.425447	10.6.4.40	72.247.210.11	60836 → 443 [ACK] Seq=2092 Ack=32472 Win=63488 Len=0 TSval=154066717 TSecr=2231141596
71	0.426687	72.247.210.11	10.6.4.40	HEADERS[5]: 200 OK, DATA[5], DATA[5] (text/css)
72	0.427213	72.247.210.11	10.6.4.40	443 → 60836 [ACK] Seq=33326 Ack=1746 Win=32256 Len=1448 TSval=2231141598 TSecr=154066703 [TCP segment of
73	0.427279	10.6.4.40	72.247.210.11	60836 → 443 [ACK] Seq=2092 Ack=34774 Win=63488 Len=0 TSval=154066719 TSecr=2231141598
74	0.427335	72.247.210.11	10.6.4.40	443 → 60836 [ACK] Seq=34774 Ack=1746 Win=32256 Len=1448 TSval=2231141598 TSecr=154066703 [TCP segment of
75	0.427459	72.247.210.11	10.6.4.40	443 → 60836 [ACK] Seq=36222 Ack=1746 Win=32256 Len=1448 TSval=2231141598 TSecr=154066703 [TCP segment of

```
▶ Frame 69: 1199 bytes on wire (9592 bits), 1199 bytes captured (9592 bits)
▶ Ethernet II, Src: Cisco_dc:71:c4 (30:e4:db:dc:71:c4), Dst: Universa_2c:dc:6c (00:1e:37:2c:dc:6c)
▶ Internet Protocol Version 4, Src: 72.247.210.11, Dst: 10.6.4.40
▶ Transmission Control Protocol, Src Port: 443, Dst Port: 60836, Seq: 31339, Ack: 1746, Len: 1133
▶ [2 Reassembled TCP Segments (2581 bytes): #68(1448), #69(1133)]
▼ Transport Layer Security
  ▼ TLSv1.3 Record Layer: Application Data Protocol: http2
      Opaque Type: Application Data (23)
      Version: TLS 1.2 (0x0303)
      Length: 2576
      [Content Type: Application Data (23)]
      Encrypted Application Data: 77f9299f91c1e3032294907016a1b4ea3a5a541fe3dc618bc9b9c27bf0f6efba97b46755…
      [Application Data Protocol: http2]
▼ HyperText Transfer Protocol 2
  ▶ Stream: HEADERS, Stream ID: 3, Length 227, 200 OK
  ▼ Stream: DATA, Stream ID: 3, Length 2305 (partial entity body)
      Length: 2305
      Type: DATA (0)
    ▶ Flags: 0x00
      0... .... .... .... .... .... .... .... = Reserved: 0x0
      .000 0000 0000 0000 0000 0000 0000 0011 = Stream Identifier: 3
      [Pad Length: 0]
      Reassembled body in frame: 69
      Data: 1f8b0800000000000000cd5ae98ee336127e156f06013241d8b0bddd8389849d3ffb1883…
  ▶ Stream: DATA, Stream ID: 3, Length 0
```

HTTP/2

- Flow Control
  - Para cada *stream*
  - HTTP/2 no fuerza a un algoritmo para el control de flujo, solo lo soporta
- Negociación
  - Mediante mensajes se puede subir una conexión de HTTP1.1 a HTTP/2

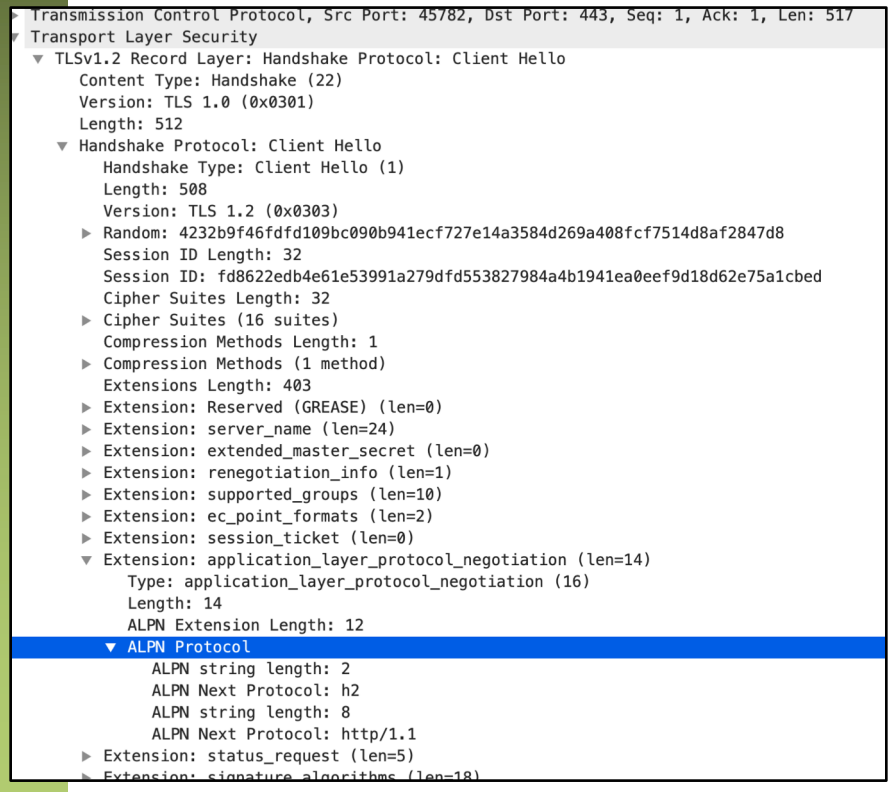
```
GET /page HTTP/1.1
Host: server.example.com
Connection: Upgrade, HTTP2-Settings
Upgrade: HTTP/2.0
HTTP2-Settings: (SETTINGS payload)
```

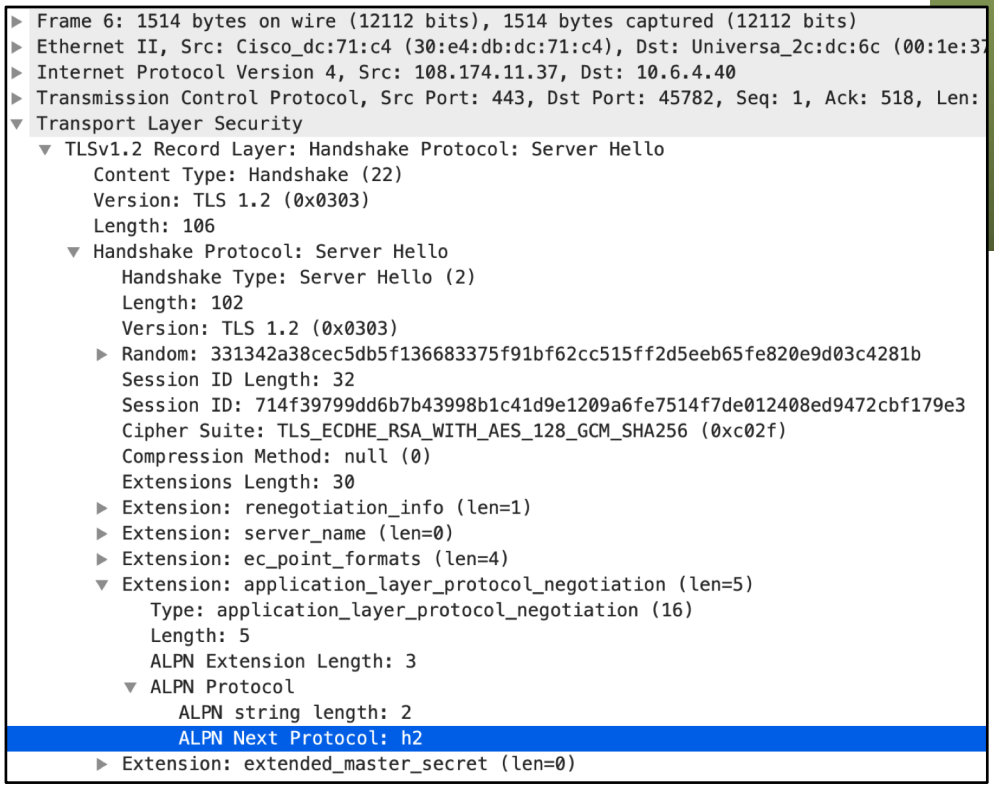
```
HTTP/1.1 200 OK
Content-length: 243
Content-type: text/html
(... HTTP 1.1 response ...)

 (or)
HTTP/1.1 101 Switching Protocols
Connection: Upgrade
Upgrade: HTTP/2.0
(... HTTP 2.0 response ...)
```

HTTP/2

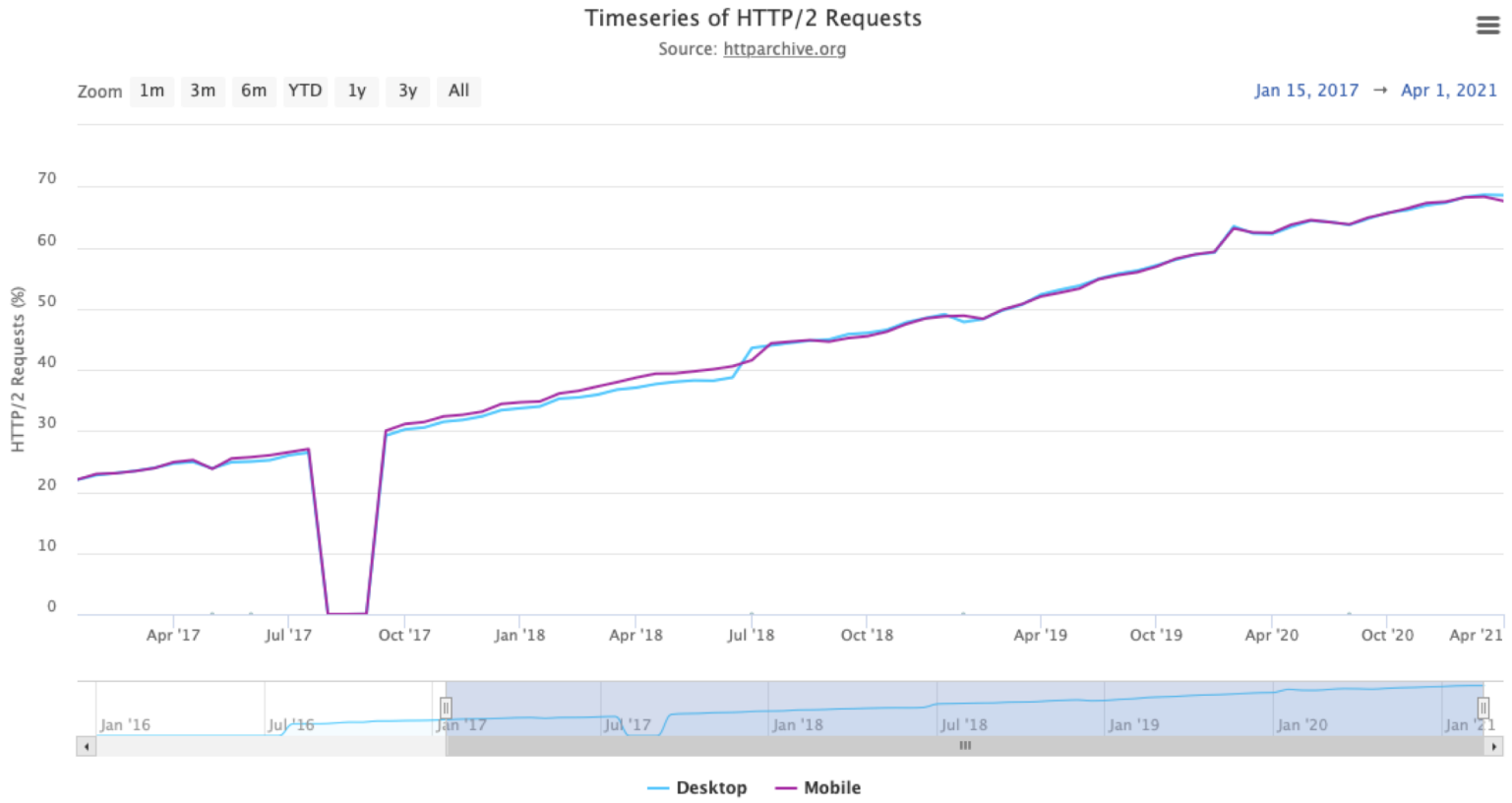
- Flow Control
  - Para cada *stream*
  - HTTP/2 no fuerza a un algoritmo para el control de flujo, solo lo soporta
- Negociación
  - Mediante mensajes se puede subir una conexión de HTTP1.1 a HTTP/2
  - Hoy en día solo sobre TLS
  - Mediante el ALPN





HTTP/2: Status

- RFC aprobada
- Múltiples implementaciones, en navegadores y servidores
- Soportado en IE11, Firefox51, Chrome49, Safari10, Opera43, etc
- Soportado ya por muchas CDNs y hostings (ej: Akamai, Azure CDN, MaxCDN, KeyCDN, CacheFly, CloudFlare, Hawk host, etc)
- Estimaciones de que es empleado en el 47.7% de websites[1]



http://isthewebhttp2yet.com/measurements/adoption.html#time

[1] https://w3techs.com/technologies/details/ce-http2/all/all (Mayo 2021)