Ingeniería de Redes (2016-2017)
Grado en Ingeniería Informática
Universidad de Granada

Trabajo de ingeniería de redes

Javier Oliva Cruz y Javier Gomez Luzón

29/10/2016

Índice

Índice de figuras

1. Introducción a SPDY

1.1. ¿Qué es SPDY?

SPDY es un protocolo de transporte diseñado como un sucesor de HTTP, desarrollado por un equipo que incluye ingenieros encargados en la creación de otros protocolos como HTTP/2, este protocolo nace a partir de la experimentación de protocolos alternativos que ayuden a reducir la latencia de las páginas web. SPDY es un protocolo que se beneficia del código abierto, la comunidad contribuye con ideas, código y resultados de test para hacer que SPDY sea el protocolo más rápido de la siguiente generación. A día de hoy SPDY reduce el tiempo de carga respecto a HTTP en alrededor de un 64

1.2. Definiciónes

- **Clientes:** Es el endpoint que inicia la sesion SPDY.
- **Conexión:** El transporte entre dos endpoints.
- **Endpoint:** El cliente o el servidor de una conexión.
- **Frame:** Una cabecera prefijada por una secuencia de bytes enviada a través de una sesión de SPDY.
- **Servidor:** El endpoint que no inicializa la sesión SPDY.
- **Stream:** flujo bidireccional de bytes a traves de un canal virtual en una sesión de SPDY.

1.3. HTTP y sus principales problemas

Para entender mejor las ventajas del protocolo SPDY hace falta entender también a HTTP y los problemas que este tiene. HTTP (Hypertext Transfer Protocol) es un protocolo de transacción de datos con el esquema cliente-servidor que se usa para la entrega de archivos y otros datos llamados 'mensajes'. Los mensajes HTTP son un texto plano, lo que los hace más legibles y fáciles de depurar, lo que conlleva que estos mensajes sean más largos. Los mensajes siguen la estructura de:

- **Línea inicial**
- **Cabecera**
- **Cuerpo del mensaje.** El cual es opcional

HTTP cuenta con una serie de "problemas" que nos obliga a buscar unos protocolos alternativos. Una de ellas es el cuello de botella que ocurre en HTTP es que solo se basa en múltiples conexiones concurrentes. Esto causa varios problemas que el HTTP "pipelining" no soluciona, ya que la conexión puede ser bloqueada en la solicitud de la línea de la cabecera. Además, muchos proxies tienen un mal soporte para el pipelining.

1.4. Principales diferencias entre SPDY y HTTP

SPDY pretende abordar esto y otros problemas relacionados con las web modernas, usando tres mejoras básicas respecto a HTTP:

- **Cabeceras comprimidas.** Los clientes actuales envían una gran cantidad de datos redundantes en las cabeceras de los HTTP, porque una sola página web puede requerir entre 50 o 100 subsolitudes. Comprimiendo las cabeceras conseguimos reducir la latencia y ancho de banda que se requiere comparado con HTTP.
- **Solicitudes multiplexadas.** No hay límite para el número de solicitudes que pueden ser emitidas concurrentemente en una sola conexión SPDY, ya que las solicitudes pueden ser intercaladas en un solo canal, la eficiencia del TCP es mucho más alta.
- **Priorizar solicitudes.** Los clientes pueden solicitar que recurso se les entregará primero. Con esto evitamos problemas de atasco en la red provocados por solicitudes no críticas que obstaculizan aquellas con mayor prioridad.

2. ¿Como se conecta SPDY?

Para las conexiones SPDY utiliza el protocolo de transporte TCP, con este las aplicaciones pueden conectarse de manera segura independientemente de las capas inferiores, lo que significa que los router solo tienen que enviar los datos en forma de datagrama. Para cumplir con un buen rendimiento, se espera que los clientes no cierren las conexiones abiertas hasta que el usuario salga de todas las web o hasta que el servidor cierre la conexión.
Los servidores están encargados de dejar la conexión abierta tanto como sea posible, pero pueden cerrar conexiones ociosas en el caso de que sea necesario.
Cuando cualquier endpoint cierre la conexion con el nivel de transporte, debe en primer lugar mandar un GOAWAY frame con lo que los endpoints pueden determinar si requiere finalizar la conexion antes de cerrarla.

3. Funcionamiento del framing

Una vez se establece la conexión, los clientes y servidores intercambian mensajes a través de 2 frames.

- **Control frames.**
- **Data frames.**

Estos frames mantienen una serie de características comunes, que explicaremos a continuación.

3.1. Control frames

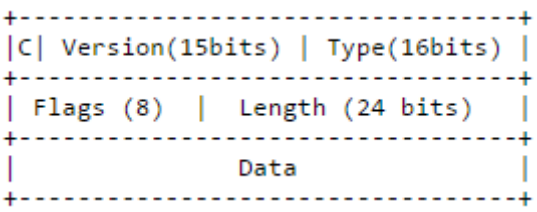

Figura 3.1: Estructura del Control frames

- **Bit de control:** La 'C' es un solo bit que indica si este es un mensaje de control. ara los frames de control este valor es siempre 1.
- **Versión:** El número de versión del protocolo SPDY.Su tamaño es de 15 bits.
- **tipo:** Es el tipo de control_frame.
- **Flags:** Sus flags son diferentes a los que tienen los data_frames.
- **Longitud:** Un valor de 24 bits representando el número de bytes de la longitud del campo.
- **Data:** El formato y longitud de estos datos está controlado por Type.

Requerimientos del control_frame:
La longitud completa de los control_frame son de 16MB, pueden ser grandes para las implementaciones con recursos limitados. En estos casos, las implementaciones pueden limitar el marco de la longitud máxima admitida. Sin embargo, todas las implementaciones deben ser capaz de recibir tramas de control de al menos 256 bytes.

3.2. Data frames

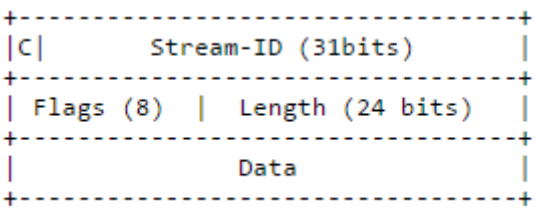

Figura 3.2: Estructura del Data frames

- **Bit de control:** Será siempre 0.

- **Stream-ID:** Un valor de que identifica al stream de 31 bits.
- **Flags:** Solo tiene un flag (0x01 =FLAG_FIN - indica que este marco representa la última trama a transmitir en esta corriente).
- **Length:** Un valor de 24 bits sin signo que representa el número de bytes después del campo de longitud. El tamaño total es 8 bytes+length.

Requisitos de procesamiento de data_frame:
Si un endpoint recibe un data_frame de un stream-id que no existe, debe devolver un RST_STREAM con INVALID_STREAM para el stream-id. Si el endpoint que crea la corriente recibe un data_frame antes de recibir un SYN_REPLY en ese stream, se trata de un error de protocolo, y el receptor debe cerrar la conexión inmediatamente. Si un endpoint recibe múltiples data_frames con streams-ids inválidos, puede terminar la sesión.

4. Funcionamiento de los streams

Los streams son secuencias independientes de datos bidireccionales divididos en marcos con varias propiedades:

- Pueden ser creados por el cliente o el servidor.
- Los stream pueden llevar un conjunto de pares de cabeceras nombre/valor.
- Son capaces de enviar datos simultaneamente intercalados con otros streams.
- Pueden ser cancelados.

4.1. Stream frames

SPDY define 3 control frames para controlar el ciclo de vida de un stream:

- **SYN_STREAM:** Abre un nuevo stream.
- **SYN_REPLY:** Libera un stream.
- **RST_STREAM:** Cierra stream.

5. Data flow

Debido a que TCP proporciona un único stream de datos el cual utiliza SPDY para mandar multiples corrientes lógicas, los clientes y servidores deben de forma inteligente intercalar los mensajes de datos durante sesiones concurrentes.

6. ¿Cuales son los control frames que utiliza SPDY?

Son los siguientes:

6.1. SYN_STREAM

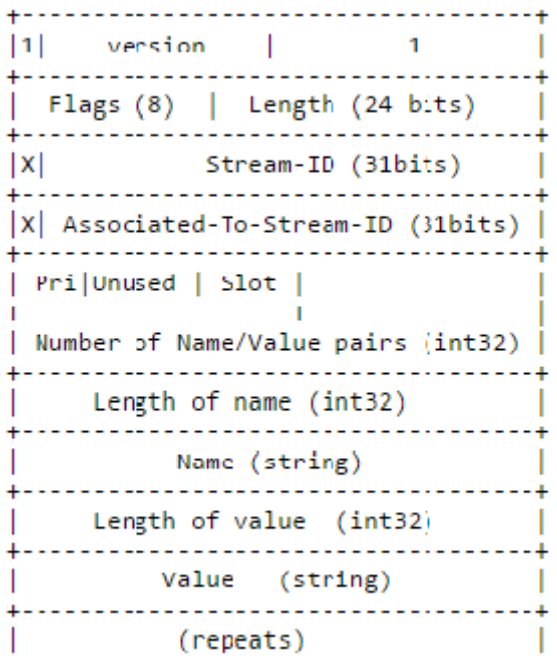

Figura 6.1: Estructura del Control frames
Permite al remitente crear un stream de forma asíncrona entre los endpoints.

- **Stream-ID:** Un identificador de 31 bits para el stream.
- **ID asociado a un stream** Un identificador de 31 bits al cual está asociado. Si el tream es independiente este valor debería ser 0.
- **Prioridad:** 3 bits con la prioridad.
- **Sin uso:** 5 bits de espacio sin uso.
- **Slot:** 8 bits de un entero sin signo especificando el índice del vector de credenciales del servidor certificado de cliente a utilizar en esta solicitud. Si hay un 0 significa que no hay significado a este stream.
- **Bloque cabecera nombre/valor** Un nombre o valor para la parte del SYN_STREAM. Si el endpoint recibe un SYN_STREAM

Si un endpoint recibe un SYN_STREAM que es más grande que los que la implementación soporta, se puede enviar un RST_STREAM con un FRAME_TOO_LARGE. Todas las implementaciones deben soportar los tamaños mínimos.

6.2. SYN_REPLY

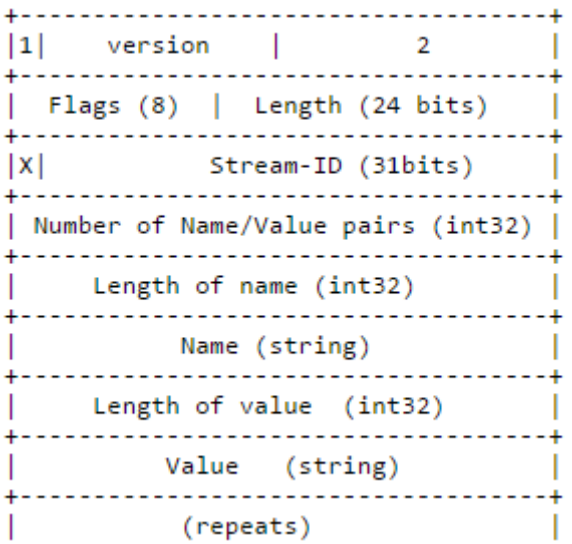

Figura 6.2: Estructura del SYN_REPLY

Permite al remitente crear un stream de forma asíncrona entre los endpoints.
Indica la aceptación de una corriente de creación por parte del receptor de un SYN_STREAM.

- **Stream-ID:** Similar al campo Stream-ID SYN_STREAM. Si un endpoint recibe varios SYN_REPLY por el mismo ID del stream activo, deberá emitir un error de stream con el código STREAM_IN_USE.
- **Bloque de cabecera nombre/valor** Similar al campo del mismo nombre del SYN_STREAM.

Si un endpoint recibe un SYN_REPLY que es más grande que lo que la implementación es capaz de soportar, se puede enviar un RST_STREAM con el código de error FRAME_TOO_LARGE. Todas las implementaciones deben soportar los límites mínimos de tamaño definidos.

6.3. RST_STREAM

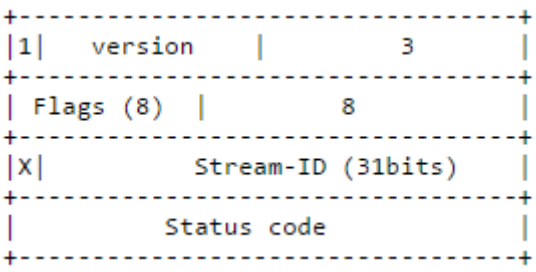

Figura 6.3: Estructura del RST_STREAM

Permite la terminación anormal de un stream. Cuando es enviada por el creador de un stream, indica que hay que cancelar el stream. Cuando enviada por el destinatario de una corriente, que indica que hay un error o que el destinatario no quería aceptar el stream, por lo que el stream se cerrará.

- **Flags:** RST_STREAM No usa ningún flag por lo que su valor es 0.
- **Longitud:** Puede llegar a tener un tamaño de 24 bits pero en el caso de RST_STREAM, este valor es siempre 8.
- **Stream-ID:** 31 bits que forman la id.
- **Status code:** Un indicador de 32 bits para cuando el stream está terminado.

  - **PROTOCOL_ERROR:** Error genérico.
  - **INVALID_STREAM:** Cuando el stream no está activo.
  - **REFUSED_STREAM:** El stream fue rechazado antes de que se hiciera ningún proceso.
  - **UNSUPPORTED_VERSION:** Indica que el destinatario de un stream no es compatible con la versión del SPDY solicitado.
  - **CANCEL:** Utilizado por el creador de un stream para indicar que el stream ya no es necesario.
  - **INTERNAL_ERROR:** Es un error genérico que puede ser usado cuando la aplicación ha fallado internamente.
  - **FLOW_CONTROL_ERROR:** Se violó el protocolo de control de flujo.
  - **STREAM_IN_USE:** El endpoint recibió un SYN_REPLY para un stream ya abierto.
  - **STREAM_ALREADY_CLOSED:** El endpoint recibe un stream de datos o SYN_REPLY de un stream que está medio cerrado.

- **INVALID_CREDENTIALS:** El servidor recibió una solicitud de un recurso cuyo origen no tiene credenciales válidas en el vector de certificado del cliente.
- **FRAME_TOO_LARGE:** El endpoint recibe un stream que la aplicación no puede soportar. Si se envía sin procesar completamente la parte comprimida, el estado de compresión estará fuera de sincronía con el otro endpoint. Entonces el remitente deberá cerrar el stream.

6.4. GOAWAY

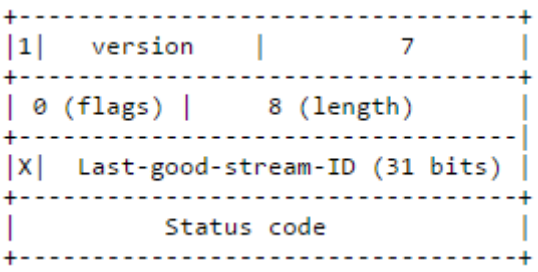

Figura 6.4: Estructura del GOAWAY

El GOAWAY control frame es un mecanismo para decir debe parar de crear stream en la sesion. Este frame de control puede ser enviado por el cliente o el servidor y, una vez enviado el remitente no responderá a ninguna SYN_STREAM en esta sesión.

Hay que tener en cuenta que algunos servidores optan por enviar el GOAWAY y finalizar inmediatamente la conexión sin esperar que el servidor de streams activos acabe. El cliente será capaz de determinar esto ya que los stream SPDY han terminado de manera abrupta lo que obliga al cliente a decidir si debe reintentar las peticiones pendientes. Los clientes deberán ser capaces de hacer frente a este caso. Los servidores más novedosos serán capaces de terminar el flujo de stream activo antes de terminar la conexión. Si un cliente que utilice SPDY cierra la conexion, tambien debería mandar un mensaje GOAWAY.

EL GOAWAY tiene una estructura muy simple, cuenta con:

- **Bit de control.**
- **La version de SPDY.**
- **Tipo.** Siempre es 7 en el caso de GOAWAY.
- **La longitud del frame.** Este es siempre 8 bytes.
- **La ID del último stream respondido.** Consta de 31 bits.

- **Razón por la que se ha cerrado la sesión.**

6.5. SETTINGS

El frame SETTINGS contiene un conjunto de pares de valores ID/valor para la comunicacion de la configuracion de datos sobre como los dos endpoints deben comunicarse.

El marco de SETTINGS puede ser enviado a cualquier momento por cualquier endpoint, su envío es opcional y totalmente asíncrona. Cuando el servidor es el remitente, este puede solicitar la configuración de datos del cliente a través de la sesión de SPDY.

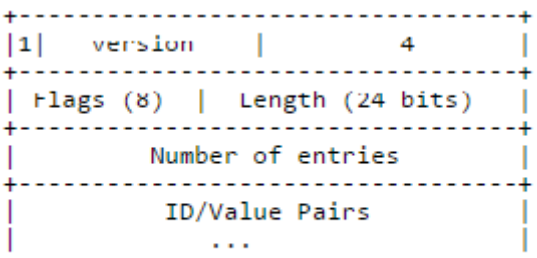

Figura 6.5: Estructura del SETTINGS

- **Bit de control.**
- **La version de SPDY.**
- **Tipo.** Es siempre es 4 en el caso de SETTINGS.
- **Un flag que puede ser:** FLAG_SETTINGS_CLEAR_SETTING o FLAG_SETTINGS_PERSIST_VALUE según si existe o no un par de ID/valor
- **La ID del último stream respondido.** Consta de 31 bits.
- **Una longitud de 24-bit.** El tamaño total de un SETTINGS frame es de 8 bytes + la longitud.
- **Número de entradas.** Un valor de 32-bit que representa el número de pares de ID/valor en el mensaje.
- **Par de ID/valor.**

Dentro del par de ID/valor encontramos una subdivisión que consta de:

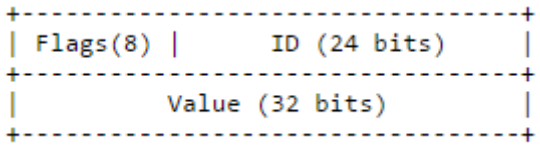

Figura 6.6: Estructura de la pareja ID/valor

- **Un flag que puede ser FLAG_SETTINGS_CLEAR_SETTING o FLAG_SETTINGS_PERSIST_VALUE** Según si existe o no un par de id/valor, es la misma descrita anteriormente.
- **ID (24-bits) :**
  - **SETTINGS_UPLOAD_BANDWIDTH:** Permite al remitente enviar el valor máximo de su ancho de banda de subida por este canal. Este número es una estimación.
  - **SETTINGS_DOWNLOAD_BANDWIDTH:** Permite al remitente enviar el valor máximo de su ancho de banda de descarga por este canal. Este número es una estimación.
  - **SETTINGS_ROUND_TRIP_TIME:** Permite enviar el tiempo mínimo que tarda una control frame en enviarse y recibir respuesta. Este valor se representa en milisegundos.
  - **SETTINGS_MAX_CONCURRENT_STREAMS:** Envía información acerca del máximo número de streams concurrentes inicializados remotamente. No debería ser menor de 100.
  - **SETTINGS_CURRENT_CWND:** Permite informar al endpoint remoto sobre el valor TCP CWND.
  - **SETTINGS_DOWNLOAD_RETRANS_RATE:** Permite al remitente enviar informacion acerca de los bytes retransmitidos.
  - **SETTINGS_INITIAL_WINDOW_SIZE:** Informa sobre el tamaño de la ventana.
  - **SETTINGS_CLIENT_CERTIFICATE_VECTOR_SIZE:** Nos indica el tamañó del vector certificado de cliente.

6.6. PING

El frame de control PING es un mecanismo de medicion del tiempo de ida y vuelta del emisor. Puede ser enviado desde un cliente o un servidor. Los destinatarios del frame de ping deben enviar el mismo frame al emisor tan pronto como sea posible. Cada ping debe utilizar su propio ID.

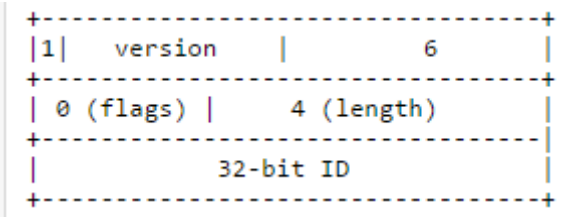

Figura 6.7: Estructura del PING

- **Bit de control.**
- **La versión de SPDY.**
- **Tipo** Siempre es 6 en el caso de PING.
- **La longitud del frame.** Este es siempre 4 bytes.
- **ID:** Es un valor de 32 bits que manda tanto el cliente como el servidor. Tiene una característica especial que se trata de que los usuarios mandan el ping con una ID par, mientras que los servidores con una impar. Esto busca evitar bucles erroneos ya que hay una posibilidad de que tanto el cliente como el servidor manden el ping a la vez. Una vez se han usado todos los IDs ($2\hat{3}1$ posibilidades) se empiezan a reusar las IDs.

6.7. HEADERS

Los frames HEADERS aumentan los streams con cabeceras adicionales. Este puede ser mandado a un stream ya existente. El bloque de la cabecera formado por nombre/valor está comprimido en este frame.
Tiene la siguiente estructura:

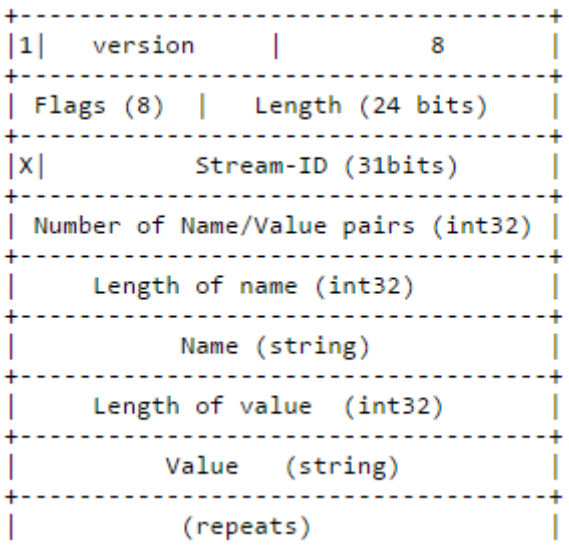

Figura 6.8: Estructura del HEADERS

- **Flags.** El flag válido para este frame es el FLAG_FIN, este flag indica cual es el ultimo frame que ha sido transmtido por el stream.
- **Longitud.** Tiene 24 bits de longitud, también tiene un valor mínimo es que es 4 bits(0000)
- **Stream-ID.** Es el stream con el que la cabecera está asociada.
- **Nombre/valor de cabecera.** Utiliza el conjunto de pares de nombre/valor que se explica en SYN_Stream

6.8. WINDOW UPDATE

El frame de control WINDOW_UPDATE se usa para controlar la corriente del stream de SPDY. El control de flujo en SPDY es por salto, es decir, solo entre los dos endpoints de una conexión SPDY. Si hay uno o más intermediarios entre el cliente y el servidor de origen, el flujo de señales de control no se reenvían explícitamente por los intermediarios.Los destinatarios deben poder soportar todos los control_frames. Si uno destinatario no puede no puede soportarlo deberá emitir un stream error con el código FLOW_CONTROL_ERROR.

El control de flujo de SPDY está implementado mediante una ventana de transferencia de datos mantenida por el emisor de cada corriente que indica el número de bytes de datos que el remitente puede transmitir. Después se crea un stream, pero antes de que se haya transmitido algún data_frame, el remitente comienza con el tamaño de ventana inicial. Este tamaño de la ventana es una medida de de la capacidad de buffering del

destinatario. El remitente no debe enviar un data_frame mayor que el tamaño de la ventana de transferencia. Después de enviar cada data_frame, el emisor disminuye su tamaño de la ventana de transferencia por la cantidad de datos transmitidos. Cuando el tamaño de la ventana se hace menor o igual a 0, el remitente debe hacer una pausa en la transmisión de data_frames. En el otro extremo de la corriente, el receptor envía un WINDOW_UPDATE para volver a notificar al remitente de que se ha consumido algunos datos y ha liberado espacio de buffering para recibir más datos.

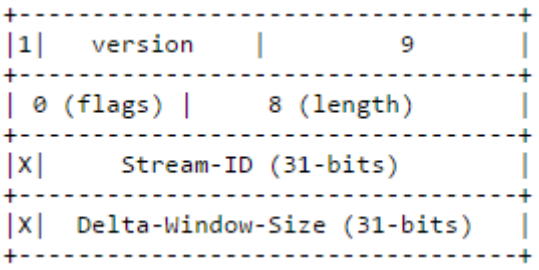

Figura 6.9: Estructura del window update

- **Bit de control:** El bit de control es siempre 1 para este mensaje.
- **Versión:** El número de versión de SPDY.
- **Type:** El tipo de mensaje para un mensaje WINDOW_UPDATE es 9.
- **Length:** El campo de longitud es siempre 8.
- **Stream-ID:** El ID del stream para el control frame WINDOW_UPDATE.
- **Delta-Window-Size:** El número adicional de bytes que el remitente puede transmitir.

7. Seguridad

- **El uso de las limitaciones del mismo origen.** Esta especificación utiliza la política de mismo origen en todos los casos donde se requiere verificación de contenido.
- **Cabeceras de HTTP y SPDY.** A nivel de aplicación , HTTP usa pares de nombre/valor en sus cabeceras.
  Debido a que PDY fusiona las cabeceras HTTP existentes con las cabeceras SPDY, hay un posibilidad de que algunas aplicaciones HTTP ya usen un nombre particular para la cabecera.
  Para evitar cualquier conflicto, todos los encabezados introducidas por HTTP sobre SPDY están prefijadas con ":":":". Esto no es una secuencia válida para el nombramiento del encabezado HTTP.

- **Ataques a través de protocolo.** Mediante la utilización de TLS, SPDY no introduce ningún nuevo protocolo de ataques. TLS cifra los contenidos de todas las transmisiones.
- **Subida de encabezados implícitos.** Subir recursos no tiene ninguna petición asociada. Para poder validar el control de caché existentes todos los recursos almacenados en caché debe tener un conjunto de solicitud de cabeceras. Por esta razón, los navegadores deben tener cuidado con no heredar encabezados de solicitud de la corriente asociada para la subida.

8. Conclusión

Como hemos visto SPDY es un protocolo desarrollado objetivamente para conseguir unos objetivos necesarios:

- Reducir un el tiempo de carga de una página. LLegado a conseguir reducirla en alrededor de un 50 %
- Minimizar la complejidad actual. SPDY utiliza TCP como capa de transporte lo que hace que no requiera cambios en la infraestructura ya existente en la red.
- Con su estructura consigue que no sea necesario cambiar nada en las web ya creadas.
- Utiliza el codigo abierto consiguiendo la ayuda de la comunidad y especialistas en la industria.

Algunas de las ventajas técnicas que nos ofrece SPDY son:

- Permitir que muchas peticiones HTTP que ocurren de manera concurrente pasen a través de una sola sesión TCP.
- Reduce el ancho de banda utilizado actualmente por HTTP mediante la compresión de cabeceras y la eliminación de las que no son necesarias.
- Es un protocolo muy fácil de implementar y eficiente a nivel de servidor. HTTP tenía el inconveniente de ser demasiado complejo.
- Hace el protocolo de transporte SSL mucho más seguro y compatible a las infraestructuras existentes en internet. Además, SSL introduce la penalización de la latencia y es necesario para asegurar que la comunicación a través de los proxies no se ha roto.
- Permite que el servidor inicie comunicaciones con el cliente y envíe los datos a este cuando sea posible.

Referencias

[1] https://docs.google.com/a/chromium.org/viewer?a=v&pid=sites&srcid=Y2hyb21pdW0ub3JnfGRldnxneDoxMzcyOWI1N2I4YzI3NzE2

[2] https://developers.google.com/speed/protocols/

[3] https://www.chromium.org/spdy/spdy-whitepaper

[4] https://gist.github.com/anonymous/2011946

[5] https://wiki.mozilla.org/Platform/Features/SPDY

[6] https://es.wikipedia.org/wiki/Transmission_Control_Protocol

[7] http://dev.chromium.org/spdy/spdy-protocol