Evolución de los protocolos web nel so 25 aniversariu

Por **David Melendi Palacio, Xabiel García Pañeda & Roberto García Fernández**
Departamentu d'Informática
Universidá d'Uviéu

Los servicios web son los más caraute-rísticos de lo que podemos considerar l'Internet modernu. Güei hai un porcen-taxe de probabilidá mui altu de que cuales-quier usuariu al que se-y entrugue polo que ye Internet, conteste con una descripción ase-meyada a lo que pue ser un serviciu web. Ye más, tolos sistemes operativos modernos traen un restolador instaláu. Ye tala la importancia d'estos servicios y de les aplicaciones que-yos sirven de base, qu'inclusive Microsoft tuvo que camudar la so política de distribución del res-tolador Internet Explorer col sistema operativu Windows per sentencia xudicial (Elzinga *et al*., 2001).

Los servicios web sírvennos pa divertinos, pa informanos, pa deprender, pa comunicanos, pa mercar, pa facer xestiones personales, etc. Ensin sabelo, les teunoloxíes que-yos sirven de base tamién s'empleguen en televisiones, mó-viles, electrodomésticos y en toa triba de dis-positivos. Pero, en resume ¿qué ye un serviciu web? Pues un serviciu web pue definise como un sistema d'intercambiu d'información multi-media. Si vamos al casu más cenciellu, nun ye más qu'un serviciu nel qu'un programa manda un mensaxe a otru pa descargar un ficheru.

Sala de sirvidores del CERN - Conseil Européen pour la Recherche Nucléaire. Semeya: Florian Hirzinger.

Fundamentalmente, estos servicios operen sobre la base de tres pegollos teunolóxicos:

- HTML: Ye un llinguaxe pa maquetar documentos. De forma declarativa, permite especificar qué parte del testu ye un parágrafu, qué ye una llista, una tabla, etc.
- URI: Ye un sistema de referencies que permiten saber el nome d'un recursu o la so ubicación.
- HTTP: Ye un protocolu que define cómo son los mensaxes qu'intercambien un restolador y un sirvidor web.

El 20 d'avientu de 1990 arranca nel CERN el primer sirvidor web y espublízase la primer páxina web

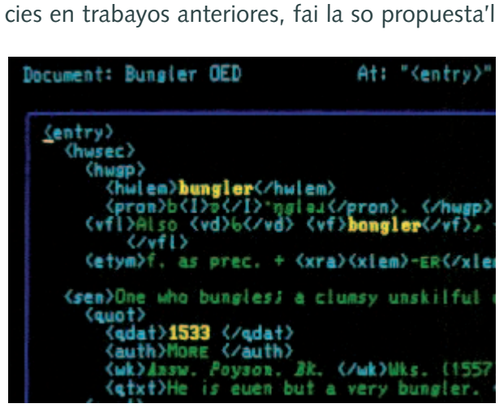
Pa comprender l'orixe d'estes teunoloxíes tenemos que pensar nel orixe mesmu de los servicios web. Todo entamó nel Centru Européu d'Investigación Nuclear (CERN) en Suiza. El problema yera cómo organizar y acceder a la bayura d'información d'investigación que xeneraba'l CERN. Un científicu británicu, Tim Berners Lee, fixo en 1989 una propuesta pa meyorar esta situación (Berners-Lee, 1989). Na figura 1 podemos ver un estractu de la propuesta orixinal, col comentariu del supervisor de Berners-Lee, Mike Sendall no cimero del documentu.

Nesta propuesta Berners-Lee describe'l problema d'accesu a la información nel CERN y propón un sistema d'información enllazada y distribuida. Yera marzu de 1989, pero'l problema nun yera nuevu. Asina, Berners-Lee inspírase en dalgunos trabayos anteriores.

Vannevar Bush yá pensare dempués de la Segunda Guerra Mundial que'l gran retu de la humanidá yera xestionar el gran crecimientu de conocimientu xeneráu no qu'él llamaba rexistros o *records* (Bush, 1945). Dende'l puntu de vista del problema pa un investigador, Vannevar Bush diseñó'l sistema MEMEX, que permitía a una persona guardar, recuperar y siguir asociaciones ente pares de documentos. Inspiraos en Vannevar Bush, los trabayos d'otros investigadores como Douglas Engelbart o Ted Nelson tamién influyeron nel enfoque de Berners-Lee. Pela so parte, Douglas Engelbart, col so NLS, pensó en disponer d'un sistema asemeyáu a MEMEX, pero que dexare facer el trabayu collaborativu ente dellos usuarios (Engelbart y English, 1968). En paralelu, la visión de Ted Nelson yera la de tener una rede xigante de documentos, con daqué estructura, enllazaos ente sí en sirvidores desplegaos a escala mundial, nel proyeutu que más tarde diba dar el nome de *Xanadú* (Nelson, 1967). Foi Nelson el qu'inventó'l términu "hipertestu".

Otros trabayos interesantes qu'influyeron nel desendolcu de Berners-Lee foron los de Goldfarb, Mosher y Lorie. Inventores de los llinguaxes d'etiquetes, diseñaron el primeru d'estos llinguaxes nel añu 1969, GML. Más tarde, Goldfarb diseña'l llinguaxe SGML que ye lo que podemos considerar precursor direutu d'HTML. Na figura 3 podemos ver un exemplu de códigu SGML.

En definitiva, Berners-Lee, tomando referencies en trabayos anteriores, fai la so propuesta'l

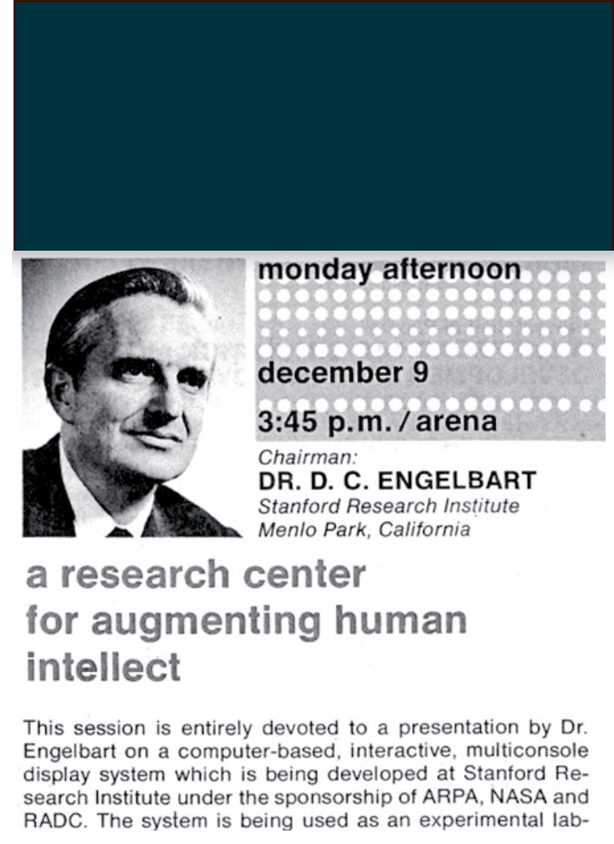


DERECHA

Figura 1. Propuesta de Tim Berners-Lee al so supervisor Mike Sendall (Berners-Lee, 1989). (CERN - Conseil Européen pour la Recherche Nucléaire).

12 de marzu de 1989. Esta apruébase y Berners-Lee ponse a trabayar nel desendolcu del so proyeutu. Cronolóxicamente, los primeros llogros conseñables son los que vienen darréu:

- El 20 d'avientu de 1990 arranca nel CERN el primer sirvidor web y espublízase la primer páxina web, tal y como s'amuesa na figura 4.
- El 10 de xineru de 1991, la web ábrese a la comunidá d'investigadores en física d'alta potencia vía la biblioteca del CERN.
- El 6 d'agostu de 1991, Berners-Lee escribe un resume del proyeutu en dellos grupos de noticies d'Internet: La web faise pública.
- El 12 d'avientu de 1991 despliégase'l el primer sirvidor fuera d'Europa, nel Stanford Linear Accelerator Center de California, EE.XX.
- El 30 d'abril de 1993, el CERN llibera la web al públicu ensin royalties, asegurando que permanecerá como un garrapiellu d'estándares abiertos. Esto tien un efeutu inmediatu y a lo

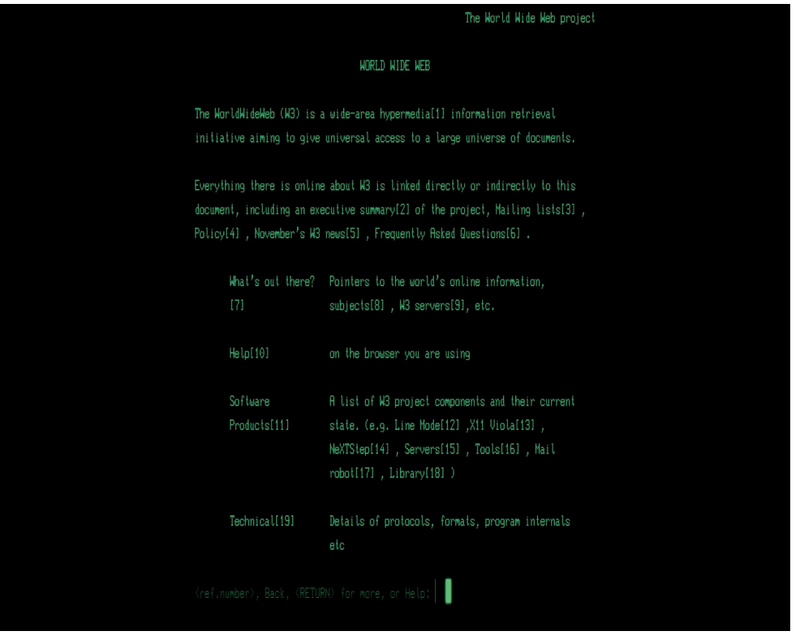


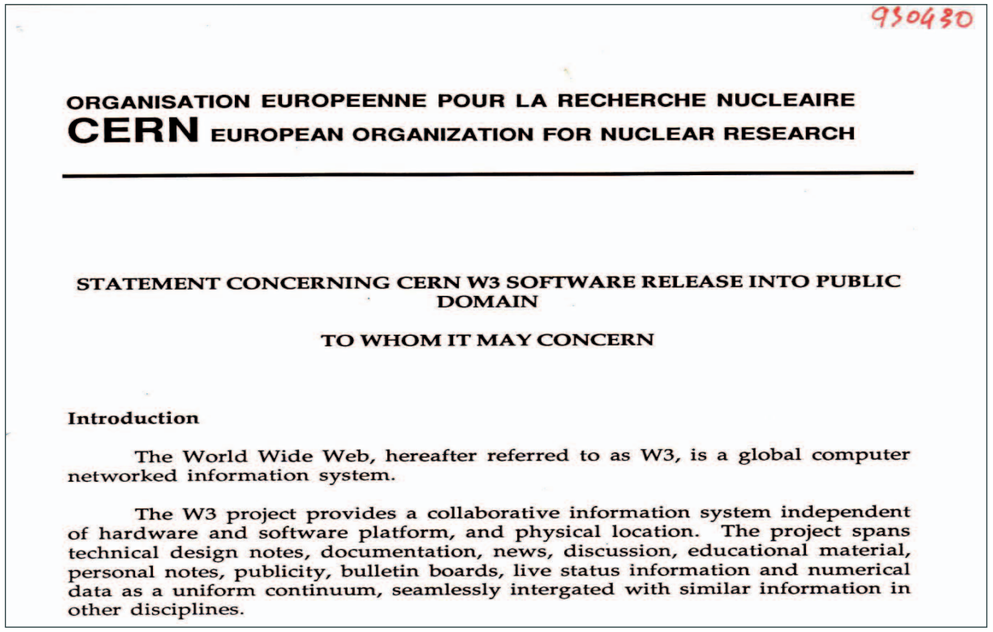
ARRIBA

Figura 2. Presentación d'Engelbart y English (1968). (Stanford Research Institute.

IZQUIERDA

Figura 3. Entrada del diccionariu Oxford d'inglés en SGML. (http://commons.wikimedia.org/wiki/File:OED-LEXX-Bungler.jpg)




930430

ORGANISATION EUROPEENNE POUR LA RECHERCHE NUCLEAIRE
CERN EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

STATEMENT CONCERNING CERN W3 SOFTWARE RELEASE INTO PUBLIC DOMAIN

TO WHOM IT MAY CONCERN

Introduction

The World Wide Web, hereafter referred to as W3, is a global computer networked information system.

The W3 project provides a collaborative information system independent of hardware and software platform, and physical location. The project spans technical design notes, documentation, news, discussion, educational material, personal notes, publicity, bulletin boards, live status information and numerical data as a uniform continuum, seamlessly intergated with similar information in other disciplines.

cabero de 1993 yá hai más de 500 sirvidores web. La declaración orixinal ye la que pue vese na figura 5.

- N'ochobre de 1994, col sofitu del CERN, el proyeutu DARPA y la Comisión Europea, Berners-Lee funda nel MIT el World Wide Web Consortium o W3C, que ye la organización que dende esti momentu pasa a encargase del desendolcu de la web.

Esto ye l'entamu d'un proyeutu ablucante que, güei, entá sigue n'evolución. Ye talu'l desendolcu teunolóxicu fechu nestos 25 años de vida del web, que nun sedríamos a reproducilo nun trabayu de les carauterístiques que se persiguen equí. Poro, vamos centranos nes seiciones siguientes pa comentar el desendolcu fechu nesti periodu de los protocolos de comunicaciones que faen qu'estos servicios puean esistir.

ARRIBA IZQUIERDA
Figura 4. Primer páxina web vista nun restolador de testu. (CERN - Conseil Européen pour la Recherche Nucléaire).

ARRIBA DERECHA
Figura 5. Documentu de lliberación de la web. (CERN - Conseil Européen pour la Recherche Nucléaire).

HTTP sufrió bien de cambeos nestos últimos años, pero entá caltién la so filosofía orixinal

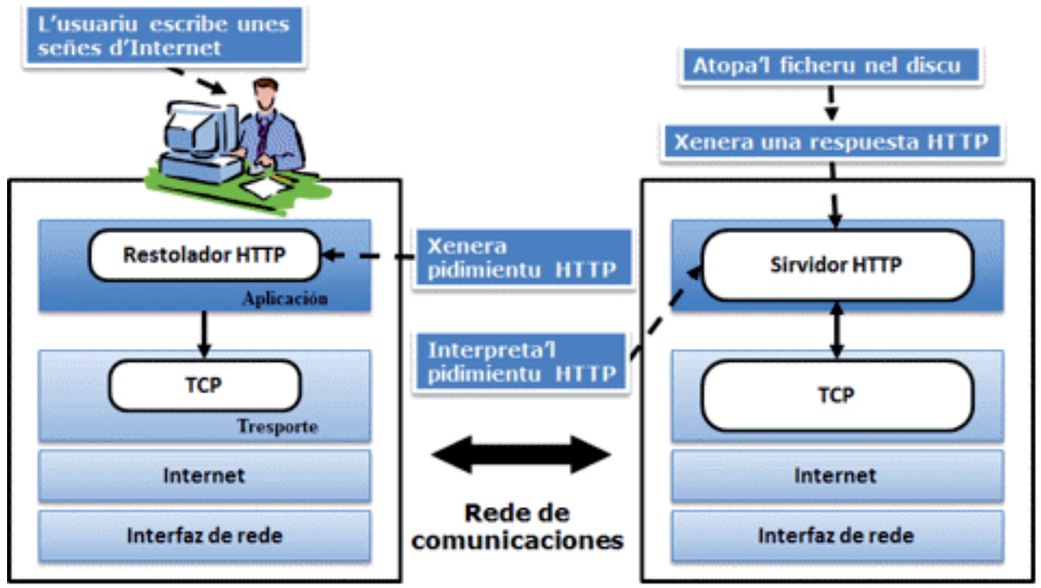
EVOLUCIÓN DE LOS PROTOCOLOS WEB

Magar que güei hai protocolos estremaos que s'usen nos servicios web, el so protocolu básicu ye HTTP o *Hypertext Transfer Protocol*.

Como s'amuesa na figura 6, la función d'esti protocolu ye la de dexar qu'un restolador puea pidir un ficheru a un sirvidor web. Cola direición que-y pasa l'usuariu, el restolador conéutase al sirvidor web y pásа-y un mensaxe nel que-y pide un ficheru. El sirvidor recibe'l mensaxe y respuende con otru mensaxe nel qu'inclúi'l ficheru solicitáu. En resume, HTTP define esti modelu d'interaición ente restolador y sirvidor, amás del formatu de los mensaxes de solicitú y respuesta.

Como vamos ver, esti protocolu sufrió bien de cambeos nestos últimos años. Sicasí, entá caltién la so filosofía orixinal: diseñóse pa ser cenciellu d'implementar. Ye un protocolu ensin estaos, poro, tolos pidimientos procésense de forma independiente. Cualesquier aplicación pue usalu con hipertestu o otros formatos de ficheru como puen ser ficheros d'audiu o videu.

ABAXO
Figura 6. Funcionamientu básicu de HTTP versión 0.9.



HTTP VERSIÓN 0.9

En 1991 dase a conocer la primer versión documentada d'esti protocolu: HTTP 0.9 (Berners-Lee, 1991). Esta versión yá afitaba les bases del modelu d'interaición que diba usase a lo llargo d'años y que s'amuesa na figura 7.

Lo que se plantea ye que cuando'l restolador necesita descargar un ficheru, abra una conexón de tresporte col sirvidor. Per esa conexón manda un mensaxe de solicitú y aguarda una respuesta. En mandando la respuesta, el sirvidor pieslla la conexón de tresporte.

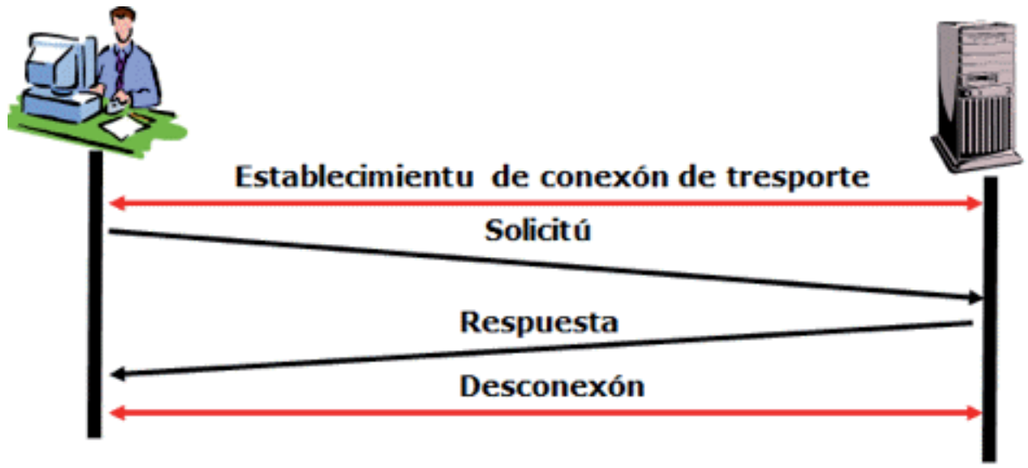
Na especificación de nivel de tresporte recomienden usar TCP, sicasí dexen la puerta abierta a otros protocolos orientaos a conexón como DECnet (Martin y Leben, 1992) o ISO TP4 (ISO/IEC 8073:1997). Tamién se fai una reserva del puertu TCP 80 pa les conexones col sirvidor.

Los documentos especifíquense con unes señes web llamaes UDI, que tienen el siguiente formatu:

http://hostname[:port]/path[?searchwords]

Onde *hostname* ye'l nome o la direición IP de la máquina na que ta agospiáu'l ficheru a descargar y *port* ye'l puertu a nivel de tresporte cuando nun s'usa'l 80. *path* depende del sirvidor, pero normalmente ye la ruta y el nome del ficheru. *searchwords* son pallabres clave que puen usase pa facer una busca.

Los mensaxes son cenciellos. Una solicitú ye la cadena *GET* siguida de la UDI del documentu que se va descargar y la combinación de símbolos *CR* –retornu de carru- y *LF* –saltu de llinia–. Una respuesta ye direutamente'l ficheru solicitáu al sirvidor. Orixinalmente'l protocolu taba pensáu namás pa sirvir ficheros HTML. Poro, si'l ficheru nun esiste o hai un error nel procesamientu de la solicitú, el sirvidor xenera un mensaxe HTML informando del error.



ARRIBA
***Figura 7.** Modelu d'interacción d'HTTP 0.9.*

HTTP VERSIÓN 1.0

Un añu dempués d'espublizar HTTP 0.9, sal la version 1.0 del protocolu, que tien que s'entender como la primer version completa del mesmu (Berners-Lee, 1992). Nesta version considérense dos formatos de pidimientu y rempuesta, amás de permitir dellos tipos de pidimientos estremaos. Los mensaxes incorporen información de control en forma de cabeceres y nes rempuestes pónense códigos de control pa saber lo que pasa coles pidimientos.

Arriendes d'ello, incorpórense dellos aspeutos del estándar MIME.

Los mensaxes siguen siendo cenciellos. Almítense mensaxes como los de la versión 0.9, no que se llama como pidimientos y respuestes cencielles. emás, permítense pidimientos y respuestes completes, qu'incorporen cabeceres de control ente otres coses. El formatu de los pidimientos y les respuestes completes ye'l que sigue:

Pidimientu-Completu = Métodu URI Versión-Protocolu CrLf
[*<Cabeceres- Pidimientu >]
[CrLf <Datos>]

Respuesta-Completa = Versión-Protocolu Códigu-Estáu Descripción-Estáu CrLf
[* Cabeceres-Entidá]
[CrLf Datos]

Como podemos ver, sigue usándose la combinación de símbolos *CR* y *LF* pa estremar les llinies del mensaxe. Nos pidimientos que xeneren los clientes, incorpórase un métodu qu'indica'l tipu de mensaxe. Nes respuestes que xenera'l sirvidor incorpórase un códigu y una descripción de lo que pasó col mensaxe al que se respuende. Los pidimientos y les respuestes puen llevar datos axuntos. El recursu que se solicita o referencia, indícase no que güei se llama URI, términu que vieno a sustituir al orixinal URI, pero que caltién un formatu similar.

Diséñense dellos tipos de pidimientos o *métodos*:

- *GET*: Pídense los datos indentificaos pola direición o URI.
- *HEAD*: Igual que *GET*, pero la rempuesta nun inclúi datos axuntos, namás les cabeceres.
- *CHECKOUT*: Igual que *GET*, pero pon un bloquéu al recursu solicitáu pa que nun lu puea modificar otru usuariu.
- *PUT*: Modifica'l recursu colos datos del pidimientu.
- *DELETE*: Borra'l recursu identificáu pola URI.
- *POST*: Constrúi un nuevu recursu colos datos del pidimientu.
- *CHECKIN*: Igual que *PUT*, pero quita'l bloquéu fechu anteriormente con un *CHECKOUT*.
- Otros: SHOWMETHOD, LINK, UNLINK, TEXTSEARCH, SPACEJUMP, SEARCH.

Les cabeceres estrémense ente cabeceres de pidimientu, xeneraes pol cliente nos sos pidimientos, y cabeceres d'entidá, xeneraes pol sirvidor nes respuestes. Les cabeceres de pidimientu incorporen información importante pa que'l sirvidor puea procesar les solicitúes como datos qu'identifiquen *(From)* o autentiquen (Authorization) al usuariu, información de formatos aceutaos pol cliente *(Accept, Accept-Encoding o Accept-Language),* información que fai una pidimientu condicional *(If-Modified-Since),* datos del software del cliente *(User-Agent)* o detalles pal pagamientu de los costos d'usar un serviciu *(Charge-To).* Les cabeceres d'entidá proporcionen información que tien que se conocer pa poder procesar la respuesta como detalles sobre los datos axuntos *(Content-Length, Content-Type, Content-Transfer-Encoding, Content-Encoding, Content-Language, Date, Expires o Title),* de les sos distintes versiones *(Last-Modified, Version o Derived-From),* del costeu d'acceder a esi recursu *(Cost)* o información sobre los métodos

Dende la so espublización en 1996, HTTP 1.0 tuvo qu'adautase a los tiempos amestando dellos anovamientos y quitando aspeutos del estándar que nun teníen interés pa la comunidá

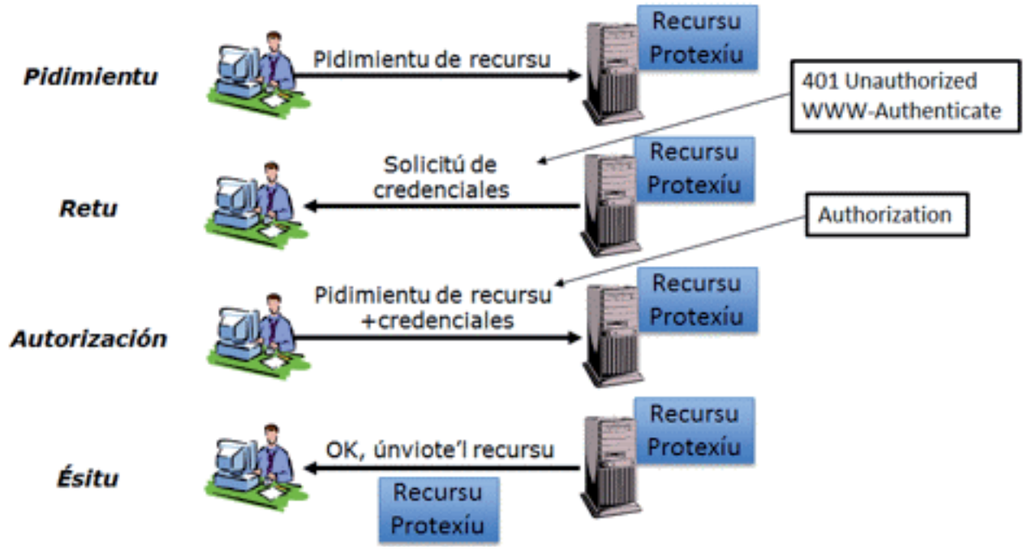
permitíos pal recursu solicitáu *(Allowed* o *Public)*. Incorpórense códigos y descripciones d'estáu nes respuestes que xenera'l sirvidor. Esta información permite a los restoladores saber qué pasó colos sos pidimientos, si hebo un error o si se procesaron con éxitu. Úsase una estructura de códigos d'error clasificaos en centenares, del mesmu mou que n'otros protocolos más vieyos como SMTP (Postel, 1981) Los códigos 200 son d'éxitu, los 300 pa facer redireiciones, los 400 son pa indicar un error del restolador (por exemplu nel formatu del so pidimientu y los 500 pa un error del sirvidor.

ACTUALIZACIÓN DE HTTP VERSIÓN 1.0

En 1996, cuatro años dempués de la espublización d'HTTP 1.0, asoléyase l'actualización cabera de la mesma versión 1.0 (Berners-Lee *et al.*, 1996). Dende'l desendolcu de la versión 1.0 orixinal, abondos desarrolladores foron amestando anovamientos nos sos productos que s'incorporen al estándar nesta versión. Del mesmu mou, viose nestos últimos años que dellos aspeutos del estándar nun tienen interés pa la comunidá. Poro, quítense del protocolu camudando en parte la so filosofía orixinal. Amás, dellos aspeutos poco claros na versión anterior especifíquense con más detalle güei.

Na actualización cabera de la versión 1.0 de HTTP almítese que l'arquitectura de serviciu basada nun restolador y un sirvidor nun ye válida en tolos casos. Ye posible que los pidimientos pasen per dellos programes intermedios. Bien por aspeutos de seguranza, funcionales o, cenciellamente, aforru de recursos, hai dalgunos programes que tamién puen tar presentes nuna arquitectura d'un serviciu web. Asina, un restolador conéutase a ún d'estos programes en cuenta coneutase direutamente al sirvidor. Dempués esti programa encárgase de procesar el pidimientu y de facelu llegar al sirvidor. Nel estándar inclúinse los casos d'un proxy, pasera y túnel.

- Un proxy (güei conocíu como *forward proxy)* vien a ser un programa que despliega'l responsable d'una rede na que los usuarios acceden a Internet, pa facer dalgún tipu de control sobre l'actividá d'esos usuarios. Esto pue ser esixir a los usuarios unes credenciales pa coneutase a Internet, restrinxir l'accesu a dellos sitios, etc. Tamién ye un programa que pue usase pa tresformar los pidimientos y respuestes cuando los restoladores y los sirvidores usen versiones estremaes del protocolu.
- Una pasera (güei conocida como *reverse proxy)* vien a ser un programa que despliega'l responsable d'una rede na que hai dellos sirvidores desplegaos. La idea ye la d'esponer esti programa como si fore un sirvidor de verdá. Los usuarios conéutense a él pensando que tán coneutándose al sirvidor real, pero en realidá fai de fachada o de *sirvidor bastión*. En recibiendo un pidimientu, la pasera pide los datos al sirvidor de verdá y dempués vuelve una respuesta al restolador. Asina, los sirvidores reales nun s'esponen a un ataque del esterior de la rede, namás s'espón la pasera. Tamién pue usase pa comunicar restoladores con sirvidores que nun usen el protocolu HTTP, faciendo una especie de traducción de mensaxes.
- Un túnel permite que dos programes qu'usen un protocolu estremáu a HTTP puean comunicase con esti protocolu. Podemos pensar en dos falantes d'asturianu que nun puen comunicase nesta llingua. Poro, usen d'intermediarios dos falantes d'asturianu ya inglés. El primer falante d'asturianu diz al intermediariu lo que quier unviar. L'intermediariu fala col otru intermediaru n'inglés y l'intermediariu caberu tresmite al segundu falante'l mensaxe. Esta utilidá ye interesante porque'l protocolu HTTP suel funcionar en toles redes, mientres qu'otros protocolos puen nun funcionar principalmente por polítiques de seguridá restrictives.



ARRIBA
Figura 8. Modelu d'autenticación na actualización cabera d'HTTP 1.0.

Nesta actualización tamién s'incorpora el conceutu de caché. Una caché permite responder a un pidimientu ensin pidir los datos al sirvidor. Por exemplu, cuando un proxy recibe un pidimientu que yá procesó nel pasáu, pue contestar con una respuesta asemeyada a l'anterior ensin pidir los datos al sirvidor. Poro, un proxy que tenga una caché dirá guardando información de tolos pidimientos que procese pa poder xenerar respuestes futures. Esta ye una midida d'aforru principalmente.

Otru aspeutu que se concreta nesta actualización ye'l mecanismu d'autenticación que nun foi a especificase en detalle na versión 1.0 orixinal. Esti mecanismu pue vese na figura 8. Si un restolador quier acceder a un recursu protexíu, el

sirvidor va responder con una respuesta d'error col códigu 401 y con detalles sobre'l formatu de les credenciales na cabecera *www-Authenticate*. En recibiendo esta respuesta, el restolador tien qu'unviar otru pidimientu incorporando les credenciales del usuariu na cabecera *Authorization*, nel formatu conseñáu pol sirvidor. Si les credenciales son correutes, el sirvidor respuende col recursu protexíu. Si les credenciales nun son correutes, el sirvidor xenera una reSpuesta como la primera qu'unviare. D'igual mou, l'estándar incorpora'l formatu d'unviu de credenciales *Basic*, que consiste n'aplicar una tresformación *Base64* a la cadena resultante de concatenar el nome d'usuariu con un ":" y la so contraseña.

Authorization: Basic+"“+Base64(usuariu+:+contraseña)

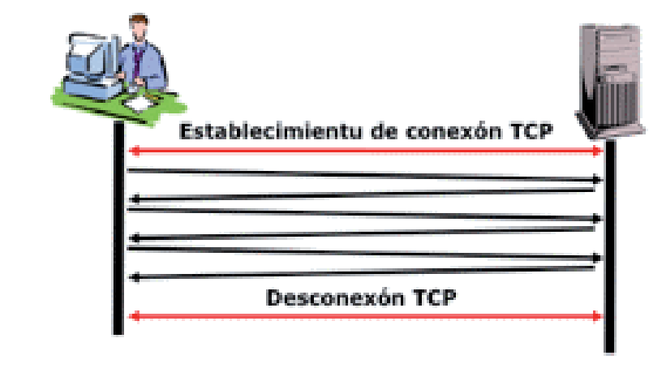
En xeneral, el formatu de los pidimientos y les respuestes caltiénse, pero fáense dellos cambeos, quedando, na so versión completa, tal y como vien darréu:

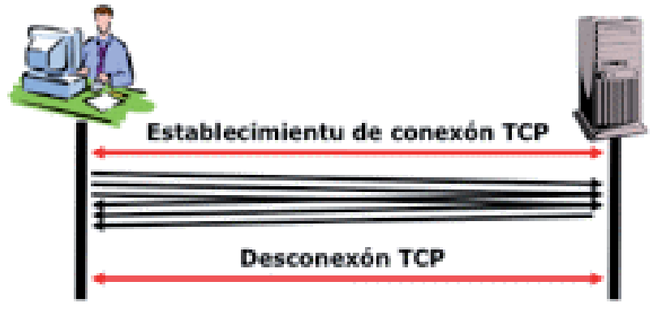
Pidimientu-Completu = Métodu URI Versión-Protocolu CrLf
*(Cabeceres-Xenerales
| Cabeceres-Petición
| Cabeceres-Entidá)
CrLf
[Cuerpu-Entidá]

Respuesta-Completa = Versión-Protocolu Códigu-Estáu Descripción-Estáu CrLf
*(Cabeceres-Xenerales
| Cabeceres-Respuesta
| Cabeceres-Entidá)
CrLf
[Cuerpu-Entidá]

Como podemos ver, respeuto a la versión orixinal los cambeos son principalmente semánticos no que cinca a la clasificación de les cabeceres.

Fáense cambeos nos métodos orixinales, escaeciendo dellos métodos, dando más importancia a un garrapiellu d'ellos y camudando la filosofía de dalgunos. Na parte principal del estándar caltiénense namás tres métodos: *GET* y *HEAD*, que caltienen la so función orixinal y *POST*, que s'usa na actualidá p'apurrir unos datos que tienen que s'aceutar como una entidá subordinada de la URI del pidimientu. Exemplos son apurrir datos d'un formulariu, amestar datos nuna base de datos o espublizar un mensaxe nun foru. Del restu de métodos namás se caltienen con dellos cambeos *DELETE*, *LINK*, *UNLINK* y *PUT*, pero separtaos a un segundu llugar al espublizalos nun anexu.





Tamién se reduz el númberu de cabeceres al descartar funcionalidaes planteaes na versión orixinal. Asina desapaecen delles cabeceres rellacionaes col control de versiones y el pagamientu de servicios. Per otru llau, delles cabeceres camuden la so función, como pue ser el casu de *Date* que güei ye pa indicar l'instante de xeneración del mensaxe. Arriendes d'ello, faise una meyor clasificación de les cabeceres en cabeceres xenerales, que puen usase nuna respuesta o nun pidimientu (*Date* o *Pragma*), cabeceres que namás se puen usar nun pidimientu (*Authorization*, *From*, *If-Modified-Since*, *Accept-Encoding*, *Accept-Language*, *User-Agent* o *Referer*) o nuna rempuesta (*Location*, *Server* o *www-Authenticate*) y cabeceres d'entidá que proporcionen detalles sobre los datos axuntos al mensaxe (*Allow*, *Content-Encoding*, *Content-Length*, *Content-Type*, *Expires* o *Last-Modified*)

Pa cabu, tamién se faen delles matizaciones na definición de dellos códigos d'estáu y surden códigos nuevos como puen ser los del grupu 100 que s'acuta pa usos futuros.

ARRIBA
Figura 8a. Conexones persistentes.

ABAXO
Figura 8b. Encadenáu de pidimentos.

HTTP VERSIÓN 1.1

HTTP 1.1 (Fielding et al., 1999) espublízase, na so revision definitiva, trés años dempués d'espublizar l'actualización cabera d'HTTP 1.0. en 1999. Esta revision dexa obsoleta una espublización preliminar d'HTTP 1.1 nel RFC 2068.

Les anovaciones más notables d'esta versión tán nel modelu d'interaición ente'l restolador y el sirvidor. Como comentábamos enantes, un restolador que se comunica con un sirvidor con HTTP, tien d'abrir una conexón TCP pela qu'unviar el pidimientu. Dempués, el sirvidor xenera una respuesta y piesslla la conexón TCP. Esti modelu pue vese na figura 7. El problema ye que, na mayoría de los casos, un restolador nun va pidir namás un ficheru al sirvidor, sinón un garrapiellu d'ellos. Poro, tien d'abrir y pesllar abondes conexones pa poder algamar tolos ficheros que necesita pa pintar la páxina web en pantalla. Esti modelu, qu'amás se procesa en secuencia, ye mui ineficiente. Pa meyoralu, surden dos cambeos na versión 1.1:

- Conexones persistentes. Cuando un restolador abre una conexón TCP, el sirvidor nun la piesslla namás devolver una respuesta. Asina, el restolador pue unviar más d'un pidimientu na mesma conexón TCP. Esti modelu, que pue vese na figura 8a, necesita que'l restolador indique al sirvidor cuándo pue pesllar la conexón. Incorpórase la cabecera *Connection* que col valor *keep-alive* quier dicir que'l sirvidor tien que caltener la conexón abierta y que col valor close indica que'l sirvidor pue pesllar la conexón. Los temporizadores internos de los sistemes operativos encargaríense de pesllar conexones de restoladores que nun lleguen a unviar un *Connection a close*.
- Encadenáu de pidimientos. Cuando s'usen conexones persistentes, el restolador tien

abierta una conexón col sirvidor pa poder unviar más d'un pidimientu. L'unviu de dellos pidimientos pue facese aguardando a recibir una respuesta enantes d'unviar el siguiente pidimientu, como s'amuesa na figura 8a. Pa meyorar entá más el rindimientu, na versión 1.1 de HTTP tamién se contempla'l casu de que, si'l restolador yá sabe qué ficheros tien que descargar del sirvidor, faiga pidimientos ensin esperar pola respuesta de caún d'ellos, como s'amuesa na figura 8b.

Tanto les conexones persistentes como l'encadenáu de pidimientos persiguen meyorar el rindimientu nel serviciu, aforrando'l pieslle y aniciu de conexones TCP innnecesaries y pudiendo unviar pidimientos ensin esperar poles respuestes correspondientes. Pero no que cinca al modelu d'interaición, na versión 1.1 del protocolu HTTP tamién s'incorpora una funcionalidá que vien a iguar l'unviu de ficheros de grandes dimensiones entre restolador y sirvidor: la codificación en cachos.

La codificación en cachos fai posible unviar un pidimientu o una respuesta en fragmentos. Ente otres ventayes d'esti modelu, ta la posibilidá d'unviar ficheros de grandes dimensiones. Si meditamos sobro'l mou de funcionamientu del serviciu, cuando'l sirvidor fai una respuesta tien que construyir un mensaxe de testu en memoria que dempués unvia pela conexón TCP entamada pol restolador. Si'l ficheru ye de grandes dimensiones, llega un momentu nel que tol ficheru, amás de les cabeceres, códigu d'estau y demás, cárgase na memoria de la máquina que correspuenda, provocando un colapsu de la mesma o una baxada notable nel so rindimientu (pola mor de la paxinación o *swapping* a memoria secundaria). La codificación en cachos funciona como se ve na figura 9. El restolador o'l sirvidor, cuando quieren usar codificación en cachos, incorporen una cabecera nomada *Transfer-Encoding* col valor *chunked*. Asina, l'otru cabu sabe que va recibir el mensaxe fragmentáu. Pela mesma conexón TCP, lo primero que manda esi restolador o sirvidor ye la primer parte del mensaxe, que suel incluir la primer llinia, les cabeceres y un fragmentu de los datos. Na parte cimera d'esi fragmentu ponse'l so tamañu en formatu hexadecimal. Siendo del mesmu mensaxe, l'unviu del siguiente fragmentu nun necesita incluyir la primer llinia de pidimientu/rempuesta ni les cabeceres. Poro, lo que s'unvia namás ye'l tamañu n'hexadecimal y el fragmentu de datos. N'unviando tolos fragmentos, mándase namás un 0 indicando que yá s'unviare tol ficheru.

Na versión de HTTP 1.1 tamién s'inclúin anovaciones relatives al usu de programes intermedios. Asina, incorpórense delles cabeceres qu'estienden el modelu d'autenticación de la figura 8 a un escenariu nel que l'autenticación tamién se pue facer nun proxy. Sobre la figura 8, la primer respuesta del proxy sedría una 407 Proxy Authentication Required con una cabecera *Proxy-Authenticate* pa indicar el formatu d'unviu de les credenciales del usuariu. Dempués el restolador unviaría les credenciales nel formatu indicáu na cabecera *Proxy-Authorization.* Per otru llau, na segunda metá de los años 90 estandarizóse'l crifráu de les comunicaciones a nivel de tresporte col protocolu SSL y la so evolución TLS. En redes nes que se desplieguen proxies, esti crifráu pue provocar problemes. Los proxies saben el sirvidor col que se tienen que coneutar garrando los datos que vienen nel pidimientu HTTP que-yos apurre'l restolador. Pero la negociación de los parámetros de crifráu, qu'implica un intercambiu de mensaxes entre'l restolador y el sirvidor, faise enantes d'unviar el pidimientu HTTP. Poro, el proxy nun ye a saber con qué sirvidor tien que se coneutar porque nun llega a ver los mensaxes HTTP.

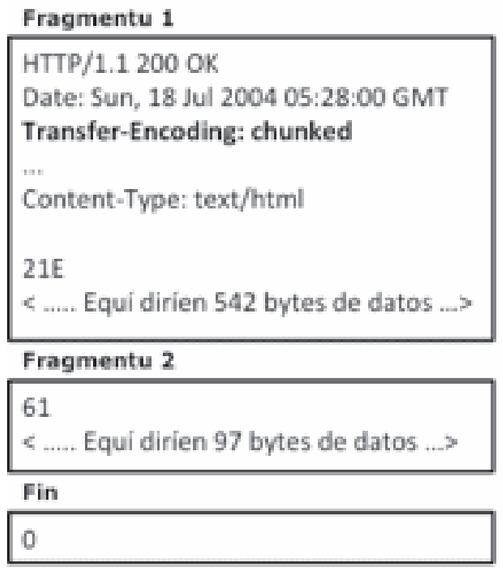
Pa iguar el problema col crifráu y los proxies, nesta versión d'HTTP incorpórase'l métodu CONNECT. Con esi métodu, el restolador informa al proxy d'a qué sirvidor tien que se coneutar. N'estableciendo la conexón col sirvidor, el proxy respuende confirmando al restolador esti fechu. Nel casu de SSL/TLS, dempués d'esta confirmación entamaría la negociación entre'l restolador y el sirvidor usando'l proxy como intermediariu. Agora yá funcionaría porque'l restolador ta coneutáu al proxy y el proxy ta coneutáu al sirvidor.

Otru preséu interesante qu'incorpora esta versión d'HTTP ye la posibilidá pal restolador de solicitar una parte namás de los datos totales qu'hai nel sirvidor (una parte d'un ficheru en cuenta del ficheru enteru). El funcionamientu ye meyor velu con un exemplu. Si un sirvidor mete nuna respuesta la cabecera *Accept-ranges* cola unidá de midida que correspuenda (por exemplu *bytes*), ta indicando al restolador que pue pidir partes d'esi recursu medibles nesa unidá. Por exemplu, si nel siguiente pidimientu'l restolador mete la cabecera *Range* col valor *bytes=0-499*, quier dicir que ta pidiendo los primeros 500 bytes del ficheru. El sirvidor entós unviaría namás esi fragmentu y na respuesta inxertaría la cabecera *Content-Range* col valor *bytes 0-499/1234*, indicando qu'unvia los primeros 500 bytes d'un total de 1234.

No que cinca al formatu de los pidimientos, caltiénse igual que na versión anterior a nivel sintáuticu, pero fáense cambeos nos métodos y nes cabeceres principalmente. De los métodos de la versión anterior, namás se caltienen GET,

Les anovaciones más notables de HTTP 1.1 tán nel modelu d'interaición ente'l restolador y el sirvidor

ABAXO
Figura 9. Codificación en cachos.



Lo que diba ser HTTP 1.2 queda namás nuna especificación de caráuter esperimental d'un mecanismu d'estensión d'HTTP

HEAD, POST, PUT y DELETE ensin camudar el so significáu. Incorpórase'l métodu CONNECT, comentáu enantes, amás *d'*OPTIONS y TRACE. OPTIONS ye pa saber los métodos que soporta un sirvidor o un recursu particular y TRACE ye una especie d'ecu pa facer depuraciones. Aumenta'l númberu de cabeceres pola incorporación de nueves funcionalidaes, delles comentaes enantes.

HTTP VERSIÓN ¿1.2?

Namás un añu dempués de la espublización de la última revisión d'HTTP 1.1, nel añu 2000, asoléyase lo qu'entama como la version 1.2 del mesmu. Y dicimos entama, porque pel camín pasa de ser HTTP 1.2 a An HTTP Extension Framework (Nielsen et al., 2000). Esto ye: un cenciellu mecanismu d'estensión d'HTTP que nun llega nin siquiera a ser estándar, sinón una especificación de caráuter esperimental.

Una vegada más, como asocediere nel pasáu, reconozse qu'hai tensiones ente'l mundu empresarial/iniciativa privada y los comités públicos d'estandarización. Les empreses, mui anovadores, proponen abondes ampliaciones al estándar. Pero estes enxamás lleguen a ponese nél o pónense mui tarde. Poro, lo que se fai con esta especificación ye diseñar un mecanismu xenéricu pa que los fabricantes puean desendolcar les anovaciones que-yos pete, ensin violar le especificaciones estándar. La filosofía ye percenciella:

- Daquién diseña una funcionalidá nueva o *estensión*, y asigna-y una URI única a mou d'identificador universal.
- Un restolador que quier usar esa estensión, pon una cabecera nel so pidimientu incluyendo esi identificador universal
- Si'l sirvidor implementa la estensión, fadrá lo que se-y pida. En cualesquier otru casu, contestará con una respuesta HTTP normal y corriente.

Los pegollos que caltienen esta filosofía son delles cabeceres y dalgunos cambeos menores que vamos comentar:

Les estensiones puen ser obligatories o optatives. La diferencia ye que si'l sirvidor nun implementa una estensión obligatoria tien que volver un error. Pa indicar la obligatoriedá o la opcionalidá d'una estensión, úsense les cabeceres *Man* o *Opt* respeutivamente. Nestes cabeceres ponse la URI de la estensión que se quier usar y el prefixu que va poner a otres cabeceres que son específiques d'esta estensión. Un sirvidor que soporta HTTP 1.2 ye a interpretar les cabeceres *Man* y *Opt* y saber lo que tien que facer, pero ¿qué pasa con un sirvidor que nun soporta HTTP 1.2? Nel casu d'un sirvidor non compatible, cuando les estensiones son *Opt* nun pasa nada porque respuende con una respuesta HTTP normal. Pero pal casu de les estensiones *Man*, lo que se fai ye poner un "M-" enantes del métodu, lo que fai que los sirvidores nun compatibles devuelvan un mensaxe d'error (por exemplu *M-GET* en vez de *GET*)

Les estensiones puen ser saltu-a-saltu o cabu-a-cabu. Les estensiones saltu-a-saltu aplíquense na más que na conexón actual (por exemplu entre'l restolador y un *proxy* o entre un *proxy* y el sirvidor) y les cabu-a-cabu aplíquense a tola cadena de conexones dende'l restolador al sirvidor (por exemplu, pasando por dellos *proxies*) Les cabeceres *Man* y *Opt* indiquen estensiones cabu-a-cabu, pero les *C-Man* y *C-Opt* indiquen estensiones obligatories y opcionales aplicaes namás saltu-a-saltu.

Nel siguiente exemplu (Nielsen *et al*., 2000) vemos un pidimientu PUT na que se pide una estensión obligatoria cabu a cabu (vemos la cabecera *Man* y el prefixu *M-* nel métodu). L'identificador de la estensión ye *http://www.copyright.org/rights-mngement* y el prefixu pa les cabeceres específiques ye *16*. Les cabeceres específiques son *16-copyright* y *16-contributions*. Un sirvidor non compatible con HTTP 1.2 respuende con un error porque nun pescancia'l métodu *M-PUT*. Un sirvidor compatible, vuelve un error si nun soporta la estensión *http://www.copyright.org/rights-mngement* porque ye obligatoria (*Man*). Un sirvidor compatible que soporta esa estensión, usará la información de les cabeceres *16-copyright* y *16-contributions* pa saber lo que tien que facer.

```
M-PUT /a-resource HTTP/1.1
    Man: "http://www.copyright.org/rights-mngement"; ns=16
    16-copyright: http://www.copyright.org/COPYRIGHT.html
    16-contributions: http://www.copyright.org/PATCH.html
    Host: www.w3.org
    Content-Length: 1203
    Content-Type: text/html
    <!doctype html ...
```

EL PROTOCOLU WEBSOCKET

Magar que la version 1.2 d'HTTP (o mecanismu d'estensión) nun llegó a usase masivamente, la version 1.1 del protocolu entá ye la version de referencia d'esti estándar. Pero, lóxicamente, dende'l so desendolcu pasaron bien d'años nos que los desarrolladores de software siguieron faciendo abondes anovaciones. Delles d'estes anovaciones puen ser la distribución d'información

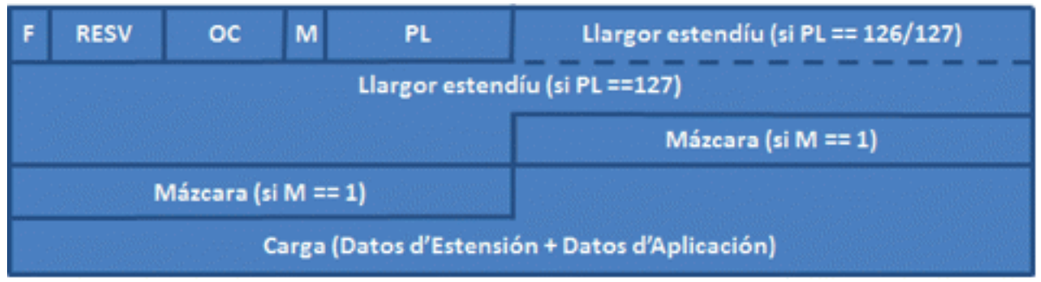
en vivo, los xuegos interautivos, los servicios de mensaxería instantánea (o comunicación interautiva) integraos en páxines web. Estes anovaciones puen implementase con HTTP, pero de forma mui poco eficiente. Por exemplu, si queremos facer un serviciu de mensaxería instántanea con HTTP y software convencional, como vemos na figura 10, el restolador tien que facer un sondéu periódicu del sirvidor cada dellos segundos pa saber si hai mensaxes nuevos pal usuariu. Si'l sirvidor tien mensaxes nuevos pa unvia-y al usuariu, tien qu'esperar a que 'l restolador d'esi usuariu entrugue por mensaxes nuevos.

L'exemplu del chat tien dellos inconvenientes. En primer llugar, el restolador de los usuarios ta faciendo pidimientos cada pocos segundos consumiendo de forma, quiciabes, innecesaria. El restolador d'un usuariu que nun recibe mensaxes ta xenerando pidimientos igualmente, pero estos son innecesarios. Per otru llau, si mengua la frecuencia del sondéu podemos reducir el númberu de recursos consumíos, pero'l serviciu pierde interactividá porque los mensaxes llegaríen tarde al usuariu. Agora bien ¿cómo meyoramos esta situación? Lo meyor ye que'l sirvidor, de la que recibe un mensaxe pa un usuariu, unvie al so restolador el mensaxe nesi momentu, pero esto con HTTP nun se pue facer porque'l sirvidor tien un rol pasivu siempre. Necesítase otru protocolu como pue ser WebSocket (Fette y Melnikov, 2011).

WebSocket ye un protocolu diseñáu pa ufrir un serviciu de comunicación simétricu y full-duplex ente un cliente y un sirvidor

IZQUIERDA
Figura 10. Desendolcu de chat con HTTP.

ABAXO
Figura 11. Formatu de mensaxe Websocket.



El protocolu WebSocket ye un protocolu de nivel d'aplicación diseñáu pa ufrir un serviciu de comunicación simétricu y *full-duplex* (bidireicional y simultaneu) ente un cliente y un sirvidor. El protocolu pue usase dende los programes JavaScript inxertaos nes páxines web. Tanto un restolador que lu use, como un sirvidor, pue unviar al otru cabu un mensaxe cuando-y pete. La conexón ente'l cliente y el sirvidor entama con un pidimientu HTTP normal. Esti pidimientu, normalmente de tipu GET, lleva una cabecera *Connection* col valor *Upgrade*, indicando que se quier camudar el protocolu HTTP por otru. Esti nuevu protocolu indícase na cabecera *Upgrade* que, nel nuesu casu, tendrá'l valor *websocket*. Asina, el sirvidor sabe que'l restolador quier usar WebSocket. Esti pidimientu tamién lleva otres cabeceres p'acutar delles opciones y pa facer comprobaciones de seguranza como *Sec-WebSocket-Key*, *Sec-WebSocket-Protocol*, o *Sec-WebSocket-Version* o *Sec-WebSocket-Extensions*. Si'l sirvidor contesta con una respuesta *101 Switching Protocols* y se cumplen les comprobaciones de seguranza, entama l'intercambiu de mensaxes WebSocket.

Los mensaxes qu'unvia'l restolador y el sirvidor tienen el formatu binariu que se pue ver na figura 11.

Como se ve na figura 11, los mensaxes tienen una estructura binaria na qu'hai dellos campos de control con estremaes funciones. Quiciabes el campu más importante ye OC o códigu d'operación que ye un campu de 4 bits qu'almite los valores que vienen darréu (preséntense en formatu hexadecimal):

- 0: Esti ye un mensaxe que lleva un fragmentu d'una secuencia de mensaxes. El campu *F* de la cabecera indica qu'esti ye'l fragmentu final.
- 1: Esti ye un mensaxe que tresporta datos en formatu testu. Nuna secuencia de fragmentos, esti valor marcaría l'entamu de la secuencia darréu que'l restu de fragmentos llevaría'l valor 0 *n'* OC.
- 2: Esti ye un mensaxe que tresporta datos en formatu binariu. Nuna secuencia de fragmentos, esti valor tamién marcaría l'entamu de la secuencia ya que'l restu de fragmentos llevaría'l valor 0 *n'*OC.
- 3-7: Son códigos reservaos.

- 8: Esti mensaxe úsase pa indicar que se quier pesllar la conexón.
- 9: L'emisor quier saber l'estáu del otru cabu (*Ping*).
- A: L'emisor contesta a un mensaxe anterior con OC igual a 9 (*Pong*).
- B-F: Son códigos reservaos.

Los valores del campu OC condensen el funcionamientu del protocolu. Si por exemplu un restolador quier unviar al sirvidor un mensaxe de testu grande y pesllar la conexón, xeneraría la siguiente secuencia: El primer mensaxe llevaría nel campu OC el valor 1, los siguientes mensaxes con datos de testu llevaríen *n'OC* el valor 0. El mensaxe caberu que tresporta datos de testu llevaría'l campu *F* a 1. Dempués d'unviar tolos mensaxes de testu mandaría otru mensaxe col campu OC a 8.

Nel exemplu del chat, n'estableciendo la conexón los restoladores y'l sirvidor tendíen una conexón abierta un tiempu indefiníu. Pa que nun se peslle sola, dacuandu intercambiaríen mensaxes con OC a 9 y a A (*ping-pong*). Si'l sirvidor tien un mensaxe pa un usuariu, mandaría-y una secuencia de mensaxes de testu. Si l'usuariu quier retrucar, el so restolador unviaría una nueva secuencia de mensaxes de testu. Si l'usuariu quier desconeutase, el restolador unviaría un mensaxe con OC a 8. Si lo que fai l'usuariu ye cenciellamente pesllar el restolador, l'ausencia d'actividá na conexón fadría que'l sirvidor la pesllare automáticamente.

ESTÁU DE LOS PROTOCOLOS ANGUAÑO (AÑU 2014)

Websocket ye un protocolu que paez que ruempe un poco la evolución tradicional d'HTTP en versiones que se van remocicando y evolucionando. Y nun ye qu'HTTP tea muertu, sinón que cenciellamente lleven munchos años ensin espublizar una nueva versión. Nel añu 2014 hai dos víes abiertes de trabayu respeuto a HTTP: una revision d'HTTP 1.1 y la espublización de la version 2 d'HTTP.

NUEVA REVISIÓN DE HTTP 1.1

Nuevamente, 15 años dempués de la última revisión estandarizada d'HTTP 1.1, faise una revisión d'esta versión del protocolu. La revisión última d'esta versión ye de xunu de 2014 y yá ta camín de convertise n'estándar. Esta revisión condensa lo que ye la base del protocolu más una refilera de funcionalidaes, pero con un nivel d'especificación más detalláu. Pol so llargor, decídese espublizalu en 6 documentos:

- Sintaxis de mensaxes y empobinamientu (Fielding y Reschke, 2014a)
- Semántica y conteníu (Fielding y Reschke, 2014b)
- Pidimientos condicionales (Fielding y Reschke, 2014c)
- Pidimientos de fragmentos (Fielding *et al*., 2014a)
- *Cachés* HTTP (Fielding *et al*., 2014b)
- Autenticación (Fielding y Reschke, 2014d).

Como podemos ver, la parte básica del protocolu (sintaxis y semántica) condénsase en dos documentos. Los otros cuatro especifiquen delles funcionalidaes yá comentaes enantes, colos cambeos propios d'esta versión.

No que cinca a la sintaxis y empobinamientu (Fielding y Reschke, 2014a), confírmase'l modelu d'interacción básicu colos cambeos inxertaos na revisión anterior d'HTTP 1.1 (Fielding *et al*., 1999), la esistencia de *proxies*, paseres y túneles, el funcionamientu de delles cabeceres y la estructura orixinal de mensaxes. Per otru llau, el nivel de detalle na especificación ye muncho mayor, aclariando delles duldes que podíen surdir nel pasáu, escaezse'l formatu cenciellu de empobinamientu 0.9 (Berners-Lee, 1991), inclúinse na especificación les URI *https* que correspuenden a mensaxes empobinamientu unviaos sobre SSL/TLS y matízase'l funcionamientu del métodu *CONNECT*.

En cuantes a la semántica y conteníu (Fielding y Reschke, 2014b), lo que se fai principalmente ye desaniciar ambigüedaes con un nivel d'especificación más detalláu. Confírmense les cabeceres yá definíes nel pasáu, faciendo delles aclaraciones sobro'l formatu de los sos valores y la interpretación de los mesmos. Tamién se caltienen los mesmos métodos que na versión anterior y los mesmos códigos d'estáu, incrementando'l nivel de detalle na especificación. Surde un códigu nuevu pa que los sirvidores puean suxerir camudar de protocolu proactivamente (*426 Upgrade Required*). Per otru llau, establezse un métodu pa estender el protocolu amestándo-y nuevos métodos, cabeceres y códigos d'estáu.

Los pidimientos condicionales son daqué que lleva nel estándar dende la versión 1.0 orixinal pola mor de la cabecera *If-Modified-Since* (Berners-Lee, 1992). Dende esa versión, foron amestándose nueves cabeceres indicando precondiciones pa poder aplica-y un métodu a un recursu determináu. Güei, estes cabeceres y la lóxica que-yos correspuende, recuéyense nun documentu específicu (Fielding y Reschke, 2014c). Nesti documentu, el nivel d'especificación ye mui superior a lo recoyío en versiones anteriores del protocolu. Por exemplu, aclariase lo qu'hai que facer en casu d'ambigüedá cuando hai delles cabeceres condicionales nel mesmu pidimientu.

Los pidimientos de fragmentos d'información yá yera daqué recoyío na versión 1.1 d'HTTP (Fielding *et al*., 1999). Anguaño, esta funcionalidá tamién se recueye nel so documentu específicu (Fielding *et al*., 2014a). Esti documentu concentra tola funcionalidá rellacionada con esti mecanismu, apurriendo más exemplos y con un nivel de detalle mayor, na llinia del restu de documentos d'esta especificación.

Les *cachés* tamién son un mecanismu compatible con HTTP dende l'actualización cabera d'HTTP 1.0 (Berners-Lee *et al*., 1996). Na revisión que se fai agora d'HTTP 1.1, dedícase un documentu enteru a esti mecanismu (Fielding *et al*., 2014b). El nivel de detalle del mesmu nun ye comparable al de les versiones anteriores del estándar no que cinca al tratamientu de les *cachés*, faciendo una bayurosa descripción de cómo se calcula'l frescor d'un recursu.

El documentu caberu d'esta nueva especificación d'HTTP 1.1 céntrase n'autenticación d'usuarios (Fielding y Reschke, 2014d). Nun se faen aportaciones considerables, como nes otres partes d'esta revisión del protocolu, puesto que

namás se condensa tolo escrito nes versiones anteriores nun documentu específicu: el modelu xeneral d'autenticación, códigos d'estáu y cabeceres tanto pa l'autenticación en sirvidores como en *proxies*, orixinalmente en Berners-Lee et al. (1996) y (Fielding *et al*. (1999).

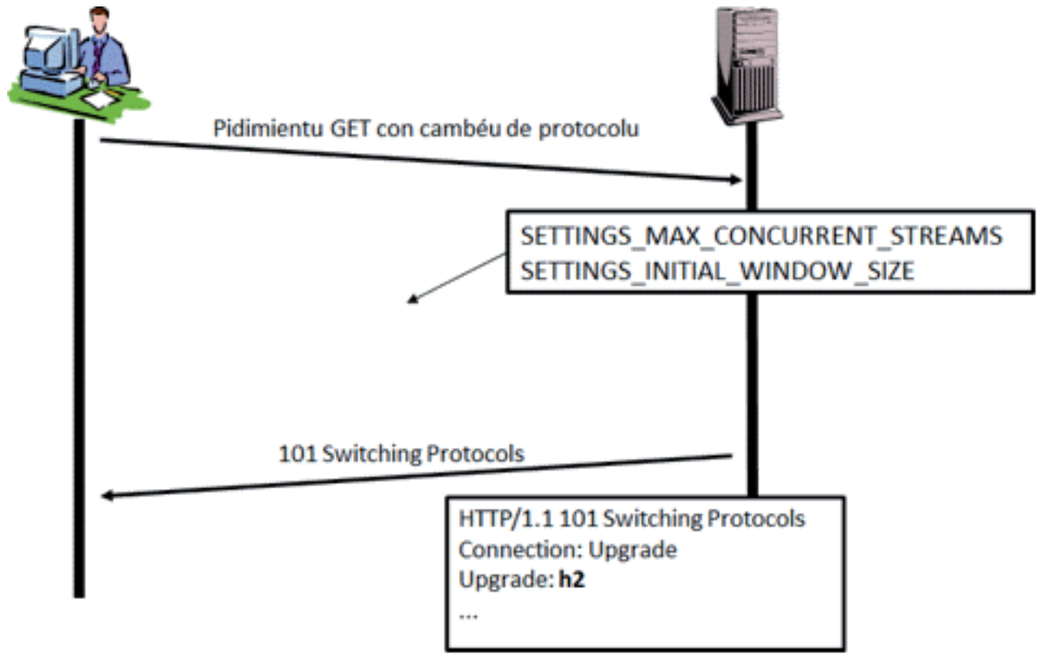
HTTP VERSIÓN 2

Entá n'estau borrador, al empar que se trabaya na nueva revisión de la versión 1.1 d'HTTP, yá ta trabayándose no que se quier llamar version 2 d'esti protocolu (Belshe et al., 2014). Como vamos ver, al contrario de lo que paez, esta versión nun compite cola anterior.

L'enfotu de la versión 2 ye facer un consumu más eficiente de los recursos de rede. Como diz la propia especificación d'esta versión del protocolu, *el formatu de los mensaxes d'HTTP 1.1 diseñóse pa que se desendolque coles ferramientes afayadizes na década de los 90 del sieglu pasáu* (Belshe *et al*., 2014). Asina, tien delles carauterístiques que, güei, puen ser ameyoraes no que cinca al rendimientu d'un serviciu web. La versión 1.1 yá amestaba al estándar delles optimizaciones coles conexones persistentes y l'encadenáu de pidimientos (Fielding *et al*., 1999). Pero, asina y too, entá con estes optimizaciones el procesamientu de los pidimientos tien que ser secuencial. La secuencialidá pue provocar bloqueos *head-of-line*, nos qu'un problema cola respuesta d'un pidimientu fai que se retrasen les respuestes posteriores. Además, les cabeceres de los mensaxes HTTP son bultables y faen que'l tamañu de los mensaxes medre enforma. Dellos estudios asitien el tamañu de les cabeceres ente 200 bytes y más de 2 kilobytes (SPDY, 2014). Poro, HTTP 2 naz col enfotu d'optimizar los servicios web dende'l puntu de vista de les comunicaciones.

Como se ve na figura 12, una conexón HTTP 2 entama col mecanismu yá utilizáu por WebSockets. El restolador entama les comunicaciones con un pidimientu HTTP normal, pero nél pídese camudar de protocolu *(Connection a Upgrade)* pa pasar de HTTP 1.1 a HTTP 2 (Upgrade a h2[1]). Si'l sirvidor contesta con una respuesta con códigu 101 Switching protocols y confirma que se fai'l cambéu a HTTP 2 *(Connection a Upgrade* y *Upgrade a h2),* entamen les comunicaciones nesta versión del protocolu.

Como se ve na figura 12, nel pidimientu del restolador métese una cabecera HTTP -Settings con dellos parámetros que van usase pa negociar les comunicaciones ente'l restolador y el sirvidor. Polo menos, esta cabecera tien que tener valores pa los parámetros SETTINGS _ MAX _ CONCURRENT _ STREAMS Y SETTINGS _ INITIAL _ WINDOW _ SIZE. Con estos parámetros, el restolador indica'l númberu de transacciones HTTP 2 concurrentes qu'almitiría y el númberu máximu de bytes que pue recibir ensin unviar una confirmación de vuelta al sirvidor.



ARRIBA
Figura 12. Entamu de comunicación HTTP 2.

Al empar que se trabaya na nueva revisión de HTTP 1.1, yá ta desendolcándose la version 2 d'esti protocolu, pero, al contrario de lo que paez, esta versión nun compite cola anterior

Esa primera transaición (pidimientu + respuesta) ye usando HTTP 1.1, pero la negociación acaba usando dellos mensaxes nel formatu propiu d'HTTP 2 unviaos pela conexón TCP yá establecida. Si'l sirvidor confirmó'l pidimientu del restolador, lo primero que fai ye unviar un mensaxe de tipu *SETTINGS* col númberu de transacciones qu'almitiría y el so "tamañu de ventana". Al mesmu tiempu, el cliente tien qu'unviar una confirmación final col mensaxe hexadecimal 505249202a20485454502f322e300d0a0d0a534d0d0a0d0a[2] y un mensaxe SETTINGS con dalgunos parámetros finales.

N'acabando esta primer fase de negociación, el restolador y el sirvidor yá tán d'alcuerdu pa usar HTTP 2 pa comunicase. Cada vegada qu'el restolador quiera unviar un mensaxe al sirvidor, usando la conexón TCP establecida (con TLS o ensin él), entama una transaición nueva y con ella unvia los datos que-y pete. Lo mesmo'l sirvidor, que pue tamién entamar transaiciones hacia'l restolador. Una transaición ye equiparable a un intercambiu de mensaxes HTTP ordinariu (pidimientu+respuesta). Na semántica del protocolu, estes transaiciones llámense fluxos

Puen abrise tantos fluxos concurrentes como se negociare orixinalmente colos *SETTINGS _ MAX _ CONCURRENT _ STREAMS*. Lo que se fai ye un tratamientu asemeyáu al encadenáu de pidimientos d'HTTP 1.1, pero rompiendo l'orde secuencial de les mesmes.

1. El valor h2 correspuende a una conexón http 2 sobro tls, mientres qu'h2c ye pa entamar una conexón HTTP 2 sobro tco direutamente (ensin cifráu).

2. pri * http/2.0 CrLf CrLf SM CrLf CrLf

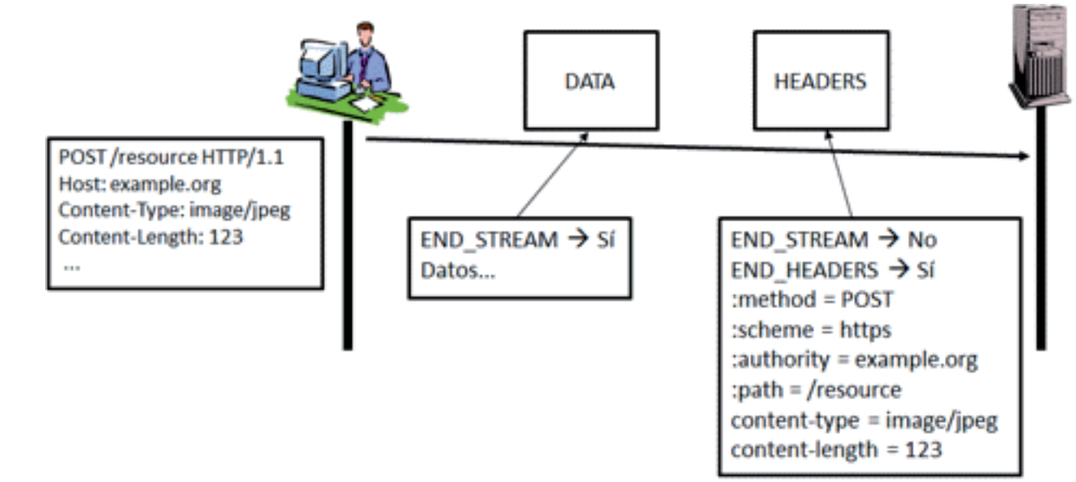
ARRIBA
Figura 13. Formatu de mensaxe HTTP 2.

Por exemplu, si un restolador necesita pidir dos ficheros, xenera dos fluxos y pide los ficheros colos mensaxes HTTP 2 que correspuenda. Estos mensaxes van pela mesma conexón TCP, pero lleven un identificador de fluxu. La diferencia ye qu'anguaño'l sirvidor pue responder col segundu ficheru enantes que col primeru: nel orde que-y apeteza. Si hai un problema con ún de los pidimientos, esti nun retrasa a los otros. Esto ye pa evitar el problema de bloqueos *head-of-line* comentáu enantes. Ye más, HTTP 2 incorpora un mecanismu de priorización que-y sirve a un cabu pa indicar al otru cómo quier que se procesen los fluxos. Poro, llevándolo a la semántica clásica d'HTTP, el procesamientu de los pidimientos yá nun ye nel orde de llegada.

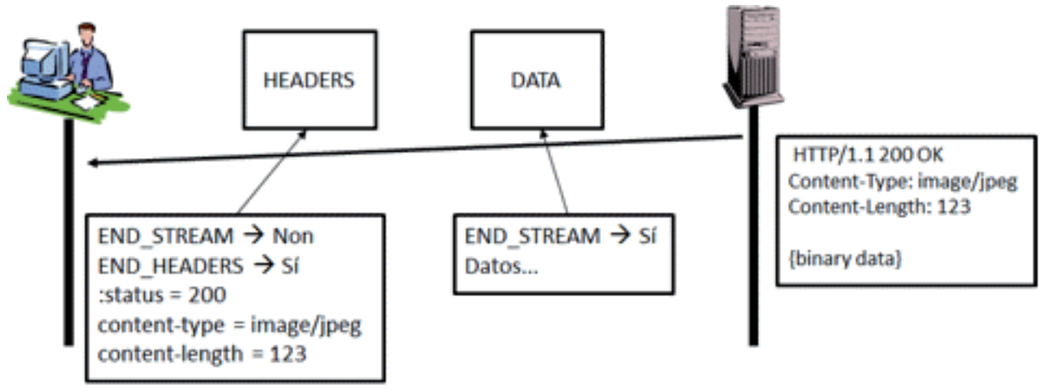
Los mensaxes HTTP 2 tienen el formatu binariu que vemos na figura 13.

Como vemos na figura 13, los mensaxes tienen una estructura na qu'hai dalgunos campos de control con estremaes funciones. El campu más interesante ye *Tipu* que ye un campu de 8 bits pa indicar el tipu de mensaxe de que se trata. Ensin entrar en detalle nel protocolu, quiciabes los más interesantes son *DATA* (0x0), que ye pa unviar datos (por exemplu una páxina o los datos d'un formulariu) y *HEADERS* (0x1), que ye pa unviar información de control (por exemplu les cabeceres clásiques de HTTP)

Pa entender el funcionamientu no que ye un contestu de navegación, vamos ver un exemplu.



DERECHA SUPERIOR
Figura 14. Pidimientu d'una páxina web con HTTP 2.



DERECHA INFERIOR
Figura 14. Unviu d'una páxina web con HTTP 2.

Na figura 14 amuésase cómo sería un pidimientu d'una páxina web.

El pidimientu HTTP 1.1 que vemos a la izquierda de la figura 14, estrémase en dos partes. La primera tien les cabeceres, que se meten nun mensaxe HEADERS na forma que se ve na figura, entamando un fluxu nuevu. Nel mensaxe HEADERS indícase qu'esti mensaxe acaba les cabeceres (END _ HEADERS→Sí), yá qu'estes podríen unviase en dellos mensaxes HEADER si fai falta. Amás, indícase que nun acaba'l fluxu (END _ STREAM→No), porque entá tien qu'unviar el mensaxe DATA. La segunda parte del pidimientu, que sedría lo que va nel cuerpu d'entidá (por exemplu los datos d'un formulariu web), mándase nun mensaxe DATA nel que s'indica qu'esta parte acabó d'unviar mensaxes pa esti fluxu (END _ STREAM→Sí).

El sirvidor respondería entós con daqué asemeyao a lo que vemos na figura 15. La respuesta HTTP 1.1 que vemos a la derecha estrémase en dos partes. La primera tien les cabeceres que s'unvien nun mensaxe HEADERS y la segunda tien el ficheru que se manda nun mensaxe DATA. La filosofía ye la mesma que la del unviu de la pidimientu.

El cuerpu d'entidá (los datos), yá se podíen comprimir en versiones anteriores del protocolu. Yera daqué que yá recoyía la versión 1.0 cola cabecera *Content-coding* (Berners-Lee *et al.*, 1996). La novedá ye qu'agora los datos de cabecera que s'unvien nos mensaxes HEADERS tamién se comprimen usando la especificación de Peon y Ruellan (2014), reduciendo'l volume d'información que s'unvia.

CONCLUSIONES

Esti trabayu reflexa la evolución de los protocolos que dan encontu a los servicios web. Estos protocolos tán diseñaos pa permitir que les aplicaciones puean comunicase, ufriendo los servicios básicos que necesiten pa comunicase.

Los servicios web entamen nun sitiu mui concretu, pero espárdense rápidamente per tol mundu apaeciendo bien d'empreses que desendolquen productos pa estos servicios. Tener protocolos estándar ye importante pa garantizar la compatibilidá ente toos estos productos. El problema ye'l desacople ente los grupos d'estandarización y les empreses. Nel aniciu, los grupos d'estandarización que facíen estos protocolos pensaben les funcionalidaes que podíen ser interesantes pa les empreses, pero yá vemos qu'esto foi un fracasu. Dellos mecanismos previstos na primer revisión de HTTP 1.0 tuvieron que se quitar porque nun se llegaron a implementar enxamás. Yá dende bien aína, la proactividá de los comités volvióse reactividá. Les anovaciones de les empreses yeren tan rápides, que los estándares dieron nuna compilación de coses que yá taben usándose. Inclusive llegó'l puntu nel que se fixo daqué pa que les empreses pudieren anovar ensin romper col estándar. Poro, acompasar el trabayu d'estandarización con un entornu tan cambiante como'l teunolóxicu ye una xera percomplexa.

Finalmente, vemos qu'enxamás se fai nada perfeuto: les coses siempre puen ameyorase. Pa los usuarios de la web lo único que se nota, quiciabes, ye que les páxines son más guapes, pero que lo demás nun camuda un res. La realidá ye bien distinta. Daqué tan bultable como facer servicios más rápidos, nun ye namás un problema de la nuesa conexón a Internet. La electrónica ye meyorable, los sistemes operativos son meyorables, los productos software son meyorables, los protocolos de comunicaciones son meyorables y too esto necesita del trabayu de munchos investigadores ya inxenieros que faen la nuesa vida cada vegada más emocionante.

Les meyores na teunoloxía de la comunicación necesiten del trabayu de munchos investigadores ya inxenieros que faen la nuesa vida cada vegada más emocionante

Referencies bibliográfiques

Belshe, M., R. Peon & M. Thomson (2014).- *Hypertext Transfer Protocol version 2.* [Versión 14 de xunetu de 2014], en http://tools.ietf.org/html/draft-ietf-httpbis-http2-14 [consultada per última vegada'l 4 de setiembre de 2014].

Peon, R. & H. Ruellan, H (2014).- HPACK - Header Compression for HTTP /2, [Versión 9 de xunetu de 2014], en http://tools.ietf.org/html/draft-ietf-httpbis-header-compression-09 [consultada per última vegada'l 4 de setiembre de 2014].

Berners-Lee, T. (1989).- Information Management: A Proposal, CERN, en http://www.w3.org/History/1989/proposal-msw.html [consultada per última vegada'l 9 de mayu de 2014.

Berners-Lee, T. (1991).- The original HTTP as defined in 1991, en http://www.w3.org/Protocols/ HTTP /AsImplemented.html [consultada per última vegada'l 2 de setiembre de 2014].

Berners-Lee, T. (1992).- Basic HTTP as defined in 1992, en http://www.w3.org/Protocols/ HTTP / HTTP.html [consultada per última vegada'l dos de setiembre de 2014].

Berners-Lee, T., R. Fielding & H. Frystyk (1996).- Hypertext Transfer Protocol – HTTP /1.0, en RFC 1945. 1996.

Bush, V. (1945).- *As we may think in The Athlantic,* xunetu de 1945.

Elzinga, K., D. Evans & A. Nichols (2001).- United States v. Microsoft: Remedy or Malady. 9 George Mason Law Review 633.

Engelbart, D.C. & W.K. English (1968).- A Research Center for Augmenting Human Intellect". *AFIPS Conference Proceedings of the 1968 Fall Joint Computer Conference,* San Francisco, CA, 33:. 395-410.

Fette, I. & A. Melnikov (2011).- The WebSocket Protocol, en RFC 6455, 2011.

Fielding, R., J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach & T.Berners-Lee (1999).- Hypertext Transfer Protocol – HTTP /1.1, en RFC 2616. 1999.

Fielding, R., Y. Lafon & J. Reschke (2014).- Hypertext Transfer Protocol (HTTP /1.1): Range Requests, en RFC 7233.

Fielding, R., M. Nottingham & J. Reschke (2014).- Hypertext Transfer Protocol (HTTP/1.1): Caching, en RFC 7234.

Fielding, R. & J. Reschke (2014a).- Hypertext Transfer Protocol (HTTP /1.1): Message Syntax and Routing, en RFC 7230.

— (2014b).-Hypertext Transfer Protocol (HTTP /1.1): Semantics and Content, en RFC 7231.

— (2014c).- Hypertext Transfer Protocol (HTTP /1.1): Conditional Requests, en RFC 7232.

— (2014d).- Hypertext Transfer Protocol (HTTP/1.1): Authentication, en RFC 7235.

ISO/IEC (1997).- Information technology –Open Systems Interconnection– Protocol for providing the connection-mode transport service. ISO/IEC 8073:1997(E).

Martin, J. & J. Leben (1992).- DECnet Phase V: An OSI Implementation. Digital Press.

Nielsen, H., P. Leach & S. Lawrence (2000).- An HTTP Extension Framework. RFC 2774.

Nelson, T.H. (1967).- Getting it out of our system. *Information Retrieval: A Critical Review.* G. Schechter, ed. Thomson Books,Washington d.c: 191-210.

Postel, J. (1981).- Simple Mail Transfer Protocol, en *RFC* 788.

SPDY: An experimental protocol for a faster web, en http://dev.chromium.org/spdy/spdy-whitepaper, [consultada per última vegada'l de setiembre de 2014].