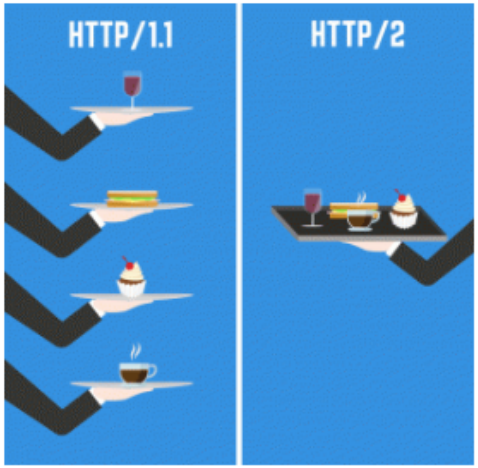


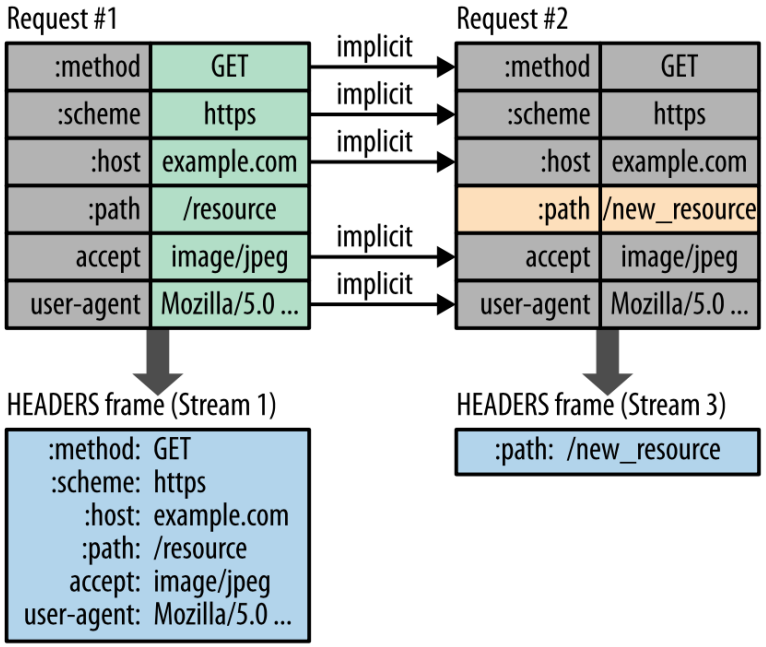
Ampliación Tema 2:
HTTP/2. Conceptos generales

- RFC 7540

- Problemas en HTTP/1.1
  - El formato de los mensajes HTTP se diseñó para su implementación con las herramientas de la década de los 90's, pero no es adecuado para el rendimiento de las aplicaciones web actuales.
  - Recordemos
    - HTTP/1.0: sólo una solicitud por conexión TCP.
    - HTTP/1.1 con pipeline aborda parcialmente la concurrencia de peticiones, pero se puede producir bloqueo en la cabecera de línea.
      - » Recordemos que las repuestas se sirven en serie, ordenadas de acuerdo a las peticiones
      - » Recursos de gran tamaño provocan que los recursos siguientes tarden en servirse, aunque sean de tamaño pequeño
      - » Por ello, se suelen utilizar varias conexiones persistentes sin pipeline.
  - En muchas peticiones las cabeceras son las mismas…..
    - » ¿Para qué repetirlas?
    - » Además, si se pudieran comprimir…..mejor

Demo: https://http2.akamai.com/demo

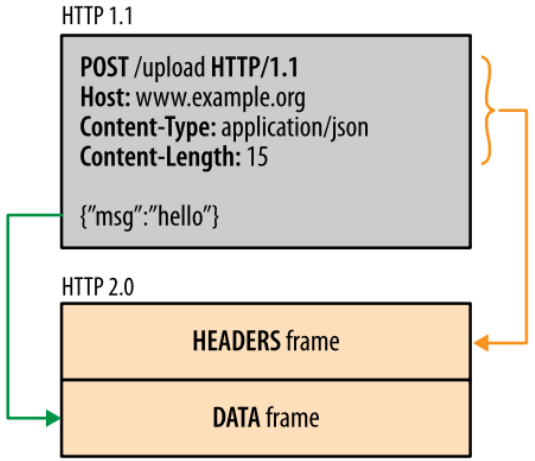
- Compresión cabeceras
  - Cada transferencia HTPP lleva un conjunto de cabeceras
    - Describen el recurso transferido y sus propiedades
      - HTTP 1.x: texto sin formato
      - Agregan entre 500 y 800 bytes por transferencia
        - » Más KB si se usan *cookies*
    - HTTP/2 comprime los datos de las cabeceras de la solicitud y respuesta
      - Evitar duplicados



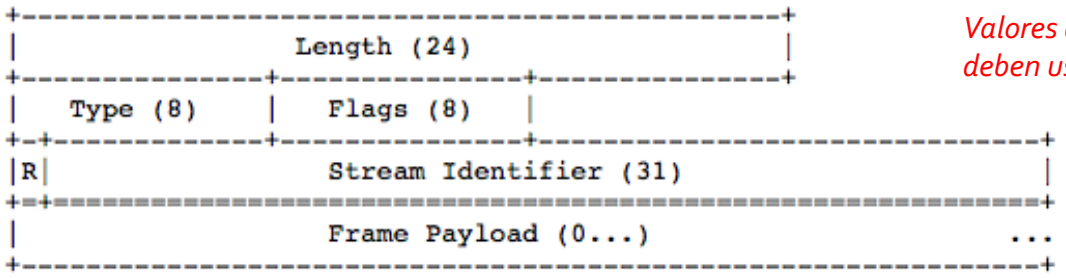
- HTTP/2 soporta todas las características básicas de HTTP/1.1, pero pretende ser más eficiente:
  - Objetivo principal: Reducir la latencia
    - ¿Cómo? Multiplexación completa de solicitudes y respuestas
      - » En lugar de crear múltiples conexiones, se crea una sola conexión para enviar toda la información
        - » Menos competencia con otros flujos y conexiones de mayor duración
          - » Mejor utilización de la capacidad de red disponible
      - » Múltiples solicitudes y respuestas en paralelo
        - » Permite que los mensajes de solicitudes y respuestas se intercalen en la misma conexión
    - Utiliza una codificación eficiente de los campos de cabecera
  - Realiza control de flujo
    - Sólo se transmite información que pueda ser utilizada por el receptor

HTTP/2

- Permite priorizar las solicitudes para mejorar el rendimiento
  - Expresa qué desea el otro extremo para el manejo de la concurrencia (flujos)
- Empleo de *Server push*
  - El servidor se anticipa y envía información antes de que el usuario la solicite
- Formato binario de las tramas
  - HTTP/1.x emplea un protocolo de texto plano delimitado por "línea nueva"
  - Ahora, en HTTP/2 la comunicación se divide en mensajes y tramas más pequeños, todos codificados en formato binario.
    - » Pueden enviarse distintos tipos de tramas y todos tiene el mismo formato



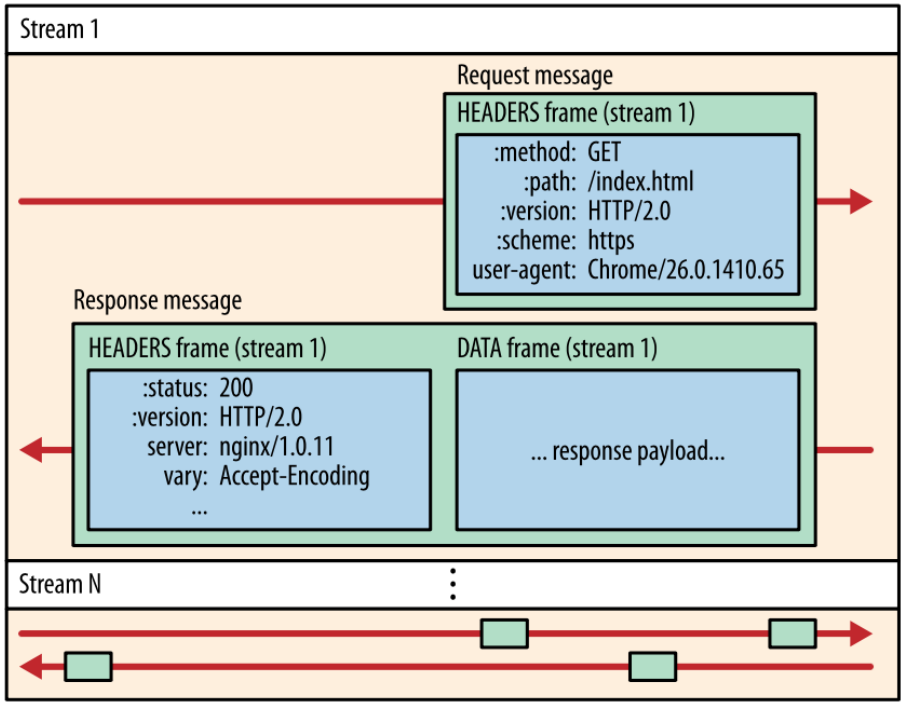
HTTP/2

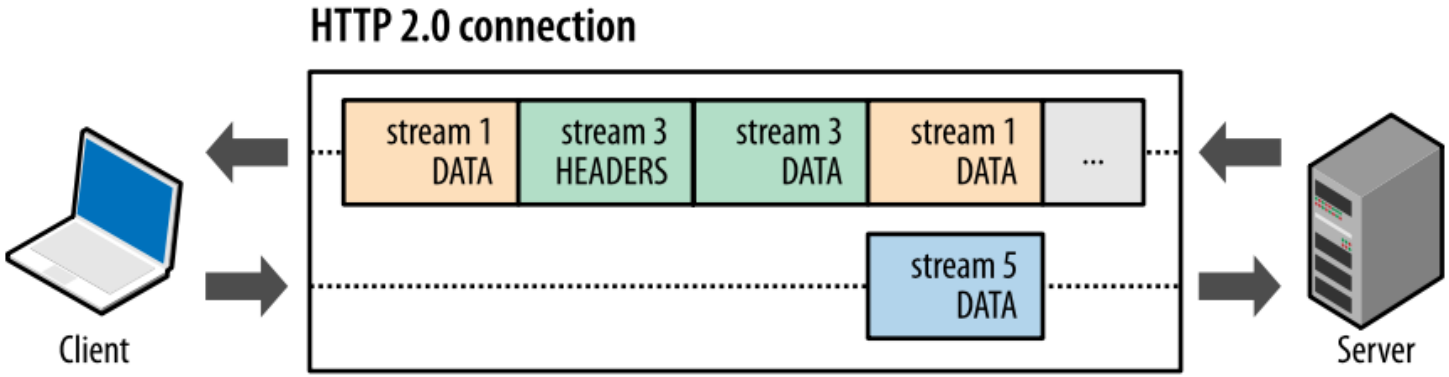


Valores de longitud > 2^{14} (16.384) no se deben usar salvo que el receptor lo indique.

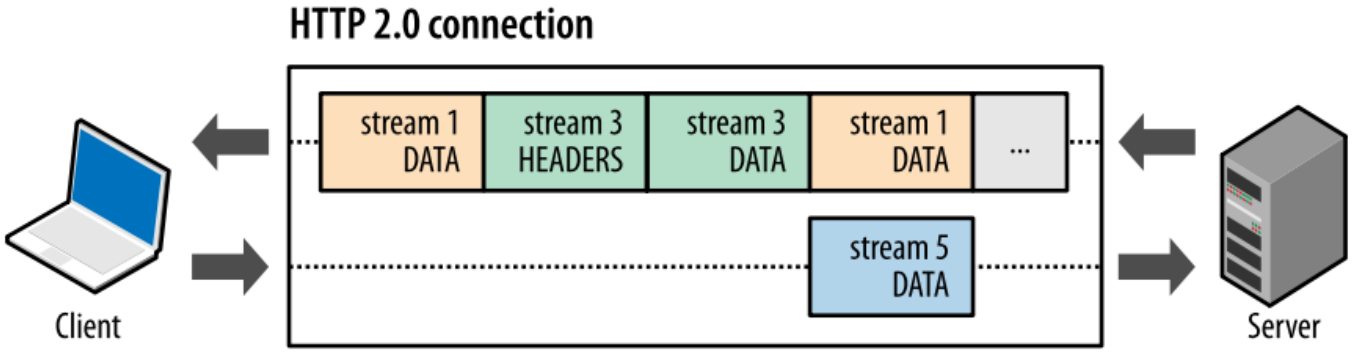
- La unidad básica del protocolo es una trama
  - Cada trama tiene un tipo y propósito distinto (HEADERS, DATA, SETTINGS, WINDOW_UPDATE, PUSH_PROMISE, …)
- Multiplexado para enviar y recibir diversos mensajes al mismo tiempo sobre una misma conexión
  - El multiplexado se logra a través de los **flujos**
    - Un flujo es una secuencia independiente bidireccional de tramas (cada par petición-respuesta se asigna a un flujo)
    - Los flujos son independientes entre sí (no se bloquean entre sí)
    - Se pueden establecer y usar unilateralmente o de forma compartida entre el cliente y el servidor
    - Se identifican por un entero asignado por el extremo que inicia el flujo (clientes inician flujo impares y los servidores, pares)

HTTP/2


Stream 1
Request message
HEADERS frame (stream 1)
:method: GET
:path: /index.html
:version: HTTP/2.0
:scheme: https
user-agent: Chrome/26.0.1410.65
Response message
HEADERS frame (stream 1)
:status: 200
:version: HTTP/2.0
server: nginx/1.0.11
vary: Accept-Encoding
...
DATA frame (stream 1)
... response payload...
Stream N


HTTP 2.0 connection
stream 1 DATA
stream 3 HEADERS
stream 3 DATA
stream 1 DATA
...
stream 5 DATA
Client
Server

- Por tanto:
  - El cliente y servidor desglosan un mensaje HTTP en tramas diferentes, con capacidad de intercalarlas y reensamblarlas en el otro extremo

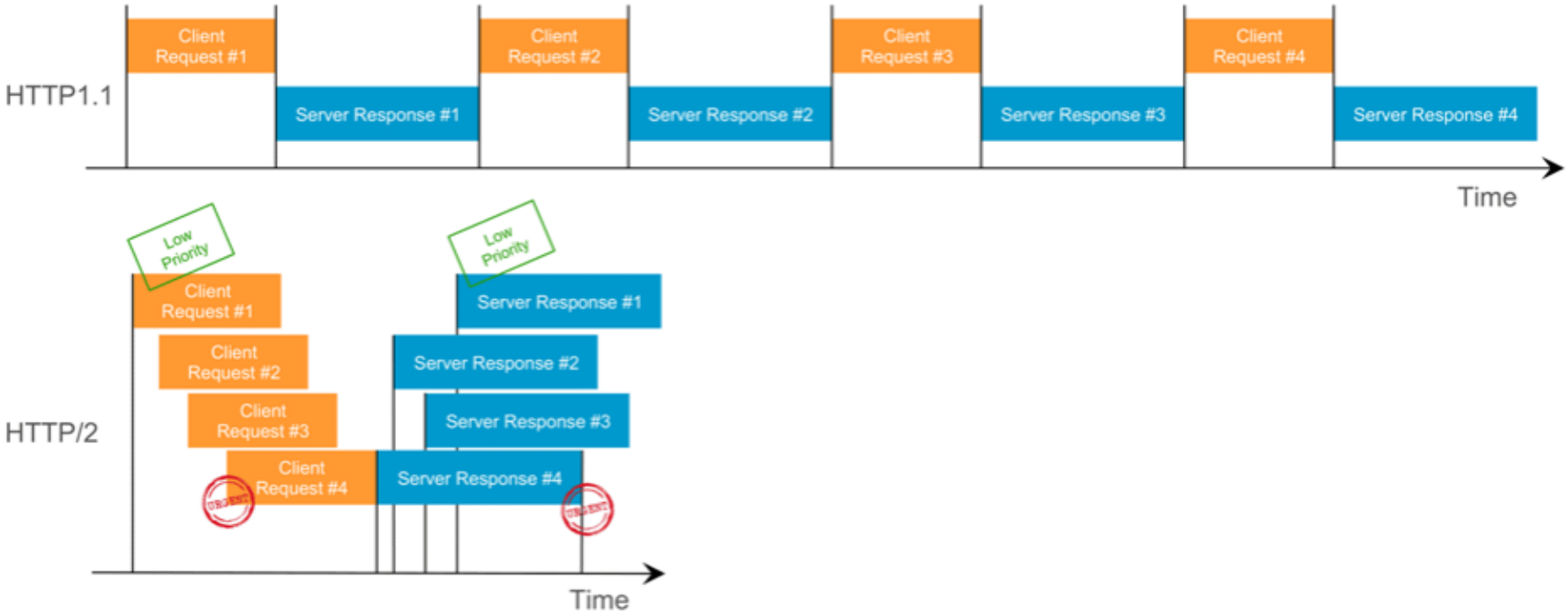


- Ej: el cliente transmite una trama DATA (stream 5) al servidor, mientras éste transmite una secuencia intercalada de tramas al clientes para las transmisiones 1 y 3
  - ➔ 3 transmisiones paralelas

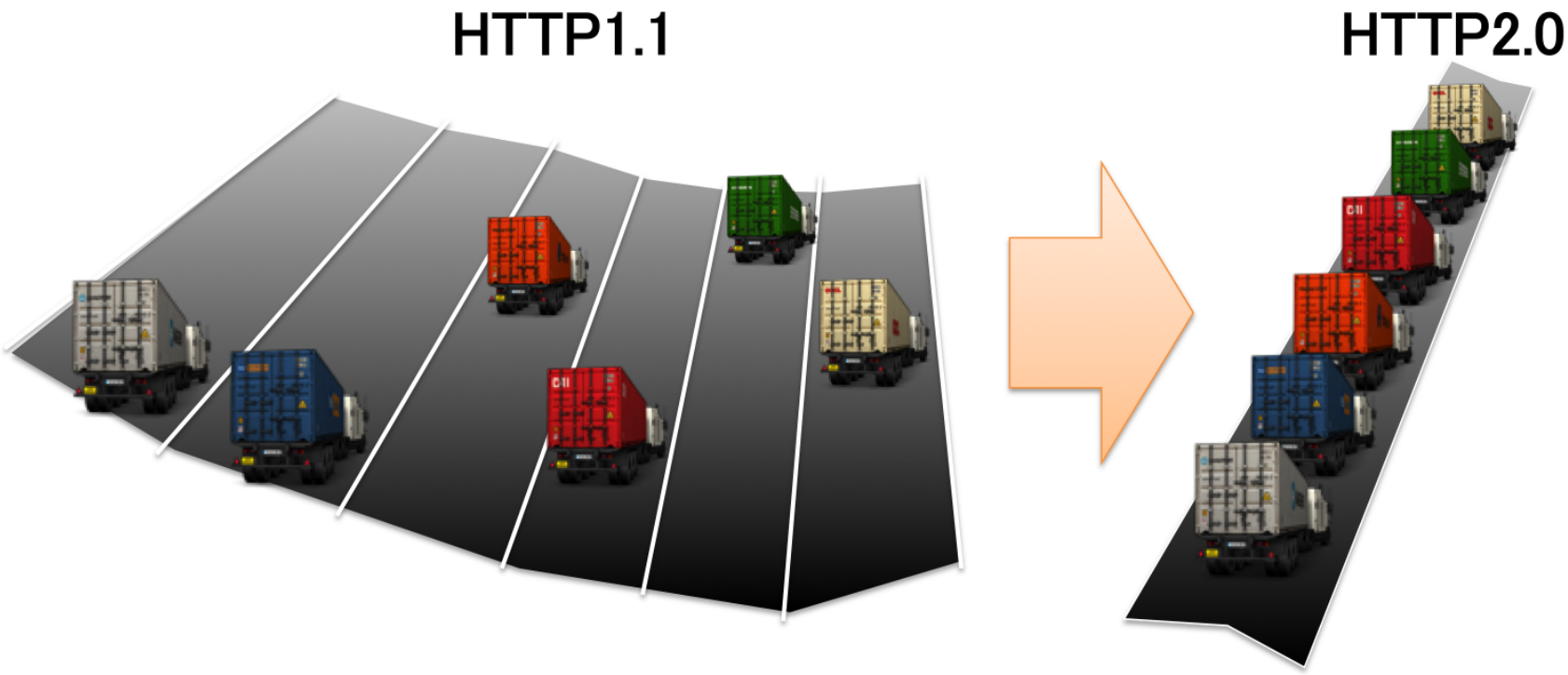
HTTP/2

- Beneficios del desglose de un mensaje HTTP en múltiples tramas independientes:
  - Intercalar múltiples solicitudes en paralelo SIN BLOQUEAR ninguna
  - Intercalar múltiples respuestas en paralelo SIN BLOQUEAR ninguna
  - Usar una única conexión para entregar múltiples solicitudes y respuestas en paralelo
  - Se eliminan métodos alternativos usados en HTTP 1.x
  - Tiempos de carga inferiores
  - Mejora la utilización de la capacidad de red disponible

HTTP/2


HTTP1.1
Client Request #1
Server Response #1
Client Request #2
Server Response #2
Client Request #3
Server Response #3
Client Request #4
Server Response #4
Time
HTTP/2
Low Priority
Client Request #1
Client Request #2
Client Request #3
Client Request #4
Low Priority
Server Response #1
Server Response #2
Server Response #3
Server Response #4
Time

HTTP/2



HTTP/2

- Los clientes son los que establecen las conexiones TCP
  - Si se realiza una petición sin conocer si el servidor soporta HTTP/2, se incluye una cabecera "Upgrade" y otra "HTTP2-Settings"

```
GET / HTTP/1.1
Host: server.example.com
Connection: Upgrade, HTTP2-Settings
Upgrade: h2c
HTTP2-Settings: <base64url encoding of HTTP/2 SETTINGS payload>
```

El servidor que no soporte HTTP/2 contestará:

```
HTTP/1.1 200 OK
Content-Length: 243
Content-Type: text/html
...
```

El servidor que sí soporte HTTP/2 contestará:

```
HTTP/1.1 101 Switching Protocols
Connection: Upgrade Upgrade: h2c

[ HTTP/2 connection ...
```

  - Si se sabe que un servidor soporta HTTP/2 no hace falta el paso anterior
    - Connection preface: 0x505249202a20485454502f322e300d0a0d0a534d0d0a0d0a
      - == "PRI * HTTP/2.0\r\n\r\nSM\r\n\r\n"