

upna

Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Redes de Nueva Generación
Área de Ingeniería Telemática

Nuevos protocolos

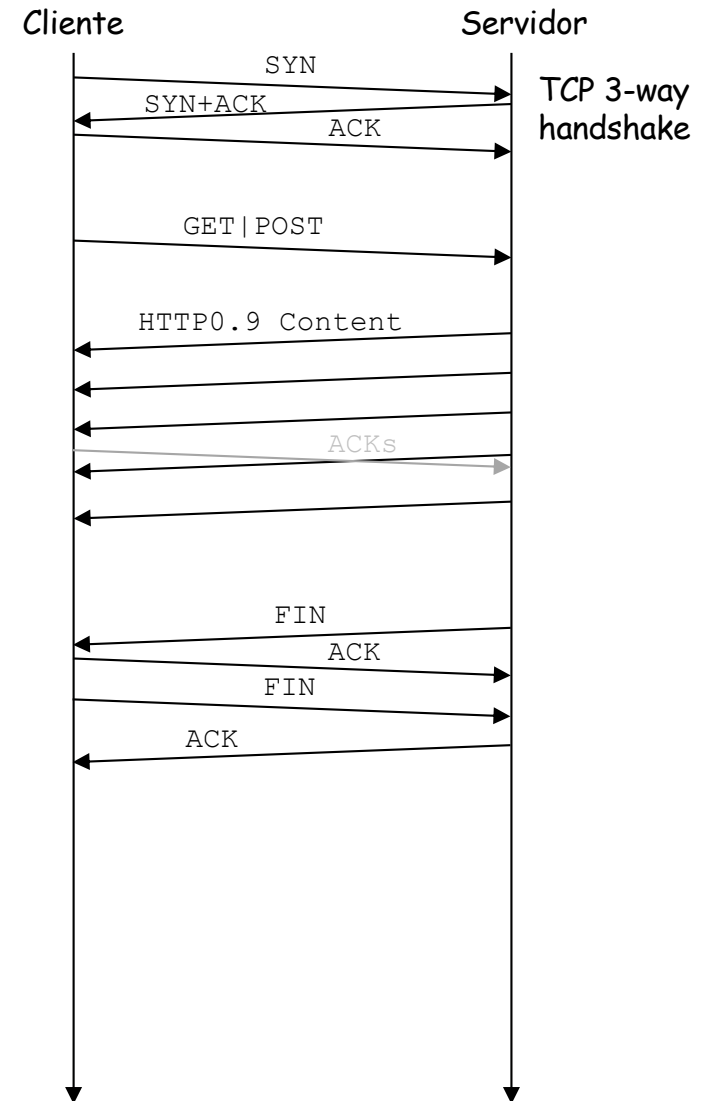
Factores en el rendimiento

- 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

HTTP 1.1 – Conexiones persistentes y pipelining

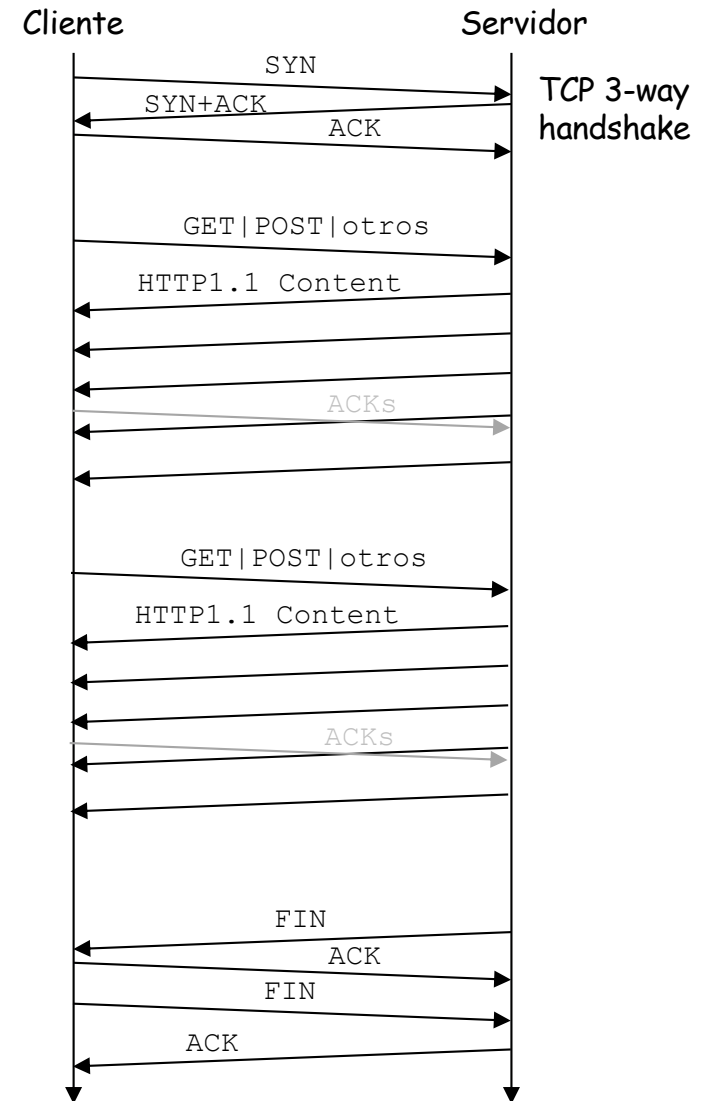
HTTP < 1.1

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



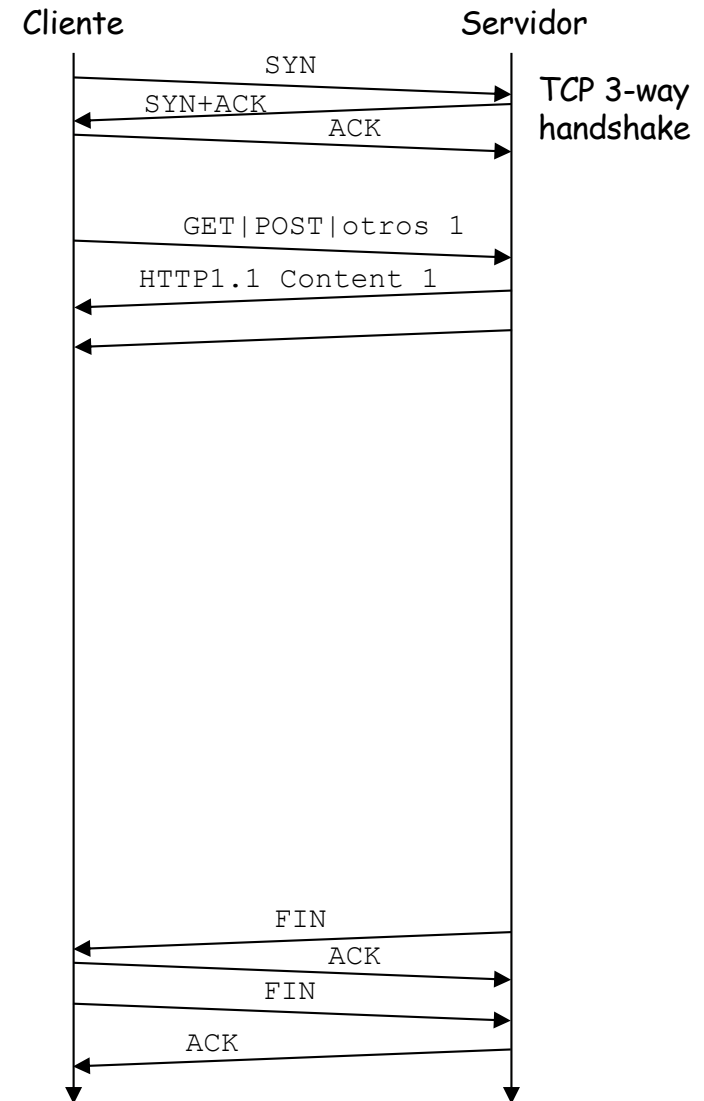
HTTP 1.1

- Conexiones persistentes



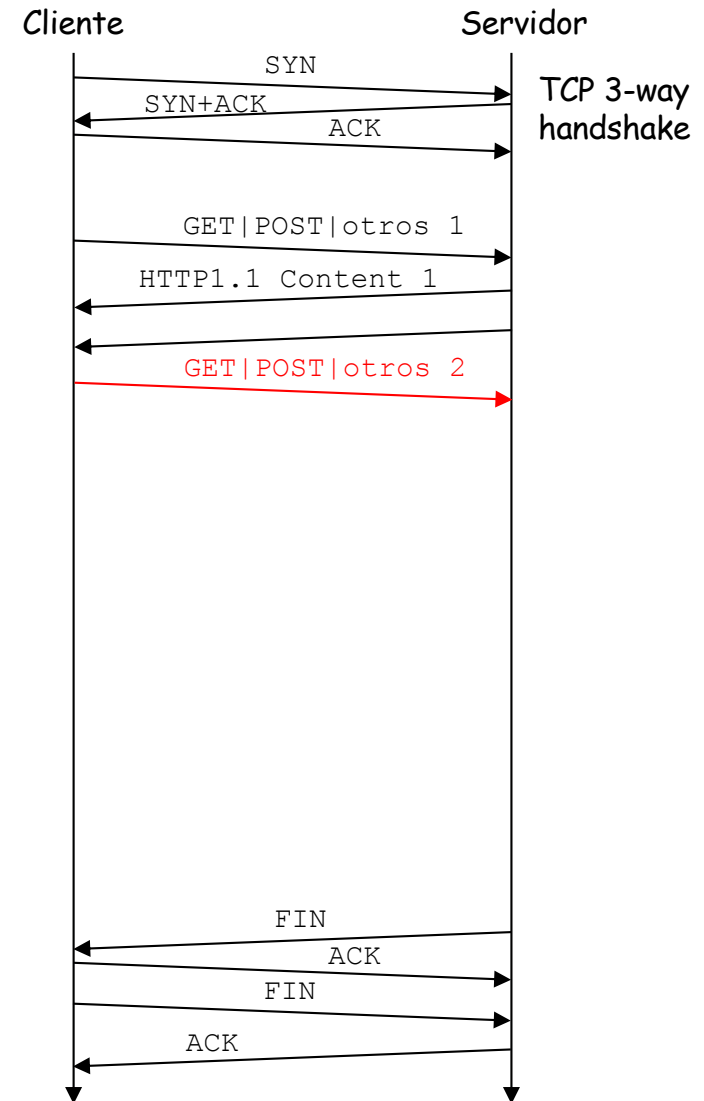
HTTP 1.1

- Conexiones persistentes
- Pipelining (...)



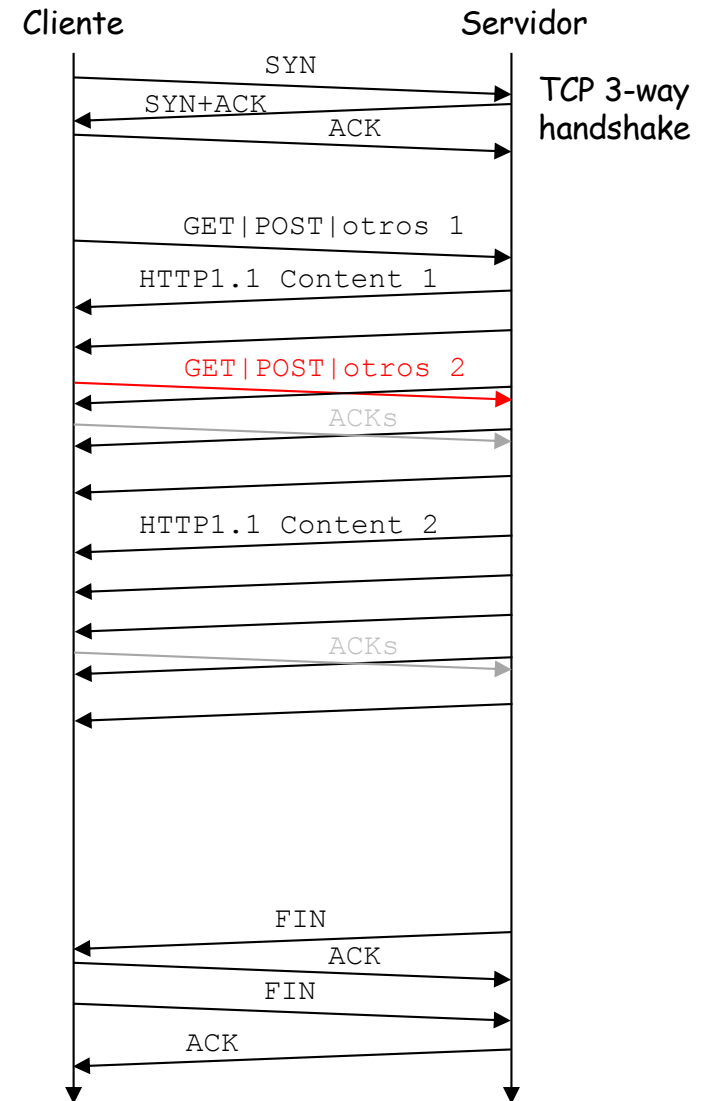
HTTP 1.1

- Conexiones persistentes
- Pipelining (...)



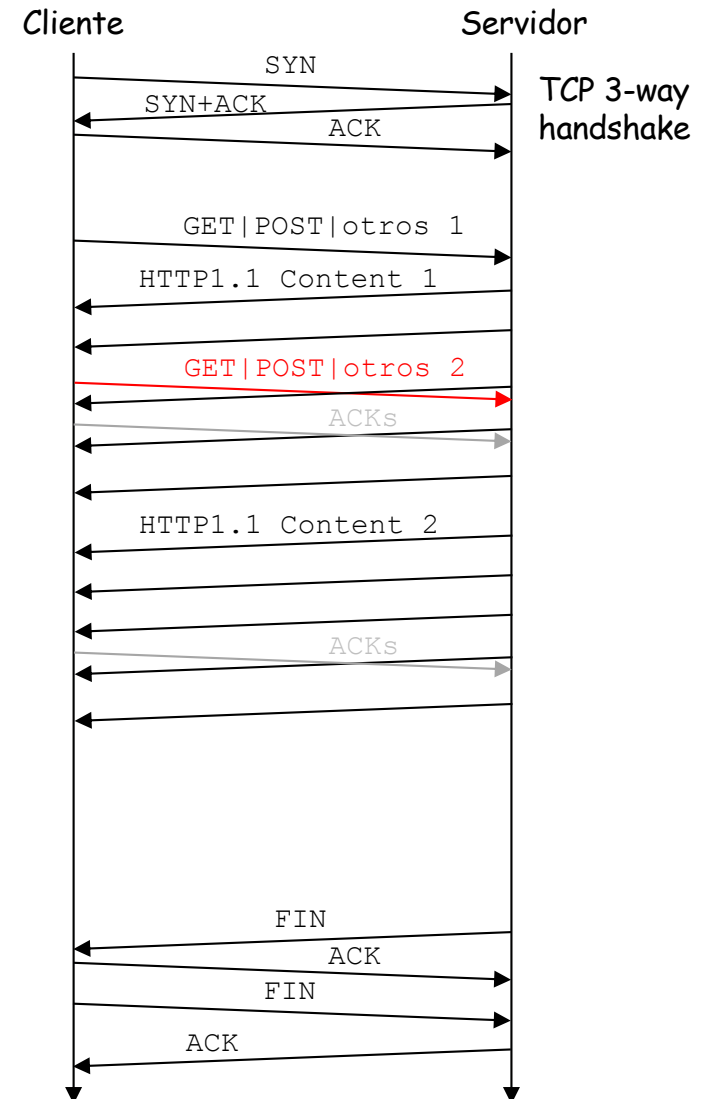
HTTP 1.1

- Conexiones persistentes
- Pipelining: respuestas en el mismo orden que las peticiones



HTTP 1.1

- Conexiones persistentes
- Pipelining: respuestas en el mismo orden que las peticiones
- Posible entrega de rangos de bytes
- RFC original 2616 (junio 1999)
- RFC 9112: HTTP/1.1 (junio 2022) deja obsoleta la RFC 7230 (y ésta la 2616)



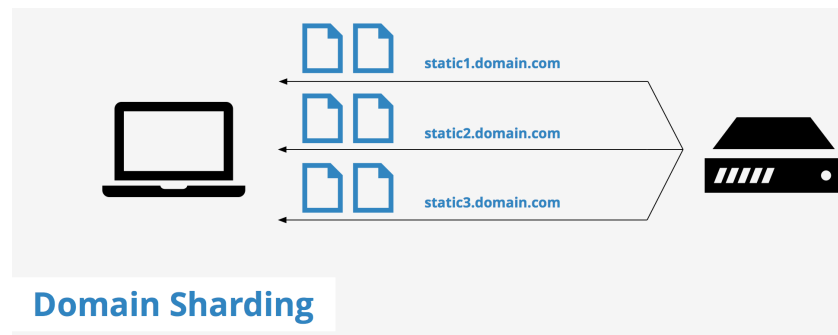
HTTP1.1

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



HTTP1.1

- 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

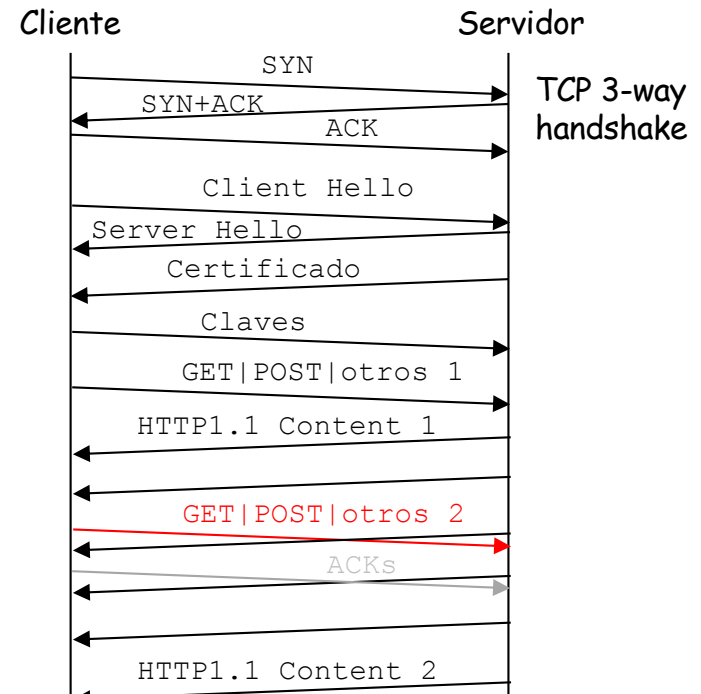
- Y algunos “hacks” de navegadores y desarrolladores
 - Conexiones en paralelo (limitadas a 5-6 para un servidor)
 - *Domain sharding* (recursos en diferentes dominios)
 - Unión de ficheros pequeños (CSS, Javascript, imágenes)
 - Inline de ficheros con el HTML
- Conexiones en paralelo y domain sharding crean mayor número de sockets, más conexiones en NATs, más DNS
- Concatenación e inlining rompe la modularidad y entorpece las caches

HTTP1.1 - Problemas

- 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

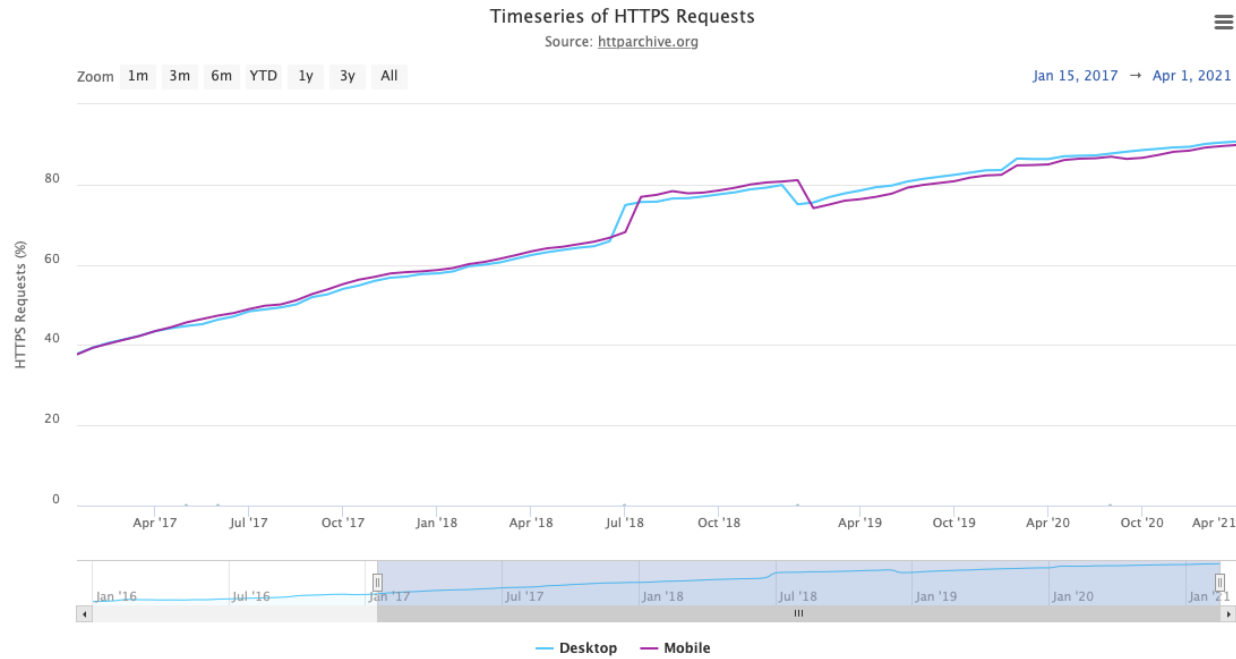
HTTP 1.1 + TLS

- 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 TSecr=
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
12	0.392847	10.6.4.40	108.174.11.37	151	Message SETTINGS[0], WINDOW_UPDATE[0]

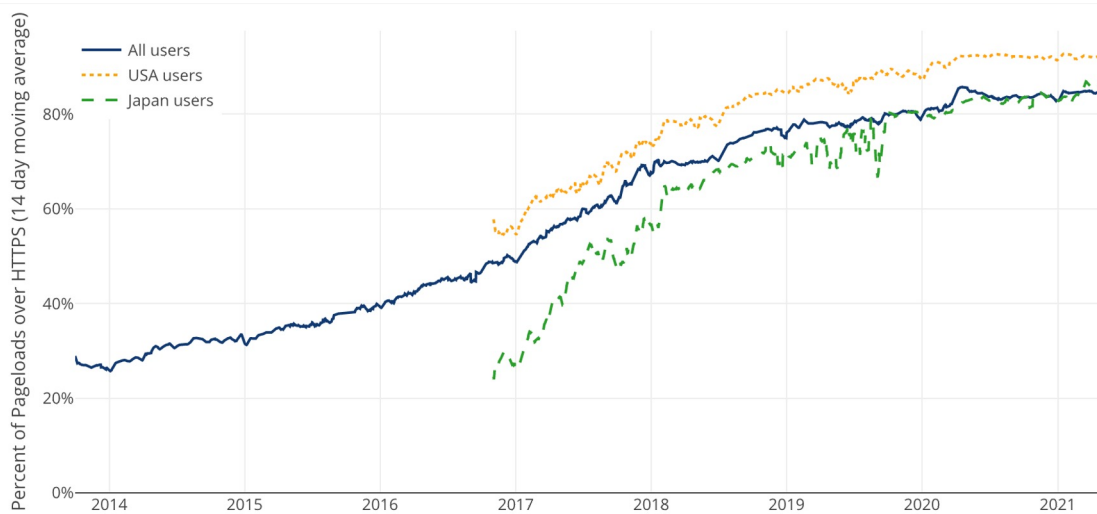
HTTP 1.1 + TLS



(Puede ser H2)

Percentage of Web Pages Loaded by Firefox Using HTTPS

(14-day moving average, source: Firefox Telemetry)



HTTP 1.1 – Conexiones persistentes y pipelining

upna

Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Redes de Nueva Generación
Área de Ingeniería Telemática

HTTP/2

HTTP/2 - Características

HTTP/2 (H2)

- RFC 9113 “HTTP/2” (Mozilla, Apple, 2022), deja obsoleta la 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

```
> Frame 6: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
> Ethernet II, Src: JuniperN_96:8d:7a (00:24:dc:96:8d:7a), Dst: Cisco_9b:d6:03 (00:0c:29:9b:d6:03)
> Internet Protocol Version 4, Src: 91.189.88.162, Dst: 130.206.158.178
> Transmission Control Protocol, Src Port: 80, Dst Port: 22591, Seq: 1, Ack: 22591, Win: 0, Len: 0
```

```
0000 00 41 d2 9b d6 ef 00 24 dc 96 8d 7a 08 00 45 48  .A....$...z..EH
0010 05 dc 1b 99 40 00 33 06 50 5b 5b bd 58 a2 82 ce  ...@.3. P[[.X...
0020 9e b2 00 50 58 3f 8e a8 0c a4 63 30 43 a9 80 10  ...PX?...c0C...
0030 00 ed 47 74 00 00 01 01 08 0a 6e 29 e3 64 cc df  ..Gt....n)d...
0040 fc fe 48 54 54 50 2f 31 2e 31 20 32 30 30 20 4f  ..HTTP/1 .1 200 0
0050 4b 0d 0a 44 61 74 65 3a 20 54 68 75 2c 20 32 34  K.Date: Thu, 24
0060 20 4a 61 6e 20 32 30 31 39 20 31 33 3a 33 32 3a  Jan 201 9 13:32:
0070 30 38 20 47 4d 54 0d 0a 53 65 72 76 65 72 3a 20  08 GMT Server:
0080 41 70 61 63 68 65 2f 32 2e 34 2e 31 38 20 28 55  Apache/2 .4.18 (U
0090 62 75 6e 74 75 29 0d 0a 4c 61 73 74 2d 4d 6f 64  buntu).. Last-Mod
00a0 69 66 69 65 64 3a 20 57 65 64 2c 20 32 33 20 4a  ified: Wed, 23 J
00b0 61 6e 20 32 30 31 39 20 31 35 3a 35 38 3a 35 36  an 2019 15:58:56
00c0 20 47 4d 54 0d 0a 45 54 61 67 3a 20 22 61 37 65  GMT ET ag: "a7e
00d0 38 2d 35 38 30 32 32 32 62 64 32 35 30 30 30 22  8-580222 bd25000"
00e0 0d 0a 41 63 63 65 70 74 2d 52 61 6e 67 65 73 3a  ..Accept-Ranges:
00f0 20 62 79 74 65 73 0d 0a 43 6f 6e 74 65 6e 74 2d  bytes Content-
0100 4c 65 6e 67 74 68 3a 20 34 32 39 38 34 0d 0a 43  Length: 42984..C
0110 61 63 68 65 2d 43 6f 6e 74 72 6f 6c 3a 20 6d 61  ache-Con trol: ma
0120 78 2d 61 67 65 3d 38 36 34 30 30 0d 0a 43 6f 6e  x-age=86 400..Con
0130 74 65 6e 74 2d 54 79 70 65 3a 20 61 70 70 6c 69  tent-Typ e: appli
0140 63 61 74 69 6f 6e 2f 78 2d 64 65 62 69 61 6e 2d  cation/x -debian-
0150 70 61 63 6b 61 67 65 0d 0a 0d 0a 21 3c 61 72 63  package...!<arc
0160 68 3e 0a 64 65 62 69 61 6e 2d 62 69 6e 61 72 79  h>..debia n-binary
```

```
> Frame 12: 5382 bytes on wire (43056 bits), 5382 bytes captured (43056 bits) on interface 0
> Ethernet II, Src: PcsCompu_ab:17:f2 (08:00:27:ab:17:f2), Dst: 0a:00:27:00:00:00
> Internet Protocol Version 4, Src: 192.168.56.101, Dst: 192.168.56.1
> Transmission Control Protocol, Src Port: 443, Dst Port: 44134, Seq: 1319, Ack: 44134, Win: 0, Len: 0
> Transport Layer Security
```

```
> [2 Reassembled TLS segments (1300 bytes): #12(1127), #12(173)]
> [2 Reassembled TLS segments (1300 bytes): #12(1127), #12(173)]
> [2 Reassembled TLS segments (1300 bytes): #12(1127), #12(173)]
< HyperText Transfer Protocol 2
  > Stream: SETTINGS, Stream ID: 0, Length 6
  > Stream: SETTINGS, Stream ID: 0, Length 0
  > Stream: WINDOW_UPDATE, Stream ID: 0, Length 4
  > Stream: HEADERS, Stream ID: 1, Length 127, 200 OK
< HyperText Transfer Protocol 2
  > Stream: DATA, Stream ID: 1, Length 1291 (partial entity body)
< HyperText Transfer Protocol 2
  > Stream: DATA, Stream ID: 1, Length 1291 (partial entity body)
```

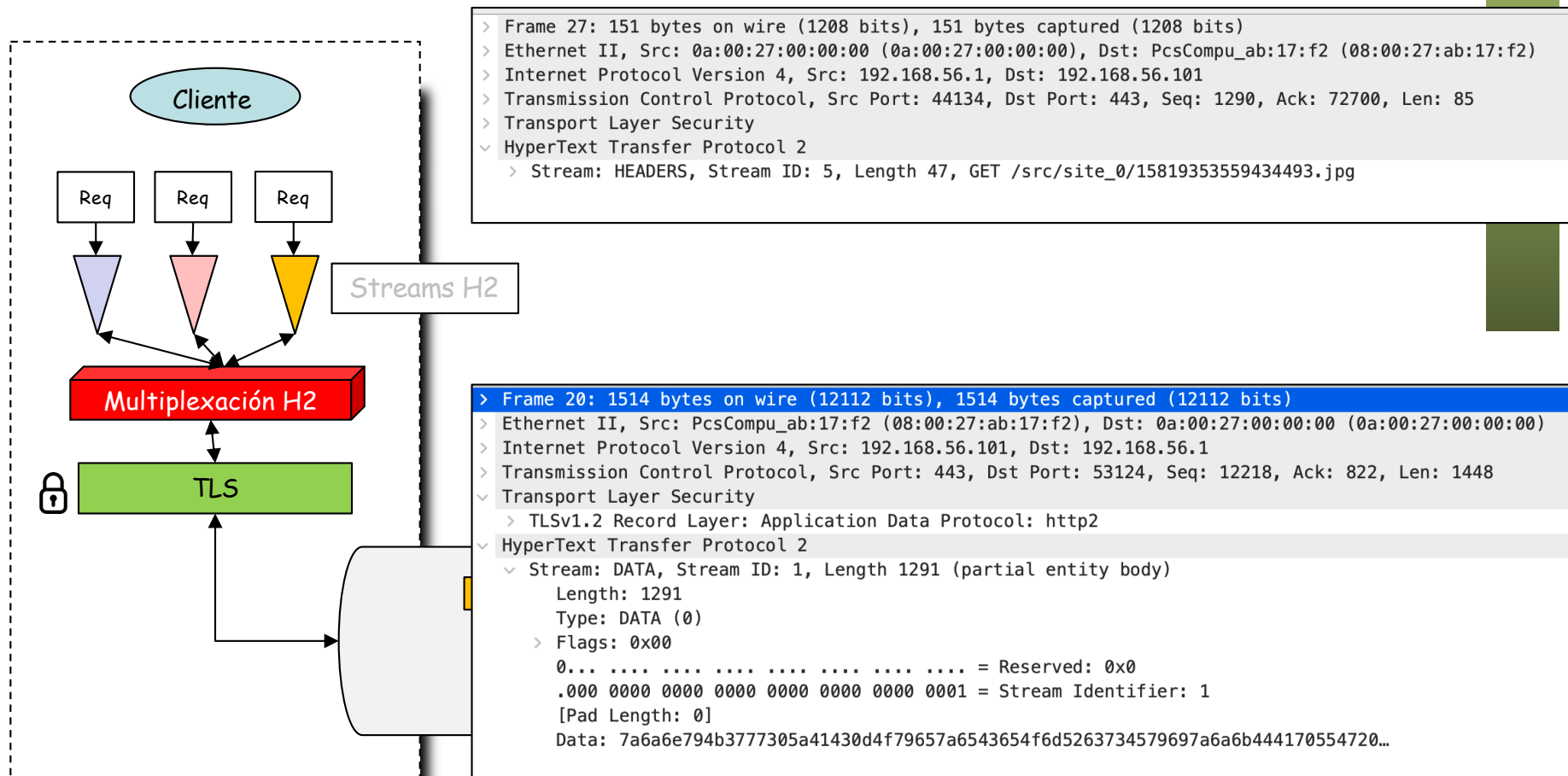
```
0000 0a 00 27 00 00 00 08 00 27 ab 17 f2 08 00 45 00  ...'.....'.....E.
0010 14 f8 d0 a2 40 00 40 06 63 a6 c0 a8 38 65 c0 a8  ...@.@.c...8e...
0020 38 01 01 bb ac 66 ac 0e cd 31 b8 2d 9f 6b 80 18  8...f...1...k...
0030 01 f7 06 a2 00 00 01 01 08 0a 85 69 42 11 e7 42  ...iB..B
0040 cf 48 17 03 03 05 2c 58 02 0f af 6b 3a 69 05 e9  .H...X...k:i...
0050 41 15 4c e3 46 d3 20 e1 e5 41 5e d4 80 ed 41 90  A.L.F...A^..A...
0060 99 be d9 5c 3e 7e 19 74 30 a2 66 33 61 bb e4 f2  ...>...t 0.f3a...
0070 a8 dd 9f b1 14 cc 32 75 af b8 03 04 b1 e7 8c 4e  ...2u.....N
0080 d9 9a ac 74 06 4a 32 83 6d 4a 0c b1 04 ee 0a 1c  ...t.J2..mJ.....
```

HTTP/2 (H2)

- RFC 9113 “HTTP/2” (Mozilla, Apple, 2022), deja obsoleta la 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

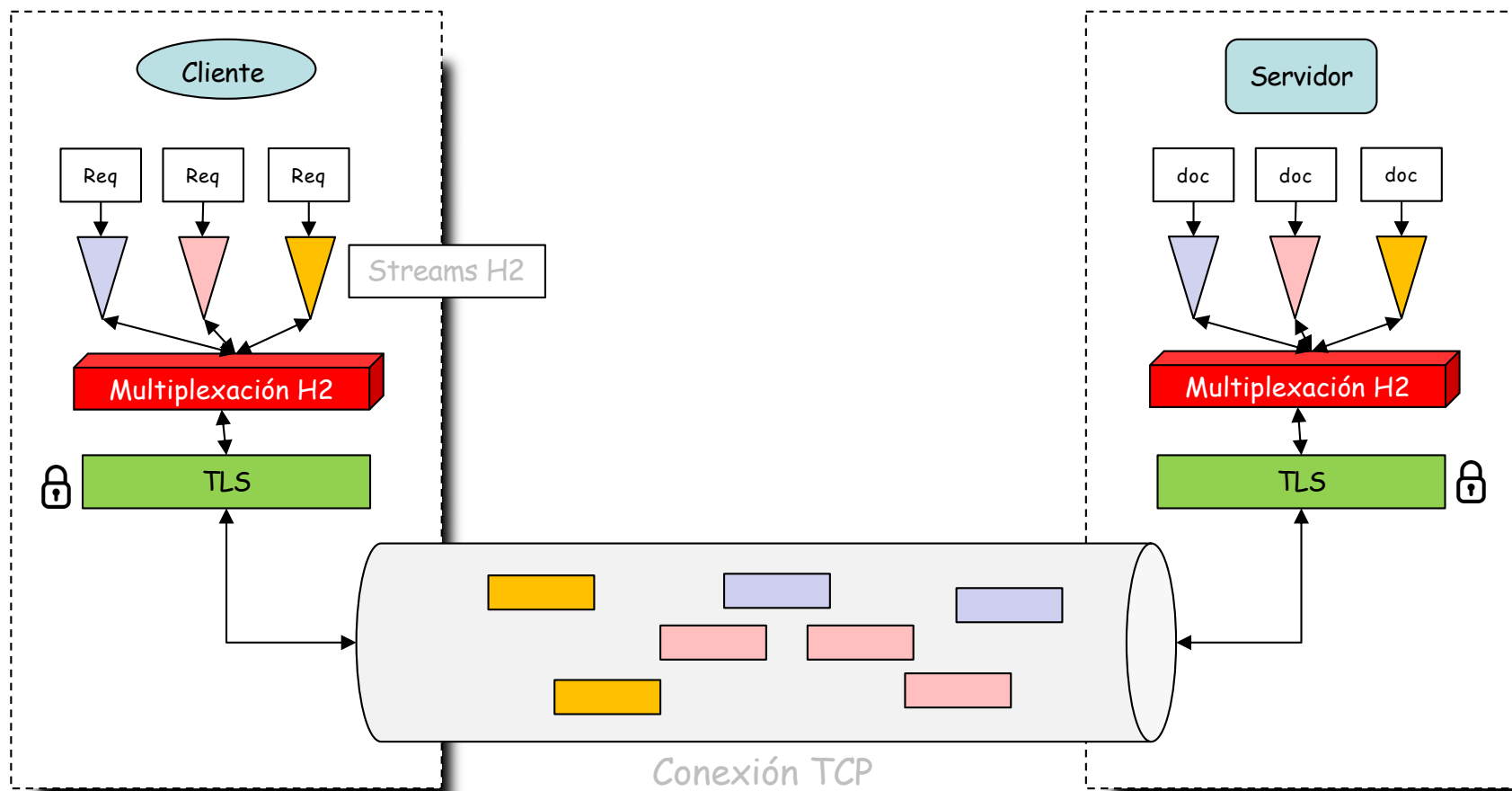
Multiplexación en H2

- Frames (nombre en H2) 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)



Multiplexación en H2

- Frames (nombre en H2) 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: 1f8b080000000000cd5ae98ee336127e156f06013241d8b0bddd8389849d3ffb1883...
 - ▶ Stream: DATA, Stream ID: 3, Length 0

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*)

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
 - Obsoleto desde RFC 9113 (2022)

```
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

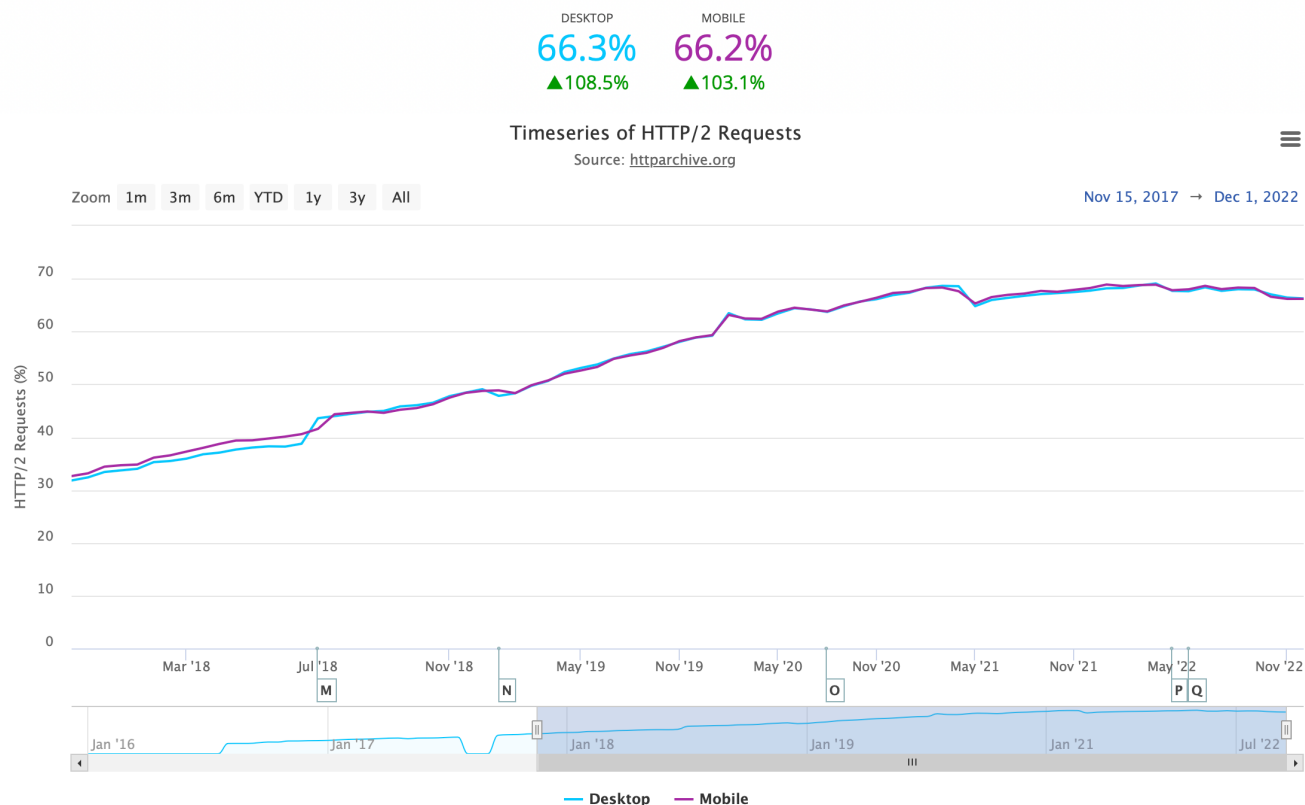
- Flow Control
 - Para cada *stream*
 - HTTP/2 no fuerza a un algoritmo para el control de flujo, solo lo soporta
- Negociación
 - Mediante mensajes se puede subir una conexión de HTTP1.1 a HTTP/2
 - Hoy en día solo se está usando HTTP/2 sobre TLS
 - Negociación mediante el ALPN en TLS

```
Transmission Control Protocol, Src Port: 45782, Dst Port: 443, Seq: 1, Ack: 1, Len: 517
Transport Layer Security
  TLSv1.2 Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 512
  Handshake Protocol: Client Hello
    Handshake Type: Client Hello (1)
    Length: 508
    Version: TLS 1.2 (0x0303)
    Random: 4232b9f46fd109bc090b941ecf727e14a3584d269a408fcf7514d8af2847d8
    Session ID Length: 32
    Session ID: fd8622edb4e61e53991a279dfd553827984a4b1941ea0eef9d18d62e75a1cbcd
    Cipher Suites Length: 32
    Cipher Suites (16 suites)
    Compression Methods Length: 1
    Compression Methods (1 method)
    Extensions Length: 403
    Extension: Reserved (GREASE) (len=0)
    Extension: server_name (len=24)
    Extension: extended_master_secret (len=0)
    Extension: renegotiation_info (len=1)
    Extension: supported_groups (len=10)
    Extension: ec_point_formats (len=2)
    Extension: session_ticket (len=0)
  Extension: application_layer_protocol_negotiation (len=14)
    Type: application_layer_protocol_negotiation (16)
    Length: 14
    ALPN Extension Length: 12
  ALPN Protocol
    ALPN string length: 2
    ALPN Next Protocol: h2
    ALPN string length: 8
    ALPN Next Protocol: http/1.1
  Extension: status_request (len=5)
  Extension: signature_algorithms (len=18)
```

```
Frame 6: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits)
Ethernet II, Src: Cisco_dc:71:c4 (30:e4:db:dc:71:c4), Dst: Universa_2c:dc:6c (00:1e:37
Internet Protocol Version 4, Src: 108.174.11.37, Dst: 10.6.4.40
Transmission Control Protocol, Src Port: 443, Dst Port: 45782, Seq: 1, Ack: 518, Len:
Transport Layer Security
  TLSv1.2 Record Layer: Handshake Protocol: Server Hello
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 106
  Handshake Protocol: Server Hello
    Handshake Type: Server Hello (2)
    Length: 102
    Version: TLS 1.2 (0x0303)
    Random: 331342a38cec5db5f136683375f91bf62cc515ff2d5eeb65fe820e9d03c4281b
    Session ID Length: 32
    Session ID: 714f39799dd6b7b43998b1c41d9e1209a6fe7514f7de012408ed9472cbf179e3
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
    Compression Method: null (0)
    Extensions Length: 30
    Extension: renegotiation_info (len=1)
    Extension: server_name (len=0)
    Extension: ec_point_formats (len=4)
  Extension: application_layer_protocol_negotiation (len=5)
    Type: application_layer_protocol_negotiation (16)
    Length: 5
    ALPN Extension Length: 3
  ALPN Protocol
    ALPN string length: 2
    ALPN Next Protocol: h2
  Extension: extended_master_secret (len=0)
```

HTTP/2: Status

- RFC aprobada (RFC7540 de 2015, *deprecated* por la 9113 en 2022)
- Múltiples implementaciones, en navegadores y servidores
- Soportado desde 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 39.8% de websites¹



<https://httparchive.org/reports/state-of-the-web#h2>

¹ <https://w3techs.com/technologies/details/ce-http2/all/all> (Enero 2023)

HTTP1.1 – HTTP/2

- Conexiones en paralelo
- *Domain sharding* (recursos en diferentes dominios)
- Unión de ficheros pequeños (CSS, Javascript, imágenes)
- Inline de ficheros con el HTML (imágenes)

HTTP1.1 – HTTP/2

- Conexiones en paralelo
 - Consume más recursos en cliente, servidor, NATs, firewalls, etc
 - Acelera descarga por evitar HoL blocking
 - En H2 la multiplexación de streams resuelve el HoL blocking
- *Domain sharding* (recursos en diferentes dominios)
 - (...)
- Unión de ficheros pequeños (CSS, Javascript, imágenes)
- Inline de ficheros con el HTML (imágenes)

HTTP1.1 – HTTP/2

- Conexiones en paralelo
 - Consume más recursos en cliente, servidor, NATs, firewalls, etc
 - Acelera descarga por evitar HoL blocking
 - En H2 la multiplexación de streams resuelve el HoL blocking
- *Domain sharding* (recursos en diferentes dominios)
 - Era para aumentar las conexiones en paralelo pero ahora están los streams
- Unión de ficheros pequeños (CSS, Javascript, imágenes)
 - (...)
- Inline de ficheros con el HTML (imágenes)

HTTP1.1 – HTTP/2

- Conexiones en paralelo
 - Consume más recursos en cliente, servidor, NATs, firewalls, etc
 - Acelera descarga por evitar HoL blocking
 - En H2 la multiplexación de streams resuelve el HoL blocking
- *Domain sharding* (recursos en diferentes dominios)
 - Era para aumentar las conexiones en paralelo pero ahora están los streams
- Unión de ficheros pequeños (CSS, Javascript, imágenes)
 - Evitaba múltiples peticiones con problema HoL que ya está resuelto en H2
- Inline de ficheros con el HTML (imágenes)
 - (...)

HTTP1.1 – HTTP/2

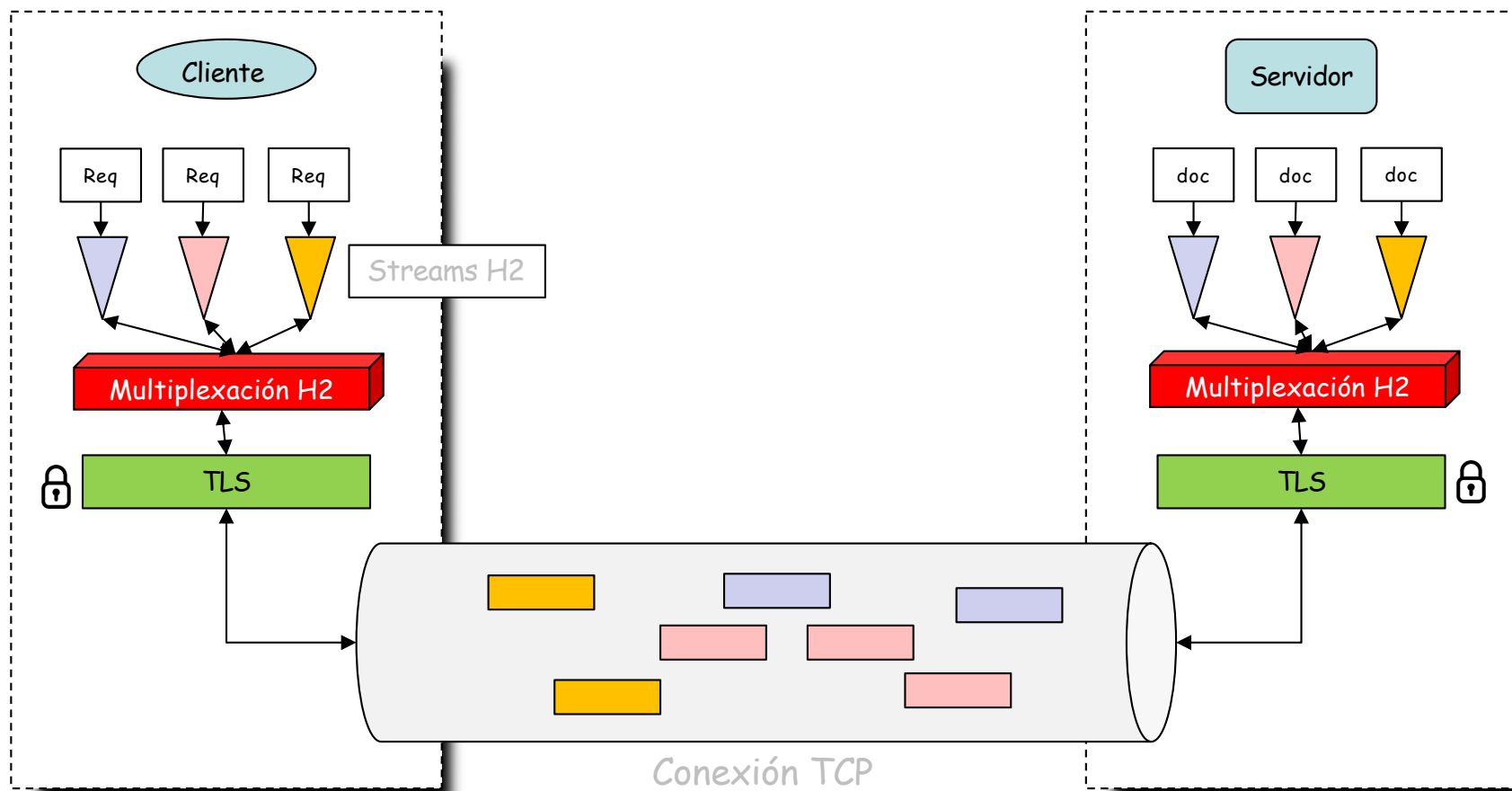
- Conexiones en paralelo
 - Consume más recursos en cliente, servidor, NATs, firewalls, etc
 - Acelera descarga por evitar HoL blocking
 - En H2 la multiplexación de streams resuelve el HoL blocking
- *Domain sharding* (recursos en diferentes dominios)
 - Era para aumentar las conexiones en paralelo pero ahora están los streams
- Unión de ficheros pequeños (CSS, Javascript, imágenes)
 - Evitaba múltiples peticiones con problema HoL que ya está resuelto en H2
- Inline de ficheros con el HTML (imágenes)
 - Evitaba tener que pedir ese recurso
 - Ahora se puede “empujar” (*push*) desde el servidor
- Otras mejoras:
 - Binario
 - Compresión de cabeceras
 - Control de flujo por stream
 - Prioridades

HTTP/2 - Características

HTTP/2 - Limitaciones

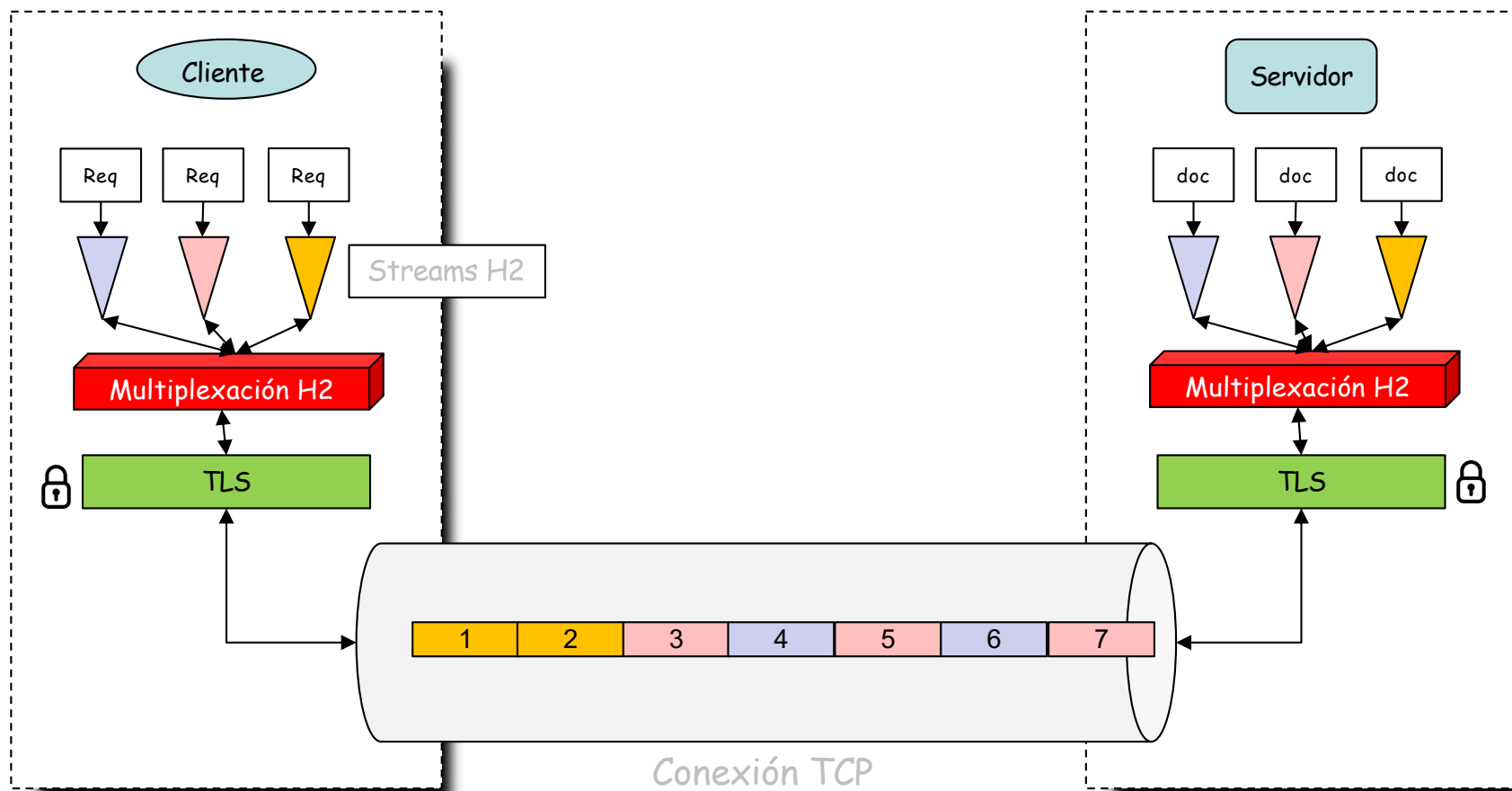
HTTP/2 (H2)

- Multiplexación de streams elimina HOL blocking
- ¿De verdad?



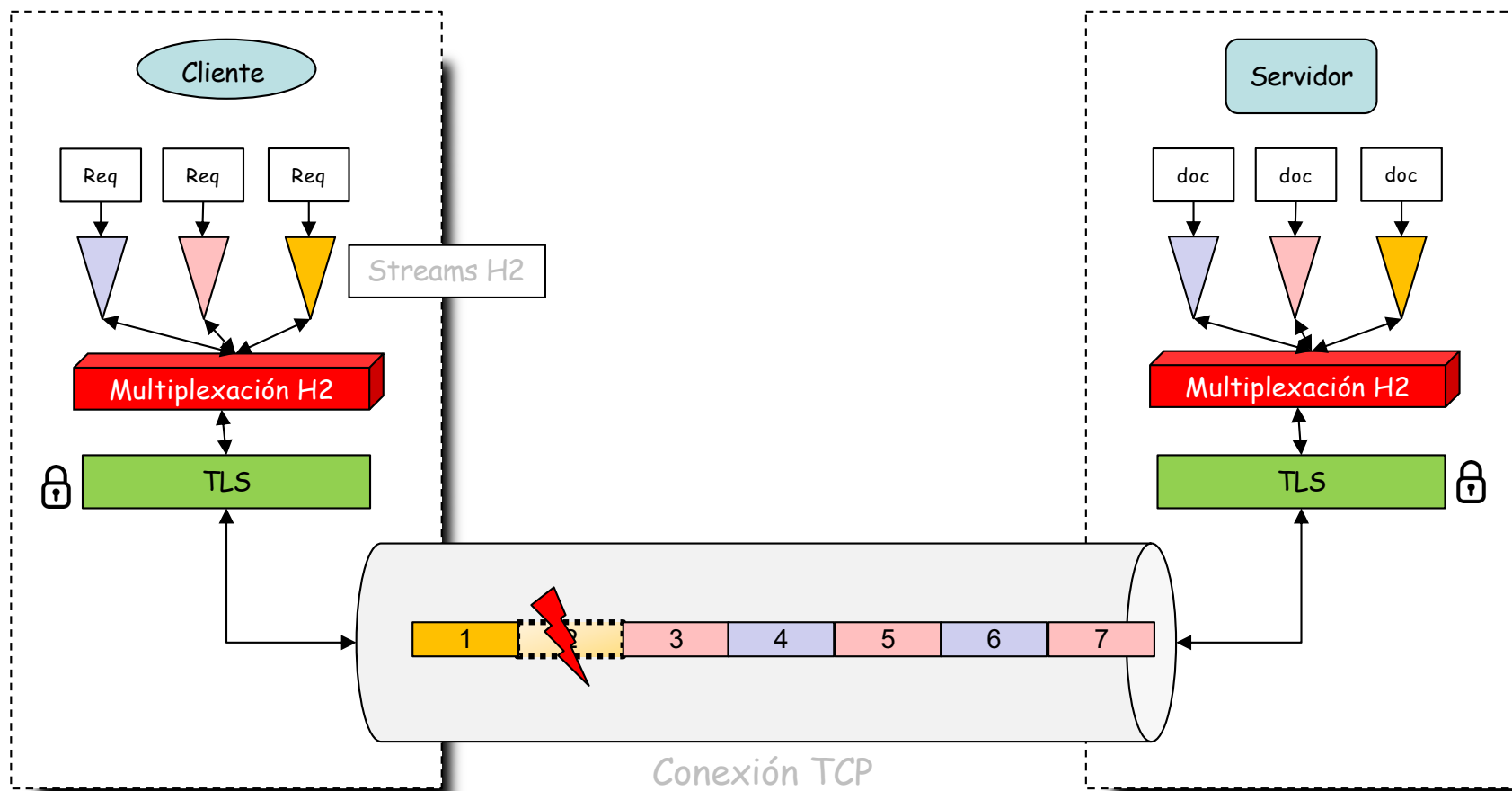
HTTP/2 (H2)

- Emplea una conexión TCP, con 2 streams (uno en cada sentido)
- Cada stream es un flujo bytes ordenados
- (...)



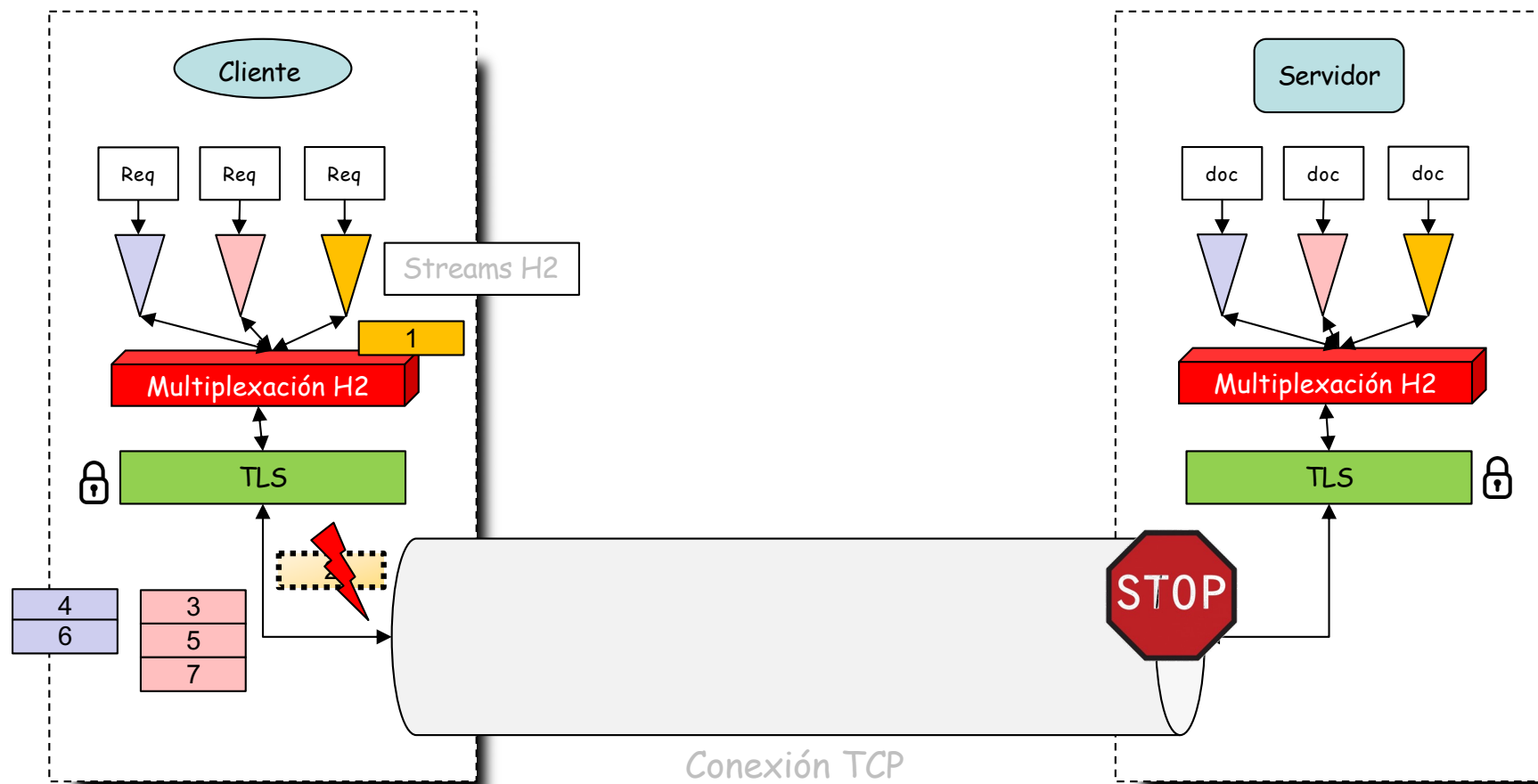
HTTP/2 (H2)

- Emplea una conexión TCP, con 2 streams (uno en cada sentido)
- Cada stream es un flujo bytes ordenados
- ¿Qué sucede si hay una pérdida?
- (...)



HTTP/2 (H2)

- No se entrega a la aplicación el resto de datos hasta “rellenar el hueco”
- La pérdida ha podido afectar solo a paquetes de un stream h2
- Pero afecta a todo el stream TCP y por lo tanto a todos los streams h2
- Además detiene todo el flujo (retransmisiones y control de congestión)
- Resuelto HOL blocking en nivel de aplicación pero no de transporte



Otras limitaciones

- Demasiados RTTs
- Tiempo de establecimiento de la conexión TCP
- Tiempo de establecimiento de la sesión TLS

RTTs con TCP

- 1 RTT establecimiento de la conexión TCP
- Aunque tenemos TCP Fast Open (RFC 7413)
 - Permite entregar a la aplicación datos que llegan con el SYN
 - Pero tiene escaso despliegue (modifica TCP y problemas con middleboxes)
- (...)

RTTs con TCP

- 2 RTTs para establecer la sesión TLS
- Aunque tenemos
 - Sesiones
 - Tickets
 - 0-RTT con TLS 1.3 (Early data)

Sesiones

No.	Time	Source	Destination	Info
34	24.1546...	192.168.1.101	192.168.1.48	51068 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=166072007
35	24.1546...	192.168.1.48	192.168.1.101	443 → 51068 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=
36	24.1548...	192.168.1.101	192.168.1.48	51068 → 443 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=1660720071 TSecr=
37	24.1557...	192.168.1.101	192.168.1.48	Client Hello
38	24.1557...	192.168.1.48	192.168.1.101	443 → 51068 [ACK] Seq=1 Ack=518 Win=30080 Len=0 TSval=2312132065 TSecr=
39	24.1563...	192.168.1.48	192.168.1.101	Server Hello, Change Cipher Spec, Finished
40	24.1564...	192.168.1.101	192.168.1.48	51068 → 443 [ACK] Seq=518 Ack=181 Win=131584 Len=0 TSval=1660720072 TS
41	24.1566...	192.168.1.101	192.168.1.48	Change Cipher Spec, Finished
42	24.1578...	192.168.1.101	192.168.1.48	GET / HTTP/1.1
43	24.1578...	192.168.1.48	192.168.1.101	443 → 51068 [ACK] Seq=181 Ack=1126 Win=31104 Len=0 TSval=2312132067 TS
44	24.1584...	192.168.1.48	192.168.1.101	HTTP/1.1 200 OK (text/html)
45	24.1586...	192.168.1.101	192.168.1.48	51068 → 443 [ACK] Seq=1126 Ack=3077 Win=128640 Len=0 TSval=1660720074
46	24.1586...	192.168.1.101	192.168.1.48	51068 → 443 [ACK] Seq=1126 Ack=3812 Win=127936 Len=0 TSval=1660720074

Early Data

No.	Time	Source	Destination	Info
1	0.000000	10.20.0.19	104.17.143.23	51052 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSV
2	0.015794	104.17.143.23	10.20.0.19	443 → 51052 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1400 SACK
3	0.015811	10.20.0.19	104.17.143.23	51052 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0
4	0.017129	10.20.0.19	104.17.143.23	Client Hello
5	0.017284	10.20.0.19	104.17.143.23	Change Cipher Spec
6	0.017343	10.20.0.19	104.17.143.23	GET /v4/assets/sidebar-lightarrow.svg HTTP/1.1
7	0.033849	104.17.143.23	10.20.0.19	443 → 51052 [ACK] Seq=1 Ack=598 Win=67584 Len=0
8	0.033859	104.17.143.23	10.20.0.19	443 → 51052 [ACK] Seq=1 Ack=604 Win=67584 Len=0
9	0.033861	104.17.143.23	10.20.0.19	443 → 51052 [ACK] Seq=1 Ack=1037 Win=68608 Len=0
10	0.038601	104.17.143.23	10.20.0.19	Server Hello, Change Cipher Spec, Encrypted Extensions, Finished
11	0.038616	10.20.0.19	104.17.143.23	51052 → 443 [ACK] Seq=1037 Ack=689 Win=64128 Len=0
12	0.039258	10.20.0.19	104.17.143.23	End of Early Data, Finished
13	0.053534	104.17.143.23	10.20.0.19	[TLS segment of a reassembled PDU]
14	0.053547	104.17.143.23	10.20.0.19	HTTP/1.1 200 OK

Otras limitaciones

- Demasiados RTTs
- Tiempo de establecimiento de la conexión TCP
- Tiempo de establecimiento de la sesión TLS
- La aplicación no puede sacar provecho a múltiples interfaces en el host

¿Soluciones?

- Cambiar el nivel de transporte
- RFC 4960 Stream Control Transmission Protocol
- Diseñado para el transporte de la señalización de la red telefónica
- Ofrece:
 - Transporte fiable (acknowledged error-free non-duplicated)
 - Multiplexación de sub-streams
 - Transporte ordenado dentro de cada stream
 - Entrega mensajes en lugar de un byte stream
 - Segmentación
 - Multi-homing
 - Congestion avoidance
 - Flow control



Problemas con SCTP

- Implementarlo en los sistemas operativos de los hosts
- Implementarlo en los NATs
- Diferente API
- HTTP/2: Protocolo de señalización empleado en la red 5G



HTTP/2 - Limitaciones

upna

Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Redes de Nueva Generación
Área de Ingeniería Telemática

HTTP/2