Capítulo 14 Capa de transporte

Glosario de términos

Capa de transporte:

Cuando distintas aplicaciones, en un dispositivo origen, quiere enviar distintos mensajes a otras aplicaciones en uno o varios dispositivo de destino(el receptor también puede ser otra aplicación en el mismo dispositivo emisor) pasa los datos a la capa de transporte de origen. La capa de transporte se encarga de gestionar el envío y recepción de la transmisión de datos entre aplicaciones de origen y destino. Lo mismo sucede en el otro extremo.

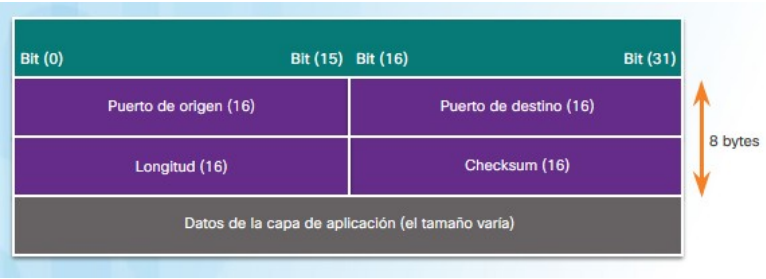
Fundamentalmente en esta capa nos vamos encontrar con dos protocolos(UDP y TCP) con objetivos distintos (no son los únicos):

UDP(Protocolo de Datagrama de Usuario). Se conoce como un protocolo de entrega de **máximo esfuerzo** por **no** ser **confiable.** Es decir, no se preocupa de sí los segmentos llegaron a su destino (El protocolo UDP de destino no acusa recibo de los segmentos recibidos. Por lo tanto, el protocolo UDP de origen **no reenvía los segmentos perdidos** dado que sin acuses de recibo no tiene conocimiento de cuales se perdieron**).** Tampoco se **reordenan** los segmento en destino.

La ventaja que le da no ser confiable es que es un protocolo de **baja sobrecarga**:

No sobrecarga las redes al tener una cabecera con los mínimos campos posibles, no enviar acuses de recibo y no reenviar segmentos perdidos.

Cabecera UDP:



No sobrecarga los procesos. Al no tener que hacer ninguna de las funciones anteriores los dispositivos están más descansados.

Con UDP no se establece sesión. Ver más abajo.

TCP(Protocolo de Control de Transmisión)

Confiable:

Acuses de recibo

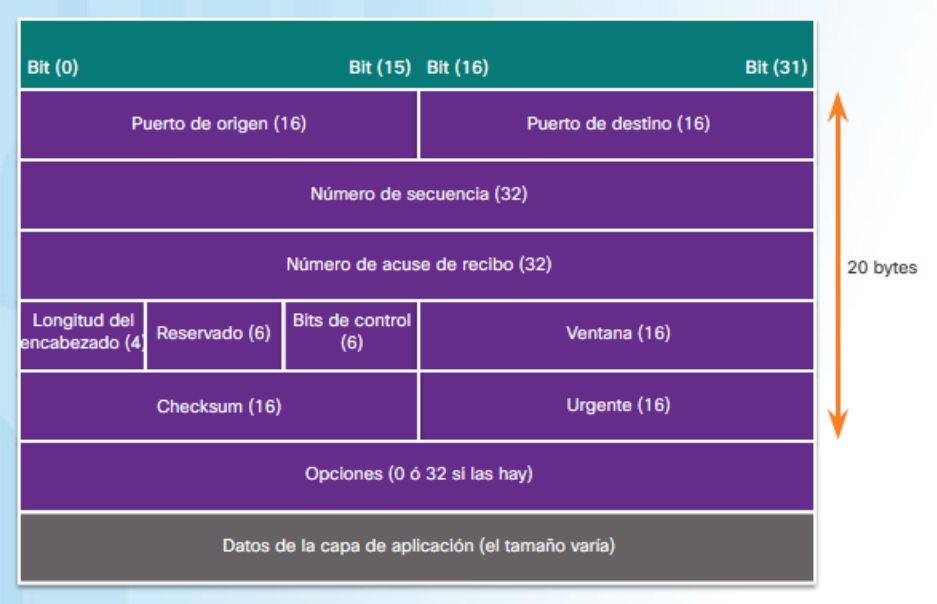
Reenvío de segmento perdidos

Reordenación de segmentos en el destino antes de pasárselos a la aplicación

Desventaja: Gran **sobrecarga**(De procesos y de tráfico. Mayor número de campos para poder controlar la confiabilidad)

Establecimiento de sesión. Con TCP el equipo que quiere enviar y recibir datos a otro establece un vinculo(sesión, enlace o apretón de manos) con el dispositivo del otro lado de la comunicación. Ver más abajo.

Encabezado TCP



Funciones de la capa de transporte:

- **Segmentación de datos**(Los datos de la aplicación se dividen en segmentos más pequeños. Lo hace tanto UDP como TCP) y **rearmado de segmentos**(TCP los reordena en el orden original antes de pasarlos a la aplicación. UDP los rearma en el orden que llegan y se los pasa a la aplicación).
- **Seguimiento de conversaciones individuales**(tanto TCP como UDP). Controla cada una de las conversaciones que tiene el ordenador de origen/destino separando cada conversación de forma eficiente e independiente:

  Cada conversación en un dispositivo está completamente identificada dentro de este, utilizando el par formado por la **IP** y el **número de puerto**. Esto se le llama **Socket**. Ejemplo: Socket 1(172.16.0.50:80), Socket 2(172.16.0.50:10500), Socket 3 [fe80::7dc0:f8d2:da3:69b0]:10001/, …

  El puerto identifica la aplicación de cada conversación.

  Una conversación entre origen y destino queda identificada de forma global(universal) por un **doble Socket** formado por la IP y el puerto de origen y la IP y el puerto de destino. Ejemplo: 197.16.100.100:10500-172.217.17.3:443

Fijate que son IPs públicas. En todo internet no hay ninguna conversación que repita todo el conjunto(las dos IPs y los dos puertos en otra conversación no pueden ser iguales. Al menos uno de los cuatro valores será distinto.)

- **Identificación de las aplicaciones**. Para ello utiliza el número de puerto indicado anteriormente.

Cuando se utiliza UDP: Utilizar UDP puede ser muy ventajoso para distintas aplicaciones.

- Cuando el **mensaje** es tan **corto** que se puede enviar con un solo segmento. Es el caso de los mensajes que intercambian servidor y cliente en el protocolo **DHCP.** También ocurre lo mismo en una consulta **DNS**. Si se pierde algún mensaje es la aplicación quien se debe preocupar en volver a solicitar la información.
- Cuando los mensajes dependen del **factor tiempo** y la perdida de un segmento no significa una perdida importante de información. Es ventajoso, sobre todo, cuando se transmitir un volumen de datos importante, la comunicación transcurre más fluida sin sobrecargas(Ej: **Videoconferencia Online**). En **VoIP** el volumen de información que se transmite no es tan grande como en el caso anterior, pero en esta caso la fluidez y la baja sobrecarga es muy beneficiosa. En cualquiera de los casos aunque se pierda algún segmento no hay perdidas de información importantes.

Cuando se utiliza TCP: Cuando es más importante asegurar la confiabilidad de la transmisión que el tiempo que conlleva transmitirla. Descarga a las aplicaciones de tener que controlar la confiabilidad.

Transacciones con bases de datos. Imagine una operación bancaria: El mensaje es pequeño pero, es más importante que el importe quede reflejado correctamente en la cuenta del cliente, y poco más tiempo puede llevar(confiabilidad).

HTTP/HTTPS Imagine que quiere ver las noticias en su navegador. Es mas importante que la pagina web se cargue correctamente que el poco tiempo de más que hay que esperar para presentar la información.

Correo electrónico(SMTP, IMAP, POP). Lo importante es que dispongamos de todos lo correos y con su información completa independientemente del tiempo en obtenerlos.

Puerto: Tanto UDP como TCP toman nota de la aplicación que le pasó cada mensaje asociando cada aplicación con un número llamado **número de puerto de origen**. La capa de transporte también pone en su cabecera el **número de puerto de destino**(corresponde a la aplicación de destino). Al igual que las IPs, los puertos de destino y origen se intercambian cuando el destino responde al origen.

Segmento: Tanto UDP como TCP dividen los mensajes(dado que suelen ser grandes) en trozos más pequeños, en partes manejables, para poder enviarlos a través de los medios (los envía de 1 en 1). Estos trozos reciben el nombre de **segmentos**. Los segmento UDP se les llama **Datagramas**.

Multiplexación: Consiste en intercalar(en el tiempo) por el medio físico(cable, fibra, ...) segmentos de distintas aplicaciones/usuarios/dispositivos. De este modo nadie acapara el

medio y todos perciben que el ancho de banda esté dedicado a él. Además la segmentación(pequeños trozos) y la multiplexación(segmentos de distintas conversaciones intercalados en el tiempo) permiten retransmitir solo aquellos trocitos que se hayan perdido. Imagine que alguien está viendo una película: Sí no se utilizara la segmentación y la multiplexación los demás usuarios tendrían que espera para transmitir sus datos y, si la transmisión de la película falla habría que volver a transmitirla toda.

Número de secuencia: TCP numera cada segmento con un número llamado **Número de secuencia** para poder reconstruir el mensaje original en el destino(cada segmento en su posición/orden correcto). El número de secuencia de cada segmento es la posición que le corresponde al primer byte de cada segmento empezando en 1(también se le llama **desplazamiento del segmento** por que indica la distancia desde el origen del mensaje). Ejemplo:

Tamaño	Número de secuencia
1º segmento 168 bytes	1 La posición del primer byte del primer segmento es siempre 1
2º segmento 200 bytes	Nº de secuencia 169 (1+168) Nº de secuencia anterior + tamaño del segmento anterior
Tercer segmento de tamaño 102 bytes	Nº de secuencia 369 (169+200) Nº de secuencia anterior + tamaño del segmento anterior

1º segmento	2º segmento	3º segmento
1	169	369

Por seguridad el número del primer byte del primer segmento no es 1 sino un número aleatorio conocido como **número de secuencia inicial(ISN)** (Arriba lo simplificamos para poder explicarlo mejor). Supongamos que en el ejemplo anterior el número aleatorio(ISN) generado es 3000. Entonce quedaría así:

Tamaño	Número de secuencia
1º segmento 168 bytes	3001 La posición del primer byte del primer segmento es siempre 0
2º segmento 200 bytes	Nº de secuencia 3169 (3001+168) Nº de secuencia anterior + tamaño del segmento anterior
Tercer segmento de tamaño 102 bytes	Nº de secuencia 3369 (3169+200) Nº de secuencia anterior + tamaño del segmento anterior

Cuando visualizamos los mensajes con una herramienta de captura de paquetes como Wireshark lo vamos a ver como en la primera tabla dado que Wireshark resta de forma automática el número aleatorio(ISN) antes de mostrar la información.

ACK(o ACKnowledgement o Acuse de recibo): Reconoce a la capa de transporte de origen los segmentos recibidos, para ello utiliza un número que se llama **ACK**. El número ACK es un valor en expectativa. Basándonos en la primera tabla, sí la capa de transporte de destino envía a la capa de transporte de origen un número ACK 370 le indica que el siguiente byte que espera recibir es el 370 y reconoce de este modo que ha recibido los 369 bytes anteriores. Se dice que es un valor en expectativa porque el destino indica al origen el número de byte a partir del cual espera recibir mas datos.
Si el emisor no recibe el reconocimiento de los segmentos en un tiempo determinado entiende que el destino no los recibió y **vuelve a transmitirlos automáticamente**.
Normalmente se reconocen un bloque de segmentos contiguos a la vez. Sí algún segmento del medio faltara no se puede reconocer el bloque entero.
En la actualidad, los hosts pueden emplear también una característica optativa llamada reconocimiento selectivo (**SACK**). Si ambos hosts admiten SACK, es posible que el destino reconozca los bytes de segmentos discontinuos, y el host solo necesitará volver a transmitir los datos perdidos(no todo el bloque).

Sesión: El protocolo **TCP** de origen dialoga con el protocolo TCP de destino para establecer lo que se conoce como una **sesión**. La sesión es un vínculo(un apretón de manos) entre origen y destino para acordar que, a partir del momento en que se establezca la sesión y hasta que ellos mismos finalicen la sesión, se pueden mandar mensajes entre sí en cualquier momento. Ambos van aceptar y responder esos mensajes. Esto se hace antes de poder transmitir ningún segmento de datos.

Flag(marcador, bandera o bits de control): La cabecera TCP tiene un campo de **6 bits** para indicar al receptor cual es la función del mensaje que se envía. Cada bit toma un valor **0** sí **no está activo** y un valor **1** sí está **activo**(Pueden estar activos más de un bit). El significado de los bit es el siguiente:

SYN (Sincronización). Sí está activo(valor 1) indica al destino que el origen quiere establecer una sesión.

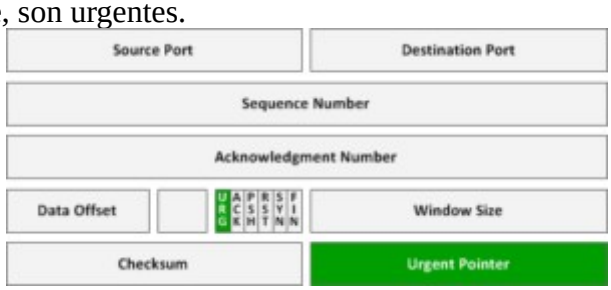
ACK (Acuse de recibo). Indica al destino que el campo Número de acuse de recibo (ACKnowledgement Number) contiene un valor en expectativa para indicar los segmentos recibidos.

PSH(PUSH) (Indica al protocolo TCP de destino que envié los datos inmediatamente a la aplicación). El protocolo TCP de origen espera a que el buffer que pone a disposición de la aplicación esté suficientemente lleno para mandar segmentos con tamaño máximo y hacer la transmisión más eficiente. Esto es así siempre que la aplicación no indique a la capa de transporte de origen que los envíe de forma inmediata. En este último caso el TCP de origen activa el flag **PUSH** de la cabecera del segmento para que el protocolo TCP de destino envíe de inmediato los datos del segmento a la aplicación de destino.
Imagine una comunicación telnet: cada comando no llena el segmento pero es necesario que se envíe inmediatamente para ser ejecutado y recibir la respuesta del mismo.

RESET (Restablecimiento de sesión). Sí se producen errores o pasa mucho tiempo sin transmitir información.

URG (URGENTE - No se usa, Indica si la comunicación es urgente o no). Este flag es utilizado para informar al extremo receptor de que ciertos datos dentro de un segmento son urgentes y deberían ser priorizados. Si la bandera está activada, la estación receptora evalúa el puntero, un campo de 16 bits en la cabecera TCP(Campo

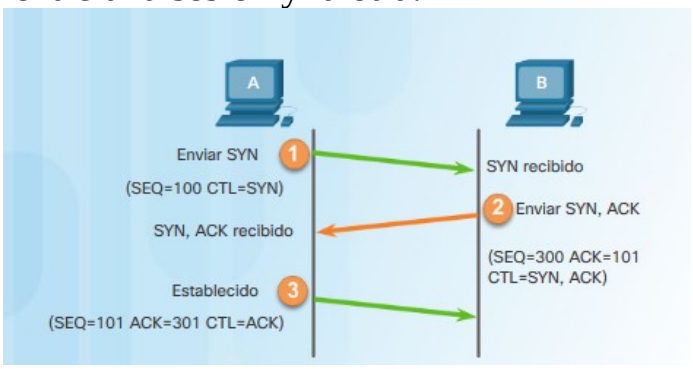
Urgente). Este puntero indica qué cantidad de datos del segmento contando desde el primer byte, son urgentes.



La bandera URG no se emplea mucho en los protocolos modernos.
FIN (Finalizar la sesión). Cuando el emisor de un segmento activa el marcador de FIN está indicando al receptor que no tiene más que transmitir y que desea terminar la sesión. En este caso el receptor enviará como respuesta un segmento con el bit(flag, marcador, …) ACK activado para confirmar el final de la sesión.

Establecimiento de sesión de tres vías TCP

Tenga en cuenta que no se establece una sola sesión. Realmente se establecen dos sesiones en tres pasos. Una sesión entre A y B, otra segunda sesión entre B y A. Además los números de secuencia no tienen nada que ver entre una sesión y la otra.



Cuando dos dispositivos se comunican utilizando el protocolo TPC, antes de transmitir datos entre si, inician una sesión. Para ello:
Punto 1: A inicia el establecimiento de sesión. A envía a B un mensaje cuyo segmento lleva activo el marcador SYN y el campo número de secuencia lleva el valor 100(ISN-Número de secuencia inicial generado por A).
Punto 2: B responde a A con un mensaje cuyo segmento lleva activo el marcador ACK y SYN.

- B acusa recibo a A. El ACK va activo para indicarle a A que B recibió el segmento de inicio de sesión(SYN) y que el próximo byte que espera recibir B por parte de A es el que figura en el valor del campo ACKnowledgement Number 101(realmente es el byte 1[101-100]).
- Por otra parte, el bit de SYN está activo porque B indica que quiere iniciar una sesión con A. Por lo tanto el número de secuencia lleva el valor 300 generado por B(es el NSI). No tiene nada que ver con el 100 que A pasó a B).

Punto 3: A responde a B con un acuse de recibo. El mensaje lleva activo el flag ACK para indicarle que está de acuerdo con el inicio de sesión de B con A. El flag ACK también indica que el campo ACKnowledgement Number lleva el valor 301, que es el primer byte que A espera recibir de B en el próximo envío de datos de B hacia A. Además el número de

secuencia del segmento es 101 porque A aún no envió ningún dato de aplicación a B. Y volverá a ser 101 cuando le envié los primeros datos.

Grupos de números de puerto

La Autoridad de Números Asignados de Internet (IANA) es el organismo normativo responsable de asignar los diferentes estándares de direccionamiento, incluidos los números de puerto.

Rango de números de puerto	Grupo de puertos
Entre 0 y 1023	Puertos bien conocidos
de 1024 a 49151	Puertos registrados
de 49152 a 65535	Puertos privados y/o dinámicos

1. **Bien conocidos.** Son lo que se conocen como puertos por defectos. Están reservados por la IANA para poner a determinados servicios en los servidores. De este modo los clientes utilizan estos puertos como puertos de destino por defecto para acceder a esos servicios:

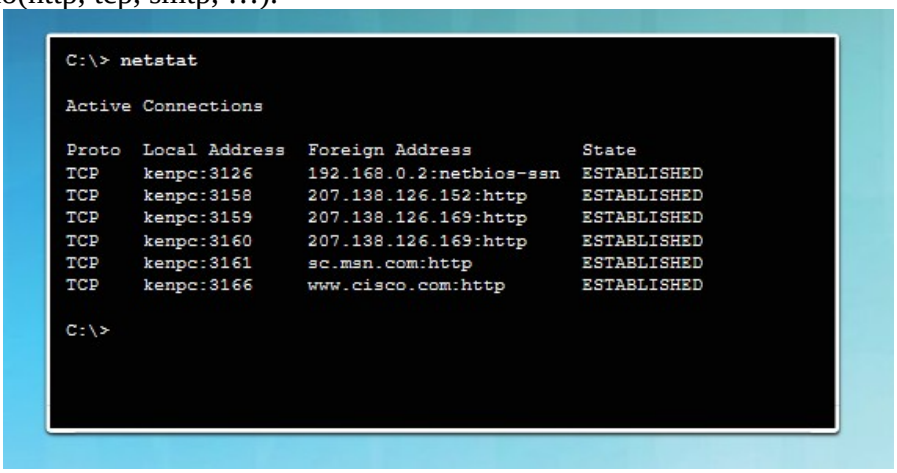
Número de puerto	Protocolo	Aplicación	Acrónimo
20	TCP	Protocolo de transferencia de archivos (datos)	FTP
21	TCP	Protocolo de transferencia de archivos (control)	FTP
22	TCP	Secure Shell	SSH
23	TCP	Telnet	–
25	TCP	Protocolo simple de transferencia de correo	SMTP
53	UDP, TCP	Servicio de nombres de dominios	DNS
67	UDP	Protocolo de configuración dinámica de host (servidor)	DHCP
68	UDP	Protocolo de configuración dinámica de host (cliente)	DHCP
69	UDP	Protocolo trivial de transferencia de archivos	TFTP
80	TCP	Protocolo de transferencia de hipertexto	HTTP
110	TCP	Protocolo de oficina de correos versión 3	POP3
143	TCP	Protocolo de acceso a mensajes de Internet	IMAP
161	UDP	Protocolo simple de administración de redes	SNMP
443	TCP	Protocolo seguro de transferencia de hipertexto	HTTPS

   Por ejemplo, cuando usamos un navegador(Chrome, Firefox, …) para obtener una página Web de un servidor web, podemos especificar el puerto de destino en el cual escucha la aplicación de servidor: “www.google.es:80” . El puerto de destino 80 no se suele especificar cuando accedemos a un servicio web porque el navegador utilizará por si solo el puerto por defecto establecido por la IANA(el 80).

Sin embargo usar los puertos bien conocidos como puertos de escucha de un servicio no es obligatorio. Imaginemos que nosotros configuramos un servidor web para que escuche las peticiones en el puerto 10000. En este caso, cada vez que un cliente intente acceder a una página web, debe especificar como puerto de destino el 10000(www.dominio.es:10000) sino el navegador intentara conectarse al puerto por defecto(80). El servicio fallará, aunque la petición llegue al servidor(la IP es correcta) no hay ningún servicio escuchando en el puerto 80.

2. **Puertos registrados** (números del 1024 al 49151): IANA asigna estos números de puerto a una entidad que los solicite para utilizar con procesos o aplicaciones específicos. Principalmente, estos procesos son aplicaciones individuales que el usuario elige instalar en lugar de aplicaciones comunes que recibiría un número de puerto bien conocido. Por ejemplo, Cisco ha registrado el puerto 1985 para su proceso Hot Standby Routing Protocol (HSRP). **MySQL(1306)**
3. **Puertos dinámicos o privados** (números 49152 a 65535): también conocidos como puertos efímeros, usualmente el SO del cliente los asigna de forma dinámica cuando se inicia una conexión a un servicio. El puerto dinámico se utiliza para identificar la aplicación cliente durante la comunicación.
   También se pueden utilizar para este fin los puertos registrados que no estén siendo utilizados en un dispositivo.

netstat: es un comando que contemplan distintos sistemas operativos, que lista los **protocolos**(de aplicación: http, ftp, smtp, ...) que se están usando, las **direcciones IP(locales y remotas),** los números de **puerto** (locales y remotos) y el **estado** de la conexión.
Puede ser interesante para controlar que algo o alguien está conectado al host local.
Sí lo utiliza sin el parámetro "-n" trata de resolver las IPs numéricas en urls y en vez de poner el número de puerto pone el nombre de la aplicación asociada al puerto bien conocido(http, tcp, smtp, …).



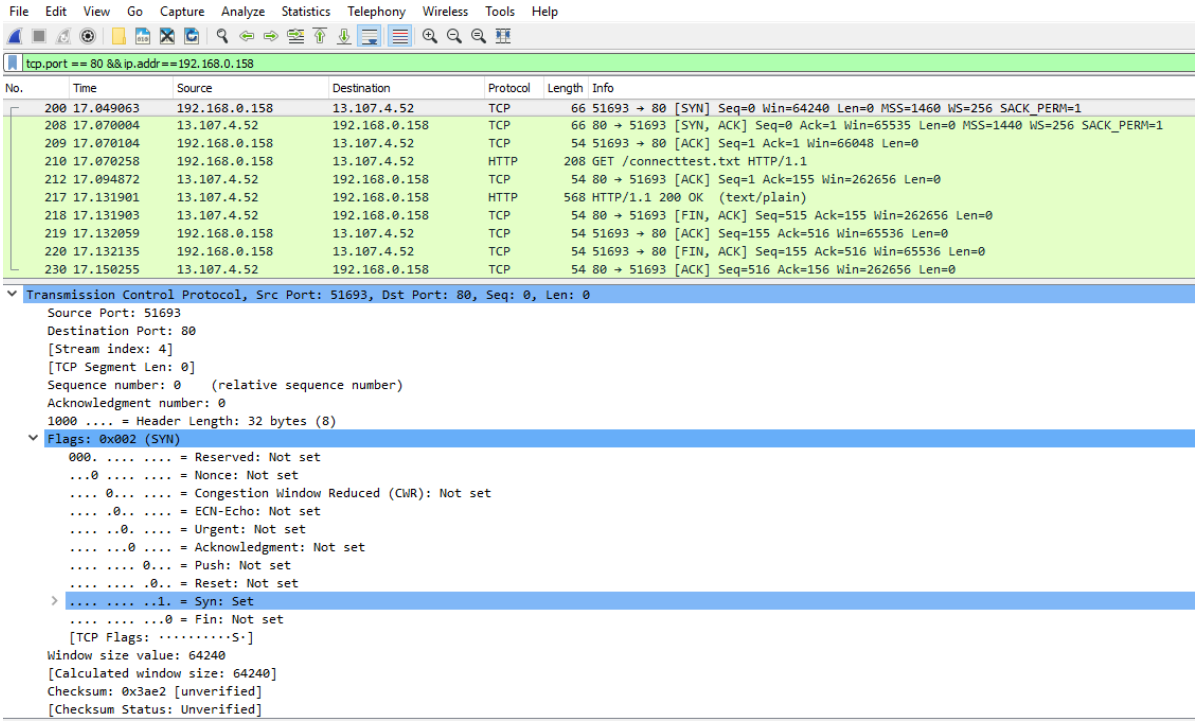
"netstat -n" tarda menos en crear la tabla al no tener que resolver los nombres.

Finalización de sesión TCP:

Para cerrar cada una de las sesiones de “una vía” TCP(Recuerde que hay dos sesiones iniciadas, una en cada sentido) se utiliza un enlace de dos vías(Es decir, un mensaje en cada sentido) por cada una de las vías(sesiones), que consta de un segmento FIN y un segmento de reconocimiento (ACK).
Los segmentos 1(activa el marcador FIN para indicar que A quiere finalizar la sesión con B) **y 2**(Activa el marcador ACK para indicar que B esta de acuerdo con finalizar la sesión) **finalizan la sesión de A con B.**
Los segmentos 3 y 4 Finalizan la sesión de B con A.

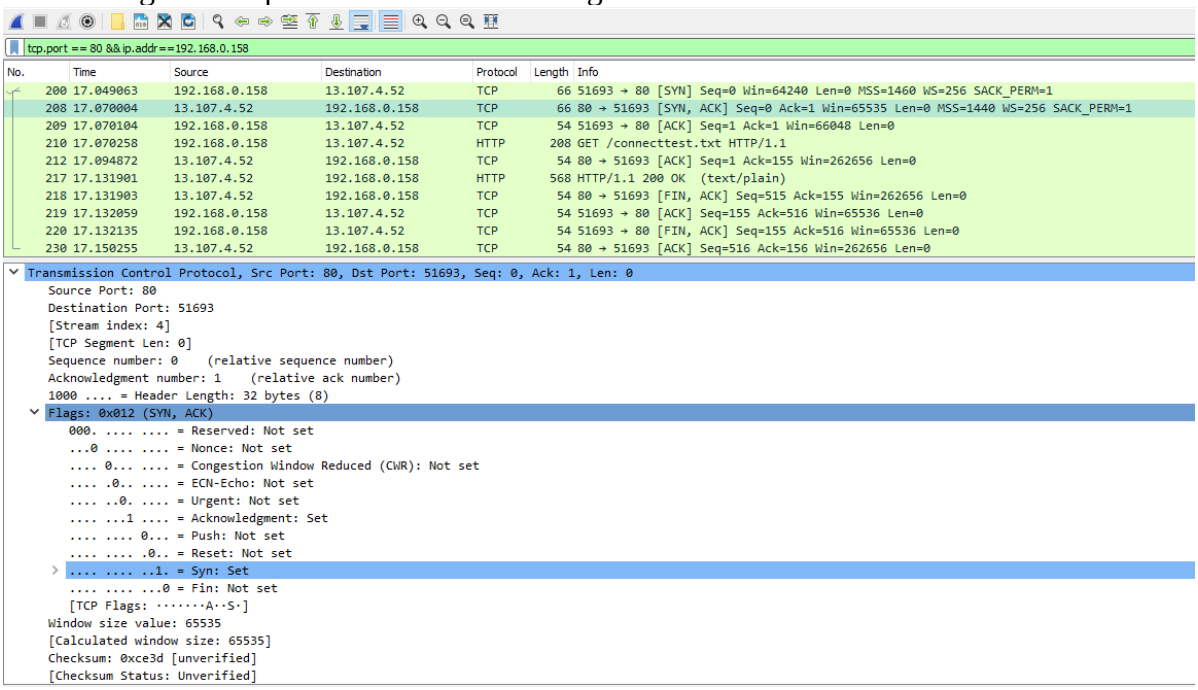
Seguimiento de una conversación real con el Wireshark

El cliente web(se utiliza un navegador) indica al servidor web que desea establecer una sesión con él. El cliente, para eso, activa el flag SYN(Syn: set), pone 0 en el número de secuencia inicial(Secuence number: 0), no envia ningún byte de datos([TCP Segment Len: 0]) y el valor de acuse de recibo es 0 (Acknowledgment number:0, Tampoco tiene importancia porque flag ACK no está activo –Acknowledgment:No set –)

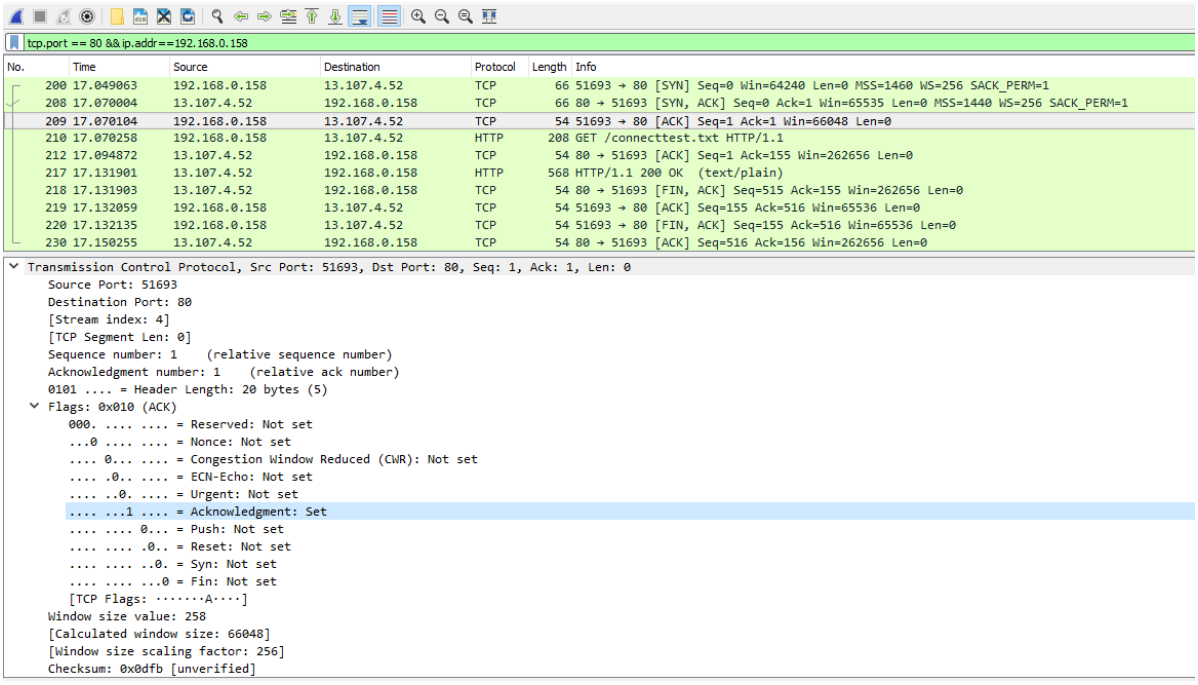


El servidor contesta al cliente que está de acuerdo(Acknowledgment:Set), que el próximo byte que espera obtener del cliente es el 1(Acknowledgment number:1), que el servidor

también quiere establecer otra sesión con el cliente(Syn: set) y que el número de secuencia inicial de servidor a cliente es 0(Secuence number: 0). Recuerde, Wireshark resta el valor inicial generado para hacer más fácil el seguimiento. Por eso es 0.



El cliente responde que está de acuerdo con la sesión que el servidor quiere mantener con él(Acknowledgment:Set) y que el próximo byte que espera obtener del servidor es el 1(Acknowledgment number:1).



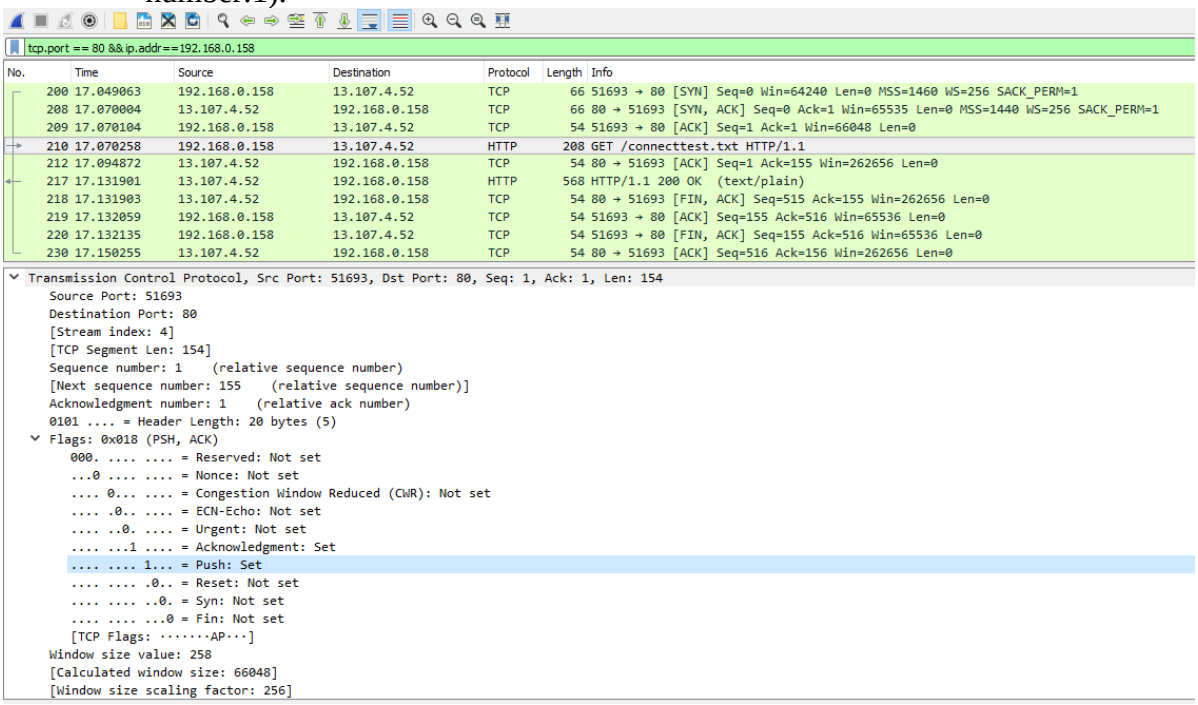
El cliente pide ahora la página web al servidor:

Para ello envía 154 bytes de datos([TCP Segment Len: 154]). En ella le indica, entre otros aspectos, cual es la pagina que quiere.

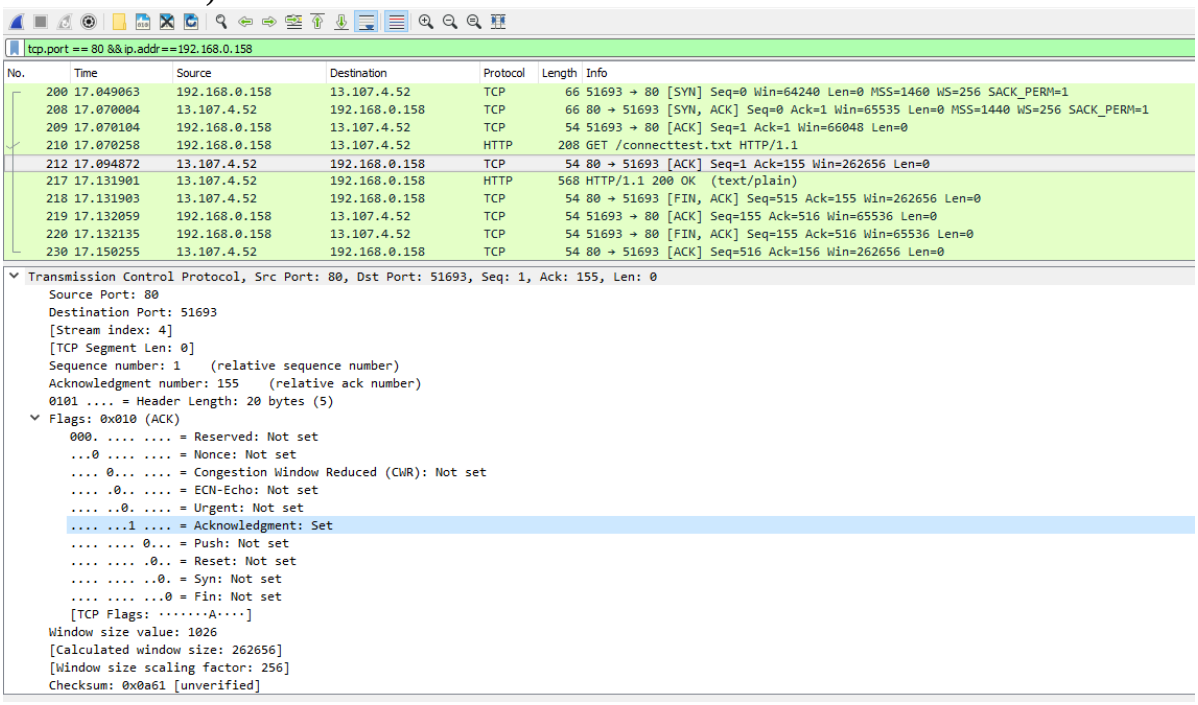
El primer byte enviado es el 1(Secuence number: 1)

Le indica al protocolo TCP que no espere por más segmentos para enviar la petición a la aplicación del servidor(Push: Set)

Sigue esperando del servidor el byte 1(Acknowledgment:Set y Acknowledgment number:1).



El servidor acusa recibo al cliente de su solicitud(Acknowledgment:Set y Acknowledgment number:155).



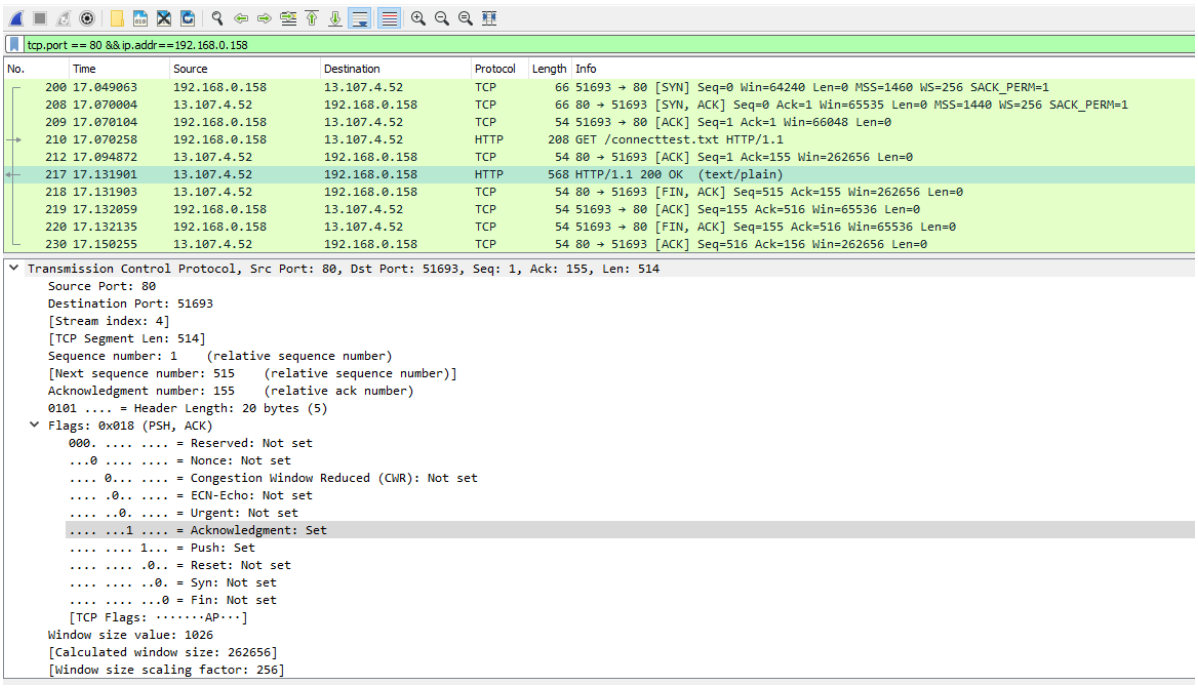
El servidor envía la pagina solicitada por el cliente:

Para ello envía 515 bytes de datos([TCP Segment Len: 515]).

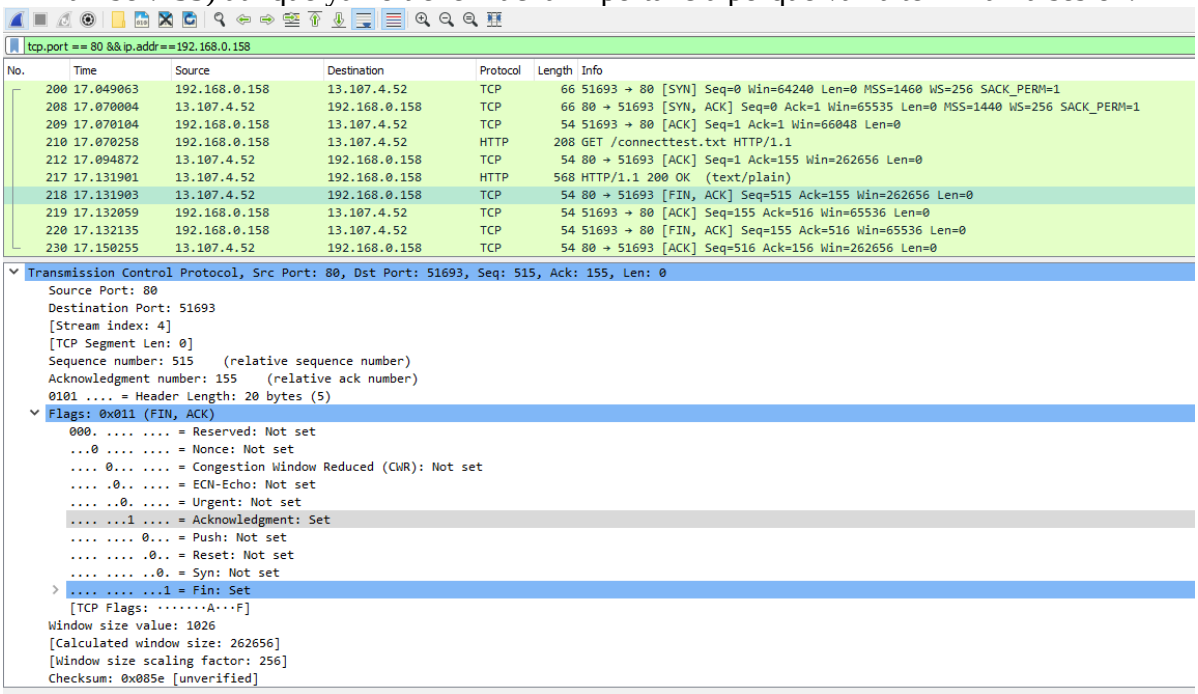
El primer byte enviado es el 1(Secuence number: 1)

Le indica al protocolo TCP que no espere por más segmentos para enviar la petición al navegador del cliente(Push: Set)

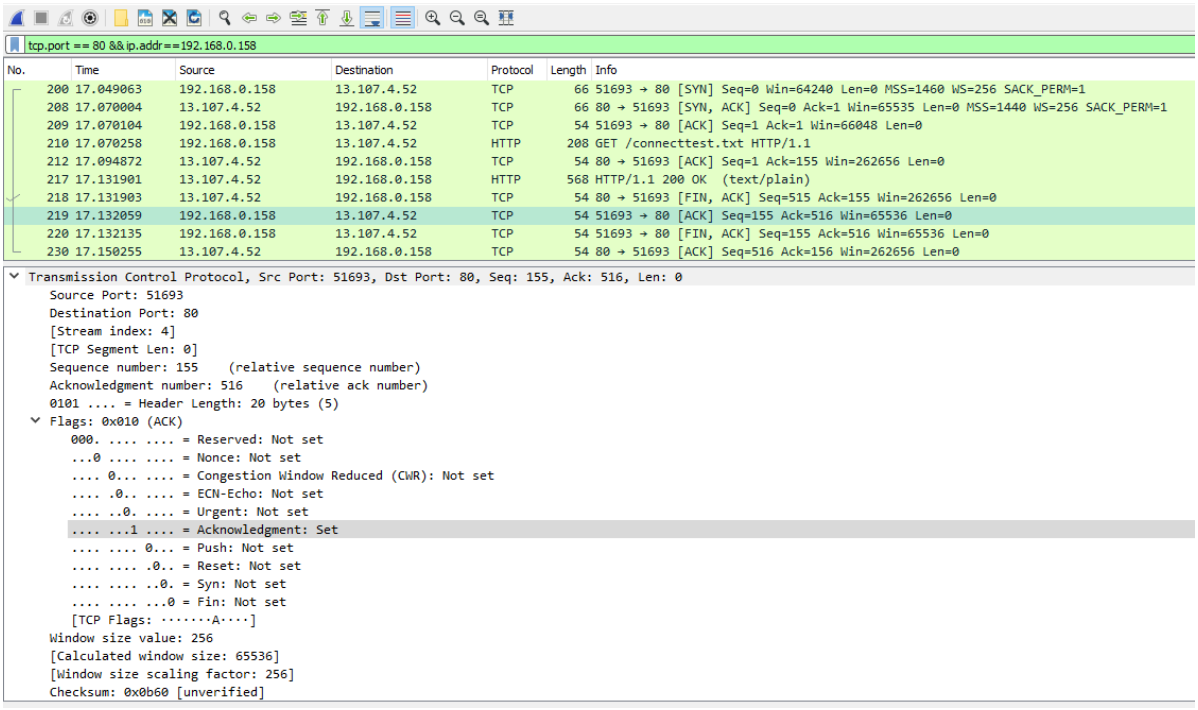
Sigue esperando del cliente el próximo byte: 155(Acknowledgment:Set y Acknowledgment number:155).



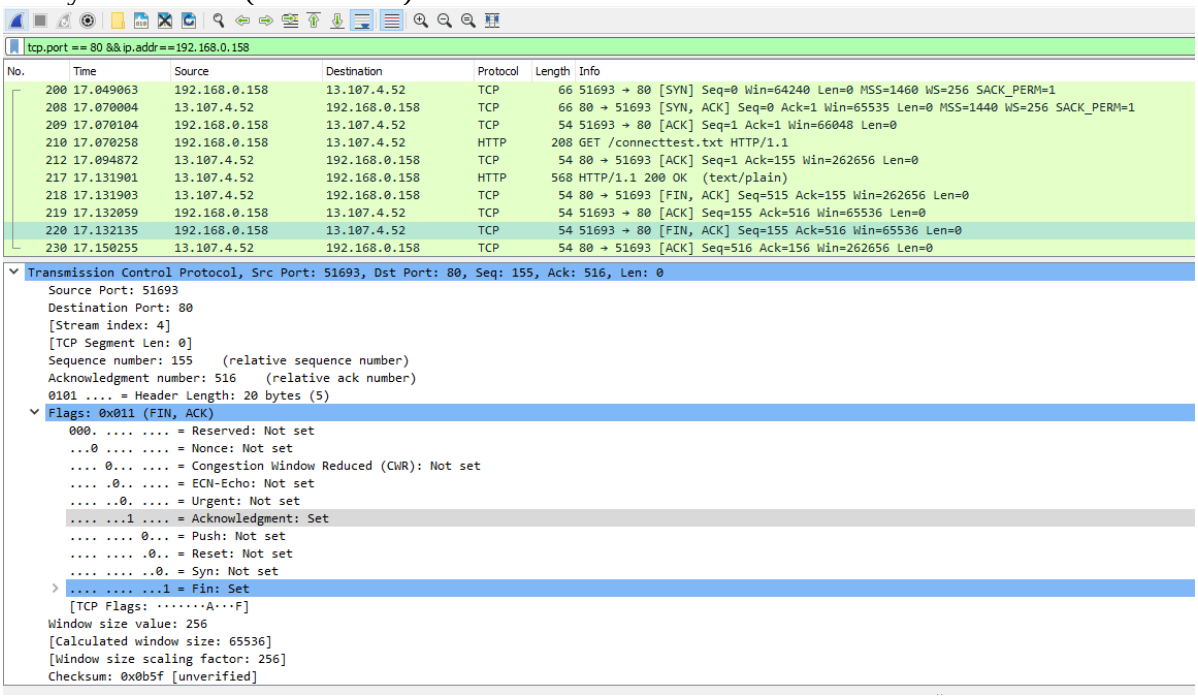
El servidor indica al cliente que no tiene más que transmitir y por tanto desea terminar la sesión(Fin: Set). También sigue acusando recibo(Acknowledgment:Set y Acknowledgment number:155) áunque ya no tiene mucha importancia porque van a terminar la sesión.



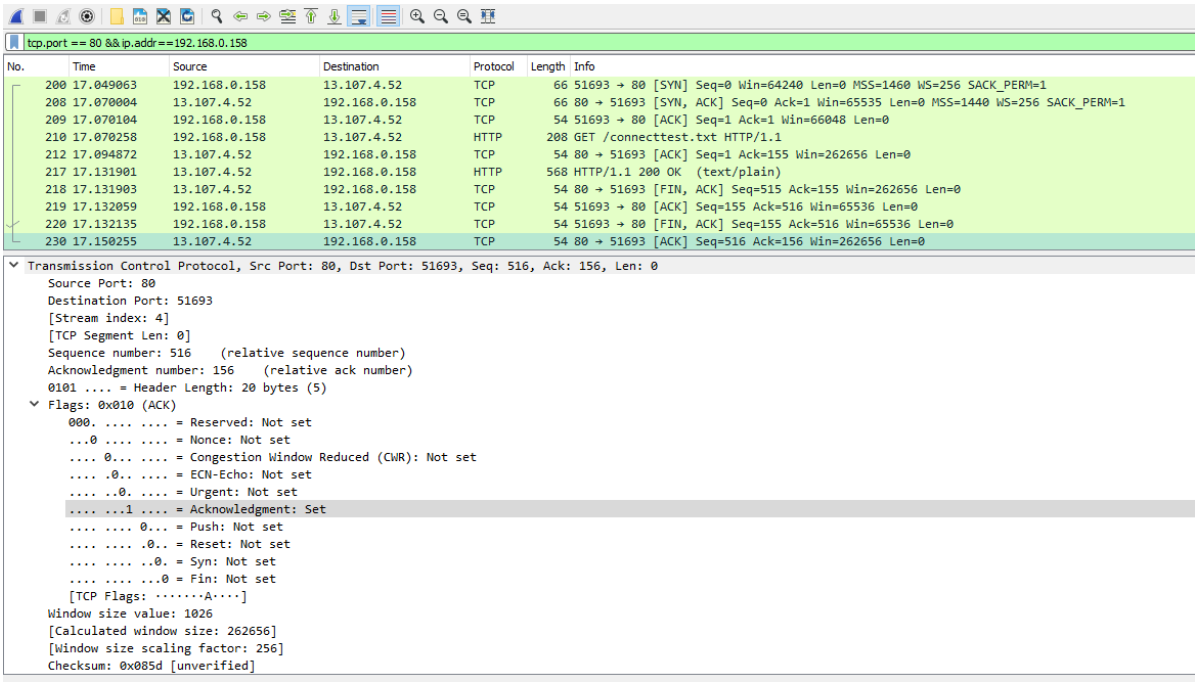
El cliente acusa recibo al servidor del FIN de sesión y del segmento recibido (Acknowledgment:Set y Acknowledgment number:516)



El cliente solicita finalizar su sesión con el servidor(Fin: Set) y sigue acusando recibo de los bytes recibidos(515+1=516)



El servidor acusa recibo del FIN de sesión al cliente(Acknowledgment:Set)



Control de flujo(Tamaño de la ventana)

Mire las imágenes anteriores.

Existe un tamaño de ventana(Window size value: xxxx)

Con este valor, el protocolo TCP del receptor de los datos, indica al emisor cuantos bytes le puede transmitir seguidos, este último al primero, sin que el primero(el receptor) le acuse recibo de los segmentos que le lleguen.

Con esto se controla la velocidad, dado que quien transmite los datos no puede transmitir más bytes hasta que le llegue un acuses de recibo de la otra parte. En todo caso pasado un tiempo puede reenviar los segmentos perdidos.

Prevención de congestiones: Cuando se produce congestión en una red, el router sobrecargado comienza a descartar paquetes. Cuando los paquetes que contienen los segmentos TCP no llegan a su destino, se quedan sin reconocimiento. Mediante la determinación de la tasa a la que se envían pero no se reconocen los segmentos TCP, el origen puede asumir un cierto nivel de congestión de la red.

Siempre que haya congestión, se producirá la retransmisión de los segmentos TCP perdidos del origen. Si la retransmisión no se controla adecuadamente, la retransmisión adicional de los segmentos TCP puede empeorar aún más la congestión. No sólo se introducen en la red los nuevos paquetes con segmentos TCP, sino que el efecto de retroalimentación de los segmentos TCP retransmitidos que se perdieron también se sumará a la congestión. Para evitar y controlar la congestión, TCP emplea varios mecanismos, temporizadores y algoritmos de manejo de la congestión.

Si el origen determina que los segmentos TCP no están siendo reconocidos o que sí son reconocidos pero no de una manera oportuna, entonces puede reducir el número de bytes que envía antes de

recibir un reconocimiento. Tenga en cuenta que es el origen el que está reduciendo el número de bytes sin reconocimiento que envía y no el tamaño de ventana determinado por el destino.

El tamaño de la ventana se establece en los dos sentidos de la comunicación. Un tamaño para cada sentido.

Esto es lo que hace que cuando descarga un archivo vea fluctuar la velocidad de transmisión. El que envía los datos está adaptando la velocidad de transmisión al tamaño de la ventana de receptor. La velocidad también puede variar por que el emisor no transmite los datos de forma uniforme por distintos motivos(red saturada, exceso de conexiones en el dispositivo,...).

El receptor de los datos puede variar el tamaño de la venta al emisor continuamente(esto se conoce como **ventanas deslizantes**) .

La ventaja de las ventanas deslizantes es que permiten que el emisor transmita continuamente segmentos mientras el receptor está acusando recibo de varios de los segmentos anteriores.

Si el receptor no da atendido la cantidad de datos que le envía el emisor(ya sea por su hardware lento, por que está saturado,…) reduce el tamaño de la ventana y viceversa.

Fijate en las imágenes como el tamaño de la venta del cliente y servidor no se corresponden (cada uno indica al otro la suya). Además no siempre es la misma porque cada uno la varía a su conveniencia.

El receptor puede acusar recibo antes de recibir la cantidad total de bytes que indica su ventana con lo cual se reinicia la cuenta del total de datos transmitidos por el receptor sin un acuse de recibo.

MMS(Tamaño máximo que pueden tener los segmentos): Transmitir 1460 bytes de datos dentro de cada segmento TCP es el tamaño de segmento máximo.

Reensamblaje de datagramas de UDP

Los datagramas que llegan desordenados no se vuelven a ordenar.

Los datagramas perdidos no se reenvían.

Protocolo de transporte que utilizan las aplicaciones:

TCP: HTTP, FTP, SMTP, TELNET

UDP: DHCP, DNS, TFTP, VoIP, IPTV

Aplicaciones que pueden utilizar los dos protocolos:

Aunque **DNS** y **SNMP** utilizan UDP de manera predeterminada, ambos también pueden utilizar TCP. DNS utilizará TCP si la solicitud DNS o la respuesta DNS tiene más de 512 bytes, como cuando una respuesta DNS incluye un gran número de resoluciones de nombres. Del mismo modo, en

algunas situaciones, el administrador de redes puede querer configurar SNMP para utilizar TCP.

Aplicaciones que manejan la confiabilidad por su cuenta:

Comunicaciones unidireccionales que no requieran control de flujo, detección de errores, reconocimientos ni recuperación de errores o que puedan ser manejados por la aplicación. Por ejemplo, SNMP y TFTP.