UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria Informàtica

HTTP3: Estudi teòric i de rendiment de la nova versió d'HTTP

Treball Fi de Grau

Grau en Enginyeria Informàtica

AUTOR/A: Pardo Minaya, Sergio

Tutor/a: Oliver Gil, José Salvador

CURS ACADÈMIC: 2021/2022

Resum

HTTP/3 serà la tercera gran versió del protocol HTTP, la importància de la qual rau en aconseguir resoldre les limitacions que patixen les versions precedents en diversos àmbits.

Aquest treball presenta, d'una banda, l'anàlisi teòrica duta a terme de les característiques i millores introduïdes per cadascuna de les tres versions d'HTTP més importants. De l'altra, es posen a prova aquests canvis mitjançant la realització de diferents tests de rendiment amb l'objectiu d'avaluar el grau de millora que presenta cada versió per a diferents pàgines web.

Paraules clau: HTTP/3, HTTP/2, HTTP/1.1, HTTP, Internet, protocol, rendiment, anàlisi.

Resumen

HTTP/3 será la tercera gran versión del protocolo HTTP, cuya importancia radica en conseguir resolver las limitaciones que sufren las versiones precedentes en diversos ámbitos.

Este trabajo presenta, por una parte, el análisis teórico llevado a cabo de las características y mejoras introducidas por cadas una de las tres versiones de HTTP más importantes. Por otra parte, se ponen a prueba estos cambios mediante la realización de diferentes tests de rendimiento con el objetivo de evaluar el grado de mejora que presenta cada versión para diferentes páginas web.

Palabras clave: HTTP/3, HTTP/2, HTTP/1.1, HTTP, Internet, protocolo, rendimiento, análisis.

Abstract

HTTP/3 will be the third major version of the HTTP protocol, whose importance lies in being able to overcome the limitations that the preceding versions suffer in various scopes.

On the one hand, this project introduces the theoretical analysis carried out about the characteristics and enhancements introduced by each one of the three most significant versions of HTTP. On the other hand, these changes are put to the test by carrying out various performance tests with the goal of evaluating the degree of improvement that each version presents for different websites.

Keywords: HTTP/3, HTTP/2, HTTP/1.1, HTTP, Internet, protocol, performance, analysis.

Índex de Continguts

Índex de Figures

Índex de Taules

1. Introducció

1.1. Motivació

Els últims vint anys, el món ha experimentat una de les transformacions tecnològiques més importants de l'Edat Contemporània amb la consolidació d'Internet com a xarxa de comunicació a nivell global i la seua adopció en un ventall cada vegada més divers d'àmbits de la vida humana.

Particularment, un dels sistemes més popularitzats gràcies a Internet ha estat el web, un espai que permet l'intercanvi d'informació i recursos a través d'una gran varietat de dispositius. És aquest espai el qual ha donat peu a la creació de grans serveis d'abast mundial orientats tant al públic general com a les empreses. És així que trobem les xarxes socials, el consum de contingut multimèdia, la computació en el núvol o l'Internet de les Coses.

La proliferació de tots aquests serveis suposa una mitjà excel·lent per al desenvolupaments de noves i millors solucions als problemes de l'actualitat, però, al mateix temps, constituïx una demanda creixent de major fiabilitat, ubiqüitat i capacitat de les comunicacions que es realitzen a través d'Internet. No hi ha dubte que la innovació en l'àrea de les telecomunicacions ha permés pal·liar aquestes limitacions amb el descobriment de noves tecnologies i mitjans de transmissió de la informació com ara la fibra òptica o les xarxes sense fils. No obstant això, donat que hi ha un límit pel que fa al desenvolupament de noves millores del maquinari i cal aspirar a aprofitar el potencial de la tecnologia i la infraestructura existents, s'ha de mirar que el programari que funciona per damunt d'aquests medis també siga el més òptim possible.

En el cas de les xarxes de computadors, açò suposa que els protocols de la pila de protocols siguen capaços de funcionar de forma eficient i no menysprear el potencial oferit pel medi pel qual actuen. Donat que el web té una rellevància especial en el món modern, és essencial que la informació es transmeta amb les majors garanties de qualitat possibles.

El protocol HTTP és l'encarregat, en la capa d'aplicació, d'encapsular totes les dades web que viatgen per la xarxa, per la qual cosa la importància de la seua optimització és fonamental. All llarg dels anys 90 i de la primera dècada del segle XXI s'han succeït una sèrie de versions d'aquest protocol encaminades a millorar la seua eficiència i a aprofitar les noves possibilitats que oferixen les xarxes de major amplada de banda.

L'última versió acabada i publicada de forma oficial és HTTP/2, que aspira a introduir una gran quantitat de canvis sobre la versió precedent, HTTP/1.1, utilitzada durant un llarg període de temps. No obstant això, encara calen certs canvis més radicals que, integrats de manera adequada amb els ja existents, donaran lloc a la versió HTTP/3, que aspira a ser la culminació de tota aquesta trajectòria de millora.

És per això que aquest treball naix de l'interés per estudiar les millores que el protocol HTTP ha patit des de la seua creació, posant un èmfasi especial en les últimes modificacions que donaran lloc a la futura versió HTTP/3, i explorar de forma directa l'eficàcia de dites millores.

1.2. Objectius

Els objectius d'aquest projecte són, en primer lloc, documentar els canvis que el protocol HTTP ha patit al llarg del temps, des de la primera fins a l'última versió; en segon lloc, analitzar les millores que aquests canvis suposen per al rendiment del protocol; finalment, l'últim objectiu és avaluar el grau de millora de la versió 3 d'HTTP sobre les versions precedents.

1.3. Context

Les orígens del protocol HTTP es remunten al 1989, amb Tim Berners-Lee com a investigador del CERN. El 1991 es va publicar el document oficial de la primera versió, HTTP/0.9, que presentava unes funcionalitats molt bàsiques [1].

Després d'uns anys en què es perfilaren les característiques addicionals del protocol i s'elaborà una especificació més extensa del mateix, es publicà la versió HTTP/1.0 el 1996. Aquesta era la primera versió que permetia crear pàgines web gràficament atractives que ajudaren a l'adopció a gran escala del web [1]. Sols un any més tard, el 1997, se'n va fer una actualització, la versió HTTP/1.1, que permetia adoptar de manera estàndard diverses capçaleres addicionals que havien sigut implementades pels desenvolupadors web i gestionar millor les connexions TCP. El període de temps durant el qual la versió HTTP/1.1 ha estat l'última disponible ha sigut llarg, ja que l'última actualització d'aquesta versió de l'estàndard HTTP es va fer el 2014.

Tot i que les característiques d'HTTP/1.1 hagen permés la generalització del seu ús a tot Internet, cal tindre en compte que el panorama tecnològic dels anys 90 i principis del 2000 és ben diferent del de la dècada del 2010. Entre aquestes diferències trobem la proliferació vertiginosa dels dispositius mòbils i la creixent presència de contingut audiovisual, estils i efectes visuals i la programació d'aplicacions web, que suposen un consum d'amplada de banda molt considerables en comparació a les pàgines estàtiques de text dels anys 90. És en aquest moment en què es comença a treballar en una

remodelació del protocol, amb la fi que aprofite millor els recursos de la xarxa al mateix temps que reduïsca la transmissió innecessària de dades.

L'any 2009, Google va començar a treballar en un projecte propi que pretenia millorar les capacitats d'HTTP/1.1, anomenat SPDY. Aquest projecte va influenciar en gran mesura el desenvolupament que va emprendre la IETF (Internet Engineering Task Force) el 2012 per a desenvolupar una nova versió de l'estàndard HTTP. Finalment, el 2015 va ser publicada la versió HTTP/2 en base a aquest treball.

Tot i que les millores d'HTTP/2 sobre les versions precedents eren considerables, es va percebre que encara hi havia certes restriccions que no podien ser resoltes amb canvis en el protocol a nivell d'aplicació. És per això que, al 2018, es va iniciar el desenvolupament d'una nova versió de l'estàndard, HTTP/3, que pren les millores ja introduïdes per HTTP/2 i les unix a les millores oferides per un nou protocol de nivell de transport anomenat QUIC. Aquest protocol de transport va començar a desenvolupar-se l'any 2013 a Google, i el fet de funcionar sobre UDP en lloc de TCP ha permés evitar les restriccions de transmissió i control de congestió imposats per aquest últim. És aquesta última versió, HTTP/3, encara basada en esborranys, en la qual anem a centrar el nostre estudi.

En les seccions següents vorem amb més detall les característiques de cadascuna d'aquestes versions. A mode d'il·lustració, a la **Figura 1** es pot observar una línia del temps amb totes les versions oficials del protocol HTTP.

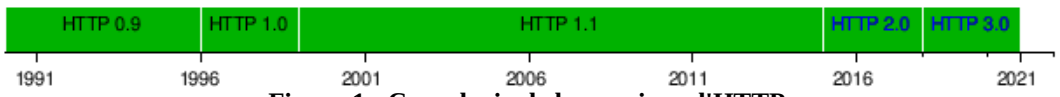

Figura 1 - Cronologia de les versions d'HTTP
[es.wikipedia.org/wiki/Protocolo_de_transferencia_de_hipertexto]

1.4. Metodologia

Per a dur a terme l'estudi teòric de les distintes versions del protocol, ha sigut necessari cercar informació a pàgines web, articles, documents oficials i llibres, ja siga per a entendre el funcionament del protocol, la seua evolució històrica o per a conéixer detalls sobre la implementació de certes característiques en components concrets com servidors i navegadors.

D'una altra banda, per a realitzar l'estudi pràctic comparatiu, s'ha fet una recerca de les millors ferramentes en línia que permeten simular diverses condicions de xarxa i

versions del protocol a usar i que permeten obtindre una ampla gamma d'informació i estadístiques sobre cadascuna de les proves.

1.5. Estructura

El treball està dividit en quatre parts, cadascuna destinada a una funció concreta, que detallarem a continuació.

La primera part correspon a una introducció del treball, en la qual es detallen els motius que han dut a la realització del mateix, els objectius que es pretenen assolir i el context tecnològic en el qual es desenvolupa l'estudi.

La segona part mostra l'anàlisi de les característiques de les versions d'HTTP més importats (HTTP/1.1, HTTP/2 i HTTP/3) i detalla quines són les millores que introduïxen per a resoldre els problemes de la versió precedent, al mateix temps que descriu les pròpies limitacions que presenta cadascuna d'elles.

La tercera part consistix en una sèrie de proves destinades a analitzar com afecten els canvis presents en cada versió al seu rendiment en diferents pàgines web.

Finalment, a la quarta part s'oferixen una sèrie de conclusions extretes a partir del treball dut a terme.

2. Estudi teòric

2.1. Característiques generals

HTTP és un protocol del nivell d'aplicació destinat a presentar una interfície uniforme als clients per a la compartició d'informació independentment del tipus de recursos proveïts pels servidors. Es tracta d'un protocol sense estat en què cada transacció està formada d'una petició i una resposta.

En aquesta transacció hi ha dos rols: client i servidor. El client és aquell programa que establix una connexió amb un servidor per a fer-li peticions, mentre que el servidor és el programa encarregat d'acceptar aquestes connexions i enviar respostes a les peticions HTTP del clients.

La informació a què es pot fer referència en les peticions i respostes està representada seguint un Identificador de Recursos Uniforme (URI), que definix la manera en què cada recurs individual proveït per un servidor és anomenat [1]. En la **Figura 2** es poden observar les parts de què es composa una URI. Entre elles trobem l'**esquema** (Scheme) que és el protocol utilitzat, l'**autoritat** (Authority) que indica el nom del servidor o l'adreça IP i el port a què s'enviarà la petició, el **camí** (Path) que indica on trobar el recurs dins del servidor, un conjunt de **paràmetres** (Parameters) en què s'envia informació al servidor en forma de parells *clau = valor* separats pel caràcter *&*, i l'**àncora** (Anchor), que fa referència a una secció dins del recurs referenciat pel camí.

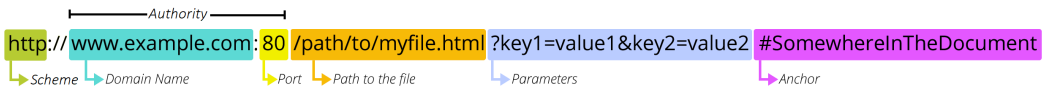

Figura 2: Parts d'una URI
[developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_URL]

El format dels missatges HTTP seguix un patró diferent depenent de si es tracta d'una petició o una resposta [1]. Aquesta distinció comença a la primera línia, que es tracta d'una línia de petició si és una petició o una línia d'estat si és una resposta.

A la **Figura 3** s'observa el format d'una petició HTTP. Es poden apreciar tres parts: la línia de petició, les línies de capçalera i el cos del missatge. La línia de petició conté tres camps: el mètode HTTP, l'URL del recurs i la versió HTTP utilitzada. A continuació es troben les línies de capçalera, que contenen informació addicional d'interés per al servidor (en el cas d'una petició) o per al client (en el cas d'una resposta). Aquestes

línies de capçalera poden contindre informació general sobre la transmissió, informació específica sobre la petició o sobre la resposta, o informació sobre el contingut del cos del missatge. Cal tindre en compte que, a banda de les capçaleres definides als estàndards, els desenvolupadors també arriben a crear les seues pròpies capçaleres. Finalment, es troba el cos del missatge, que conté les dades a enviar al servidor.

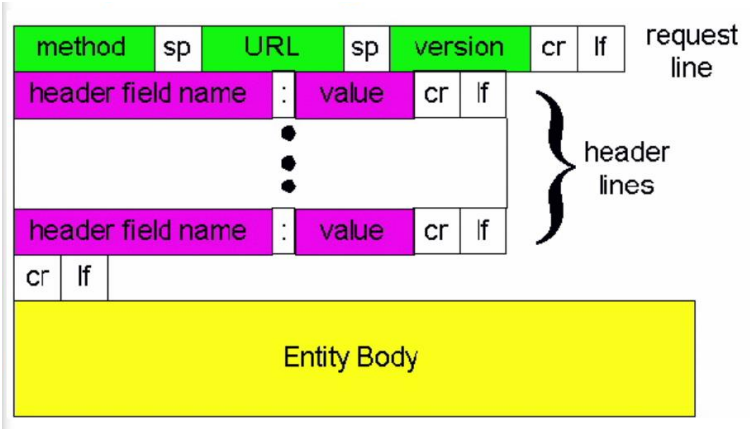

Figura 3 - Format d'una petició HTTP

[www.cs.umd.edu/~shankar/417-F01/Slides/chapter2a-aus/sld017.htm]

El format general d'una resposta HTTP es pot apreciar a la **Figura 4**. La resposta està formada per tres parts: la línia d'estat, les línies de capçalera i el cos del missatge. La línia d'estat conté tres camps: la versió del protocol HTTP, el codi d'estat i la frase textual que descriu el codi d'estat. A continuació es troben les línies de capçalera, que com en el cas de les peticions, poden ser molt variades. Finalment, també com en el cas de la petició, hi ha el cos del missatge, que conté les dades que el servidor envia al client.

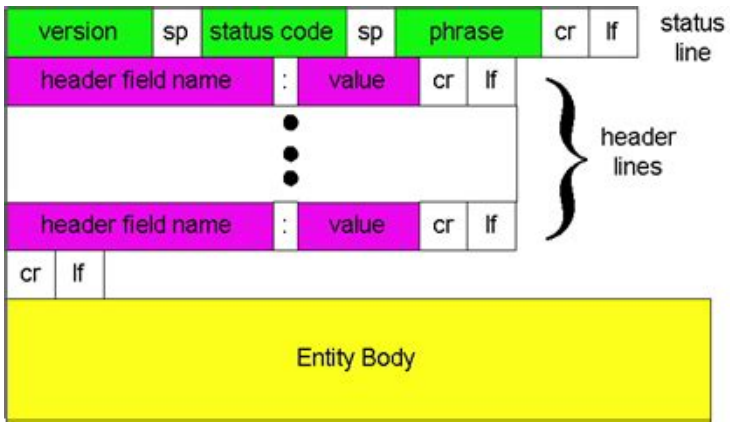

Figura 4 - Format d'una resposta HTTP

[www2.ic.uff.br/~michael/kr1999/2-application/2_02-http.htm]

Hi ha diferents mètodes disponibles a la línia de petició d'una petició HTTP, com ara GET, HEAD, POST, PUT, DELETE o OPTIONS [1]. Existixen també altres mètodes com CONNECT o TRACE amb un ús menys comú. A continuació descrivim els mètodes més importants.

GET servix per a demanar la transferència d'un recurs identificat per l'URI de la línia de petició.

HEAD té el mateix significat que GET, però, a diferència d'aquest (en què el servidor inclou en el cos del missatge les dades referenciades per l'URI de la petició) el servidor deixa el cos del missatge buit. D'aquesta manera, només s'inclouen les capçaleres que estarien presents en una resposta a un GET sense enviar el recurs sol·licitat. Açò permet dur a terme tasques de depuració web sense haver de transmetre recursos potencialment voluminosos amb cada prova.

POST és un mètode que permet enviar dades que siguen processades pel servidor d'acord a l'URI proporcionada. L'efecte habitual que es produïx és la creació d'un recurs determinat dins de la col·lecció referenciada per l'URI. Aquest mètode no és idempotent, és a dir, si es du a terme vàries vegades, acabaran creant-se diversos recursos al servidor.

PUT també permet enviar dades a processar pel servidor d'acord a l'URI especificada, però a diferència de POST, l'URI identifica un recurs particular en lloc d'una col·lecció, de manera que, si el recurs no existix, el crea, però si ja existix, l'actualitza. Aquest mètode és idempotent, és a dir, si es du a terme vàries vegades sempre s'obtindrà el mateix resultat, la creació o actualització d'un recurs determinat.

DELETE servix per a demanar al servidor que elimine el recurs identificat per l'URI.

OPTIONS és un mètode que permet demanar informació al servidor sobre quins mètodes estan disponibles per al recurs especificat.

Com hem vist anteriorment, hi ha uns camps a la línia d'estat de la resposta que permet especificar el codi d'estat i el seu nom [1]. Aquests codis de tres xifres poden ser de cinc classes depenent del nombre pel qual comencen. Els codis 1XX servixen per a donar informació al client, els 2XX s'empren per a identificar que la petició ha tingut èxit, les 3XX servixen per a indicar una redirecció, els 4XX s'utilitzen per a indicar que hi ha hagut un error a la part del client, i els 5XX indiquen que s'ha produït un error a la part del servidor.

La gestió de les connexions en HTTP pot analitzar-se en funció de diversos aspectes, tals com el protocol que s'utilitza al nivell inferior, la persistència de les connexions, la canalització (*pipelining*) de les transmissions o la concurrència.

Normalment, i fins a la versió HTTP/2 inclosa, el protocol de transport utilitzat per sota d'HTTP ha sigut TCP, que permet establir connexions amb transport fiable i mecanismes de control de congestió. Justament aquests aspectes del protocol TCP el fan poc adient per a certs tipus de transmissió de contingut, per la qual cosa a la versió HTTP/3 es va decidir triar un altre protocol de transport.

La persistència de les connexions depén de si aquesta es tanca després de cada parell petició-resposta (no persistent) o, per contra, la connexió es manté per a la realització de subsegüents enviaments de peticions i respostes (persistent) [1]. A la **Figura 5** s'observa un diagrama temporal en què s'exemplifica la diferència del temps requerit per a una sèrie de transmissions amb connexions no persistents i persistents.

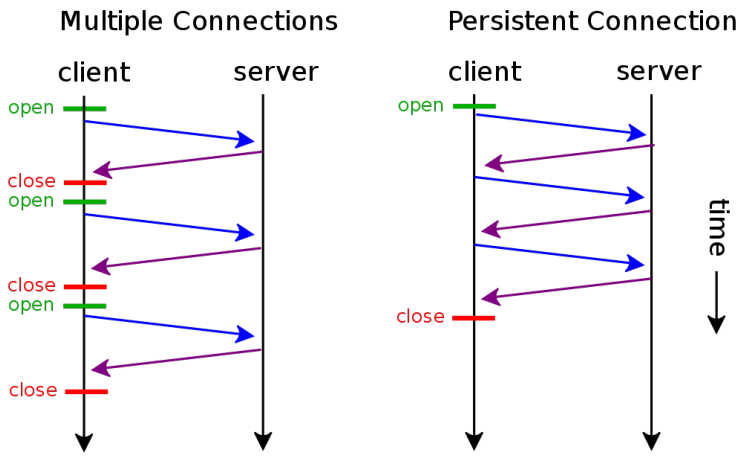

Figura 5: Comparació d'una connexió no persistent i persistent [en.wikipedia.org/wiki/HTTP_persistent_connection]

La canalització o *pipelining* de les transmissions és un mecanisme vinculat a les connexions persistents que permet al client enviar diverses peticions, una seguida de l'altra, sense esperar que el servidor acabe d'enviar la resposta a aquestes peticions [1] (**Figura 6**). Cal tindre en compte que només les peticions amb mètodes idempotents (GET, HEAD, PUT i DELETE) poden ser canalitzades, de manera que si ocorreguera un error es poguera tornar a enviar sense produir canvis inesperats.

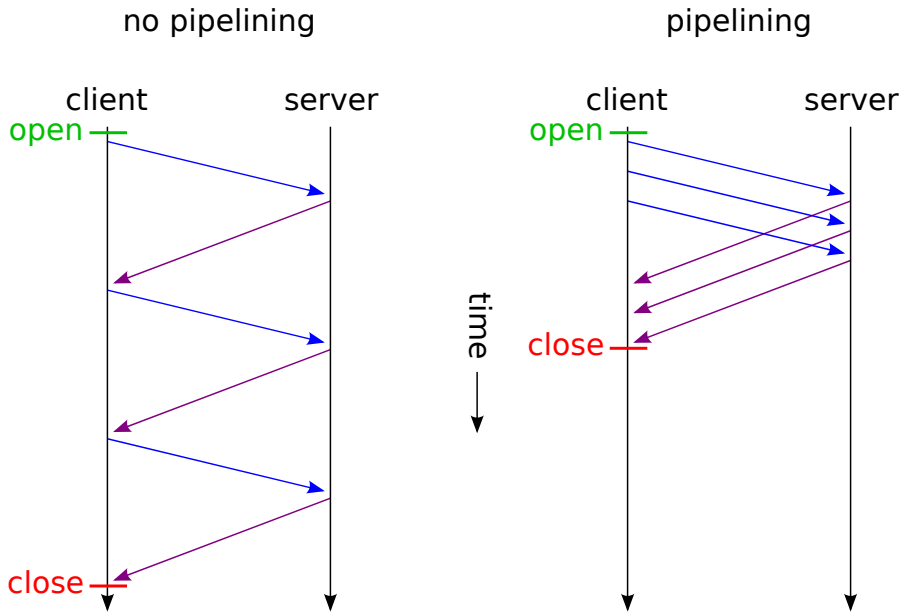

Figura 6: Comparació d'una connexió sense i amb canalització [en.wikipedia.org/wiki/HTTP_pipelining]

La concurrència de les connexions fa referència a la capacitat que tenen els clients d'establir més d'una única connexió TCP al mateix temps, siga amb el mateix o amb un altre servidor [1]. Aquest mecanisme permet carregar els recursos d'una pàgina més ràpidament i pot ajuda a evitar el bloqueig de cap de línia (*head-of-line blocking*) que es dóna quan una petició requerix un gran processament a la part servidora que impedix el processament de subsegüents peticions pertanyents a la mateixa connexió. Tot i això, els clients han de limitar el nombre màxim de connexions que establixen amb un servidor concret a fi d'evitar saturar els seus recursos.

En els apartats següents, explorarem més profundament alguns aspectes concrets del protocol HTTP, especialment les diferències entre versions.

2.2. HTTP/1.1

La versió HTTP/1.1 està generalment basada en la versió 1.0 a la qual es van afegir certes característiques addicionals.

En primer lloc, HTTP/1.0 presentava un nombre molt limitat de codis d'estat que va ser ampliat i classificat amb HTTP/1.1 [2].

En segon lloc, fins a la versió 1.0 d'HTTP només es podia tindre un servidor a una màquina física, és a dir, una màquina física servidora només podia tindre una adreça IP

[2]. Per a superar aquesta limitació, és obligatori incloure la línia de capçalera *Host* perquè així es puga identificar a quin servidor s'està fent la petició de tots els presents a una màquina física determinada. Aquest és un gran avantatge que permet la instal·lació de molts servidors virtuals en una mateixa màquina, una pràctica extremadament habitual hui en dia.

En tercer lloc, hi ha prou diferències en la manera de gestionar les connexions en HTTP/1.0 i HTTP/1.1. Mentre que a la versió 1.0 les connexions establides entre client i servidor sempre es tancaven després de l'intercanvi d'un parell petició-resposta (no eren persistents), a la versió 1.1 es pren la perspectiva contrària: les connexions es mantenen obertes per defecte [2]. D'aquesta manera, es pot regular el comportament de les connexions amb la capçalera *Connection* que pot prendre els valors *close* (es tanca la connexió després de cada petició-resposta) o *keep-alive* (la connexió es manté oberta per a permetre posteriors transmissions de parells petició-resposta). Aquesta és una característica que permet reduir l'impacte que té en el servidor haver de crear cada vegada noves connexions TCP.

A més, la persistència de les connexions permet aprofitar la canalització o *pipelining*, que ajuda a reduir encara més els temps de càrrega de les pàgines web mitjançant el processament en paral·lel de diverses peticions.

2.3. HTTP/2

HTTP/2 introduïx una gran quantitat de canvis respecte de la versió precedent que anem a detallar.

En primer lloc, cal tindre en compte que fins a HTTP/1.1, tots els missatges es transmeten en text pla, de manera que poden ser llegits a simple vista per un humà. No obstant això, aquesta aproximació presenta algunes limitacions i problemes d'optimització a l'hora de processar i optimitzar els missatges, per la qual cosa, es va acordar un canvi en la versió 2 d'HTTP que consistix a conservar la semàntica ja existent d'HTTP/1.1 (mètodes, capçaleres, etc.) però canviar la sintaxi de format textual a format binari perquè així es puga agilitzar el seu processament [3]. D'aquesta manera, les aplicacions continuen generant missatges HTTP en text pla, però aquests són convertits a binari quan són processats per la capa del protocol HTTP/2. A la **Figura 7** es pot vore com es processa un missatge en text pla.

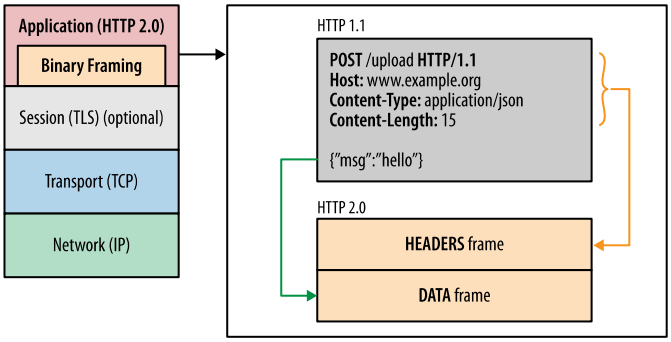

Figura 7: Transformació de text pla a binari en HTTP/2
[developpaper.com/some-key-performance-improvement-points-of-http-2/]

Un problema important que presenta HTTP/1.1 és el coll de botella conegut com a bloqueig de cap de línia, en el qual hi ha peticions independents entre si que romanen bloquejades a causa de la petició que està sent processada, ja siga perquè la seua resposta requerix un gran processament al servidor o s'ha perdut algun paquet TCP. Tot i que s'intenta resoldre aquest problema amb l'establiment de vàries connexions TCP concurrents, hi ha un límit en el màxim nombre d'aquests. A més, donat que HTTP/1.1 no posseïx una capa de multiplexació, s'han d'obrir diverses connexions TCP simultànies si es desitja augmentar el paral·lelisme de l'obtenció de recursos. No obstant això, aquesta aproximació presenta l'inconvenient que TCP només és capaç de gestionar el control de flux i de pèrdues a nivell de connexió, cosa que suposa un impacte negatiu en la congestió i l'eficiència de la xarxa [3].

Per a resoldre aquesta situació, es va adoptar un punt de vista ben diferent en el disseny de les transmissions d'HTTP/2. Així, en HTTP/2 es crea una única connexió TCP entre el client i el servidor. A través d'aquesta connexió passaran diversos fluxos o *streams*, corresponents a l'intercanvi d'una sèrie de missatges, cadascun d'ells partits en trossos xicotets anomenats trames o *frames* [3]. Des del punt de vista de la connexió TCP, el que es veu és un seguit de missatges menuts que es transmeten entre el client i el servidor. Tota la logística per assignar les trames als missatges i aquests als fluxos es fa de manera lògica per mitjà d'identificadors o etiquetes. A la **Figura 8** es pot observar de forma gràfica aquest disseny. És així com es poden transmetre xicotets trossos de diversos missatges al mateix temps (un procés anomenat multiplexació), evitant en gran mesura els problemes del bloqueig de cap de línia.

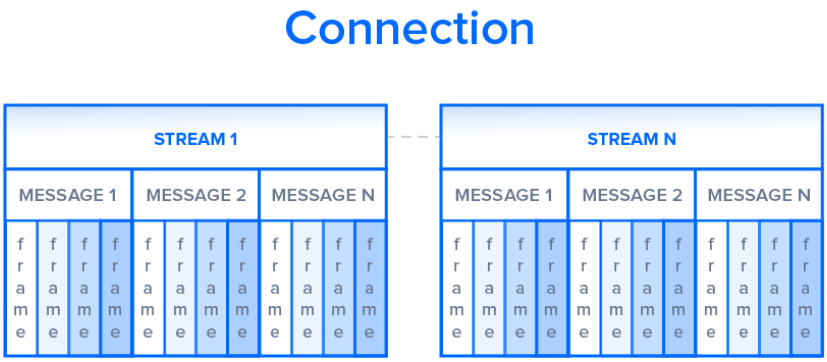

Figura 8: Organització lògica d'una comunicació en HTTP/2 [www.digitalocean.com/community/tutorials/http-1-1-vs-http-2-what-s-the-difference]

Per a entendre millor cada divisió d'una connexió HTTP/2, anem a explicar amb més detall en què consistixen. Un *stream* o flux és una seqüència independent i bidireccional de trames intercanviades entre el client i el servidor. En una connexió normal HTTP/2 es poden trobar diversos fluxos funcionant concurrentment, cadascun d'ells amb un identificador. El nombre de fluxos funcionant concurrentment pot ser configurat mitjançant els paràmetres de la connexió [4].

Els missatges corresponen a les peticions i respostes ja existents de la versió HTTP/1.1, doncs la semàntica d'aquests no s'ha alterat. Açò permet que els programes no hagen de preocupar-se per canviar la seua forma de construir els missatges HTTP, ja que serà la capa HTTP la qual s'encarregarà de fer tota la transformació a binari i la gestió d'assignació a fluxos i partició en trames.

El format de les trames, que són la unitat atòmica de transmissió en HTTP/2, es mostra a la **Figura 9**. Els camps que hi ha presents són: Longitud (longitud del cos de la trama), Tipus (el tipus de trama), Indicadors (indicadors variats depenent del tipus), un bit Reservat i l'Identificador del Flux. Els tipus de trames que podem trobar són: DATA, HEADERS, PRIORITY, RST_STREAM, SETTINGS, PUSH_PROMISE, etc. D'aquests, alguns destacats són HEADERS (trama amb capçaleres d'un missatge HTTP), DATA (trama amb les dades d'un missatge HTTP), SETTINGS (trama amb configuracions de la connexió HTTP) i PUSH_PROMISE (trama amb informació dels recursos que el servidor vol enviar al client abans que aquest li'ls demane) [4].

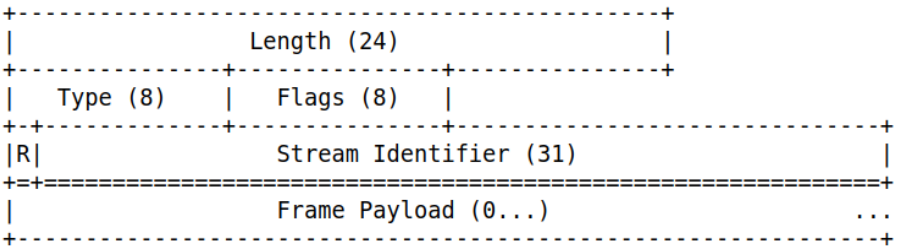

Figura 9: Format d'una trama HTTP/2
[https://www.rfc-editor.org/rfc/rfc7540.txt]

HTTP/2 posseïx altres mecanismes destinats a millorar la transmissió òptima de la informació. Un d'aquests és la priorització de fluxos. Aquest procediment consistix en donar un pes a les trames pertanyents a un flux entre 1 i 256, sent aquest últim el valor de major prioritat. A més del pes, també es pot indicar la dependència entre fluxos mitjançant l'identificador únic que posseïx cadascun d'ells. Açò permet al servidor atendre les peticions en funció de les dependències i la importància de cada flux [4]. A la **Figura 10** es pot apreciar un arbre de dependències en què s'indica l'identificador de cada flux junt amb el seu pes. Quan el servidor ha de processar les peticions de cada flux, anirà de dalt a baix de l'arbre, processant un nivell cada vegada. Quan hi ha diversos fluxos en un mateix nivell, els recursos es repartixen proporcionalment al seu pes [3].

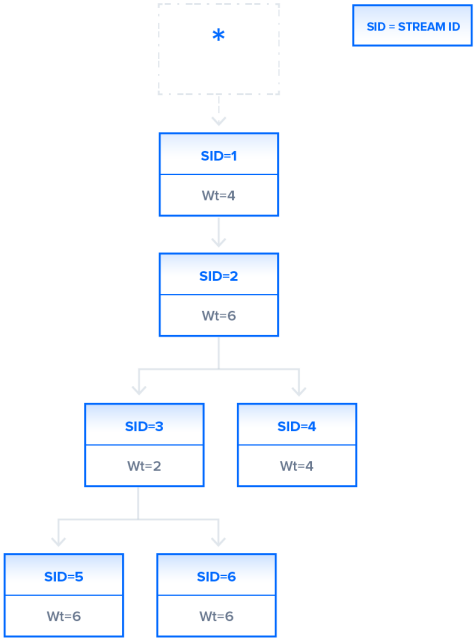

Figura 10: Arbre de dependències dels fluxos en HTTP/2
[www.digitalocean.com/community/tutorials/http-1-1-vs-http-2-what-s-the-difference]

Un altre mecanisme que permet al programador augmentar el control que se li atorga per a optimitzar la transmissió de la informació consistix a establir en el nivell d'aplicació un sistema de control de flux independent d'aquell utilitzat al nivell de transport. D'aquesta manera, les aplicacions client i servidora poden comunicar-se la finestra de recepció disponible per als fluxos [3].

HTTP/2 implementa un procediment anomenat *server push*, que permet al programa servidor enviar recursos que considera que el client va a necessitar sense que aquest els haja demanat explícitament amb anterioritat [4]. Quan el client rep aquest recurs pot decidir de mantindre'l o eliminar-lo. Per a informar al client dels recursos que el servidor desitja enviar-li i així evitar que els torne a demanar (a més de permetre-li rebutjar l'enviament), el servidor envia una trama especial (PUSH_PROMISE) al client amb aquesta informació.

Una característica important introduïda per HTTP/2 destinada a no malbaratar l'amplada de banda de manera innecessària consistix en la compressió de les capçaleres. Aquesta compressió actua de dues maneres. D'una banda, és capaç de comprimir les capçaleres de cada missatge, reduint així el seu tamany de forma considerable. D'una altra banda, si hi ha missatges que compartixen les mateixes línies de capçalera, és possible construir un índex compartit entre el client i el servidor en el qual s'emmagatzemen aquestes capçaleres compartides entre missatges. Com a conseqüència, quan s'envien missatges amb aquestes mateixes capçaleres, en lloc d'incloure tota la informació només caldrà incloure-hi el número de l'entrada de l'índex [3].

Finalment, en termes de seguretat, mentre que HTTP/2 no obliga a l'ús d'una connexió segura (mitjançant TLS), totes les implementacions del protocol forcen l'establiment d'un canal de comunicació xifrat mitjançant el protocol TLS, de manera que, tot i que tècnicament es podrien acceptar comunicacions no xifrades, els desenvolupadors han preferit que es rebutge qualsevol intent de connexió insegur.

2.4. HTTP/3

Fins a la versió HTTP/2, el protocol de nivell de transport utilitzat ha sigut TCP, donat que l'establiment formal de la connexió entre els extrems, l'entrega fiable dels paquets i els mecanismes de control de flux el convertixen en una bona opció per a transmetre la informació del Web.

No obstant això, al llarg dels anys han aparegut nous tipus de contingut servits a través d'Internet, ja siga la reproducció multimèdia, connexions en temps real, etc. que tenen uns requeriments d'entrega que no són completament satisfets per TCP. Fins ara, l'única

alternativa com a protocol de transport ha sigut UDP, que, no obstant això, no proporciona cap garantia ni control de la comunicació.

A més, no es preveia un panorama de les comunicacions tan ubic, on la quantitat de dispositius mòbils supera fins i tot la d'ordinadors. Aquest model de connexió presenta nous reptes, ja que és prou habitual que un dispositiu mòbil canvie d'una xarxa durant la seua utilització al llarg del dia. TCP presenta clars desavantatges en aquestes situacions, ja que no està pensat per a suportar connexions tan volàtils entre dispositius [5].

Al mateix temps, el model de transmissió implementat per HTTP/2, segons el qual s'utilitza una única connexió TCP per la qual passen diversos fluxos, té un límit en el benefici que és capaç de proporcionar, ja que la naturalesa paral·lela de la multiplexació d'HTTP/2 no és percebuda pels mecanismes d'entrega fiable de TCP, per la qual cosa, si es produïx una pèrdua d'un paquet pertanyent a un flux, tots els altres fluxos experimenten una parada encara que cap dels seus paquets haja sigut afectat per una pèrdua [6]. Açò suposa una limitació molt considerable en la capacitat que posseïx HTTP/2 per a millorar l'eficiència de les comunicacions des de la capa d'aplicació.

Per aquest motiu, es va fer palesa la necessitat de dur a terme canvis en capes més profundes, de manera que es va decidir adoptar un nou protocol de transport anomenat QUIC, sobre el qual es transmetrien els missatges del protocol HTTP. De nou, s'ha procurat mantindre la semàntica del protocol HTTP, mentre que la sintaxi canviarà en utilitzar-se un nou protocol [7]. Cal tindre en compte que no es perden tots els avanços desenvolupats en la versió HTTP/2; al contrari, l'ús de QUIC permet aprofitar i delegar tota la gestió de fluxos, missatges i trames a la capa de transport, recuperant així la divisió per capes parcialment difuminada amb HTTP/2. A la **Figura 11** s'aprecia una comparativa de la pila de protocols entre HTTP/1.1, HTTP/2 i HTTP/3.

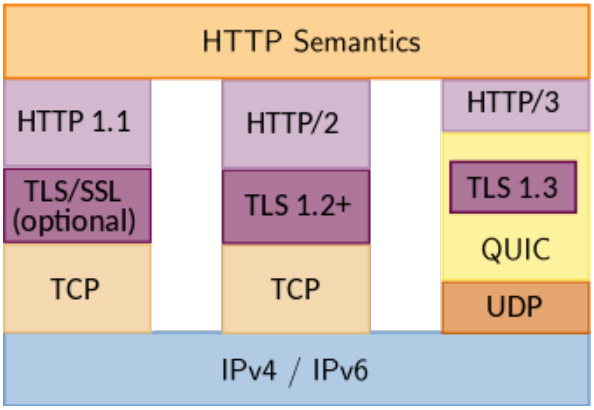

Figura 11: Pila de protocols amb HTTP/1.1, HTTP/2 i HTTP/3 [https://en.wikipedia.org/wiki/HTTP/3]

En general, HTTP/3 utilitza un mecanisme de gestió de la transmissió molt semblant al desenvolupat per HTTP/2, tot i que amb algunes modificacions per a adaptar-lo a QUIC, un protocol estretament lligat a HTTP/3 que tot i això pot ser usat per altres protocols d'aplicació com DNS [8]. Una de les majors diferències rau en el fet que el protocol de transport usat per HTTP/3, QUIC, funciona sobre UDP, a partir del qual es van implementant tots els mecanismes necessaris addicionals, mentre que HTTP/2 implementava el sistema de multiplexació sobre TCP. Açò proporciona un avantatge considerable a HTTP/3 sobre la seua versió precedent, ja que la possibilitat de proveir fiabilitat a nivell de flux i control de congestió a nivell de connexió millora el rendiment d'HTTP en comparació a l'ús de TCP. Gràcies a açò, desapareix el problema del bloqueig de cap de línia, ja que la transmissió i la gestió de les trames de cada flux és independent de la resta.

Pel que fa a la seguretat, mentre que a l'estàndard d'HTTP/2 no s'obligava a l'ús de comunicacions xifrades (tot i que en la pràctica sí que existia aquesta obligació), en HTTP/3 sí que se'n força l'ús, sent en aquest cas la versió 1.3 de TLS que ve integrada al protocol de transport QUIC [9]. Aquesta integració no només permet reduir la latència de les operacions criptogràfiques, sinó que també permet reduir els RTT necessaris per a establir una connexió segura.

Si aprofundim un poc en TLS, la versió 1.3 proporciona millores de rendiment front a les versions precedents que es resumixen en la **Figura 12**.

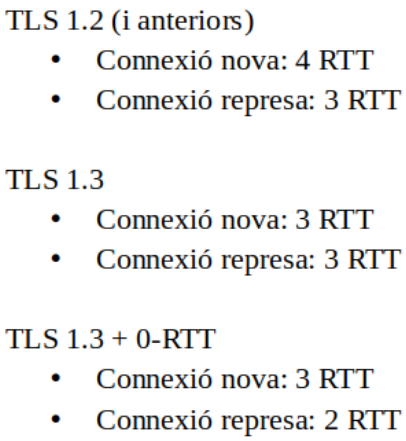

Figura 12: Comparació dels RTT de diferents versions de TLS

El progrés de TLS 1.2 a 1.3 ha sigut, en l'àmbit dels RTT, la reducció d'1 RTT per a les connexions noves, mentre que les connexions represes continuen consumint 3 RTT. Tenint en compte que el 60% de les connexions a un lloc web són noves i el 40% són connexions represes [10], sols s'aporten avantatges a les connexions noves. És per això

que, per a millorar també aquest 40% de connexions es va implementar el 0-RTT, que permet reduir 1 RTT addicional per a aquestes connexions represes.

El problema que presenta aquest mecanisme és que oferix unes garanties de seguretat lleugerament inferiors que si no s'emprara. Concretament, les peticions podrien ser vulnerables a un atac de repetició, segons el qual un atacant podria prendre la petició que fa ús del mecanisme 0-RTT i tornar a enviar-la al servidor, cosa que podria canviar l'estat d'aquest de forma no desitjada [10].

Fins a la versió HTTP/1.1 es pressuposava que la versió estàndard del protocol HTTP a utilitzar anava a ser HTTP/1.1, donat que es considerava que aquesta versió feia obsoletes les anteriors. No obstant això, no ocorre el mateix en el cas de les versions posteriors HTTP/2 i HTTP/3, ja que els seus models de funcionament tan diferenciats i la manca d'una adopció clarament predominant sobre HTTP/1.1 els obliga a conviure. És per aquest motiu que s'ha hagut de desenvolupar un mecanisme de negociació de la versió d'HTTP, donat que no es pot conéixer per endavant l'última versió suportada entre els extrems de la comunicació si no s'ha produït cap contacte previ entre el client i el servidor.

En el cas d'HTTP/2, aquesta negociació es du a terme al nivell de TLS, ja que, en la pràctica, totes les comunicacions estan obligades a utilitzar aquest protocol de xifratge (tot i que també existix un mecanisme de negociació a nivell d'HTTP mitjançant la capçalera Upgrade per a connexions no xifrades). No obstant això, aquesta negociació no servix per a HTTP/3, donat que funciona amb un protocol de transport diferent de TCP, i en el moment d'aquesta negociació TLS ja s'està utilitzant una connexió TCP [11].

Per aquest motiu, el procediment a seguir és establir una connexió TCP entre el client i el servidor, teòricament emprant la versió HTTP/2, després de la qual el servidor podrà avisar al client que disposa de la capacitat d'establir una connexió sobre QUIC i usar HTTP/3 mitjançant la capçalera alt-svc [11]. Després d'aquest descobriment inicial, el client pot recordar amb quin servidor va ser capaç d'establir una connexió QUIC i, en futures comunicacions, utilitzar directament aquest protocol amb dit servidor.

3. Estudi pràctic

3.1. Anàlisi dels cronogrames de càrrega

Aquest apartat ens permetrà d'observar com es reflectixen les diferències en la gestió de les peticions i lliuraments de recursos en les tres versions d'HTTP.

En concret, es presenten una sèrie de cronogrames que representen com es van demanant i servint els recursos entre un client i un servidor mitjançant cadascuna de les versions d'HTTP al llarg del temps.

A la **Figura 13** podem observar el cronograma corresponent a una connexió client - servidor que utilitza la versió HTTP/1.1; a la **Figura 14**, una connexió amb HTTP/2 i a la **Figura 15**, una connexió amb HTTP/3.

Un dels trets que destaquen si comparem els tres cronogrames és el fet que es pot percebre com al cronograma d'HTTP/1.1 els recursos van demanant-se un darrere de l'altre amb un cert espai de temps entre les peticions, sent aquest buit més evident en algunes ocasions que altres, mentre que als cronogrames d'HTTP/2 i HTTP/3 es veu clarament com es demanen els recursos en bloc, de manera que tots ells es demanen alhora.

Açò és degut al sistema de gestió de connexions present en cadascuna d'aquestes versions. Així, per exemple, mentre que HTTP/1.1 preveu la canalització i la concurrència de les connexions, hi ha un límit en la capacitat d'aquests mecanismes de paral·lelitzar la petició i recepció dels recursos. D'aquesta manera, HTTP/2 i HTTP/3 posseïxen mecanismes de gestió de fluxos independents entre sí, cosa que permet fer moltes més peticions simultànies que només estaran limitades principalment per la dependència existent entre els recursos.

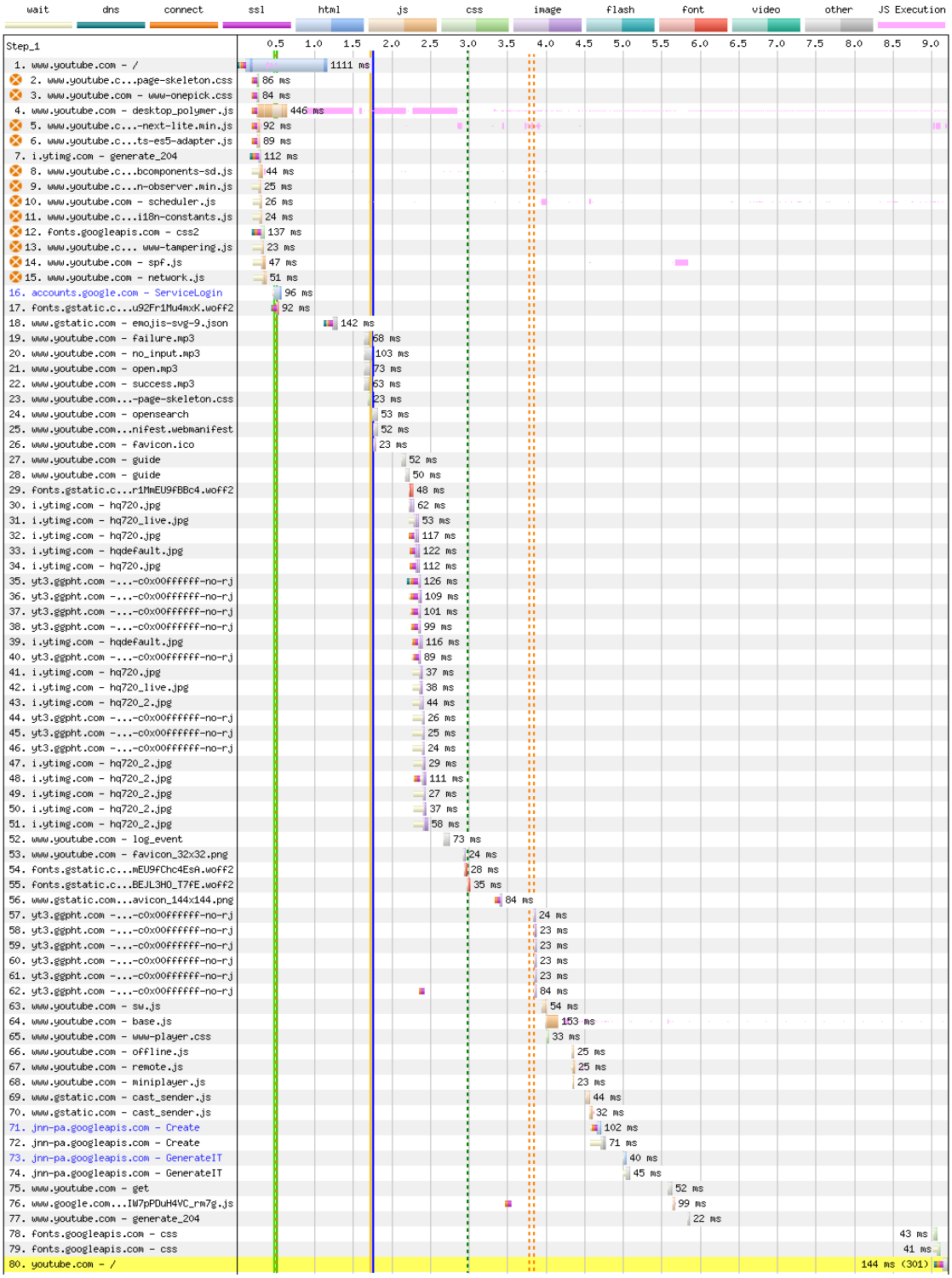

Figura 13: Cronograma d'una connexió HTTP/1.1

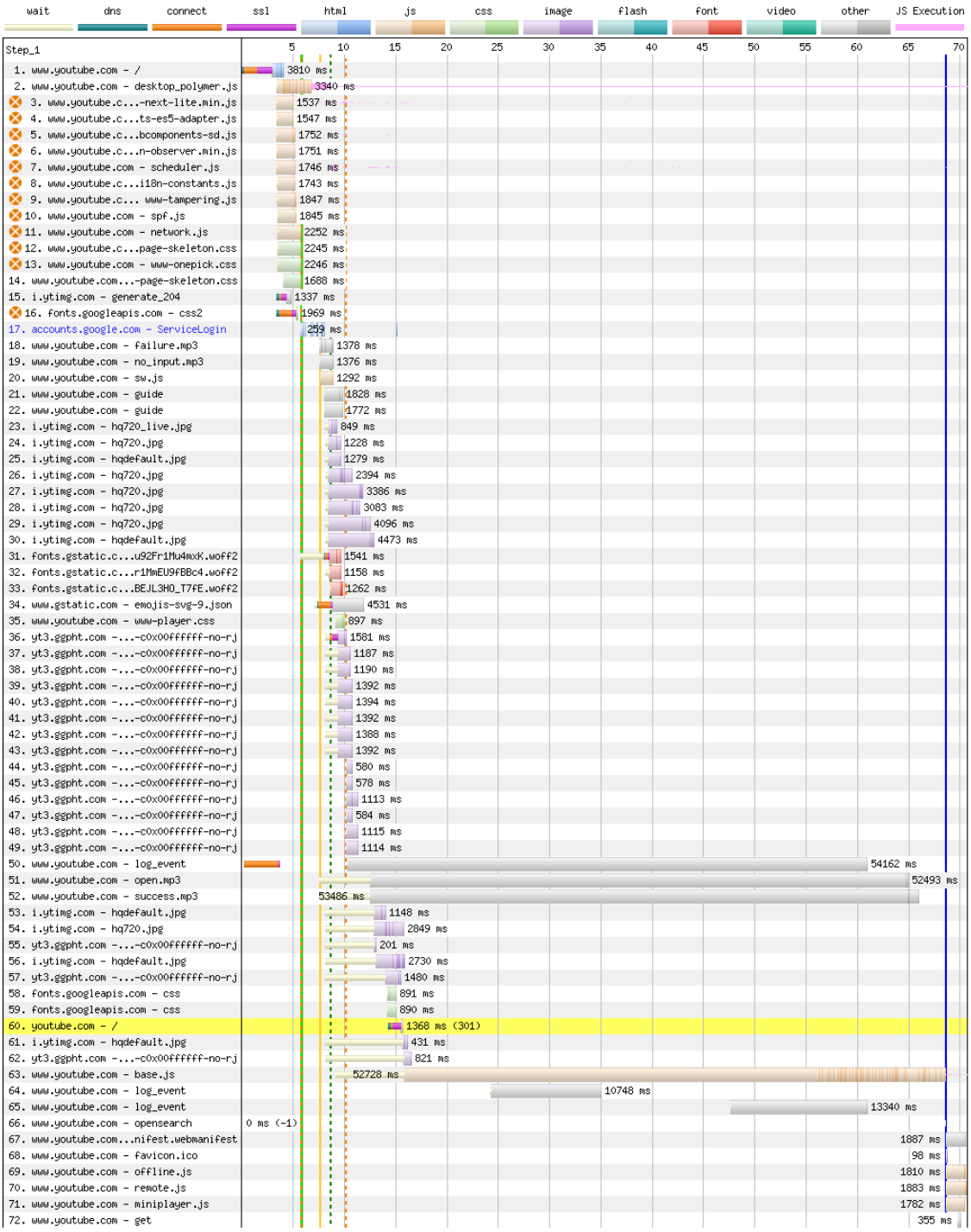

Figura 14: Cronograma d'una connexió HTTP/2

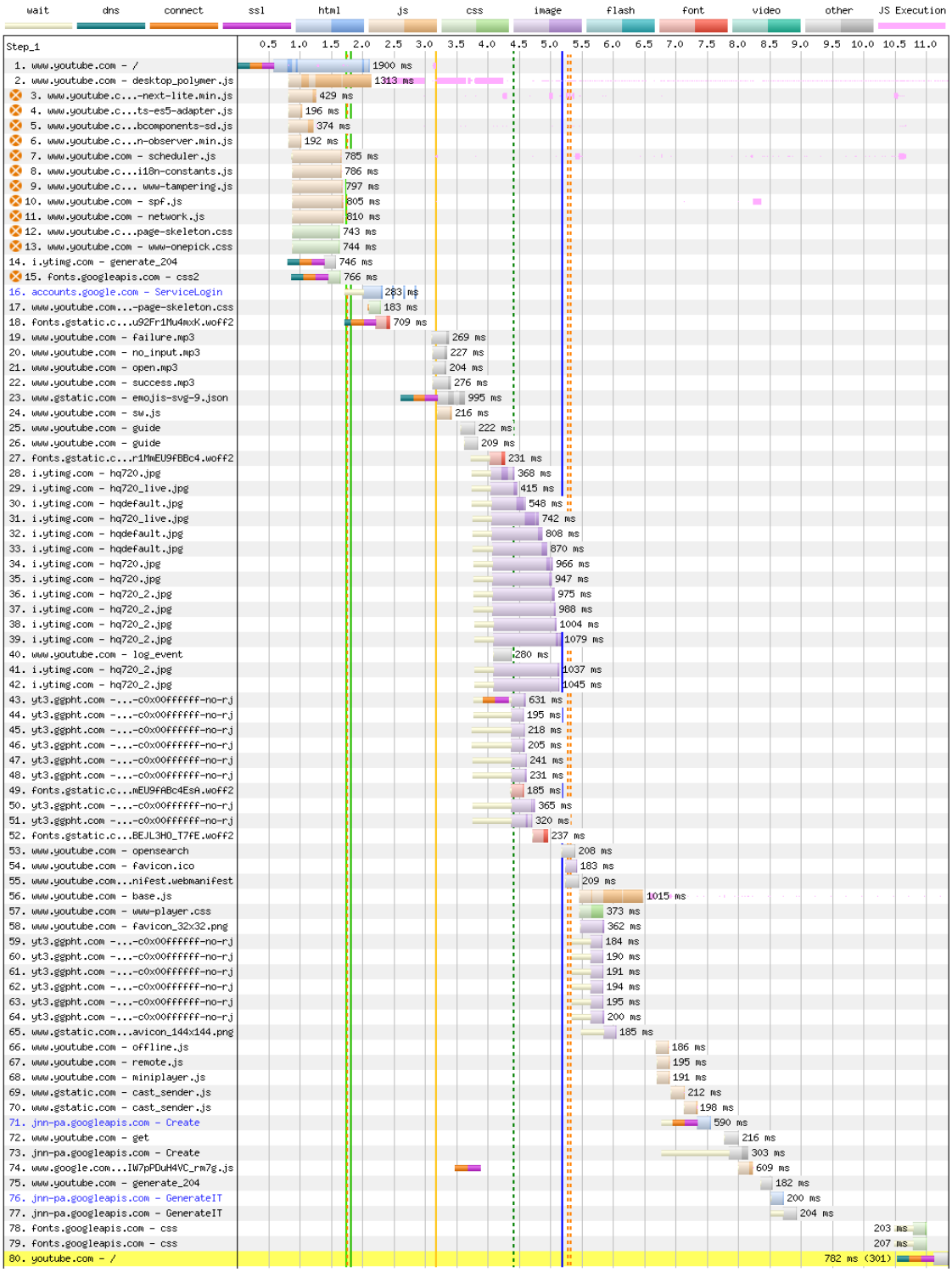

Figura 15: Cronograma d'una connexió HTTP/3

3.2. Anàlisi del temps de càrrega

En aquest apartat anem a exposar i explicar els resultats de rendiment obtinguts després d'analitzar dues pàgines web (www.pexels.com i www.youtube.com) en diverses situacions. Els casos que hem triat per a l'anàlisi han sigut la simulació d'una connexió ADSL, 4G i Fibra Òptica amb una latència baixa i alta que, al seu torn, pot presentar o no pèrdues de paquets per a cadascuna de les 3 versions principals d'HTTP (HTTP/1.1, HTTP/2 i HTTP/3).

Aquestes proves han sigut dutes a terme mitjançant l'eina en línia www.webpagetest.org, que ens permet triar el navegador que actua com a client, la localització del client, l'amplada de banda de pujada i baixada de dades, la latència, el percentatge de pèrdua de paquets i comandaments de línia addicionals per al navegador.

D'aquestes característiques, aquelles que no van a canviar al llarg de totes les proves són el navegador a usar, que en el nostre cas és Chrome, la localització del servidor que actua de navegador, que en el nostre cas és Milà (Itàlia), el percentatge de pèrdua de paquets, que pot ser 0% o 30%, o les opcions de línia de comandaments. En aquest últim cas, si volem fer ús d'HTTP/3 no hem d'especificar-ne cap; si volem usar HTTP/2 hem d'especificar *--disable-quic* (que desactiva HTTP/3); finalment, si volem usar HTTP/1.1 hem d'especificar *--disable-quic --disable-http2* (que desactiva HTTP/3 i HTTP/2 respectivament). Ens hauria agradat molt més haver pogut usar el navegador Firefox, ja que es tracta d'un projecte de codi obert, però malauradament no hi ha una manera fàcil de forçar l'ús d'una versió HTTP concreta a través de webpagetest.org a diferència de Chrome, al qual podem especificar-li les opcions anteriorment mencionades.

Per a dur a terme les proves, s'han de configurar altres aspectes com ara l'amplada de banda o les latències alta i baixa que depenen de la tecnologia concreta que es pretén simular. Perquè les dades recollides siguen el més representatives possibles de la realitat, hem triat el nombre màxim de repeticions dels tests, 9, que ens permetia la ferramenta per a calcular la mitjana de les mètriques analitzades. A continuació, es mostraran els resultats obtinguts del temps de càrrega per a cadascuna d'aquestes.

3.2.1. ADSL

La primera simulació consistix en una connexió client - servidor mitjançant un servei ADSL, amb una amplada de banda configurada de 5Mb/s per a la baixada i 1Mb/s per a la pujada. Així mateix, la latència baixa ha sigut configurada a 50ms, mentre que la latència alta, a 150ms.

En primer lloc, podem observar el temps de càrrega en ADSL per a una latència baixa a la **Figura 16**. El fet més destacat que ens permet observar és la gran diferència en el temps de càrrega que suposa patir o no pèrdua de paquets durant la transmissió.

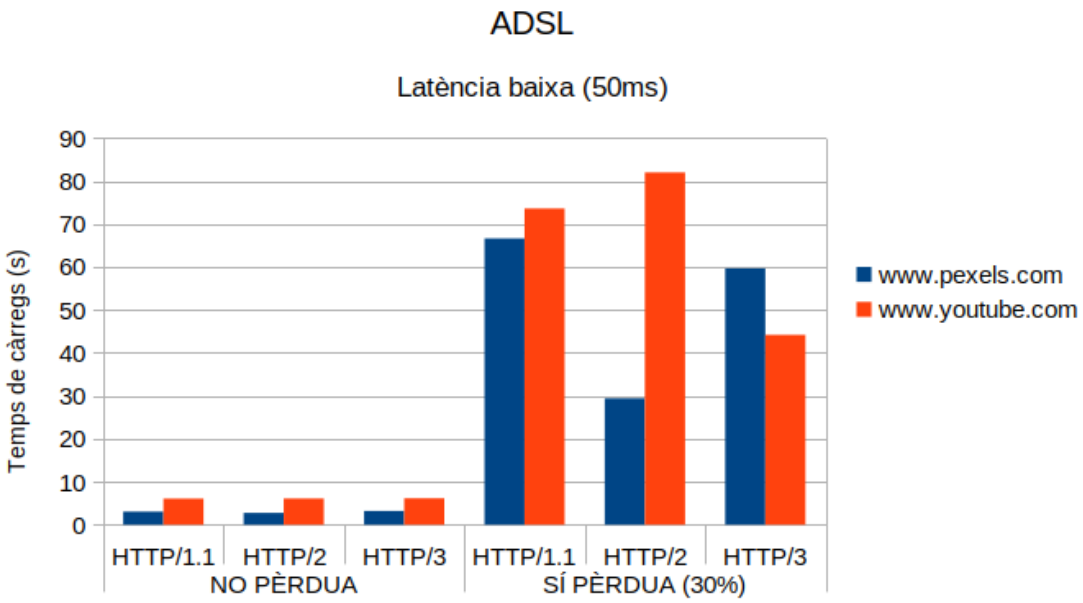

Figura 16: Temps de càrrega en ADSL amb latència baixa

Si comparem les gràfiques de la **Figura 17** i la **Figura 18**, que detallen el temps de càrrega en ADSL amb latència baixa sense i amb pèrdua, observem que, mentre que sense pèrdues no hi ha grans variacions de rendiment, sent la versió HTTP/2 lleugerament més ràpida en ambdós llocs web, quan en produïxen pèrdues hi ha una versió d'HTTP que és clarament més eficient que les altres, HTTP/2 en el cas de www.pexels.com i HTTP/3 en el cas de www.youtube.com.

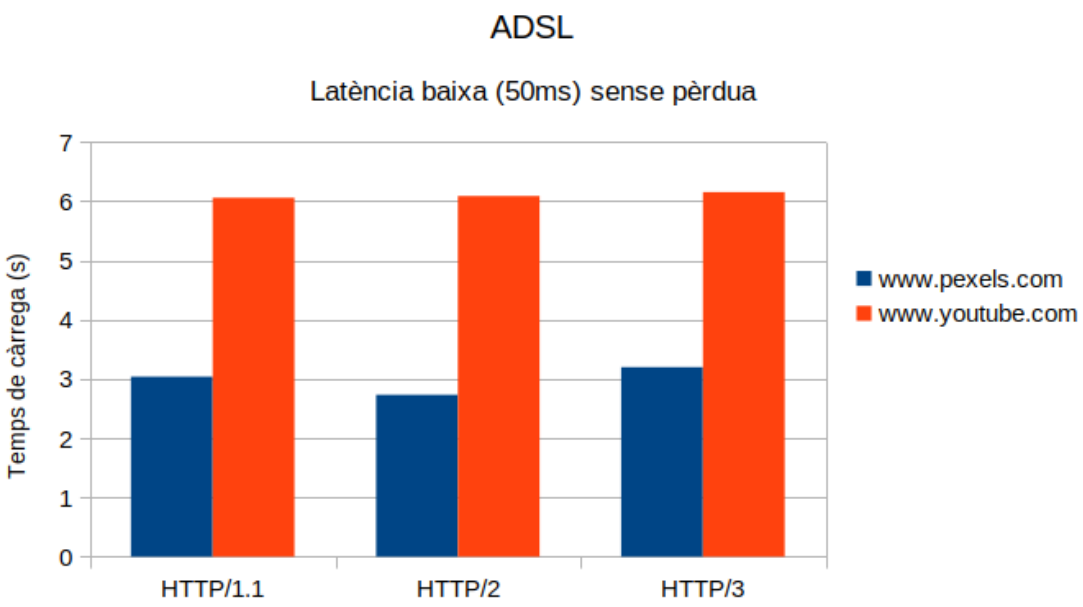

Figura 17: Temps de càrrega en ADSL amb latència baixa sense pèrdua

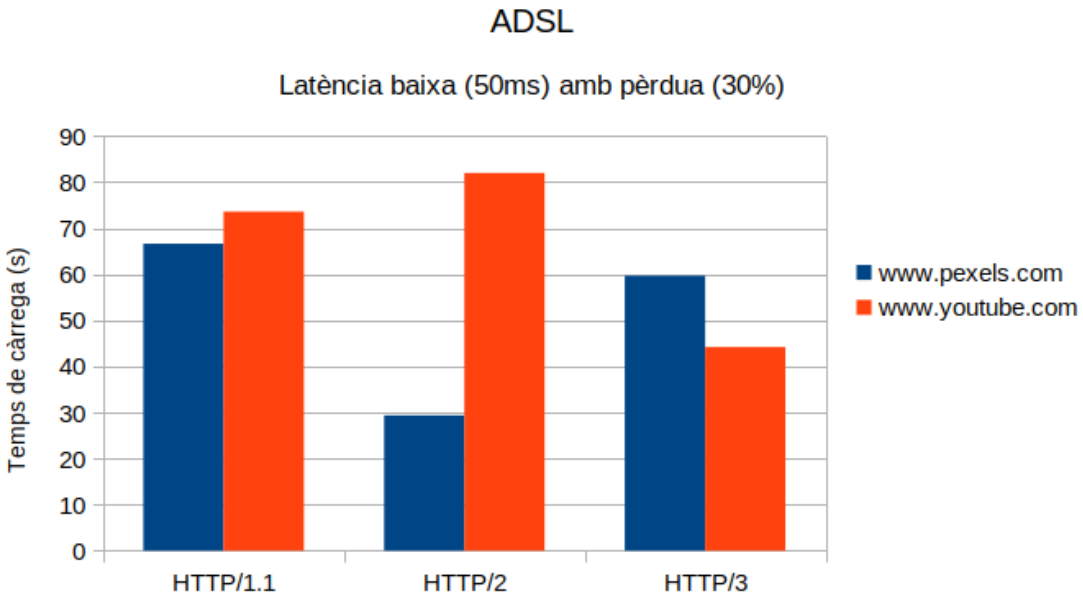

Figura 18: Temps de càrrega en ADSL amb latència baixa amb pèrdua

En segon lloc, podem observar el temps de càrrega en ADSL amb latència alta present a la **Figura 19**.

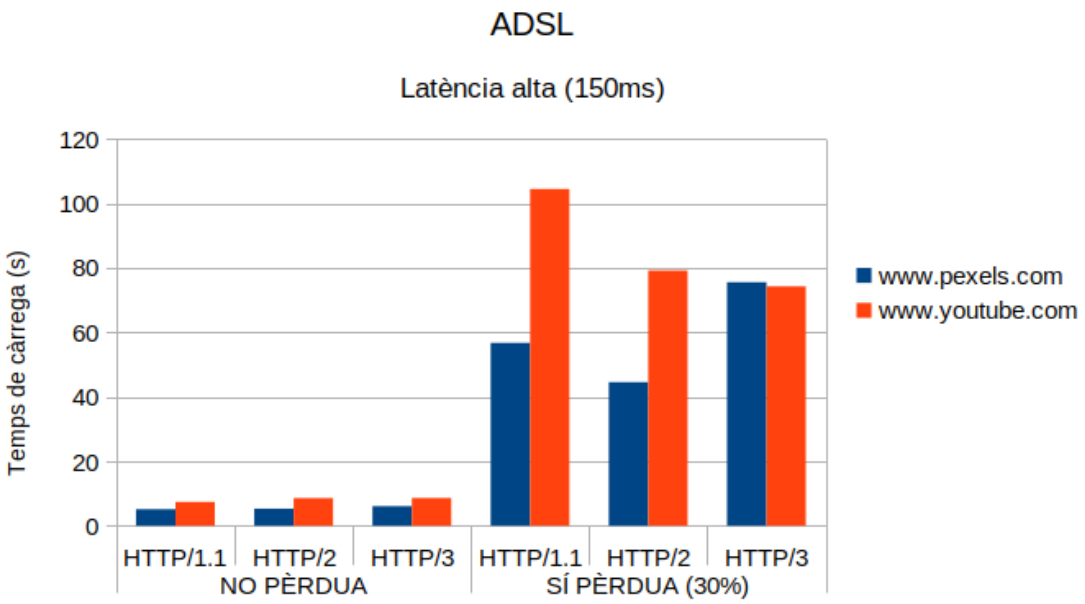

Figura 19: Temps de càrrega en ADSL amb latència alta

Si ara ens centrem en cada cas particular, podem vore a la **Figura 20** els resultats del temps de càrrega en ADSL amb latència alta i sense pèrdues amb més detall, on s'aprecia que la versió HTTP/1.1 és la més eficient per als dos llocs web.

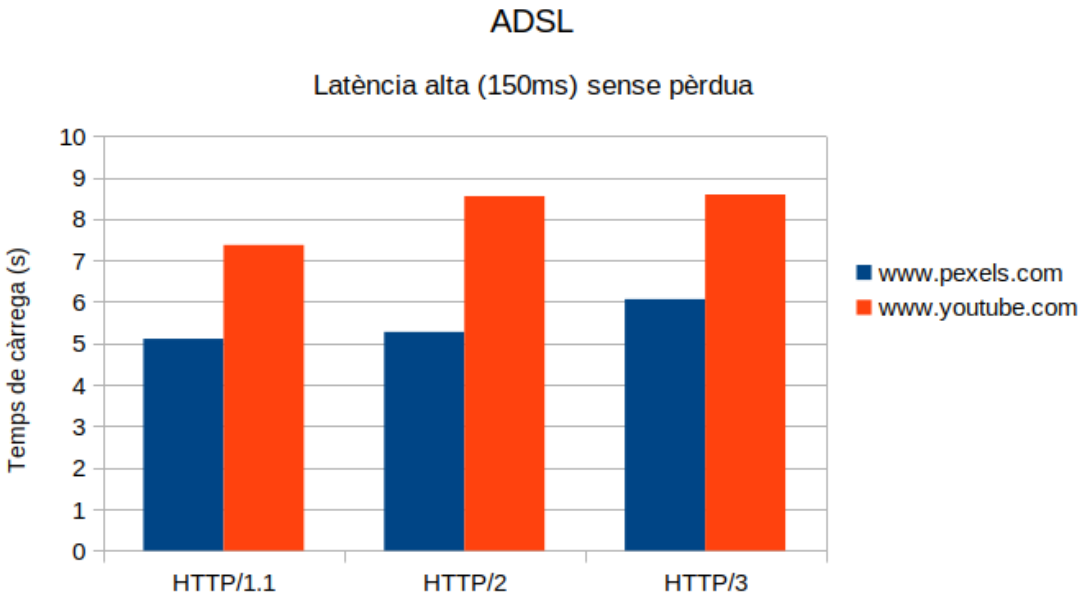

Figura 20: Temps de càrrega en ADSL amb latència alta sense pèrdua

En canvi, a la **Figura 21**, on tenim el temps de càrrega en ADSL amb latència alta i amb pèrdues, s'observa que la versió més eficient és HTTP/2 per a www.pexels.com i HTTP/3 per a www.youtube.com.

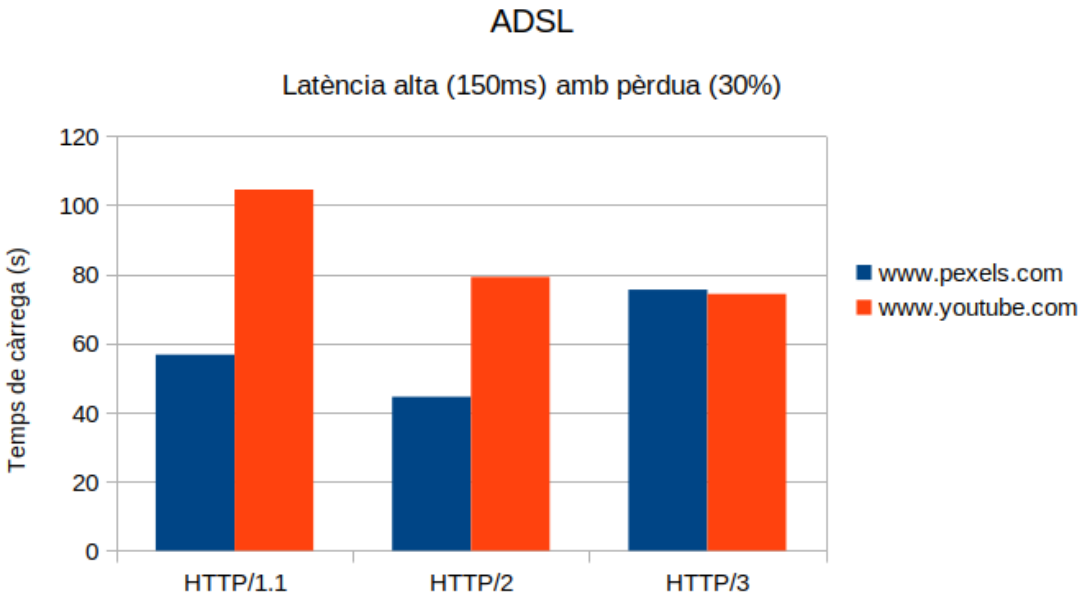

Figura 21: Temps de càrrega en ADSL amb latència alta amb pèrdua

3.2.2. 4G

La segona simulació consistix en una connexió client - servidor mitjançant un servei 4G, amb una amplada de banda configurada de 20Mb/s per a la baixada i 5Mb/s per a la pujada. Així mateix, la latència baixa ha sigut configurada a 30ms, mentre que la latència alta, a 170ms.

En primer lloc, podem observar el temps de càrrega en 4G per a una latència baixa a la **Figura 22**.

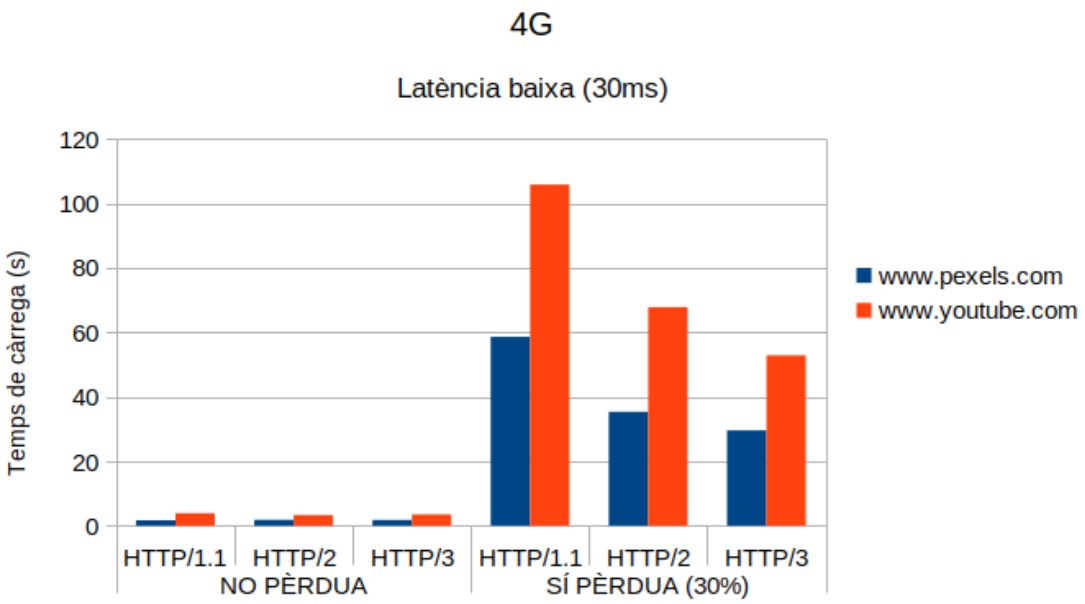

Figura 22: Temps de càrrega en 4G amb latència baixa

Si ens centrem en els resultats sense pèrdues de la **Figura 23**, es pot observar que, tot i que no hi ha grans variacions entre les versions d'HTTP, HTTP/1 és lleugerament més ràpid en www.pexels.com mentre que HTTP/2 ho és en www.youtube.com.

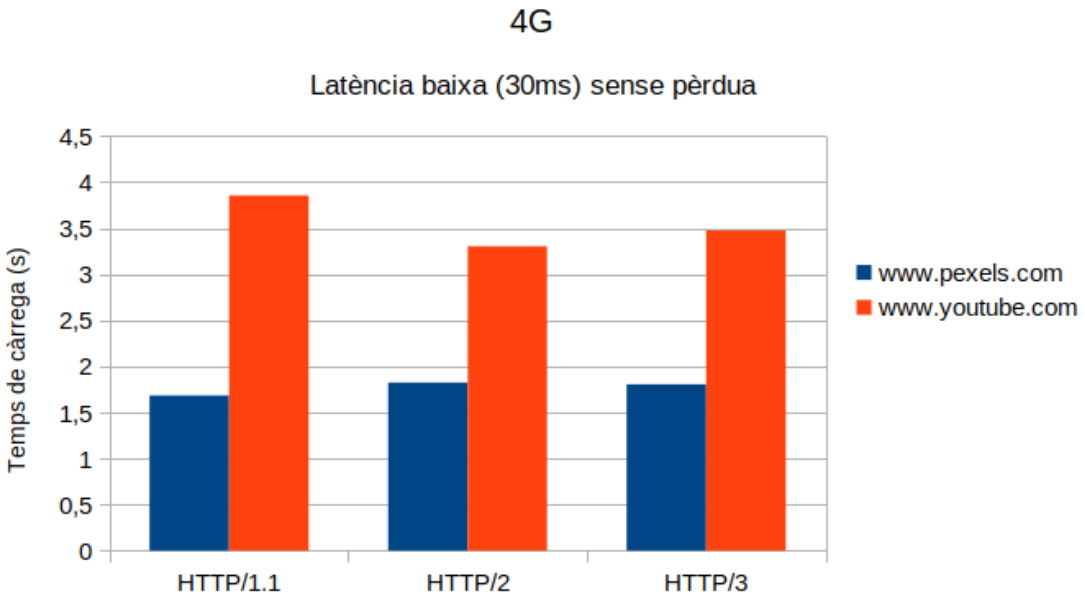

Figura 23: Temps de càrrega en 4G amb latència baixa sense pèrdua

En canvi, en els resultats amb pèrdues de la **Figura 24**, veiem una gran millora en el temps de càrrega de www.youtube.com per a HTTP/3, que també és la versió més eficient per a www.pexels.com tot i que amb una millora menor.

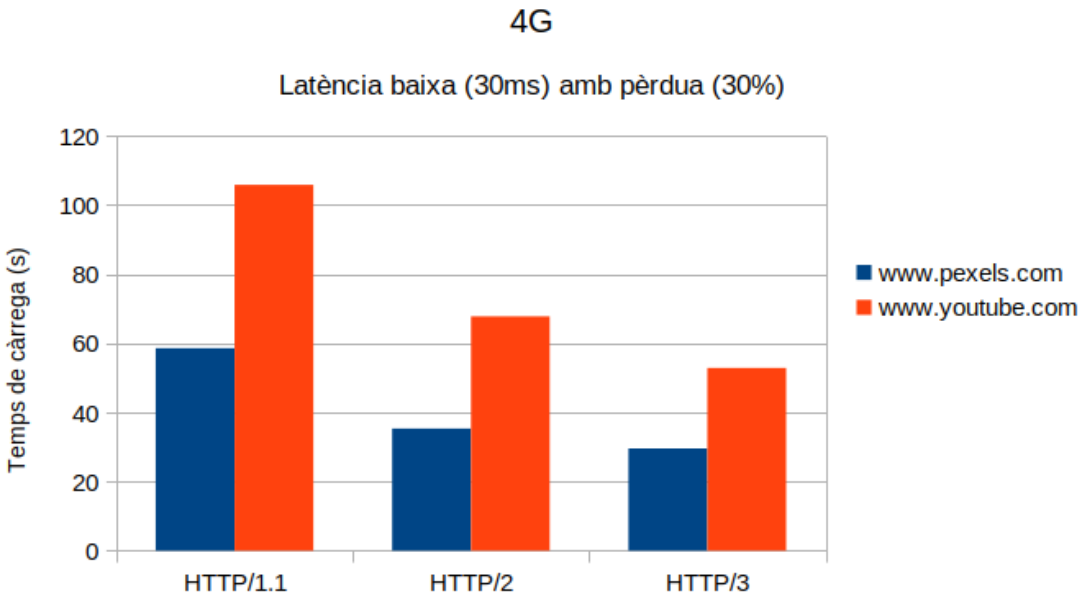

Figura 24: Temps de càrrega en 4G amb latència baixa amb pèrdua

En segon lloc, podem observar el temps de càrrega en 4G amb latència alta present a la **Figura 25**.

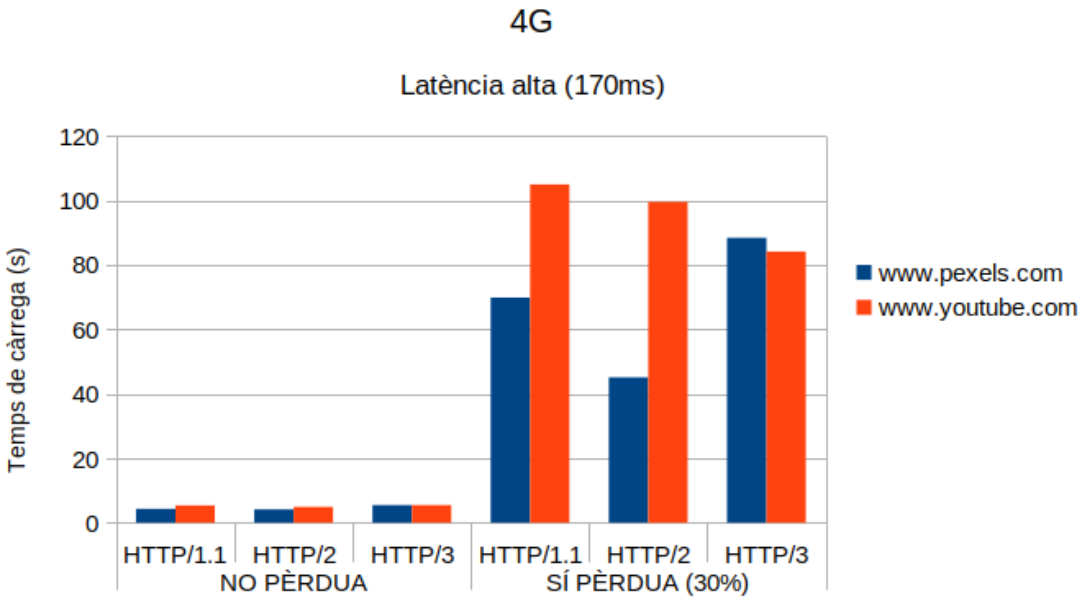

Figura 25: Temps de càrrega en 4G amb latència alta

Si ens centrem en els resultats sense pèrdues de la **Figura 26**, destaca el fet que la versió HTTP/3 presenta un temps de càrrega pitjor en tots dos llocs web, mentre que HTTP/2 és la versió més eficient per als dos.

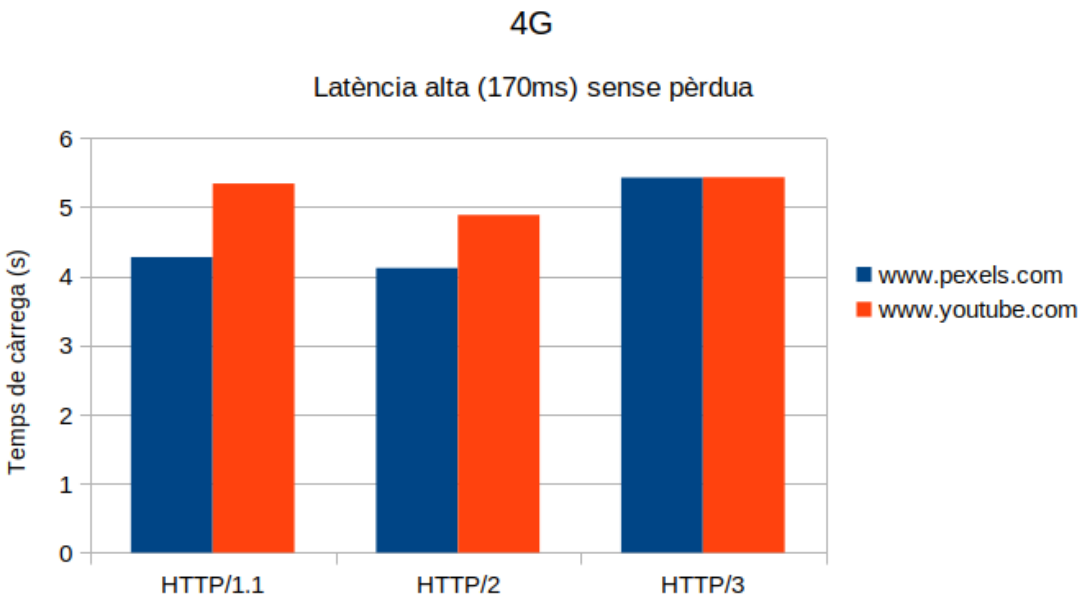

Figura 26: Temps de càrrega en 4G amb latència alta sense pèrdua

En els resultats amb pèrdues de la **Figura 27**, veiem que es repetix el patró segons el qual la versió HTTP/3 és la més eficient per a www.youtube.com mentre que la versió HTTP/2 l'és per a www.pexels.com.

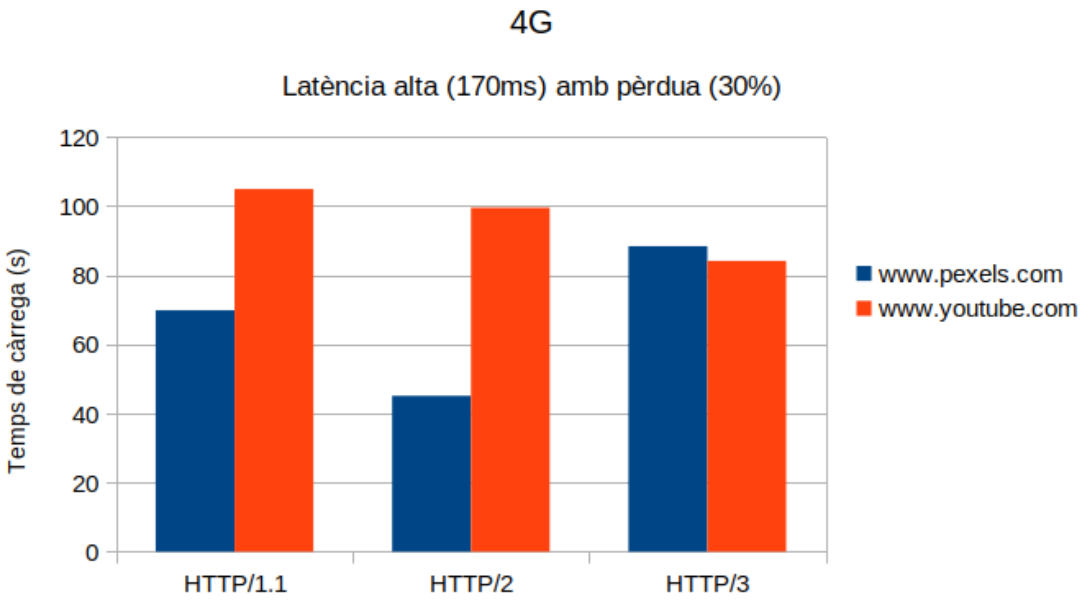

Figura 27: Temps de càrrega en 4G amb latència alta amb pèrdua

3.2.3. Fibra Òptica

La tercera simulació consistix en una connexió client - servidor mitjançant un servei de fibra òptica, amb una amplada de banda configurada de 300Mb/s per a la baixada i 300Mb/s per a la pujada. Així mateix, la latència baixa ha sigut configurada a 20ms, mentre que la latència alta, a 180ms.

En primer lloc, podem observar el temps de càrrega en fibra òptica per a una latència baixa a la **Figura 28**.

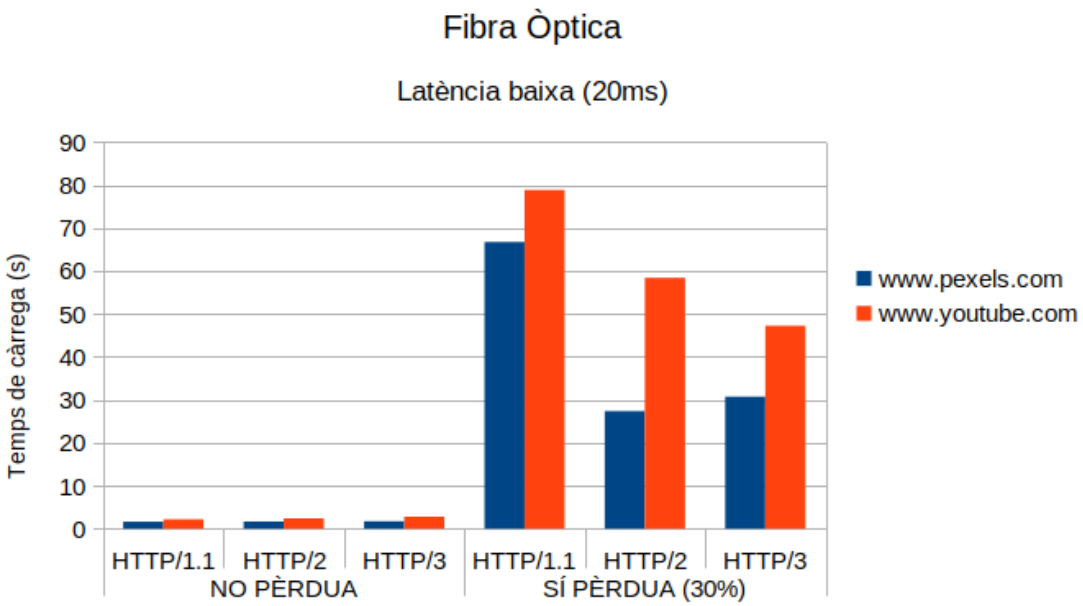

Figura 28: Temps de càrrega en fibra òptica amb latència baixa

Si ens centrem en els resultats sense pèrdues de la **Figura 29** destaca que el temps de càrrega siga lleugerament superior a mesura que augmentem de versió, tendència més acusada en www.youtube.com que a www.pexels.com. Així, en ambdós casos, tot i que amb una diferència relativament menuda, la versió més ràpida és HTTP/1.1.

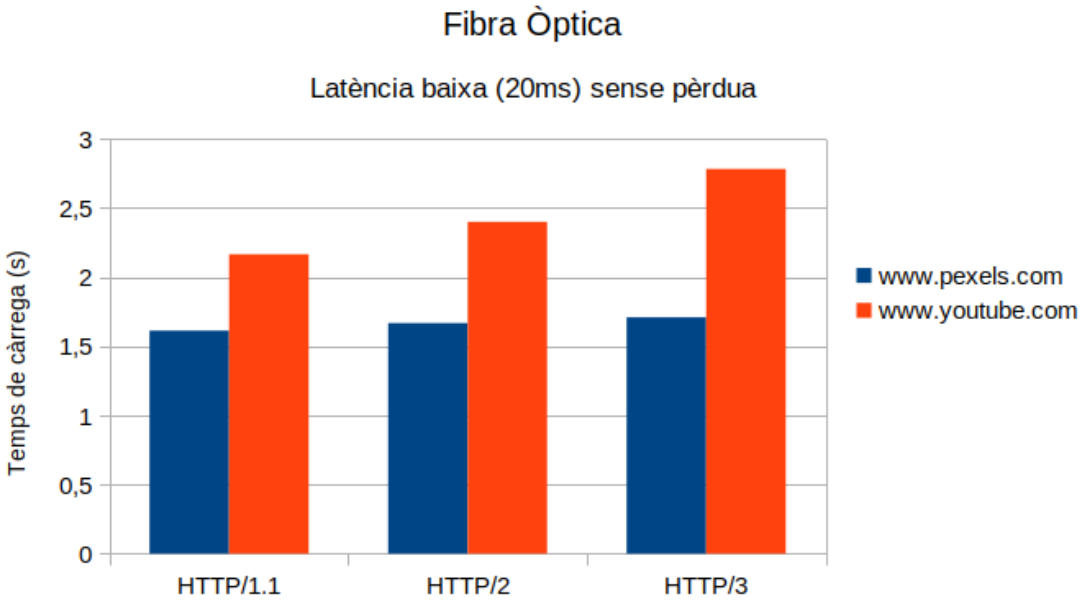

Figura 29: Temps de càrrega en fibra òptica amb latència baixa sense pèrdua

D'una altra banda, quan ens trobem en una situació de pèrdua de paquets com en la **Figura 30**, s'aprecia el procés contrari, de forma que per a www.youtube.com la versió amb un menor temps de càrrega és HTTP/3, mentre que per a www.pexels.com, HTTP/2 guanya lleugerament a la versió 3.

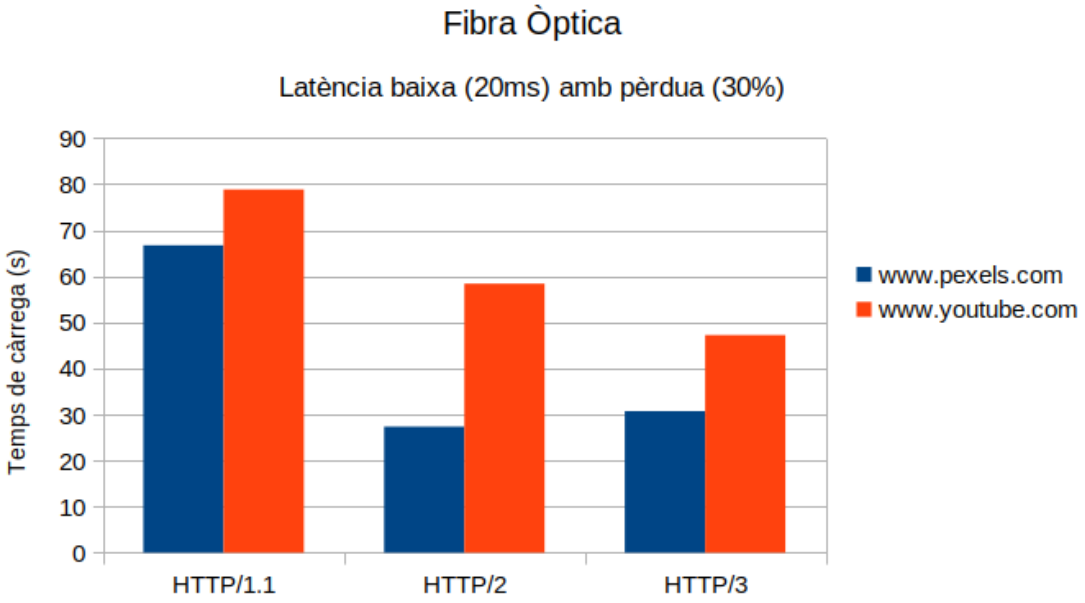

Figura 30: Temps de càrrega en fibra òptica amb latència baixa amb pèrdua

En segon lloc, podem observar el temps de càrrega en fibra òptica amb latència alta present a la **Figura 31**.

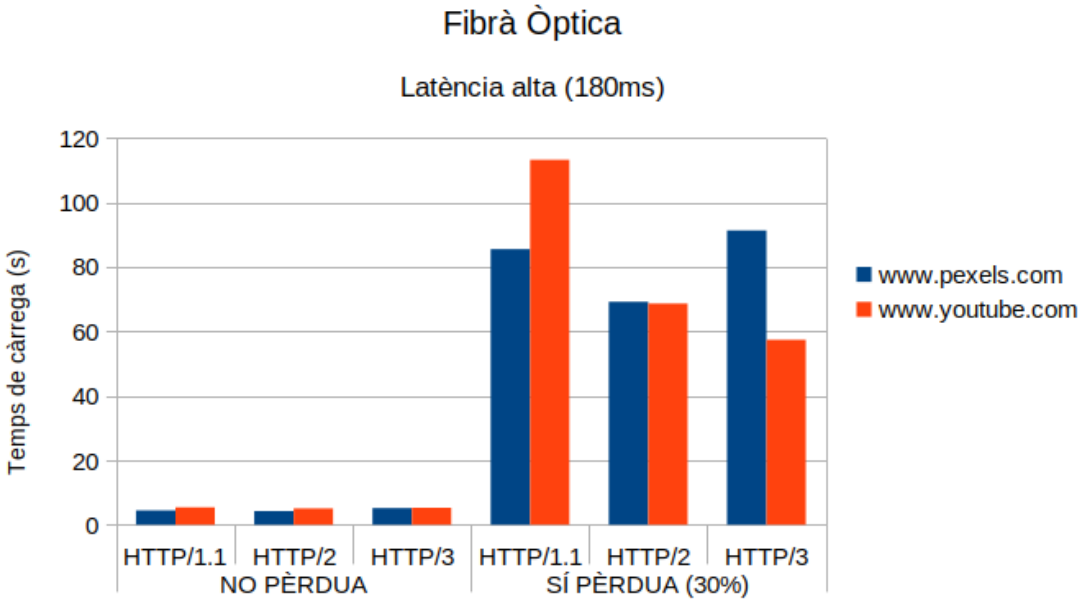

Figura 31: Temps de càrrega en fibra òptica amb latència alta

Si analitzem en els resultats sense pèrdues de la **Figura 32**, s'observa que la versió HTTP/2 és lleugerament més ràpida per als dos llocs web.

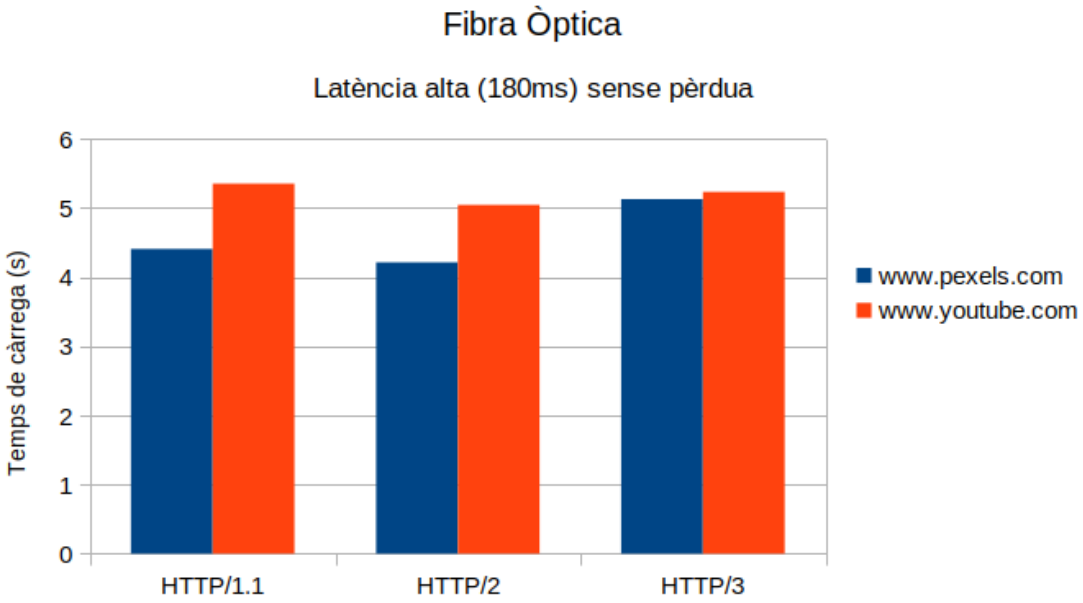

Figura 32: Temps de càrrega en fibra òptica amb latència alta sense pèrdua

No obstant això, en el cas de pèrdua de paquets de la **Figura 33** ens tornem a trobar la situació en què la versió HTTP/3 és la més eficient per a www.youtube.com i la versió HTTP/2 l'és per a www.pexels.com.

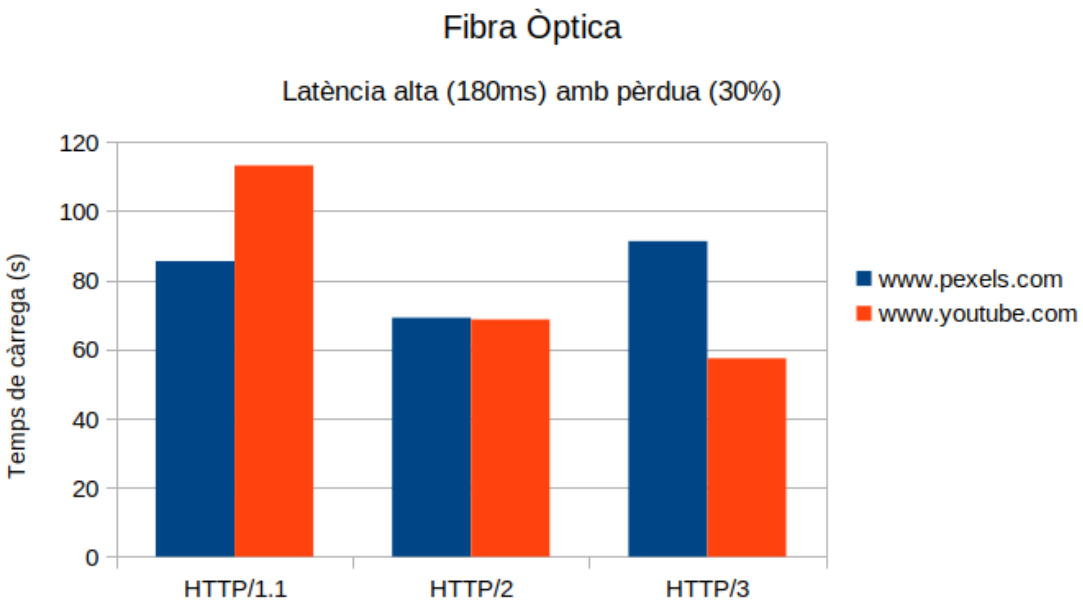

Figura 33: Temps de càrrega en fibra òptica amb latència alta amb pèrdua

3.3. Anàlisi dels resultats

En l'anàlisi dels cronogrames de càrrega s'han obtingut uns resultats que concorden amb el funcionament de cadascuna de les versions, ja que, com s'ha comentat a l'apartat 3.1., la gestió de les connexions i els recursos permet que hi haja molts més recursos que es poden demanar al mateix temps en les versions 2 i 3 a diferència de la versió 1.1, cosa que explica les diferències en la forma dels cronogrames; el de les versions 2 i 3 presenta grans blocs de recursos demanats tots alhora, mentre que en la versió 1.1 es percep una corba més definida, ja que els recursos s'han de demanar un darrere l'altre quan s'arriba al màxim de connexions TCP establides.

En l'anàlisi del temps de càrrega, no obstant això, s'han obtingut uns resultats que no eren totalment els esperats. Donades les millores que una versió d'HTTP introduïx en comparació amb les precedents, és sensat pensar que en augmentar la versió el temps d'espera serà menor i, per tant, el funcionament del protocol estarà més optimitzat. Tanmateix, s'han obtingut resultats dispars per als dos llocs web analitzats en aquesta secció. Així, es pot observar que la versió HTTP/2 funciona millor en www.pexels.com mentre que HTTP/3 funciona millor en www.youtube.com. Açò ens fa pensar que no només són importants les millores que aporten els protocols en si, sinó també que la

implementació dels servidors estiga optimitzada al màxim per a esta versió. D'aquesta manera, és molt probable que www.pexels.com estiga més optimitzat per a HTTP/2, mentre que www.youtube.com estiga més optimitzat per a HTTP/3.

De fet, segons un article de CloudFlare, «el funcionament d'HTTP/3 suposa una sobrecàrrega a la part client i servidora addicional que caldrà optimitzar amb el pas del temps» [5]. Aquesta sobrecàrrega pot deure's en gran mesura al fet que totes les connexions han de ser xifrades obligatòriament, procés que suposa un consum addicional de temps, dades intercanviades i capacitat de processament per a l'algoritme de xifratge de la informació. Però no només això, ja que s'ha de tindre en compte l'optimització duta a terme en els sistemes operatius que implementen tant TCP com UDP, ja que, donat que TCP ha rebut molt més ús i atenció, està més optimitzat que UDP (sobre el qual es basa QUIC) [12].

Si observem el tipus de servidors de què disposen www.pexels.com i www.youtube.com (per exemple mirant el valor de la capçalera «server» en les respostes a les peticions), podem adonar-nos que el primer utilitza servidors de CloudFlare, mentre que el segon empra servidors anomenats «ssfe», que hem descobert que es tracta de servidors propis de Google. Amb aquesta informació podríem deduir que la implementació del protocol HTTP/3 junt amb QUIC ha aconseguit un nivell d'optimització acceptable, doncs, al menys, és més eficient que la seua implementació de la versió HTTP/2. En canvi, en el cas de CloudFlare, donat que el rendiment d'HTTP/3 amb QUIC ha sigut generalment pitjor que amb HTTP/2, podem dir que no han acabat d'optimitzar-lo satisfactòriament. De fet, donat que Google ha tingut un paper essencial en el desenvolupament de la versió original de QUIC a l'any 2012, és lògic pensar que té molta més experiència amb el maneig d'aquest protocol i, per tant, més recorregut a l'hora d'optimitzar-lo. Per contra, CloudFlare no ha introduït suport a HTTP/3 fins al 2020, cosa que li dóna un clar desavantatge a l'hora de saber com optimitzar-lo al mateix nivell de Google.

L'optimització serà un factor important per a l'èxit del protocol, ja que si no es demostren millores significatives, es podria desencoratjar l'adopció del protocol.

4. Conclusions

HTTP/3 representa la culminació de dècades de canvis destinats a millorar la implementació original del protocol HTTP sorgida als anys 90. Canviar el funcionament dels protocols de la pila de protocols que formen Internet no és una tasca fàcil, ja que una immensitat de serveis construïts a sobre en depenen. No obstant això, HTTP és un protocol situat al nivell d'aplicació, que al cap i a la fi és al cim de la pila de protocols, per la qual cosa presenta una major facilitat per a aplicar-li canvis.

De la versió HTTP/1.1 hem obtingut una implementació completa, estable i prou flexible per a permetre el seu funcionament en un context tan canviant com el web; de fet, així ho avalen el llarg període en què ha sigut l'última versió oficial del protocol. Tanmateix, al llarg dels anys han sorgit nous casos d'ús que ni tan sols s'imaginaven al moment de la seua concepció, la qual cosa ha fet necessària una remodelació del seu funcionament.

Després d'anys de desenvolupament, va sorgir la versió HTTP/2, que introduïx una visió radicalment diferent en el model de connexió i la gestió de la comunicació entre el client i el servidor. Això sí, encara sobre la mateixa base que la versió precedent, és a dir, TCP. Açò va suposar l'aparició de diverses limitacions produïdes no pel disseny d'HTTP/2, sinó pel funcionament intrínsec del protocol de transport subjacent.

Per aquest motiu, es va cercar un mode de sobrepassar les limitacions de TCP mitjançant el desenvolupament d'un nou protocol basat sobre UDP, QUIC. Aquest protocol oferix totes les característiques que, acoblades a les noves funcionalitats ja introduïdes per d'HTTP/2, permeten aprofitar-les al màxim. Aquesta nova versió d'HTTP basada en HTTP/2 i QUIC ha rebut el nom d'HTTP/3.

Així doncs, l'anàlisi teòrica del funcionament de les versions d'HTTP ha sigut de gran ajuda a l'hora d'entendre l'evolució de la implementació de cadascuna d'aquestes i entendre com aquests canvis poden afectar al millor funcionament del protocol.

Finalment, l'anàlisi pràctica, al seu torn, ha sigut necessària a l'hora de comprovar experimentalment l'impacte que aquestes modificacions tenen en el rendiment. En aquest sentit, ens ha sorprés observar la importància de les implementacions de les versions d'HTTP als servidors i no sols el desenvolupament dels estàndards, ja que moltes característiques que s'aprecien com a millores poden acabar no funcionant tan bé com es preveia en un principi degut a la manca d'un bon procés d'optimització.

5. Referències

[1] David Gourley i Brian Totty i Marjorie Sayer i Anshu Aggarwal i Sailu Reddy, 2002. *HTTP: The Definitive Guide*. [en línia] O'Reilly Media, Inc. Disponible en: https://learning.oreilly.com/library/view/http-the-definitive/1565925092/

[2] R. Fielding, Ed. i J. Reschke, Ed., 2014. *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Rounting, RFC 7230*. [en línia] RFC Editor. Disponible en: https://www.rfc-editor.org/rfc/rfc7230.txt

[3] ABCOM, 2022. *HTTP/1.1 vs HTTP/2: What's the Difference?*. [en línia] DigitalOcean. Disponible en: https://www.digitalocean.com/community/tutorials/http-1-1-vs-http-2-what-s-the-difference

[4] M. Belshe i R. Peon i M. Thomson, Ed., 2015. *Hypertext Transfer Protocol Version 2 (HTTP/2), RFC 7540*. [en línia] RFC Editor. Disponible en: https://www.rfc-editor.org/rfc/rfc7540.txt

[5] CloudFlare, n.d. *What is HTTP/3?*. [en línia] CloudFlare. Disponible en: https://www.cloudflare.com/learning/performance/what-is-http3/

[6] M. Bishop, Ed., 2022. *HTTP/3*. [en línia] RFC Editor. Disponible en: https://www.rfc-editor.org/rfc/rfc9114.txt

[7] Lucas Pardue, 2019. *HTTP/3: From root to tip*. [en línia] CloudFlare. Disponible en: https://blog.cloudflare.com/http-3-from-root-to-tip/

[8] Mike Bishop, 2021. *HTTP/3 and QUIC: Past, Present, and Future*. [en línia] Akamai. Disponible en: https://www.akamai.com/blog/performance/http3-and-quic-past-present-and-future

[9] IONOS, 2020. *HTTP/3: the next Hypertext Transfer Protocol explained simply*. [en línia] IONOS. Disponible en: https://www.ionos.com/digitalguide/hosting/technical-matters/http3-explained/

[10] Nick Sullivan, 2017. *Introducing Zero Round Trip Time Resumption (0-RTT)*. [en línia] CloudFlare. Disponible en: https://blog.cloudflare.com/introducing-0-rtt/

[11] Viscomi, R. i Meyer, E.A. i Manna, S. i Goel, N. i Lakatos, A. i Alderson, J. i Volpini, A. i Portis, E. i Sillars, D. i Stepanyan, I. i altres, 2021. *The 2021 Web Almanac: HTTP Archive's annual state of the web report*. [en línia] HTTP Archive. Disponible en: https://books.google.es/books?id=7zFYEAAAQBAJ

[12] Tonino Jankov, 2022. *Qué Es HTTP/3 – Información Sobre el Nuevo y Rápido Protocolo Basado en UDP*. [en línia] Kinsta. Disponible en: https://kinsta.com/es/blog/que-es-http3/

6. Annexos

Annex 1: Objectius de Desenvolupament Sostenible

Grau de relació del treball amb els Objectius de Desenvolupament Sostenible (ODS).

Objectius de Desenvolupament Sostenible		Alt	Mitjà	Baix	No Procedix
1	Fi de la pobresa.				x
2	Fam zero.				x
3	Salut i benestar.		x		
4	Educació de qualitat.	x			
5	Igualtat de gènere.				x
6	Aigua neta i sanejament.				x
7	Energia assequible i no contaminant.				x
8	Treball decent i creixement econòmic.		x		
9	Indústria, innovació i infraestructures.		x		
10	Reducció de les desigualtats.			x	
11	Ciutats i comunitats sostenibles.		x		
12	Producció i consum responsables.				x
13	Acció pel clima.		x		
14	Vida submarina.				x
15	Vida d'ecosistemes terrestres.				x
16	Pau, justícia i institucions sòlides.				x
17	Aliances per a assolir objectius.				x

Taula 1: Relació del treball amb els ODS

Els Objectius de Desenvolupament Sostenible (ODS) van ser adoptats per les Nacions Unides l'any 2015 com una crida universal a actuar per acabar amb la pobresa, protegir el planeta, i assegurar que per al 2030 tothom gaudisca de pau i prosperitat. Els 17 ODS estan integrats entre si, és a dir, l'actuació sobre un d'ells té repercussions sobre altres. A més, els països han acordat ajudar en major mesura aquells que encara estiguen més lluny d'assolir aquests objectius.

En la **Taula 1** se n'han llistat els 17, marcant el grau de relació que tenen cadascun d'ells amb aquest treball. Aquells que estan més relacionats són:

- **Salut i benestar**, l'ús de la versió 3 d'HTTP pot permetre l'intercanvi d'informació entre sistemes d'una manera més ràpida i eficient, la qual cosa suposa un millor funcionament de programes i sistemes mèdics que facen ús de la xarxa, cosa que permet oferir un millor servei als pacients i investigadors.

- **Educació de qualitat**, HTTP/3 suposa una reducció de les latències presents en les comunicacions a través d'Internet, la qual cosa pot millorar l'experiència en l'aprenentatge dels alumnes, ja que aplicacions destinades a l'aprenentatge en línia (videoconferències, compartició de pantalla, pissarres interactives, enviament de documents, etc.) no patiran tants problemes de connexió i tindran una major resiliència a possibles situacions de xarxa desfavorables.

- **Treball decent i creixement econòmic**, no hi ha dubte que el teletreball ha patit un augment vertiginós quant al seu ús els darrers anys degut al canvi brusc que ens hem vist obligats a fer a causa de la pandèmia, la qual cosa ha suposat una sobrecàrrega de les eines de teletreball durant els primers mesos en què es va popularitzar el seu ús. Gràcies a aquesta nova versió del protocol es podrien evitar molts dels problemes de sobrecàrrega tant dels servidors com de la xarxa. A més, l'oportunitat de dur a terme activitats econòmiques en línia permet una ubiqüitat dels negocis que no estava tan estesa anteriorment, possibilitat així noves oportunitats per a fer negocis.

- **Indústria, innovació i infraestructures**, aquest ODS està relacionat amb el present treball perquè la major eficiència del protocol permet agilitzar les comunicacions i processos industrials, al mateix temps que contribuïx a una millora de l'intercanvi d'informació a nivell científic i tecnològic.

- **Reducció de les desigualtats**, les millores d'aquesta versió del protocol poden ajudar les persones més desfavorides, que es troben en llocs amb una connexió dolenta i amb pocs recursos per a adquirir dispositius més potents, a aprofitar els recursos i oportunitats que oferix Internet per a comunicar-se, formar-se i desenvolupar el seu treball d'una manera més eficient, reduint, per exemple, la barrera entre els més rics i els més pobres o ajudant a disminuir la segregació de certs grups socials.

- **Ciutats i comunitats sostenibles**, la millora de la fiabilitat de les comunicacions comporta un major ús de ferramentes de teletreball i cooperació a distància, cosa que permetrà, per exemple, disminuir els desplaçaments que no siguen necessaris, amb la qual cosa el trànsit serà menor.

- **Acció pel clima**, ja que la reducció de les latències i l'aprofitament de l'amplada de banda permetrà disminuir l'energia utilitzada pels sistemes.