Lunes 13 de Agosto, 2018

Protocolo HTTP/2

Ian Roberts 201603030-1

Gustavo Matamala 201730020-5

Joaquín Opazo 201730039-6

Ignacio Díaz 201721002-8

➢ Resumen

Hoy en día el intercambio de paquetes en internet es prácticamente gobernado por sólo un protocolo, el protocolo HTTP. Si bien, este protocolo lleva más de 20 años funcionando bastante bien, se ha tenido que innovar a su alrededor para evitar que este caiga en la obsolescencia; incluso creando verdaderos “Hacks”, para obtener el óptimo uso de los recursos disponibles. Es por esto, que se creó una nueva versión del protocolo mencionado, la cual estudiaremos en este trabajo, llamado HTTP/2. Para esto, inicialmente investigaremos y hablaremos sobre cuáles son sus principales características y diferencias con la versión anterior. Luego observaremos cómo se comporta este nuevo protocolo, primero de manera práctica observando la carga de un mismo objeto utilizando ambos protocolos, y después analizando más detalladamente los paquetes enviados en la carga de este objeto. A través de esto podremos ver como efectivamente HTTP/2 utilizando una conexión única permite mejor rendimiento en la red.

➢ Introducción:

El protocolo HTTP es un protocolo creado formalmente en 1991 con HTTP/0.9 que nació siendo prácticamente un protocolo obsoleto, pero con mucho espacio para crecer. Este no contenía ni siquiera el comando POST por lo que el usuario no podía interactuar realmente con el servidor. Luego en 1996 nace HTTP/1.0, este protocolo es verdaderamente revisado y permite una interacción del usuario con el servidor. Incluso esto le permite ser utilizado hasta el día de hoy, en ciertos servicios, como en los servidores proxy. Aun así, este protocolo presentaba ciertas ineficiencias, que para la red creciente del momento, empezaron a hacerse notar. Por esto nace el protocolo HTTP/1.1 en 1999. Este protocolo es el más usado en la web hasta hoy en día y fue el que analizamos previamente en este ramo. [(1)]

Pero esto ya sucedió hace casi 20 años, lo que es una eternidad para la ciencia de la computación. En comparación, los discos duros de fines del siglo pasado tenían menos 40 GB de capacidad, mientras que hoy en día los discos duros alcanzan sin problemas los 5000 GB. Entonces si comparamos, ¿cómo es que en 20 años no hemos diseñado un nuevo protocolo? Y más aún, ¿qué efecto tiene en la web el utilizar un protocolo casi

"arcaico"? Es por esto que queremos presentarles lo que sería el futuro cercano de la red. El protocolo HTTP/2.

➢ ¿Por qué implementar un nuevo protocolo?

Bueno, como dijimos previamente este es un protocolo bastante viejo. Pero si la red funciona correctamente con él. ¿Por qué tenemos que modificarlo? Bueno, como podemos ver en el video de la charla de Simone Bordet[(2)], la velocidad a la que carga la página genera una gran diferencia hoy en día. Antiguamente las empresas se enfocaban en mantener una presencia en la red sin invertir mucho en el mantenimiento o renovación de sus aparatos. Hoy en día estamos viendo que cada vez es más importante hacer que el acceso a la red para los usuarios sea lo más rápido posible. Esto porque le permite a las empresas tener un mejor "ranking" en Google, lo que quiere decir mejores ingresos y/o mayor notoriedad. Esto es visible en los diversos estudios presentados en la charla, como por ejemplo el caso de *"Shopzilla"*, empresa que mejorando el tiempo de carga de su página de 6 [s] a 1[s] lograron mejorar sus ingresos de 12%. Otro caso que vale la pena resaltar es el de Mozilla, la empresa diseñadora del conocido browser, quien disminuyendo en 2,2[s] el tiempo de carga de su página principal *(www.mozilla.org)* lograron aumentar sus descargas por año en 60 millones.

Entonces queda claro que tener un protocolo obsoleto, el cual tiene ciertas deficiencias en el uso de la red moderna, va a generar un mal uso de las capacidades de las nuevas tecnologías. Y como esto se traduce a una pérdida de posibles consumidores, queda claro que no solo en el ámbito del desarrollo de la ciencia, sino que en el ámbito económico, es necesario aprender, mejorar y desarrollar un mejor protocolo.

➢ Descripción de HTTP/2

El protocolo HTTP/2 entra al mercado no como un protocolo completamente nuevo, sino más bien como una serie de mejoras que eran necesarias implementar en el protocolo HTTP/1.1. Es por esto que los métodos, los códigos de estatus y las semánticas se mantienen. Donde sí había que generar cambios era en el desempeño del protocolo antiguo, y aquí es donde HTTP/2 brilla; reduciendo la latencia que percibe el usuario, así como mejorando la utilización de recursos de la red y de los servidores. ¿Pero, cómo

logra esto? Dentro de las mejoras que se pueden apreciar de HTTP/2 sobre HTTP/1.x (cualquier versión de HTTP/1) están:

a) **Es multiplexado**: Uno de los mayores problemas en HTTP/1.1 era el de “Head Of Line Blocking (HOL)” que consistía en que el host no podía enviar más requests hasta que hubiera llegado la respuesta al request que ya había enviado. Esto producía un encolamiento al momento de comunicar host-servidor, lo que aumentaba el tiempo de carga de una página web. Este problema intentó solucionarse con el paralelismo, el que habría múltiples conexiones TCP para pedir los elementos para cargar cierta página de forma paralela, pero tanto host como servidor utilizaban más recursos. En HTTP/2 este problema se resuelve gracias a la multiplexación, que permite que se envien múltiples frames de HTTP/2 sobre una misma conexión TCP, sin importar el orden en que estos lleguen de vuelta. (Esta es la principal diferencia con el pipelining de HTTP/1.1).
b) **Compresión de cabecera**: En HTTP/1.1 se presentaba el problema de que en las cabeceras se repetía mucha información (principalmente User-Agent y Cookies, que son estáticas), lo que agregaba bytes innecesarios al paquete. Además, los headers no podían ser comprimidos, (a diferencia del data), pero si consideramos headers de 400 bytes (cuando utilizan cookies), al final estamos estamos dejando bastante información sin comprimir. HTTP/2 logra reducir de gran manera toda esta información extra que se envía, utilizando un solo paquete de HEADER, el cual luego es referenciado por los siguientes paquetes.
c) **Server-Push**: Al momento de hacer el request de una página web, casi siempre se pide el documento index.html, que contiene toda la página. Al recibir este documento, el host “lo lee” y comienza a pedir lo necesario para cargar por completo el sitio. Esto genera un retardo entre recibir y leer, antes de comenzar a pedir nuevamente. Esta situación es arreglada por HTTP/2 con la opción de Server-Push, la cual permite al servidor, luego de recibir un request por index.html, el enviar los datos que sabe que el host va a pedir luego, con el fin de reducir este retardo.
d) **Es binario**: Esto tiene que ver con el formato en que es presentada la información (principalmente los headers) dentro de un paquete HTTP/2. Esto hace que sea un

poco más difícil de leer para los humanos, pero para las máquinas facilita bastante el proceso, lo que permite nuevamente reducir latencia y recursos utilizados, que eran los objetivos de este protocolo.

➢ Análisis de capturas.

```
> Frame 117: 401 bytes on wire (3208 bits), 401 bytes captured (3208 bits) on interface 0
> Ethernet II, Src: Dell_20:21:8b (74:86:7a:20:21:8b), Dst: ArrisGro_82:30:f8 (40:0d:10:82:30:f8)
> Internet Protocol Version 4, Src: 192.168.0.16, Dst: 37.252.251.43
> Transmission Control Protocol, Src Port: 58867, Dst Port: 80, Seq: 1, Ack: 1, Len: 347
> Hypertext Transfer Protocol
```

Paquete HTTP/1.1 muestra distinto puerto de origen para los siguientes paquetes HTTP (esto se verá en la demostración práctica), lo que muestra el paralelismo del protocolo.

```
> Frame 1977: 85 bytes on wire (680 bits), 85 bytes captured (680 bits) on interface 0
> Ethernet II, Src: Dell_20:21:8b (74:86:7a:20:21:8b), Dst: ArrisGro_82:30:f8 (40:0d:10:82:30:f8)
> Internet Protocol Version 4, Src: 192.168.0.16, Dst: 37.252.251.43
> Transmission Control Protocol, Src Port: 58883, Dst Port: 443, Seq: 1050, Ack: 5271, Len: 31
> Secure Sockets Layer
> HyperText Transfer Protocol 2
```

Mismo puerto de origen, (para todos los paquetes que siguen, acá se muestra 1 solo) lo que muestra multiplexación.

```
v Hypertext Transfer Protocol
  > GET / HTTP/1.1\r\n
    Host: www.http2demo.io\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:61.0) Gecko/20100101 Firefox/61.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
    Accept-Language: en,en-US;q=0.8,es-MX;q=0.5,es;q=0.3\r\n
    Accept-Encoding: gzip, deflate\r\n
    Connection: keep-alive\r\n
    Upgrade-Insecure-Requests: 1\r\n
    \r\n
```

HTTP/1.1 que muestra el header, tipo texto, el cual contiene mucha información en pocos campos, lo que dificulta la lectura de datos por la maquina.

```
> Header: :method: GET
> Header: :path: /demo/index_sp.html?num=20
> Header: :authority: http2.akamai.com
> Header: :scheme: https
> Header: user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:61.0) Gecko/20100101 Firefox/61.0
> Header: accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
> Header: accept-language: en,en-US;q=0.8,es-MX;q=0.5,es;q=0.3
> Header: accept-encoding: gzip, deflate, br
> Header: referer: https://http2.akamai.com/demo/http2-lab.html
> Header: upgrade-insecure-requests: 1
```

HTTP/2 que muestra la forma binaria del header, donde se puede ver la división del campo y su respectivo dato.

```
▾ Hypertext Transfer Protocol 2
  ▾ Stream: PRIORITY, Stream ID: 4, Length 5
      Length: 5
      Type: PRIORITY (2)
    ▾ Flags: 0x00
        0000 0000 = Unused: 0x00
      0... .... .... .... .... .... .... .... = Reserved: 0x0
      .000 0000 0000 0000 0000 0000 0000 0100 = Stream Identifier: 4
      0... .... .... .... .... .... .... .... = Exclusive: False
      .000 0000 0000 0000 0000 0000 0000 0000 = Stream Dependency: 0
      Weight: 1
      [Weight real: 2]
```

Paquete HTTP/2 que muestra la versión comprimida del header (no hay header a pesar de esta enviando a la IP de la página web).

770	11.274789	192.168.0.16	200.83.1.4	DNS	84	Standard query 0xd6c0 A api.accounts.firefox.
771	11.293833	104.114.232.157	192.168.0.16	HTTP2	104	PUSH_PROMISE[25]: GET /resources/push.css
772	11.294393	192.168.0.16	104.114.232.157	HTTP2	97	PRIORITY[4]
773	11.296967	200.83.1.4	192.168.0.16	DNS	132	Standard query response 0xd6c0 A api.accounts

Paquete HTTP/2 que muestra un servidor avisando al host sobre la posibilidad de usar el Server-Push.

➢ Conclusión

Como conclusión, en el presente trabajo de investigación hemos logrado demostrar las enormes ventajas que posee HTTP/2, en comparación a su antecesor, por lo cual podemos decir que este protocolo es mucho más útil, ya que por ejemplo aporta muchas cosas útiles como: una única conexión para ofrecer múltiples solicitudes y respuestas en paralelo, la eliminación de información redundante cuyo objetivo es evitar el envío de datos repetidos durante una misma conexión, o la multiplexación la cual permite enviar y recibir varios mensajes al mismo tiempo optimizando la comunicación y así se consigue reducir el número de conexiones mejorando considerablemente la velocidad de carga y disminuyendo la congestión de los servidores web, entre otros.

Al adentrarnos más en el tema del protocolo HTTP/2, logramos determinar que si era mucho más rápido y conveniente que el HTTP/1.1 y obviamente que el HTTP/1.0, y todo esto debido a que gracias a la forma en que estaba programado servía para aligerar la carga del envío y transporte de datos, por lo cual reducía la congestión y hacia mas optimo el sistema de conexiones entre redes.

➢ Referencias

❖ (1) Wikipedia – Protocolo de transferencia de hipertexto https://es.wikipedia.org/wiki/Protocolo_de_transferencia_de_hipertexto, Sub índice: “Versión”.

❖ (2) Youtube - HTTP 2.0: why and how by Simone Bordet https://youtu.be/UVwI1K0M7P0

❖ (3) https://tools.ietf.org/html/rfc7540

❖ (4) Wireshark