JESUÏTES educació

DAWM06UF4
COMUNICACIÓ ASÍNCRONA CLIENT-SERVIDOR

UF4.7 HTTP/2

CFGS Desenvolupament d'Aplicacions Web
M06. Desenvolupament web en entorn client
Fundació Jesuïtes Educació - Escola del Clot
Sergi Grau sergi.grau@fje.edu

OBJECTIUS

- Aprendre com funciona el protocol HTTP/2

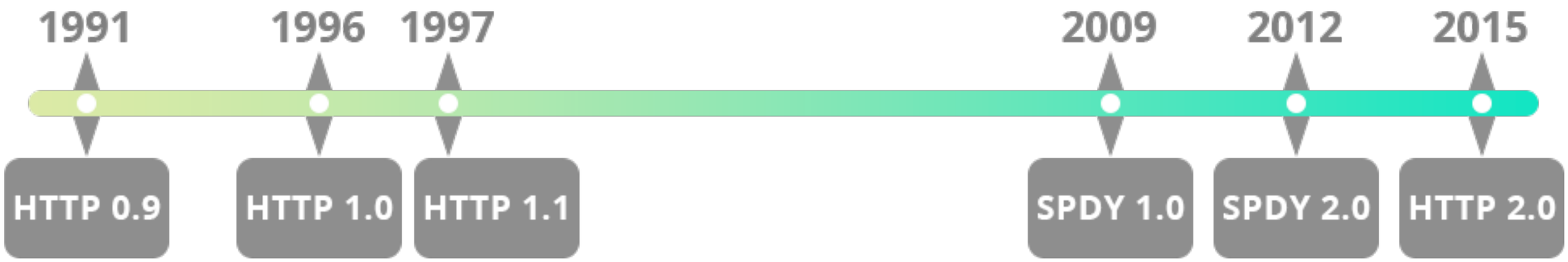
ORIGEN

- HTTP/2 (Protocol de Transferència d'Hipertext, versió 2) és un protocol de xarxa utilitzat per la World Wide Web que arriba amb l'objectiu d'actualitzar el protocol HTTP / 1.1, amb el qual és compatible.

- HTTP 2.0 no modifica la semàntica d'aplicació de Http. Tots els conceptes bàsics, com ara els mètodes HTTP, codis d'estat, URI, i camps de capçalera, es mantenen sense canvis; però, HTTP 2.0 introdueix innombrables millores com l'ús d'una única connexió, la compressió de capçaleres o el servei 'server push'.

- Inicialment va sorgir el protocol SPDY per implementar HTTP tenia com a objectiu reduir la latència (suma de retards temporals dins d'una xarxa).

ORIGEN


1991
1996
1997
2009
2012
2015
HTTP 0.9
HTTP 1.0
HTTP 1.1
SPDY 1.0
SPDY 2.0
HTTP 2.0

SPDY DE GOOGLE

- A principis de l'any 2012 basant-se en el projecte SPDY, l'IETF (Internet Engineering Task Force) va crear un equip per al desenvolupament d'un nou protocol anomenat HTTP 2.0 o HTTP/2. L'esborrany inicial d'HTTP/2 es va publicar al novembre de 2012 però no és fins l'any 2015 quan els navegadors comencen a utilitzar-lo com a suport.

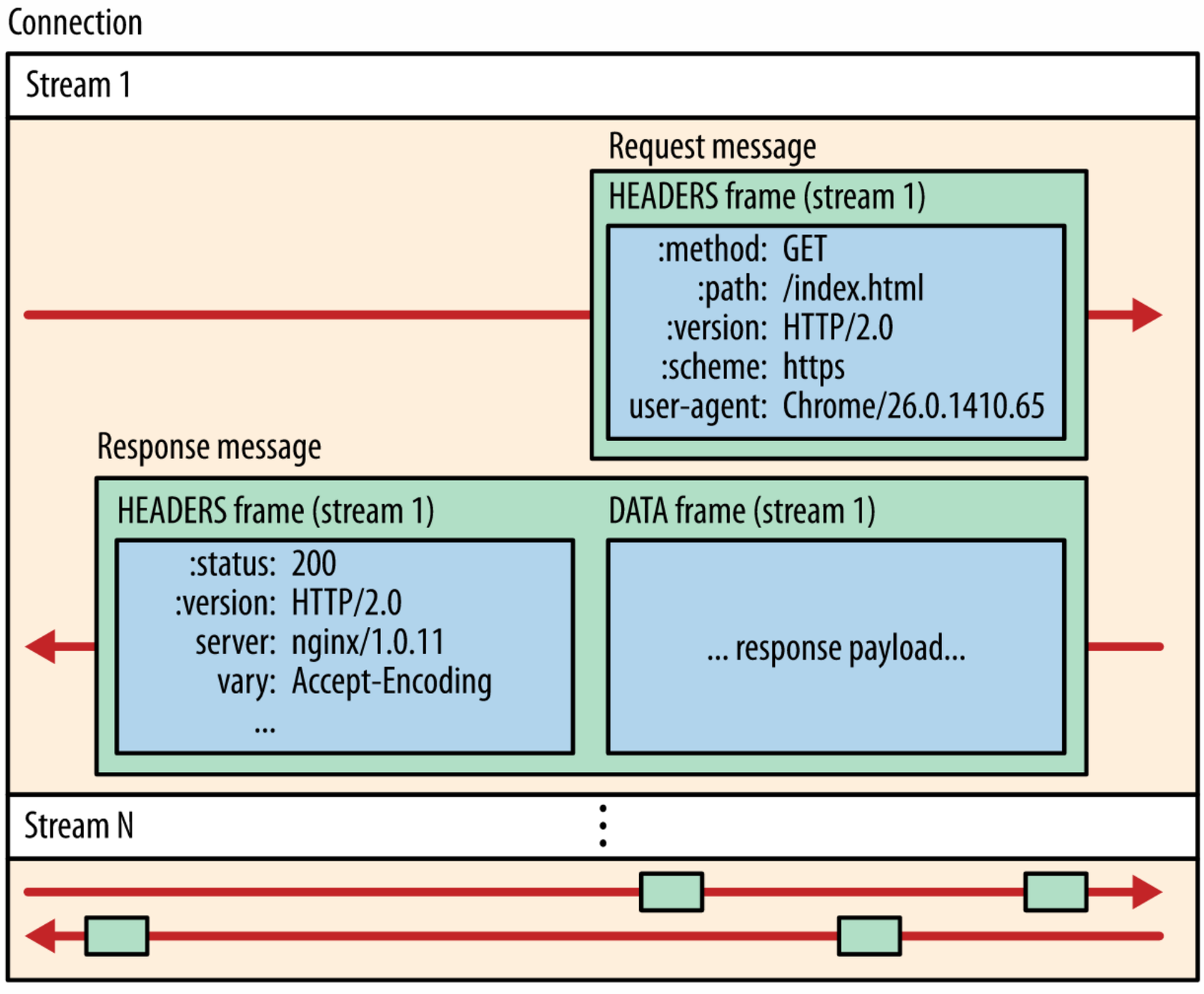
- SPDY s'abandona a favor d'HTTP 2.0 a causa de que la majoria d'avantatges que aporta SPDY també es troben en HTTP 2.0. HTTP / 2 és més ràpid que SPDY, en part a causa de que els seus missatges de sol·licitud són més petits gràcies a la compressió de capçaleres HPACK.

- https://tools.ietf.org/html/rfc7540

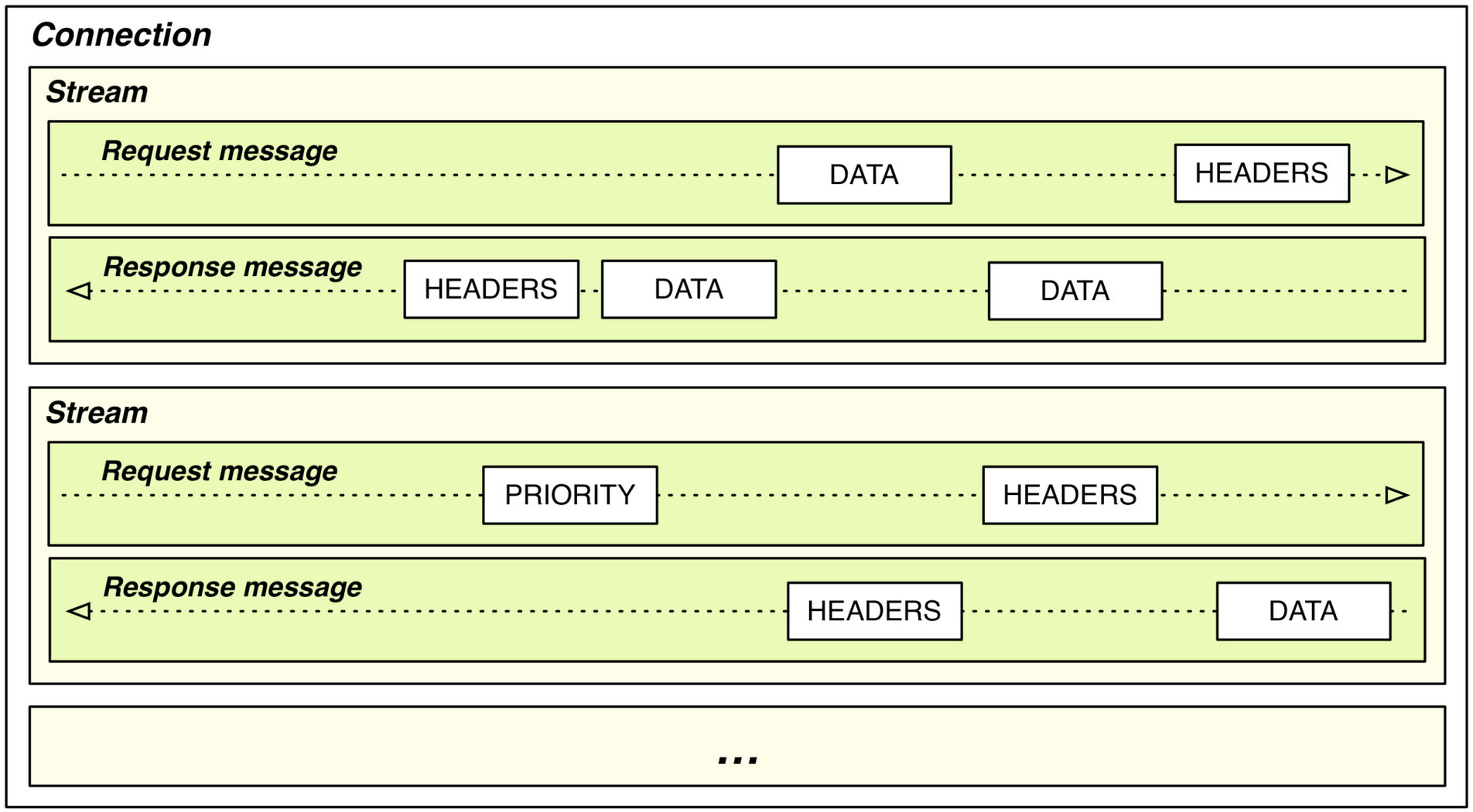
CARACTERÍSTIQUES DEL PROTOCOL

- **Una única connexió:** Amb HTTP / 1.x per carregar qualsevol contingut web és necessari l'ús de múltiples connexions TCP simultànies per poder descarregar tots els elements d'aquesta web. En canvi, HTTP 2.0 utilitza una única connexió per oferir múltiples sol·licituds i respostes en paral·lel. Tenint en compte que cada pàgina web pot contenir objectes HTML, CSS, JavaScript, imatges, vídeo ... la diferència de treball entre utilitzar una única connexió o utilitzar diverses és elevada.

CARACTERÍSTIQUES DEL PROTOCOL



CARACTERÍSTIQUES DEL PROTOCOL



CARACTERÍSTIQUES DEL PROTOCOL

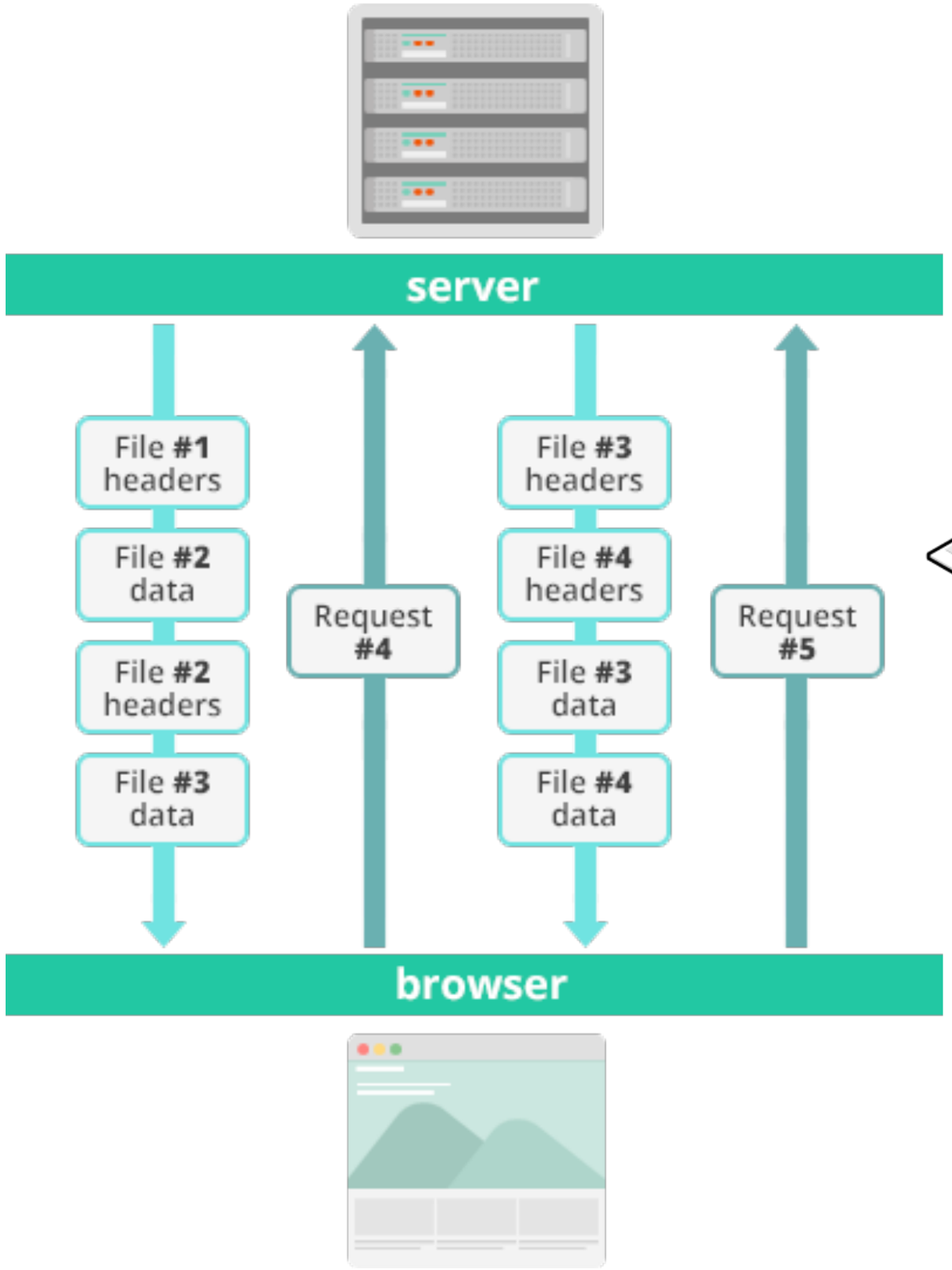
+ **Eliminació d'informació redundant:** Un altre canvi de gran rellevància en HTTP 2.0 és l'eliminació d'informació redundant l'objectiu és evitar l'enviament de dades repetides durant una mateixa connexió, així aconseguirem que es consumeixin menys recursos, obtenint una menor latència.

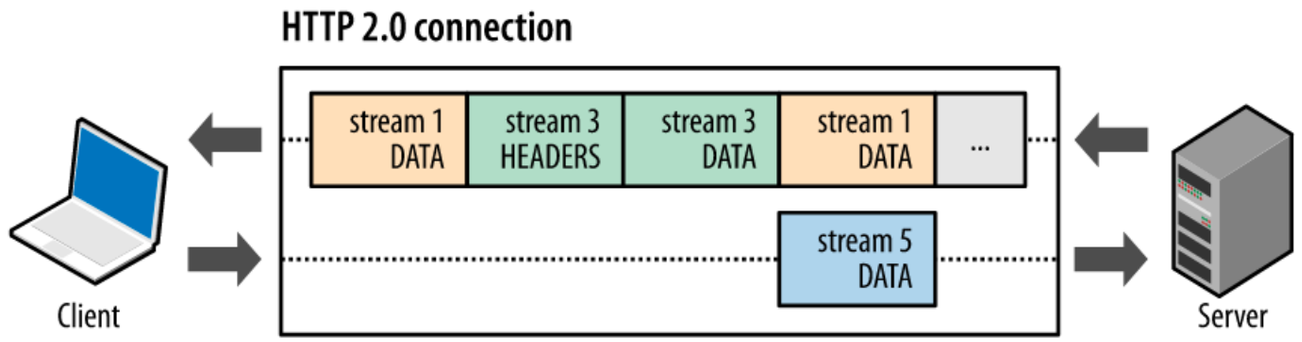
MULTIPLEXACIÓ

- **multiplexació:** Amb HTTP / 1.1 el navegador envia una petició i ha d'esperar la resposta del servidor per poder enviar la següent sol·licitud. El problema és que les webs modernes solen tenir més de 100 objectes, de manera que el retard és gran. La solució que introdueix HTTP 2.0 a aquest problema és la denominada Multiplexació.

- La multiplexació permet enviar i rebre diversos missatges al mateix temps optimitzant la comunicació. Amb la multiplexació s'aconsegueix reduir el nombre de connexions millorant considerablement la velocitat de càrrega i disminuint la congestió dels servidors web.

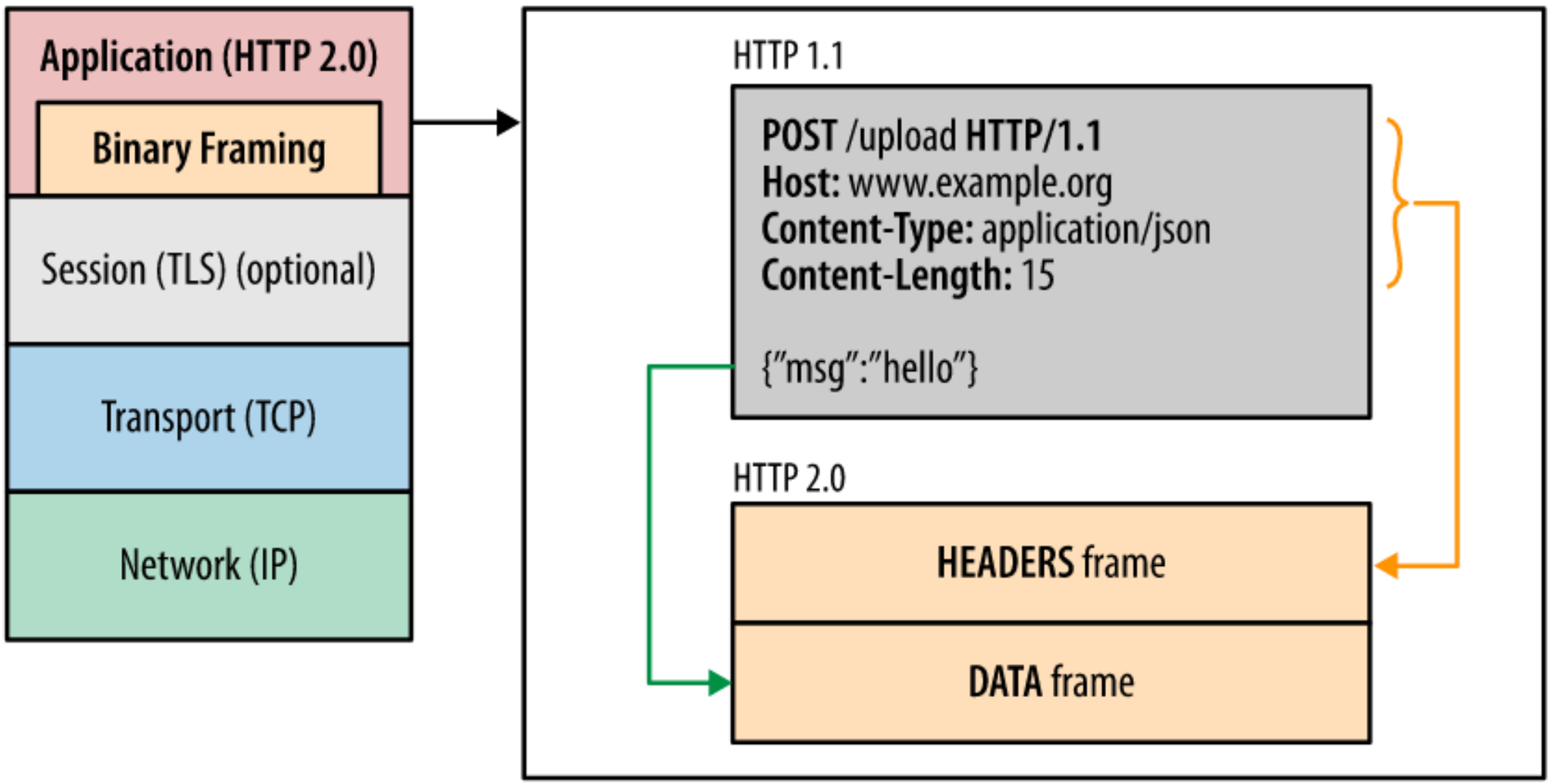
MULTIPLEXACIÓ





- **HTTP 2.0 és un protocol binari:** L'avantatge que té l'ús d'un protocol binari és la facilitat per trobar el començament i el final de cada frame, que és una cosa realment complicat en qualsevol protocol de text. A més, els protocols binaris són molt més simples i per tant són menys propensos a tenir errors que els protocols de text utilitzats per les versions anteriors a HTTP 2.0.

PROTOCOL BINARI



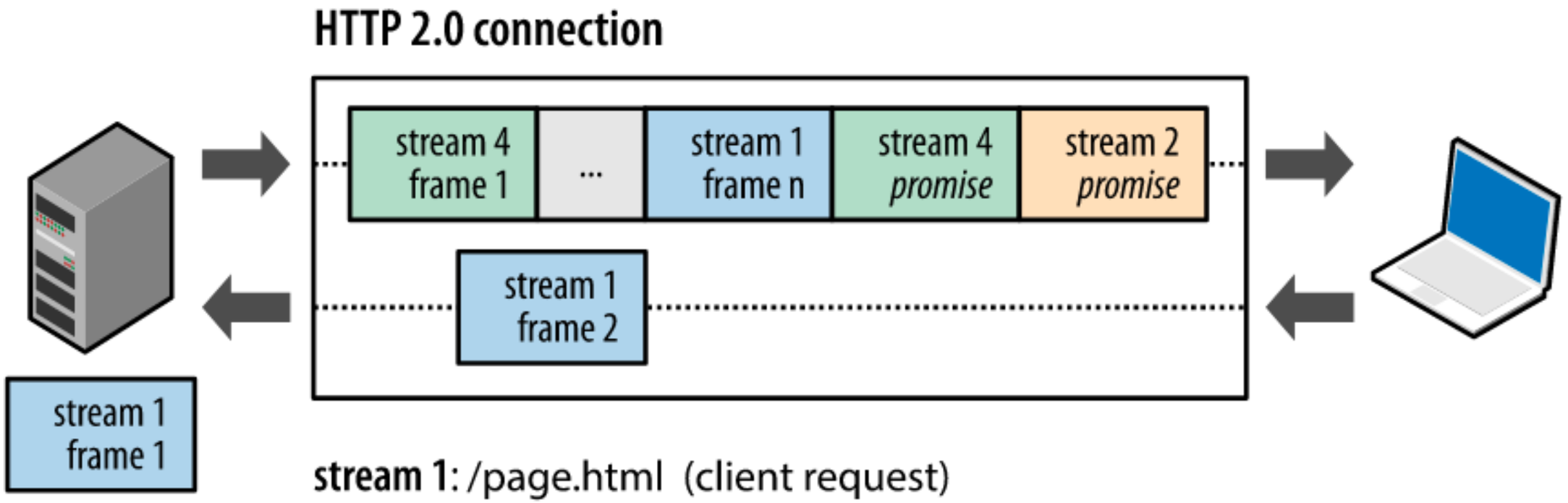
SERVER PUSH

- **Servei 'server push’:** El servei "server push" també conegut com "memòria cau push", es basa en estimacions perquè el servidor sigui capaç d'enviar informació a l'usuari abans que aquest la demani perquè la informació estigui disponible de forma immediata. La forma d'actuar del servidor és enviar diverses respostes a una única sol·licitud del client, és a dir, a més de la resposta a la sol·licitud original, el servidor pot enviar recursos addicionals. Això és així perquè una pàgina web està formada per desenes d'arxius referenciats que gràcies al servei "server push" el servidor envia després de rebre una única sol·licitud estalviant missatges innecessaris.

- HTTP 2.0 conté un camp anomenat 'Ajustaments' amb el qual el client pot indicar si vol obtenir els recursos que proporciona el servei 'server push'.

SERVER PUSH

- Els recursos push poden ser:
  - Desat en memòria cau pel client
  - Reutilitzada en diferents pàgines
  - Multiplexat al costat d'altres recursos
  - Prioritzat pel servidor
  - Rebutjat pel client



stream 1: /page.html (client request)
stream 2: /script.js (push promise)
stream 4: /style.css (push promise)

COMPRESSIÓ DE CAPÇALERES PER TRANSMETRE MENYS INFORMACIÓ

- Amb les versions anteriors a HTTP 2.0, les capçaleres dels missatges de sol·licitud eren de text clar, sense cap tipus de compressió. El problema apareix com a conseqüència de l'increment de mida que pateixen aquestes capçaleres pels agent d'dels navegadors, a l'ús de cookies (també han d'aparèixer en els missatges de sol·licitud), etc. A més, quan HTTP / 1.1 envia una petició, ha d'esperar la resposta del servidor per poder enviar la següent sol·licitud, augmentat molt el retard sofert.

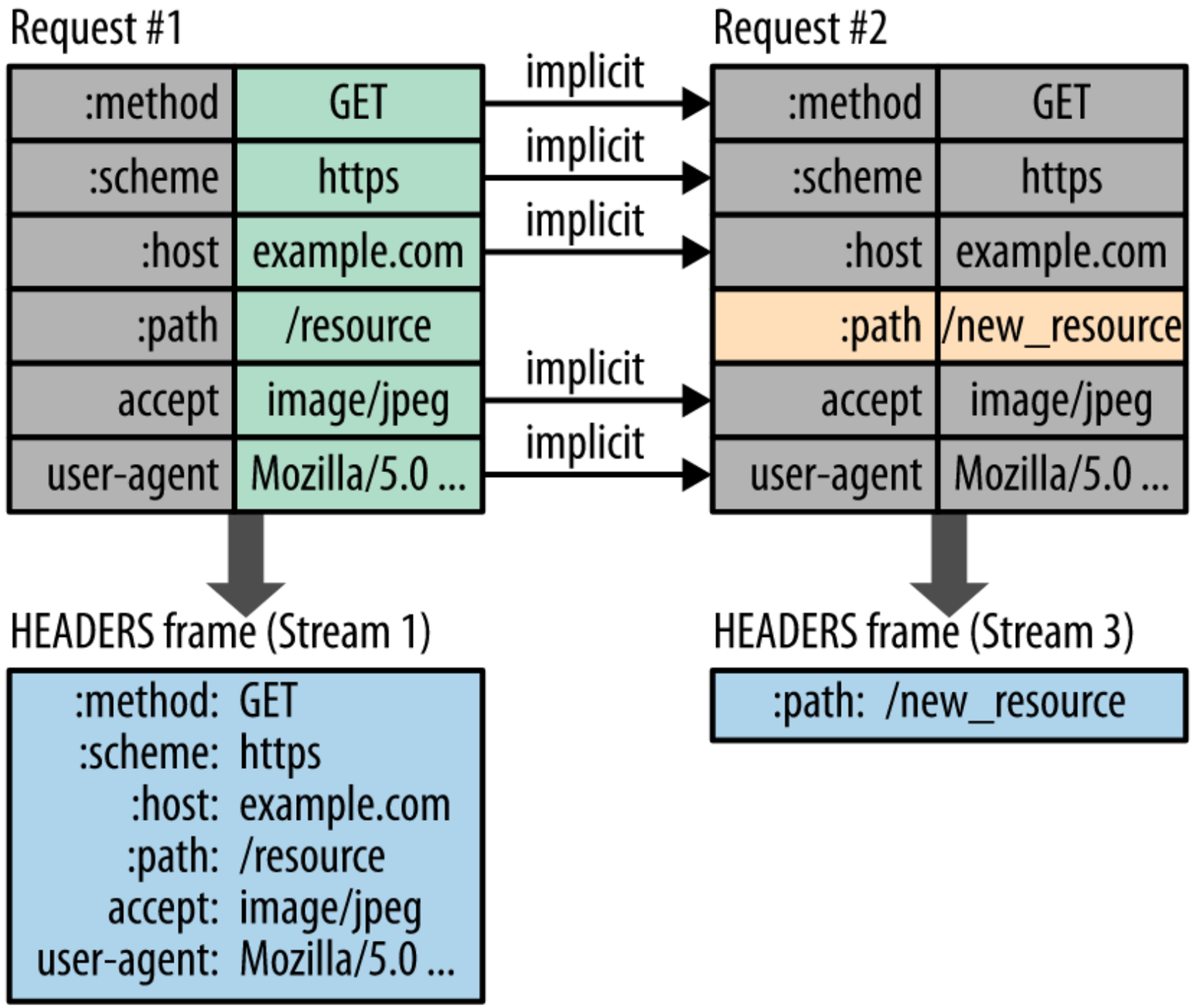
- Així mateix, cal tenir en compte que quan un client demana nombroses peticions a un mateix servidor, els encapçalaments amb prou feines canvien els uns dels altres, de manera que s'envia molta informació redundant.

COMPRESSIÓ DE CAPÇALERES PER TRANSMETRE MENYS INFORMACIÓ

- Amb HTTP 2.0 les capçaleres experimenten compressions, amb el que s'obtenen millors temps de resposta i també es millora l'eficiència (sobretot en terminals mòbils). L'algoritme emprat per realitzar la compressió de capçaleres és HPACK.

- HPACK és un algoritme simple i poc flexible que es basa en eliminar camps de capçalera redundants, a més de prevenir possibles vulnerabilitats.

COMPRESSIÓ DE CAPÇALERES PER TRANSMETRE MENYS INFORMACIÓ



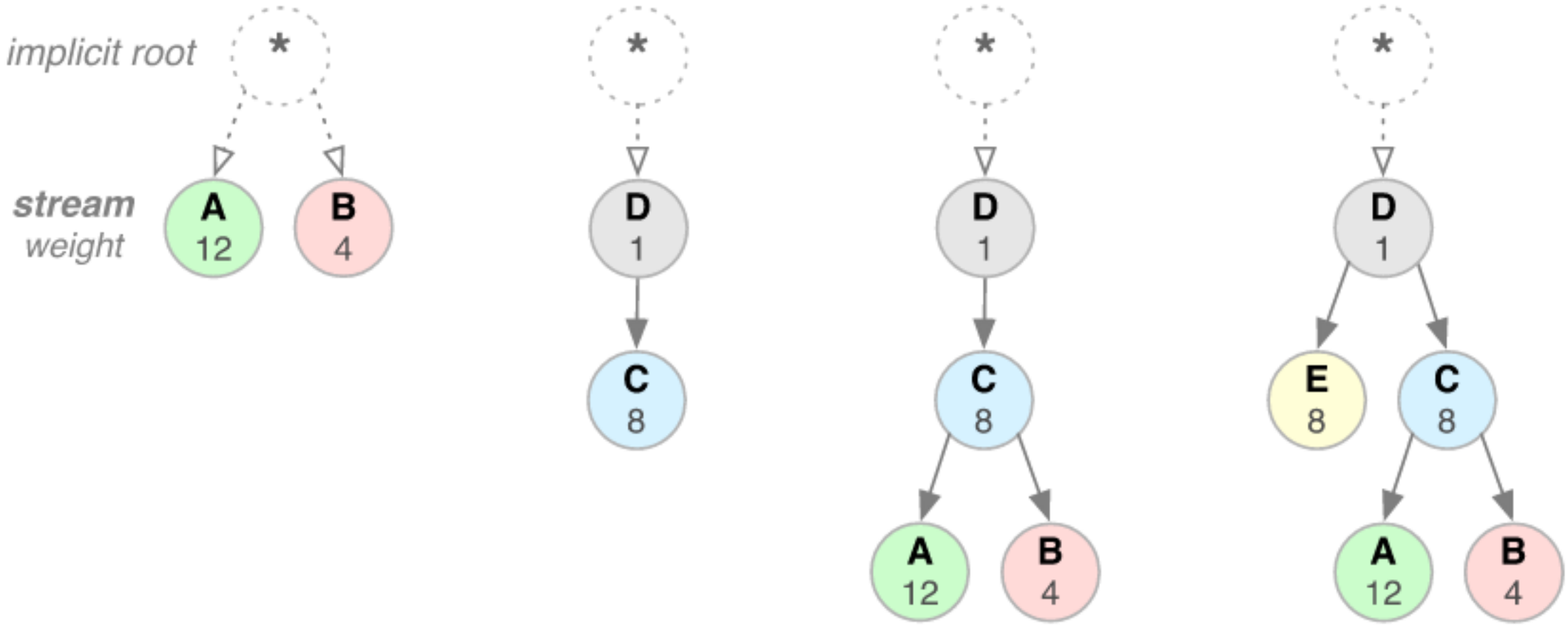
PRIORITZACIÓ DE FLUXOS

- Un missatge HTTP es pot dividir en múltiples fragments en el seu recorregut des del client fins al servidor o des de servidor al client. L'ordre i el retard amb què aquestes trames arriben al seu destí són fonamentals, atès que alguns objectes de les webs són més importants que altres. Ens interessarà que els objectes més rellevants comptin amb algun tipus de prioritat.

- Per poder 'controlar' la prioritat que tenen les trames, HTTP 2.0 permet assignar a cada flux un pes (entre 1 i 256) i una dependència. Hem de ser conscients que les prioritats poden variar durant l'execució.

- Amb les prioritats i la dependència es fa un arbre de prioritats. exemple; Si A té un pes de 12 i B té un pes de 4: A disposarà del 75% dels recursos i B es quedarà amb el 25% restant.

PRIORITZACIÓ DE FLUXOS



FORMAT DE LA TRAMA

- Els trames poden tenir múltiples mides, fins a un límit màxim de 16 kb, llevat que el client i el servidor arribin a un acord per utilitzar una mida de frame major, en aquest cas el màxim mida a la que pot arribar és de 16 MB . Malgrat aquesta possibilitat les frames no solen superar els 16 kb de grandària.

- Totes les trames comencen amb una capçalera de 9 octets fixos seguits de la càrrega útil de longitud variable. El format és el següent:

- Length: 24 bits que indiquen la longitud de l'esmentat frame sense comptar als 9 bytes de la capçalera.

- Type: 8 bits que ens informen de com s'interpreta la resta de la trama. Tot i que avui dia hi ha pocs tipus de frames disponibles, els 8 bits deixen espai per a 256 formats diferents de frames.

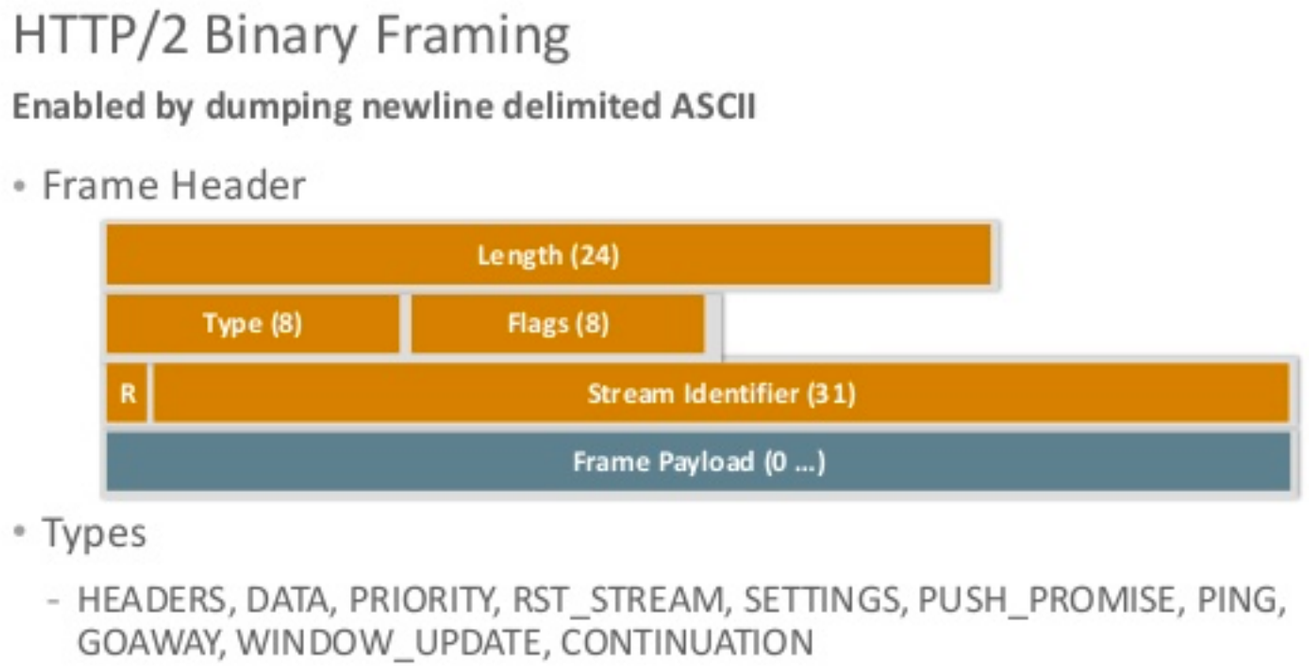
- Flags: 8 bits que determinen el contingut dels flags.

FORMAT DE LA TRAMA

- R: es tracta d'un bit reservat. Ha de romandre amb valor '0' en ser enviat i ha de ser ignorat quan es rep.

- Stream Identifier: Es tracta d'un identificador de flux de 32 bits. El bit més significatiu sempre és 0.

- Payload: conté les dades, és de longitud variable



HTTP/1.1 VS HTTP/2

HTTP request	Frame
GET /index.html HTTP/1.1 Host: example.com Accept: text/html	HEADERS + END_STREAM + END_HEADERS :method: GET :scheme: http :path: /index.html :authority: example.com accept: text/html

HTTP response	Frames
HTTP/1.1 200 OK Content-Length: 11 Content-Type: text/html May The Force Be With You	HEADERS - END_STREAM + END_HEADERS :status: 200 content-length: 11 content-type: text/html
	DATA + END_STREAM May The Force Be With You

BIBLIOGRAFIA

+ https://developers.google.com/web/fundamentals/performance/http2/
+ https://www.ibm.com/developerworks/library/wa-http2-under-the-hood/index.html
+ https://kinsta.com/learn/what-is-http2/
+ https://es.wikipedia.org/wiki/HTTP/2