PROTOCOLO HTTP

DIANA LAURA AVILES ELIZALDE

ARQUITECTURA DE INFORMACIÓN EN WEB Y LABORATORIO

ANTONIO CARLOS CARDEÑA MATAMOROS

2 DE FEBRERO DE 2021

PRIMAVERA 2021

Índice

Ilustraciones

Resumen

El protocolo HTTP (Protocolo de Transferencia de Hipertexto) es el más utilizado de internet, con la finalidad de transferir datos a través de internet. La versión más actualizada te permite la transferencia de mensaje con encabezados que describen el contenido de los mensajes mediante una codificación. En esta tarea se describe más a detalle su funcionalidad, los principales métodos que utiliza y cuál es la ventaja de cada versión de este protocolo.

Contenido

El http es el protocolo de transmisión de información de la World Wide Web, significa que el código que se establece para entre computadores puedan “comunicarse” en un mismo idioma a la hora de transmitir información por la red. Este establece criterios de sintaxis y semántica informática para establecer dicha comunicación con los diferentes elementos que conforman a la arquitectura web como los son servidores, clientes, proxies, etc. En pocas palabras, http regula cómo el servidor envía este recurso a su cliente. Es importante destacar el funcionamiento de este protocolo, así como las características y ventajas de cada versión para observar como a lo largo del tiempo estos protocolos se mantionen actualizados conforme avanzan las nuevas tecnologías y las peticiones de usuarios. Con esta investigación se podrá observar que nos ofrece la versión más actualizada de este protocolo y empezar a trabajar sobre ello a futuro. Algunos conocimientos se tenían por la materia de Redes, en esta investigación se detalla un poco más lo aprendido y entender su aplicación.

Funcionamiento

Al escribir una dirección web en tu navegador y te muestra la página, es porque el navegador se comunicó con el servidor web por medio de HTTP. Para entender de una mejor manera se explicará a continuación qué pasa cuando se abre una página web.

Proceso

1. Al escribir en la barra de direcciones del navegador de su preferencia, el usuario teclea, por ejemplo: ibero.mx
2. El navegador envía esa solicitud, es decir, la **petición HTTP**, al servidor que administra ibero.mx. La solicitud enviada puede ver como un "¿Tienes este archivo?"
3. Posteriormente el servidor web recibe la **solicitud HTTP**, busca la página ibero.mx**,** este busca su archivo index.html, y envía un header (más adelante se hablarán de los métodos de http), este comunica al cliente mediante un código de estado, es decir, el resultado de la búsqueda.
4. Si encuentra el archivo solicitado (index.html) y el cliente a solicitado recibirlo (no solo hay que verificar que existe), el servidor web envía por el header el cuerpo del mensaje, es decir, el contenido solicitado.
5. Por último, el navegador lo recibe y abre la página web.

Para poderlo visualizar de una mejor forma observa la siguiente imagen.

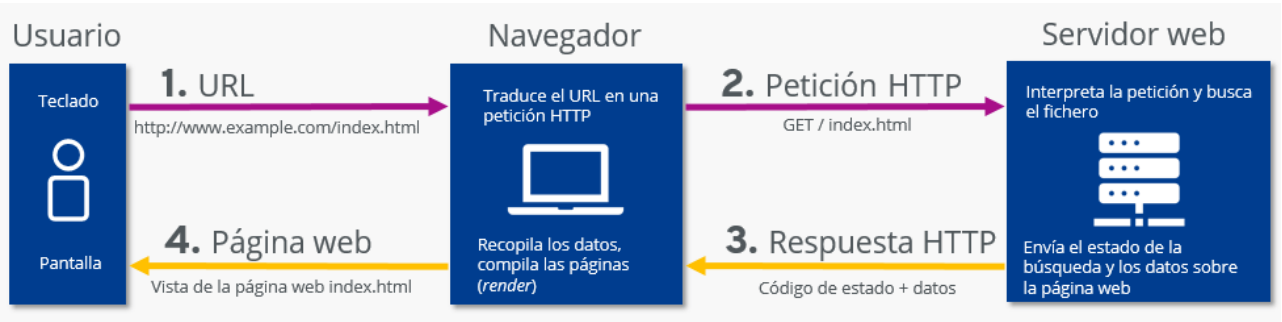

Ilustración 1 – Proceso de comunicación http[1]

[1] https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/protocolo-http/

Métodos

HTTP determina un conjuto de métodos de petición para mostrar la acción deseada para un determinado recurso. Algunos los conocen como **HTTP verbs.** Cada uno implementa una semántica diferente, pero algunos tienen características similares.

- **Get**

  Solicita una representación de un recurso en específico. Solo deben recuperar datos

- **Head**

  Solicita una respuesta idéntica a la de una petición GET, pero sin el cuerpo de la respuesta.

- **Post**

  Envía un recurso en específico, causando a menudo un cambio en el estado o efectos secundarios en el servidor.

- **Put**

  Reemplaza todas las representaciones actuales del recurso de destino con la carga útil de la petición.

- **Delete**

  Borra un recurso en específico.

- **Connect**

  Establece un túnel hacia el servidor identificado por el recurso.

- **Options**

  Describe las opciones de comunicación para el recurso destino.

- **Trace**

  Realiza una prueba de bucle de retorno de mensaje a lo largo de la ruta al recurso destino.

- **Patch**

  Aplica modificaciones parciales a un recurso.

Se pueden observar de mejor manera en la siguiente imagen.

Ilustración 2 - Peticiones HTTP

Ventajas por versión

A continuación se explicarán las características de cada versión comenzando desde la primera hasta la más actual de hoy en día.

HTTP/1

Comenzó en 1989. Su versión inicial de HTTP fue **versión 0.9,** solo era una línea y únicamente permitía solicitar un archivo HTML del servidor cada vez. Esta versión solo manejaba archivos de HTML.

En 1996, Internet Engineering Task Force (IETF) especificó en el memorando RFC1945, la version HTTP/1 como propuesta. Incluía el header que señalaba tanto solicitud del cliente cmo la respuesta del servidor; también la introducción de Content-Type que transfería otros tipos de archivos que no eran HTML, entre sus ventajas:

1. **Sin estado**

   Al finalizar la comunicación, no se recuerda, de que dicho cliente ya le ha enviado solicitudes antes.

2. **Independiente del tipo de archivo**

   Este transfiere otro tipo de archivos, pero ambas partes deben saber manejar este tipo de archivos.

3. **Conexión efímera**

   Al establecer una conexión con el servidor, se envían y se reciben solicitudes, al estar hecho, se termina dicha conexión. Para una nueva solicitud, se debe volver a establer una nueva conexión, volviendose costoso porque la mayoría de páginas web continen muchos archivos que deben recogerse por solicitudes uno a uno.

1997 se publicó **HTTP/1.1,** especificado en el RFC2068, fue el primer estándar oficial. Se sigue usando actualmente y sus ventajas son:

1. **Especificación del host**

   En el **header** la solicitud de HTTP funciona para varios dominios alojados en la misma IP. Es el caso de muchas páginas por el shared hosting[2].

2. **Caché**

   Nuevos mecanismo para guardar contenido temporalmente

3. **Rastreo de ruta**

   Con el método TRACE , se puede rastrear la ruta entre cliente y el servidor web

4. **Transferencia de datos**

   Se transfieren datos del cliente al servidor

[2] https://www.ionos.mx/digitalguide/hosting/cuestiones-tecnicas/en-que-consiste-el-hosting-compartido/

5. **Uso de MIME**

   En los chats, el navegador puede actualizar la ventana usando MIME multiparte/replace.[3]

6. **Técnica de HTTP pipelining**

   Permite al cliente enviar la siguiente solicitud sin tener que esperar a recibir la respuesta de la primera.

7. **Keepalive**

   El cliente decide mantener la conexión más allá de la solicitud, añadiendo al header el comando keepalive (manterner con vida).

[3] https://developer.mozilla.org/en-US/docs/Learn/Server-side/Configuring_server_MIME_types

HTTP/2

Al cargar una página web, este solicita mucho MB de datos y enviar más de cien solicitudes HTTP/1.1, esta procesando una solicitud tras otra en una misma conexión, en cuanto más compleja la página web, más tarde en cargarse y mostrarse en el navegador. Por esta razón google desarrollo SPDY o Speedy, permitiendo que en 2015 se publicara la versión HTTP/2. Su objetivo es acelerar la carga de las páginas web, entre sus otras ventajas son:

1. **Multiplex**

   Entre cliente y servidor se puede enviar y procesar varias solicitudes HTTP de forma simultánea.

2. **Compresión**

   Headers se comprimen, al ser algunos idénticos en muchas solicitudes HTTP, evitando así redundancias.

3. **Server Push**

   Al prever que datos le pedirá el cliente, los envía directamente a la caché del cliente, sin esperar a recibir la solicitud HTTP que corresponda.

4. **Datos binarios**

   Trabaja con este tipo de datos en lugar de archivos de texto.

Esta versión se entendió rápidamente y páginas con mucho tráfico fueron de las primeras en adoptar esta versión. En enero de 2020 según W3Techs, un 42% de las páginas web utilizan HTTP/2. Se puede ver en la siguiente ilustración la diferencia en funcionamiento de HTTP.

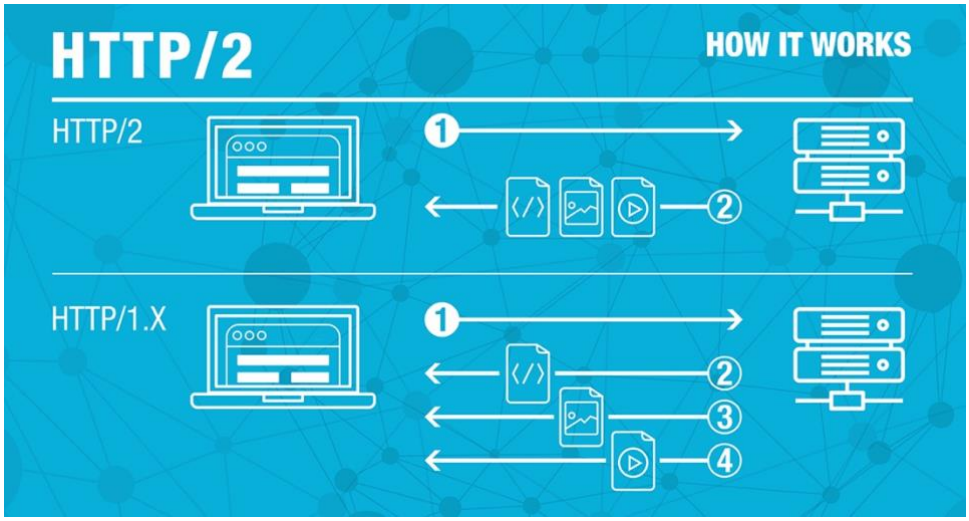

Ilustración 3 -HTTP diferencias

HTTP/3

Un punto débil de HTTP hasta ahora es el **protocolo de control de transmisión** (TCP) en el que se basan, requiere que el receptor de cada paquete de datos confirme la recepción antes de que pueda enviarse el siguiente paquete. Cuando se pierde un paquete para que todos los demás tengan que esperar a que dicho paquete sea transmitido de nuevo. Entre especialistas, estos casos son llamados **head-of-line blocking**. Para evitarlos, la nueva versión **HTTP/3** no funcionará con TCP, sino con UDP, que no aplica este tipo de medidas correctivas. A partir de UDP, se ha creado el protocolo QUIC (Quick UDP Internet Connections), que será la base de **HTTP/3**.

Google desarrolló QUIC, este evita la congestión de carga por TCP recurriendo a la transferencia por UDP, que se basa en datagramas y no requiere conexión. Al igual que TCP, UDP trabaja en la capa de transporte, pero, por el contrario, renuncia a las confirmaciones de emisor y receptor. Cada transferencia no tiene que esperar a la anterior y los trayectos de ida y vuelta entre cliente y servidor se acortan considerablemente. El IETF reconoció las ventajas de este nuevo protocolo y lo presentó en 2018 como HTTP-over-QUIC, la versión que sustituía a HTTP/2.

Básicamente, el protocolo de transporte HTTP se mantiene igual. Se sigue componiendo de un encabezado (*header*) y un cuerpo de mensaje (*body*), y emplea verbos, *cookies* y caché. La diferencia reside en la forma de transferir los datos y en la disponibilidad del cifrado integrado.

W3Techs[4] el 3% de las páginas web ya utilizan HTTP/3. Este ya se ha implementado en los navegadores Google Chrome, Microsoft Edge y Firefox, así como en Safari desde abril de 2020.

Entre sus principales ventajas son:

1. **Utilización exclusica de URL de tipo HTTPS**

   Un URL desfasado y dudoso se marca como no seguro y/o no se cifra. Con HTTP/3 sustituye el paso del cifrado TLS a nivel tcp y utiliza un cifrado TLS 1.3 automáticamente.

2. **Conexión constante y estable**

   Aunque se cambie red en una transferencia

3. **Reducción de los paquetes de datos**

   La transferencia de paquetes se hace de manera paralela.

4. **Correción de errores**

   Un forward error connection ya que tiene lugar a nivel QUIC

5. **Aumento de velocidad de transferencia**

6. **Redución de tiempos de carga**

7. **Renuncia a los handshakes preliminares**

   Que verififcan la seguridad de conexión. Este reduce la duración del establecimiento de conexión de dos pasos a uno solo.

8. **Utiliza identificadores de conexión individuales**

[4] https://w3techs.com/technologies/history_overview/site_element/all

9. **Mejora de navegación**

   Sobre todo para usuarios móviles, mejora la conexión a mñas estable, flexible y rápida.

Se considera UDP como un protocolo de carencias, los usuarios se benefician de este nuevo protocolo, pero los proveedores se enfretan a nuevos retos para cambiar de TCP y TLS a UPD Y QUIC, es el gran problema de ahora para su implementación.

Los proveedores temen que en el control de seguridad y cifrado, al no utilizar TLS, con UDP se tiene que enviar tantos paquetes en poco tiempo entonces será díficil comprobar minuciosamente el tráfico de datos por la ausencia de la autenticación TLS. En muy riesgosa es la parte de seguridad, el riesgo de entrada de malware en el flujo de datos es mucho mayor.

A continuación se muestra un gráfico de su funcionamiento y si reducción de tiempos.

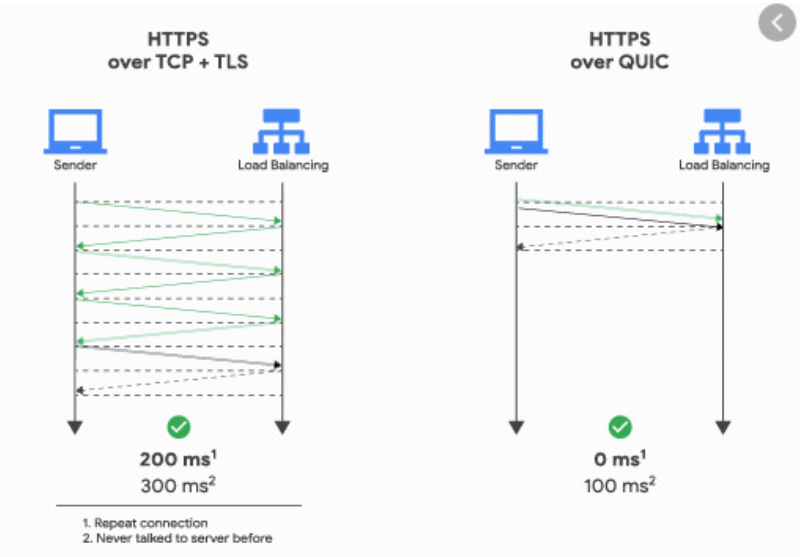

Ilustración 4 - HTTP/2 vs HTTP/3

Para complementar la información esta línea de tiempo marca la estadarización por cada versión HTTP.

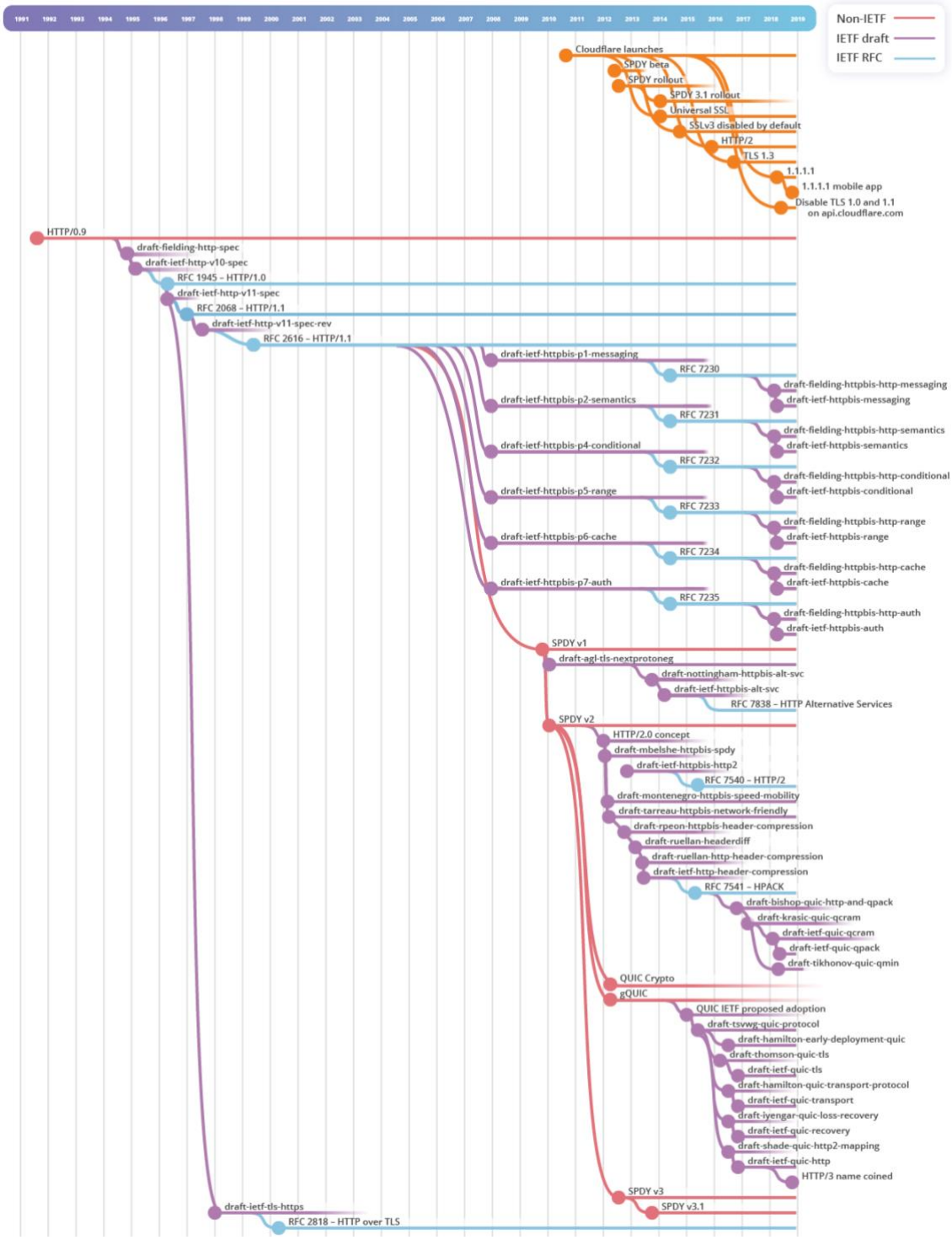

Ilustración 5 - Estadarizaciones de HTTP [5]

[5] https://blog.cloudflare.com/es/http-3-from-root-to-tip-es/

Conclusiones

Conocer las versiones más actuales de los protocolos, herramientas, softwares, etc; nos ayudan a observar como podemos resolver un problema actual con la tecnología nueva, ya que por lo general son mejoras que agilizan un proceso de alguna función o problema. Actualmente la parte web tiene gran futuro, saber como poder afrontar problemas de conexión y lo más importante de la seguridad web, ya que conforme avanza la tecnología esta tiene que estar actualizada para mayor confiabilidad. Ahora con la idea de UDP, como se mencionó anteriormente será un reto para los proveedores, se tendrá que buscar la forma de abordar ese problema ya sea con una nueva tecnología o con alguna que ya exista. En mi opinión es por eso que se ha tardado un poco en implementarse en nuevos sitios, HTTP/3 beneficia al usuario pero ¿Cómo mantendrán los proovedores de internet esa nueva tecnología?, sin duda es un gran reto y alguien que le agrade la parte de seguridad en web, podría enfocarse en este problema.

Bibliografía

1. 1&1 IONOS España S.L.U. (2020, 9 diciembre). *protocolo HTTP*. IONOS Digitalguide.https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/protocolo-http/
2. Fernández, Y. (2020, 6 marzo). *HTTP/3: qué es, de dónde viene, y qué es lo que cambia para buscar un Internet más rápido*. Xataka. https://www.xataka.com/basics/http-3-que-donde-viene-que-que-cambia-para-buscar-internet-rapido
3. *Generalidades del protocolo HTTP - HTTP | MDN*. (2020, 7 agosto). MDN Web Docs. https://developer.mozilla.org/es/docs/Web/HTTP/Overview
4. *Historical trends in the usage statistics of site elements for websites, February 2021*. (s. f.). W3Techs. Recuperado 1 de febrero de 2021, de https://w3techs.com/technologies/history_overview/site_element/all
5. *HTTP/3: las claves del nuevo protocolo de transferencia de hipertexto*. (2020, 26 agosto). IONOS Digitalguide. https://www.ionos.mx/digitalguide/hosting/cuestiones-tecnicas/http3/
6. *Métodos de petición HTTP - HTTP | MDN*. (2020, 15 octubre). MDN Web Docs. https://developer.mozilla.org/es/docs/Web/HTTP/Methods
7. Pardue, L. (2020, 24 marzo). *HTTP/3: De arriba a abajo*. The Cloudflare Blog. https://blog.cloudflare.com/es/http-3-from-root-to-tip-es/