Práctica 8: Análisis del protocolo TCP

1. Trabajo previo

Para el correcto aprovechamiento de la práctica es conveniente que antes de acudir al laboratorio hayas estudiado el funcionamiento del protocolo TCP. Para ello, además de repasar los conceptos vistos en las clases de teoría y problemas, también debes estudiar el **capítulo 3 de Kurose**.

Por otra parte, como en esta práctica se va a usar de forma intensiva el analizador de protocolos Wireshark, debes repasar el contenido de la práctica 1.

2. Objetivos de la práctica

Al acabar esta práctica serás capaz de interpretar en una captura de tráfico los mecanismos fundamentales involucrados en el funcionamiento del protocolo TCP, como el establecimiento y cierre de la conexión y el uso de las opciones TCP. Además, utilizarás nuevas herramientas del analizador de protocolos Wireshark que te permitirán crear gráficos para mostrar el intercambio de segmentos TCP.

3. Introducción

A pesar de que en esta práctica nos dedicaremos a analizar los segmentos TCP del nivel de transporte, ver los mensajes del protocolo de aplicación puede ayudarnos a seguir el diálogo cliente/servidor. Para ello, asegúrate al utilizar el programa Wireshark que en la opción **Analyze → Enabled Protocols**, el protocolo HTTP está habilitado, tal y como se muestra en la Figura 1.

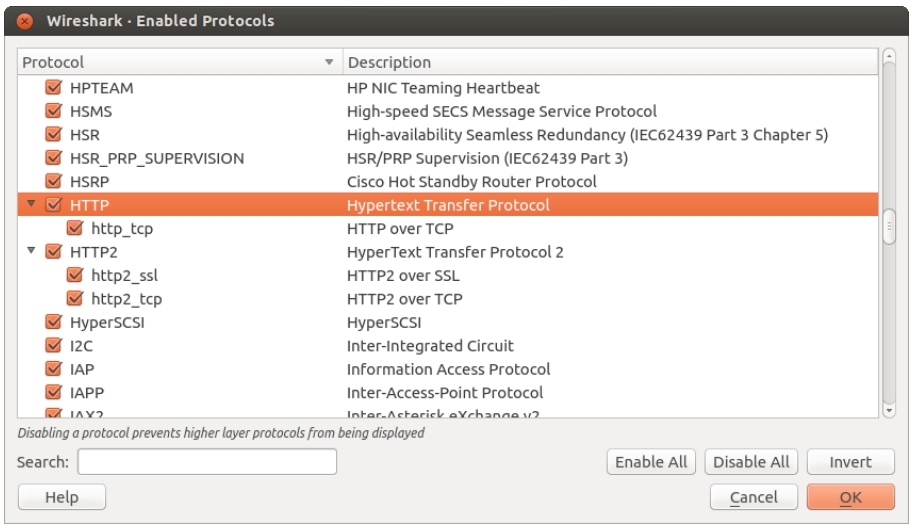


*Figura 1. Opción **Analyze → Enabled Protocols**. Marca el protocolo HTTP.*

4. Establecimiento y cierre de la conexión TCP

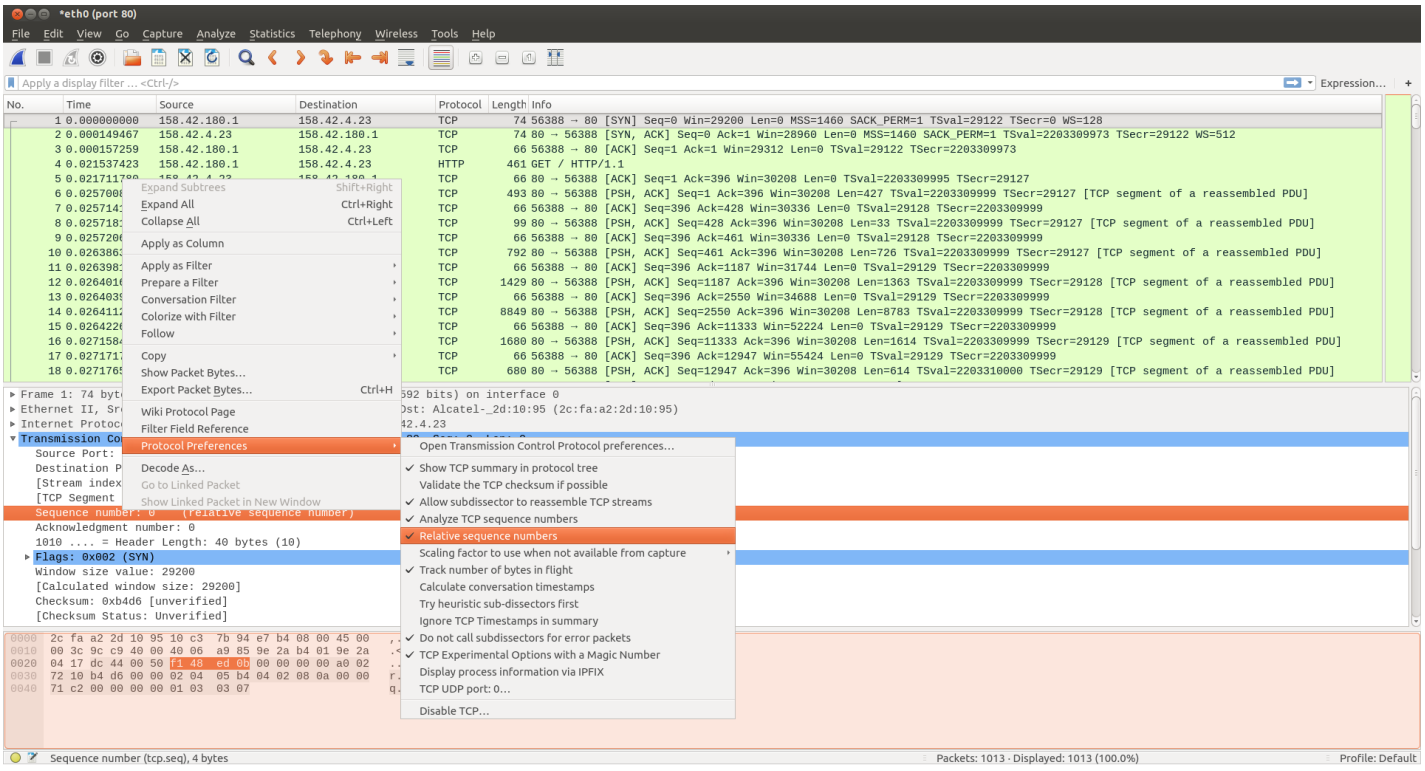
En esta sección vamos a analizar tráfico TCP con el objetivo de revisar el establecimiento de la conexión y las opciones incluidas en los segmentos de SYN. También vamos a ver cómo cada extremo de la conexión usa el flag FIN para indicar que desea cerrar la conexión.

Ejercicio 1:

Realiza una captura, mediante el analizador Wireshark, de una conexión a la página web de la Universidad (http://www.upv.es). Puede ser útil establecer un filtro de captura ("port 80 and host www.upv.es") desde el menú **Capture → Options** o un filtro de visualización ("tcp.port == 80 and ip.addr==158.42.4.23") desde el filtro de la ventana principal, para facilitar la interpretación de los datos.

a) Analiza el establecimiento de la conexión e identifica el protocolo en tres fases. ¿Qué MSS se elige para realizar la transferencia? ¿En qué segmentos aparece la opción MSS? ¿Qué otras opciones establecen cada uno de los extremos? ¿Siguen apareciendo una vez establecida la conexión?

b) Determina los números de secuencia iniciales de cada extremo. Diferencia entre el número de secuencia relativo, que muestra el analizador para realizar un seguimiento más cómodo, del real que aparece en el segmento (pulsa sobre el número de secuencia relativo y verás el número real que figura en el segmento transferido en la ventana inferior de la ventana principal -contenido en hexadecimal). De igual forma, seleccionando un segmento TCP, con el botón derecho del ratón (Protocol preferences) podemos indicarle que no queremos usar números relativos (ver Figura 2), aunque es más cómodo seguir usando en el resto de la práctica los números relativos.



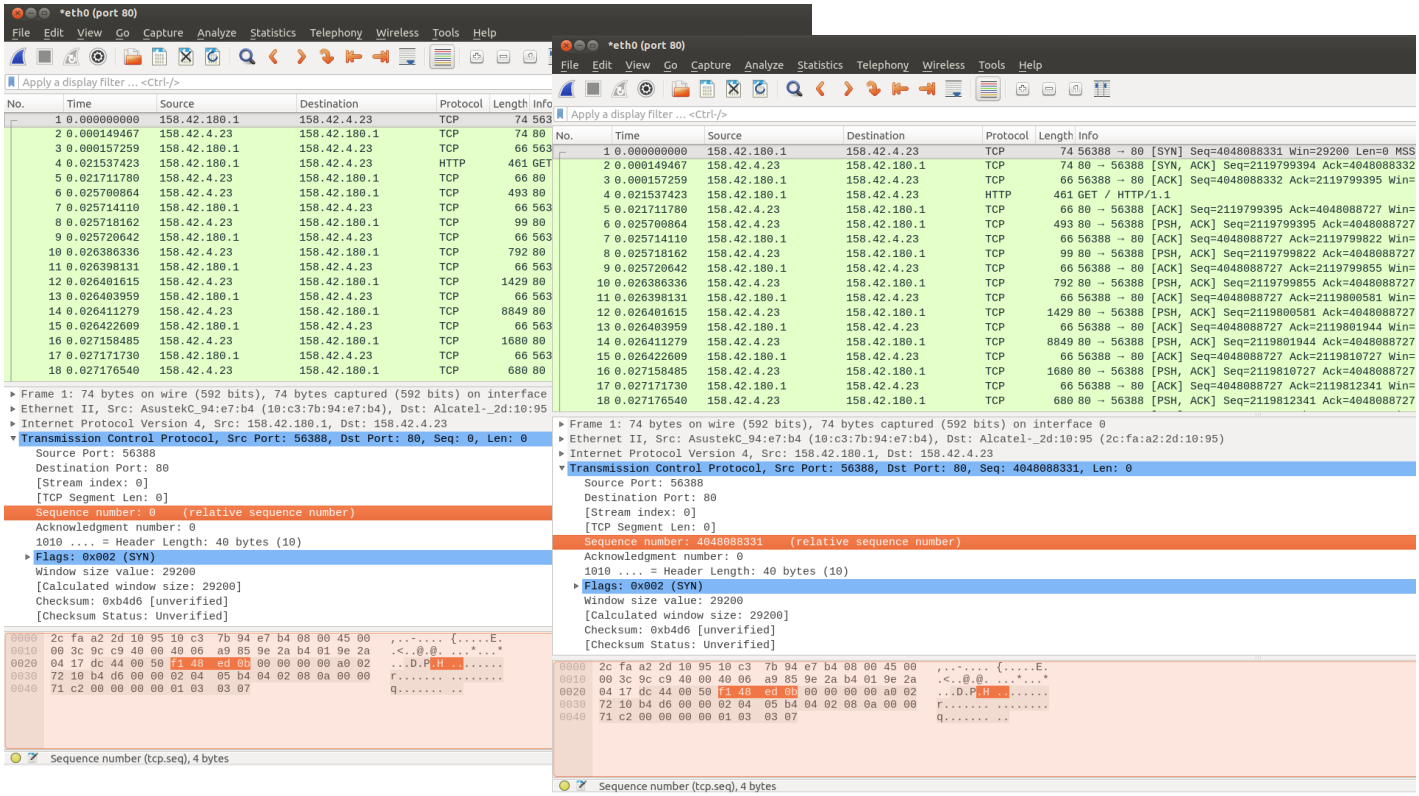


Figura 2. Números de secuencia absolutos y relativos.

Ejercicio 1 (continuación):

c) En este apartado vamos a analizar el cierre de la conexión. Utilizaremos el primer flujo TCP (el flujo con el que se solicita el fichero HTML). Para ver únicamente este flujo y ocultar el resto selecciona el segmento que transporta la petición "GET / HTTP/1.1" en la lista de paquetes capturados y a continuación aplica el filtro "**Follow TCP Stream**" del menú "**Analize**". Wireshark mostrará una nueva ventana mostrando sólo ese flujo. Cierra la ventana y desplázate al final de la pantalla principal, que es donde verás el final de la conexión. ¿Quién toma la iniciativa? ¿Es un cierre en tres o cuatro fases? ¿Cómo se modifican los números de secuencia y los de reconocimiento durante el cierre de la conexión?

d) Seguimos analizando la misma conexión del apartado anterior pero vamos a centrarnos ahora en la opción MSS (Maximum Segment Size). Recuerda que, en realidad, el MSS indica el tamaño máximo del campo de datos del segmento TCP, no el tamaño total del segmento. Compara el MSS indicado al inicio de la conexión con el tamaño máximo del campo de datos utilizado durante la misma. Realiza esta comparación para diversos segmentos. La longitud de los datos contenidos en los segmentos recibidos ¿coincide con el MSS? ¿Es mayor o menor que el MSS? ¿Puede ser mayor?

Ejercicio 1 (continuación):

Como habrás visto, para numerosos segmentos mostrados por Wireshark en la captura que has realizado, el tamaño de los datos contenidos en el segmento es claramente mayor que el MSS. Esto contradice, a priori, el significado de la opción MSS. ¿Qué está ocurriendo? En redes de alta velocidad, muchas veces las tarjetas de red tienen capacidad para procesar por ellas mismas los segmentos TCP, descargando de esta manera a la CPU del ordenador. Es lo que sucede en los ordenadores involucrados en la captura. Con esto se consigue una importante mejora de prestaciones global. Entre otras acciones, estas tarjetas de red son capaces de agregar varios segmentos consecutivos de forma que entregan al sistema operativo un segmento de mayor tamaño. Es decir, aunque por la red viajan segmentos con un campo de datos menor que el MSS, estas tarjetas de red concatenan los datos de varios de estos segmentos para entregarlos juntos al sistema operativo como si fuera un único segmento. Por este motivo, Wireshark observa segmentos de tamaño mayor que el MSS establecido al inicio de la conexión. Como la tarjeta de red genera una interrupción al procesador cada vez que debe pasarle un segmento, uno de los beneficios de esta concatenación de segmentos es que reduce el número de interrupciones necesarias. Este menor número de interrupciones mejora las prestaciones globales del sistema dado que el procesador es interrumpido menos veces.

5. Frecuencia de envío de reconocimientos y tamaño de la ventana de congestión

En esta sección vamos a ver cómo la frecuencia de envío de reconocimientos puede ayudar a crecer más rápido a la ventana de congestión.

Ejercicio 2:

a) Usando el mismo flujo TCP del ejercicio 1 c), ejecuta la opción del menú **Statistics → Flow Graph**. A continuación selecciona el flujo TCP de la ventana principal pinchando en la opción "**Limit to display filter**" que aparece en la nueva ventana (Figura 3). Selecciona también en la nueva ventana la opción "**Flow type: TCP flows**". ¿Qué es lo que aparece?

b) A continuación analiza la evolución de los reconocimientos (ACK). Para ello fíjate en los segmentos que envía el servidor web, su longitud y número de secuencia y observa cuándo el cliente (tu ordenador) envía los ACK y hasta qué byte están reconociendo. Esto lo puedes ver tanto en la ventana de **Statistics → Flow Graph** que has abierto en el apartado previo como en la principal con el flujo seleccionado. Ten en cuenta que cuando marcas un segmento en la ventana de **Statistics → Flow Graph**, se selecciona también en la ventana principal. (Nota: para saber cuál es el segmento del servidor que está reconociendo el cliente, analiza el número de secuencia del transmisor y súmale la longitud enviada; busca entonces el ACK que lleve ese número de reconocimiento cuando envía el cliente).

c) ¿Hay un ACK cada dos segmentos TCP? ¿Por qué? Intenta encontrar una explicación (Razona sobre las posibles ventajas de evitar el uso de ACK retardados durante los primeros RTT, cuando está en ejecución el algoritmo de arranque lento).

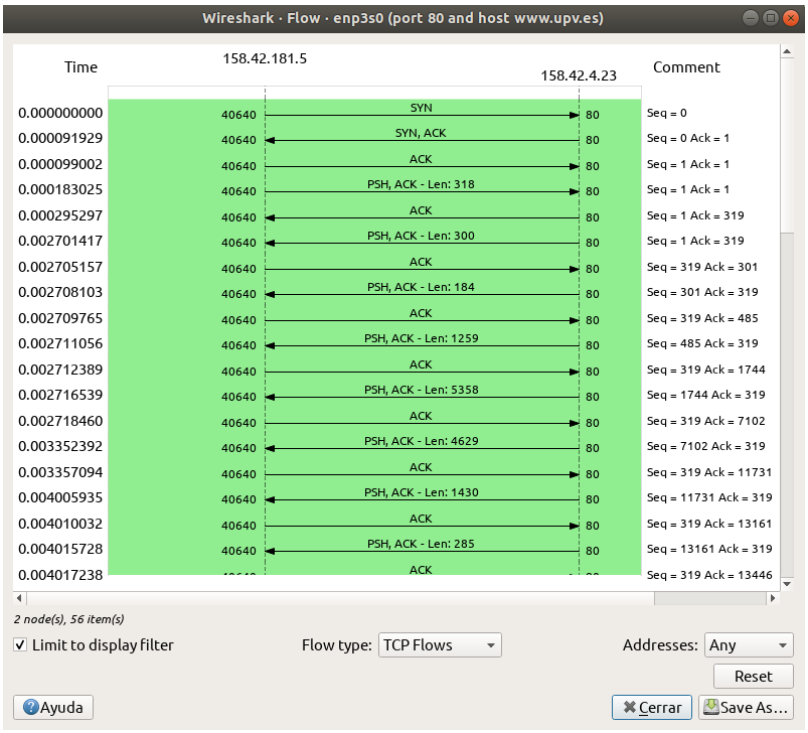


*Figura 3. Opción Wireshark **Statistics → Flow Graph**.*

*Durante el inicio de una conexión, algunos sistemas tienen un modo denominado **QuickACK**. En este caso no se entra en el envío diferido o retardado de ACK hasta que se han transmitido varios segmentos (por ejemplo 16 en algunas implementaciones). ¿Qué se consigue con esto? El efecto es enviar un ACK por cada segmento recibido y conseguir que la ventana de congestión crezca más rápido que si el receptor envía un ACK cada dos segmentos.*

*Esto último también es matizable: hay implementaciones que incorporan la técnica denominada **TCP Congestion Control with Appropriate Byte Counting** (RFC 3465). En los sistemas que la implementan no se incrementa la ventana de congestión de forma constante con la recepción de cada ACK, sino que se incrementa tanto como la información reconocida por los segmentos de ACK. Así, si solo llega un ACK, pero reconoce 2, 3 ó 4 segmentos, la ventana se incrementa de forma acorde con la información reconocida. De esta forma no es necesario enviar en el inicio un ACK por segmento para incrementar más rápido la ventana de congestión. Estas dos técnicas no son excluyentes y podemos encontrar implementaciones que utilicen las dos (por ejemplo, piensa que un sistema puede usar QuickACK para que el otro extremo –que no sabe si usa Appropriate Byte Counting o no- incremente en cualquier caso la ventana de congestión más rápido).*

Por último, también es interesante saber que en las implementaciones actuales el número de segmentos que se envían al iniciarse una conexión no tiene que ser siempre igual a 2 (la norma RFC 5681 aporta algunas recomendaciones). Hay estudios que determinan, además, que sería mejor incrementar el número de segmentos que se transmiten inicialmente y muchos sistemas permiten enviar hasta 10 segmentos en el arranque de una conexión.

6. Flag RST

En esta sección, analizaremos el uso del flag RST.

El efecto del RST es cerrar inmediatamente la conexión. Esto es ligeramente diferente de un FIN, que solo dice que el extremo que lo envía ya no va a transmitir ningún nuevo dato, aunque aún puede recibir algunos. Hay dos tipos de eventos que hacen que se transmita un RST:

a) Que la conexión se cancele explícitamente de forma abrupta. Por ejemplo, se mata el proceso que tiene abierto el socket.
b) Que TCP reciba ciertos tipos de segmentos no válidos. Por ejemplo, un segmento SYN destinado a un puerto que no está abierto en el sistema o un segmento destinado a una conexión que no existe.

Ejercicio 3:

a) Realiza una captura mediante el analizador Wireshark de una conexión al puerto 81 del servidor web de la Universidad de Valencia (ojo, no de la UPV). Para ello en el navegador usa la URL http://www.uv.es:81. **Ten cuidado si tienes algún filtro activo y establécelo al puerto 81**. Analiza la captura realizada. ¿Cómo se indica que no hay un servicio en ese puerto? ¿Cómo se cierra la conexión? ¿Qué le contesta tu navegador? ¿Vuelve a intentar tu navegador la conexión? En caso afirmativo ¿cuántas veces?

b) Repite lo mismo pero esta vez intentando hacer la conexión con el servidor de nuestra universidad: http://www.upv.es:81. ¿Pasa lo mismo que en el caso anterior? ¿Qué motivos puedes encontrar para esos comportamientos?

Como se puede comprobar, ambos servidores se comportan de manera diferente. En el primer caso, como el puerto está cerrado (no hay un servidor escuchando en el puerto 81), TCP identifica la solicitud de conexión del cliente como un segmento no válido y, por lo tanto, responde con un segmento RST. Sin embargo, en el segundo caso se está utilizando un cortafuegos (*firewall*) que impide el paso de los segmentos SYN destinados a puertos cerrados en el sistema. Estos segmentos se descartarán sin avisar a TCP. Este filtrado realizado por el cortafuegos dificulta el trabajo de los atacantes al proporcionar menos información que una respuesta con un segmento RST.

7. Reconocimientos selectivos (SACK)

En el establecimiento de la conexión del ejercicio 1 hemos visto que se hace uso de reconocimientos selectivos. La opción SACK permite al receptor informar al emisor de que ha recibido un bloque de datos cuyo número de secuencia está dentro de la ventana de recepción, pero no coincide con el límite inferior de esa ventana. Es decir, ha recibido un

segmento con un número de secuencia posterior al que debería haber llegado. Si la situación se repite y se pierden otra vez segmentos, el receptor puede acabar almacenando bloques de datos no contiguos en su buffer. Por cada bloque de datos recibido fuera de orden el receptor enviará un nuevo segmento con la opción SACK, que incluirá los bloques de datos posteriores al esperado recibidos más recientemente. De esta forma el emisor puede retransmitir los segmentos que se han perdido sin esperar a que venza su temporizador de retransmisión. Cuando lleguen los datos que rellenan el hueco, el receptor reconocerá los datos normalmente y avanzará el borde izquierdo de su ventana de recepción. El campo **Número de reconocimiento** de la cabecera TCP reconocerá todo el bloque de datos completo de forma global.

En la figura siguiente puedes ver un ejemplo donde se pierden dos segmentos de datos, los que llevan número de secuencia 4.000 y 7.000. El receptor utilizará la opción SACK para avisar de que ha recibido otros bloques posteriores.

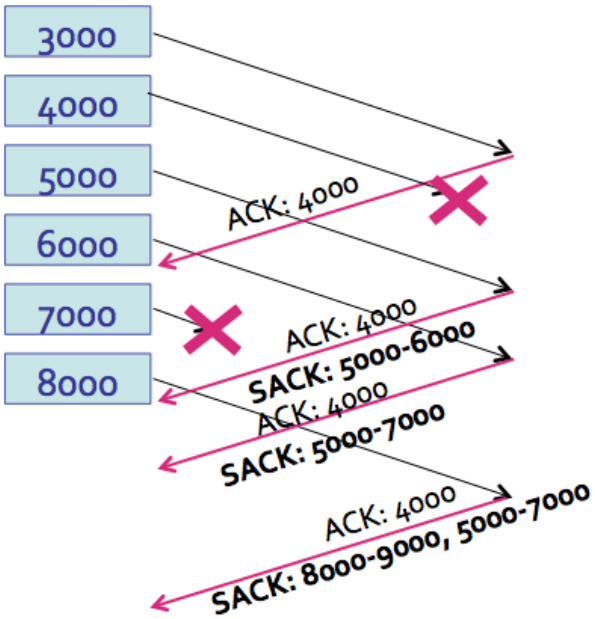


Figura 4. Ejemplo de uso de la opción SACK del protocolo TCP.

Es importante tener en cuenta que la opción SACK no modifica el significado del campo **Número de reconocimiento**, cuyo valor aún especificará el borde izquierdo de la ventana de recepción.

Ejercicio 4:

Descarga el fichero **`CapturaPráctica150.pcapng`** que está disponible en Poliformat y cárgalo en Wireshark.

a) En primer lugar vamos a fijarnos en un detalle de la captura que no está relacionado con los reconocimientos selectivos. Al igual que has hecho en el ejercicio 2b), usa la opción **Statistics → Flow Graph** y selecciona en la parte inferior de la nueva ventana la opción "**Flow type: TCP flows**". Vamos a centrarnos en los primeros 40 segmentos de la conexión y comprobar si se emplean ACKs retrasados. Analiza la proporción de segmentos enviados por el cliente y los reconocimientos recibidos desde el servidor. Es decir, ¿cada cuántos segmentos enviados por el cliente se recibe un reconocimiento desde el servidor? A la vista de los resultados, ¿se están empleando reconocimientos retrasados? Ahora fíjate en los números de reconocimiento enviados por el servidor. A la vista de los números de reconocimiento enviados por el servidor al cliente, ¿cuántos segmentos enviados por el cliente son reconocidos por cada ACK enviado por el servidor? ¿Por qué en la captura los ACK aparecen mucho después de los envíos que reconocen?

b) Cierra la ventana que se ha abierto con **Statistics → Flow Graph** y analiza en las tramas de la captura la evolución del tamaño de la **ventana de recepción del servidor**. ¿Cómo evoluciona el tamaño de dicha ventana? ¿Qué piensas que está ocurriendo?

c) En este apartado vamos a ver en funcionamiento la opción SACK. Para ello analiza el segmento número 88 de la ventana principal. ¿Cuál es el valor del ACK? Ahora ve al segmento 91. Independientemente de que Wireshark lo cataloga como Window Update, analiza el resto de la información de ese segmento. ¿Cuál es el valor del ACK? Ve al campo SACK. ¿Qué está indicando el receptor? Mira el campo SACK en los siguientes segmentos enviados desde el servidor al cliente (segmentos 93, 95, 97...), ¿qué está pasando?