UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Establecimiento de una conexión TCP

Apellidos, nombre	Baydal Cardona, María Elvira (mebaydal@disca.upv.es)
Departamento	Informática de Sistemas y Computadores
Centro	Universitat Politècnica de València

1 Resumen de las ideas clave

En este artículo vamos a explicar cómo el protocolo TCP (*Transport Control Protocol*) es capaz de establecer una conexión fiable apoyándose en el servicio proporcionado por una red de conmutación de paquetes que, en el caso peor, puede proporcionar a TCP un servicio sin garantías, capaz de perder paquetes, entregarlos desordenados o incluso hacer llegar paquetes duplicados al receptor.

Para ello describiremos brevemente porque pueden aparecer todos estos problemas y las herramientas elegidas en TCP para conseguir evitarlos. Esto nos permitirá justificar los diferentes aspectos de la estrategia que se utiliza en el establecimiento de una conexión TCP.

2 Objetivos

Una vez que leas con detenimiento este documento, serás capaz de explicar cómo se establece una conexión TCP, diferenciando los papeles realizados por el cliente y por el servidor. En particular, podrás:

- Describir qué bits de la cabecera TCP se activan en los segmentos TCP que participan en el establecimiento de la conexión.
- Justificar por qué se eligen números de secuencia iniciales aleatorios.
- Justificar por qué los primeros dos segmentos TCP no pueden incluir datos.
- Describir distintas situaciones en las que TCP es capaz de evitar problemas ante la recepción de duplicados de paquetes de establecimiento de conexión.

3 Introducción

TCP, uno de los 2 protocolos de transporte empleados en la arquitectura TCP/IP y uno de los principales de la arquitectura, proporciona a las aplicaciones un servicio orientado a la conexión. Por lo tanto, necesita definir los mecanismos necesarios para establecer y liberar las conexiones que emplea.

Además, el tipo de servicio que TCP proporciona a las aplicaciones es fiable y ordenado, para lo que utiliza mecanismos de control de error basados en reconocimientos y retransmisiones. Estas retransmisiones, aunque son necesarias, pueden generar duplicados de paquetes ya enviados y plantear problemas a la hora de establecer una conexión (un paquete duplicado es una copia de un paquete que ya se había recibido correctamente en el receptor).

La gestión de la conexión TCP incluye los mecanismos necesarios para establecer la conexión en primer lugar, y para liberarla una vez finalizada la transmisión de los datos. En este artículo nos centraremos en el establecimiento.

4 Desarrollo

La forma más simple de establecer una conexión entre dos extremos es, evidentemente, mediante un protocolo en dos fases (Figura 1). Un extremo solicita la conexión y el otro la acepta o la rechaza. Este esquema funciona bien en muchos casos. En las redes de computadores se utiliza para establecer conexiones entre extremos directamente conectados, como ocurre en el nivel de enlace de datos.

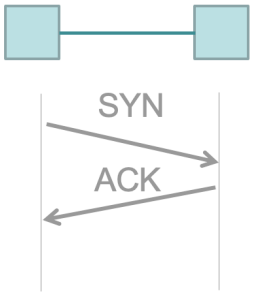

Figura 1. Establecimiento de conexión en dos fases

Sin embargo, en el caso del nivel de transporte, y en particular de TCP, los dos extremos se comunican a través de una red insegura que puede perder paquetes, por lo que pueden generarse retransmisiones y posibles duplicados de paquetes. Se asume que la red tampoco garantiza la entrega en orden.

En esta situación, un mismo segmento TCP puede recibirse varias veces, típicamente cuando se ha perdido su confirmación y el emisor lo ha reenviado. Por este motivo un protocolo de conexión en dos fases puede dar problemas. Los segmentos duplicados podrían provocar establecimientos de conexión erróneos, malgastando recursos de los servidores. Para evitar este problema se utiliza un esquema más elaborado, conocido como protocolo o acuerdo en tres fases.

En un protocolo en tres fases la conexión quedará establecida en el cliente al recibir la confirmación del servidor, como ya ocurría en el establecimiento en dos fases. Sin embargo, para que la conexión quede establecida en el servidor, éste deberá recibir una confirmación adicional del cliente (Figura 2). Es decir, se requiere un tercer segmento para el establecimiento de la conexión. Por lo tanto, ambos extremos deben solicitar y confirmar la solicitud del otro. Además, cada extremo elige un número que identifica de forma única cada intento de conexión. De esa forma se evita el riesgo de aceptar como válido algún segmento duplicado de una conexión anterior.

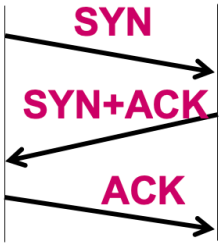

Figura 2. Ejemplo de establecimiento de una conexión en tres fases

4.1 Intercambio de segmentos

Veamos con más detalle el intercambio de segmentos y la gestión de los números de secuencia para establecer una conexión TCP. En la figura 3 se han indicado también las instrucciones en java que darían lugar al establecimiento de la conexión, tanto en el cliente como en el servidor.

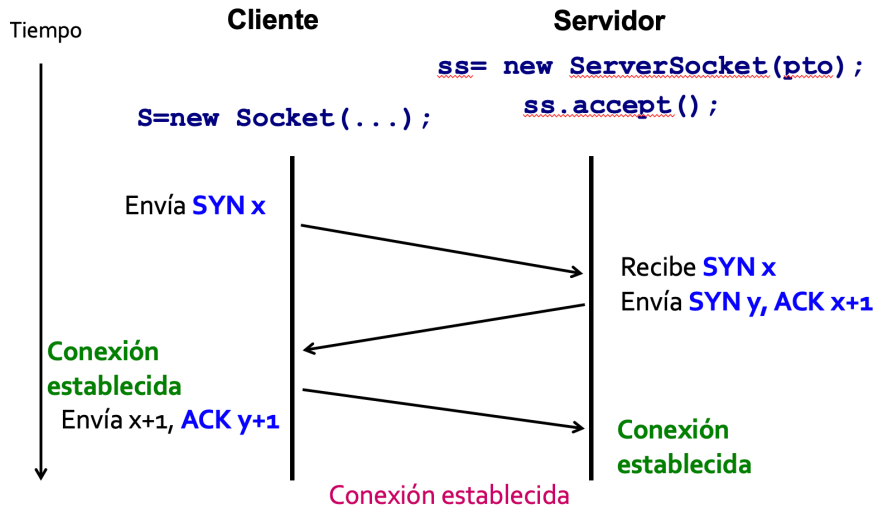

Figura 3. Establecimiento de la conexión TCP en tres fases

Como puede observarse el servidor debe estar escuchando previamente en un puerto esperando a recibir la petición del cliente. Esta escucha en java se realiza invocando el método *accept()* de la clase *ServerSocket*. Por su parte el cliente inicia el establecimiento de la conexión, que en java tiene lugar cuando se instancia un objeto de tipo *Socket* y se especifica un servidor destino.

A nivel de TCP tendrán lugar las siguientes acciones:

1. El cliente elige para cada intento de conexión un número de secuencia único, que hemos representado por **x**, y lo envía al servidor. Además, el segmento enviado debe llevar el bit SYN de la cabecera TCP activado para indicar que se trata de una solicitud de conexión.
2. Tras recibir el segmento SYN del cliente, el servidor elige su propio número de secuencia inicial que en este caso hemos llamado **y**, lo envía al cliente en un segmento que también lleva el bit SYN activado, y que además reconoce el número de secuencia enviado por el cliente. Observa que el reconocimiento del servidor indica que el próximo segmento del cliente deberá numerarse con el valor **x+1**.
3. El cliente al recibir la respuesta del servidor considera establecida la conexión. A continuación, envía un tercer segmento que confirma el del servidor. Al recibir este tercer segmento, el servidor considera también la conexión ya establecida.

Como hemos visto la conexión se ha establecido en el cliente al recibir la confirmación del segmento enviado, pero ten en cuenta que el cliente no tiene problemas de ambigüedad. El TCP del cliente sabe perfectamente si tiene o no una solicitud de conexión pendiente de confirmar o no, y qué número de reconocimiento debe llevar el segmento recibido del servidor.

4.2 Problemas debidos a duplicados enviados por el cliente

Para entender mejor cómo funciona el esquema propuesto y cómo soluciona distintos problemas que pueden plantearse, veamos ahora lo que ocurre si debido a un duplicado como el de la figura 4 hay un intento de conexión erróneo.

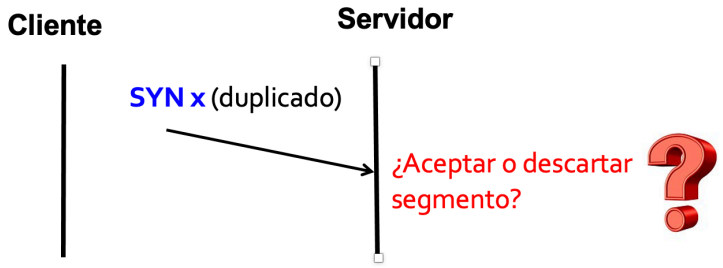

Figura 4. Recepción de un duplicado de establecimiento de conexión en el servidor

La primera cuestión que debemos plantearnos es si el servidor puede detectar que se trata de un segmento duplicado o no. Por lo tanto, si debe a aceptar la conexión o simplemente ignorar el duplicado. Piensa en las tres situaciones posibles en las que puede encontrarse el servidor:

1. Aún no se ha completado el establecimiento de la conexión, porque el servidor no ha recibido el tercer segmento.
2. La conexión ya está establecida pero todavía no se ha cerrado.
3. La conexión ya ha finalizado y está cerrada.

Razona cómo habría que actuar en cada una de ellas. Recuerda que también las respuestas que envía el servidor pueden perderse, por lo que puede ser necesario que el servidor tenga que retransmitir algún segmento ya enviado. Para cada uno de los tres casos anteriores piensa si el servidor debe aceptar o descartar el segmento, y si debe o no retransmitir algún segmento ya enviado previamente. No sigas leyendo hasta que hayas obtenido tus propias conclusiones.

Vamos a comprobar ahora los aciertos has conseguido. Veamos en primer lugar lo que ocurre si el servidor recibe un segmento SYN duplicado mientras se está estableciendo la conexión (Figura 5). Este duplicado podría generarse, por ejemplo, si se perdiese la respuesta del servidor (SYN + ACK). Al recibir la solicitud por segunda vez, el servidor debe volver a aceptarla y a enviar una nueva confirmación. Piensa por qué. Esta situación realmente no es problemática, aunque si el servidor no reenvía de nuevo el segmento SYN + ACK el cliente seguirá retransmitiendo una y otra vez el primer segmento de establecimiento de la conexión sin poder seguir adelante.

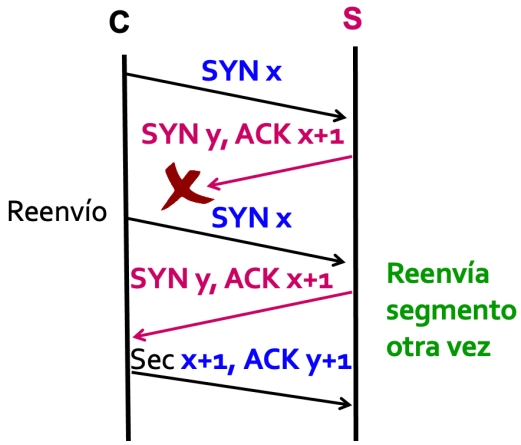

Figura 5. Recepción de un duplicado SYN cuando la conexión TCP aún no se ha establecido

El segundo caso donde uno de dos los segmentos SYN enviados por el cliente llega al servidor cuando la conexión ya está establecida y sigue aún activa, tampoco es problemático (Figura 6). El servidor detectará que ya tiene establecida una conexión entre los mismos extremos, con los mismos identificadores de direcciones IP y números de puerto, y descartará el segmento recibido por ser un duplicado.

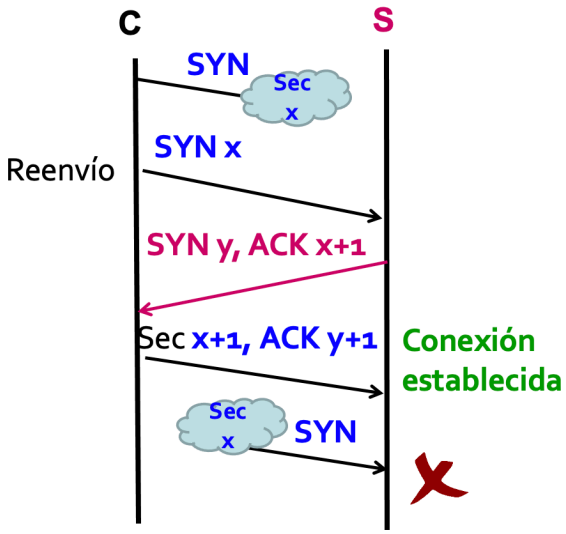

Figura 6. Recepción de duplicado SYN mientras la conexión sigue en curso

Veamos ahora lo que ocurre en el tercer caso, si el segmento SYN duplicado llega al servidor cuando ya ha finalizado la conexión que lo originó (Figura 7).

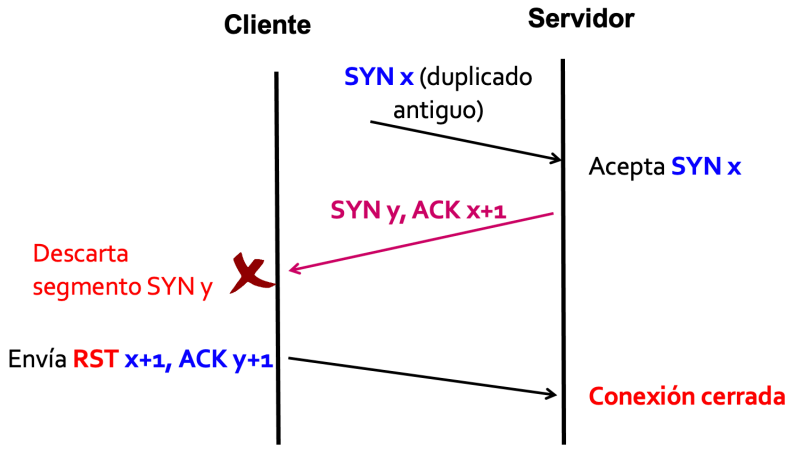

Figura 7. Recepción de duplicado cuando la conexión TCP ya está cerrada

El servidor lo aceptará como válido y enviará la confirmación al cliente. Cuando el cliente lo reciba, su TCP comprobará que no hay ninguna solicitud pendiente y lo descartará por tratarse de un error. Además, piensa en lo que ocurrirá si lo descarta silenciosamente. El pobre servidor, que no sabría nada, se hartaría de retrasmitir el segmento descartado creyendo que el paquete que lo lleva ha tenido problemas en la red y no ha llegado al cliente. Sin embargo, ¡lo que estamos buscando es mantener los recursos del servidor tan libres como sea posible! Así, que el cliente debe avisar al servidor del intento erróneo. Estos avisos se realizan mediante un segmento con el bit RST de la cabecera TCP activado, y además como puedes ver en la imagen utilizan un número de secuencia y de reconocimiento coherentes con el segmento recibido del servidor. Esto permite al servidor determinar sin ambigüedad cuál es el segmento que ha generado el problema, incluso aún en el caso de que tenga varios establecimientos de conexión en curso con el mismo el mismo ordenador donde se ejecuta el cliente.

4.3 El bit RST

Los segmentos RST siempre avisan de un problema. Además del caso anterior, otro ejemplo típico de uso se da cuando un cliente TCP intenta establecer una conexión pero el puerto destino está cerrado. Es decir, no hay un servidor escuchando en ese puerto (Figura 8). También se utilizan también cuando se recibe un segmento con un número de secuencia imposible para una conexión que ya estaba abierta.

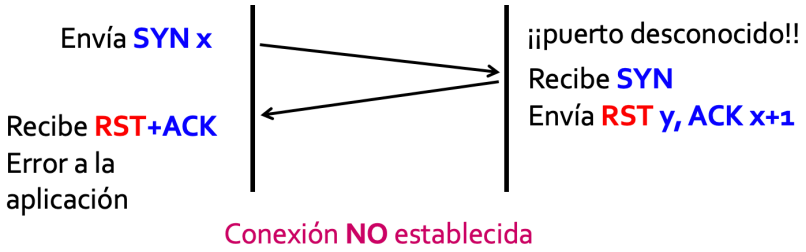

Figura 8. Envío de segmento RST porque el puerto destino del servidor está cerrado

4.4 El número de secuencia inicial

El número de secuencia es un campo de 32 bits que aparece en las cabeceras de todos los segmentos TCP desde que se inicia una conexión hasta que se cierra. Lleva la cuenta de los bytes de datos transmitidos durante la conexión en módulo 32. Es decir, el contador vuelve a cero cuando se alcanza el valor máximo.

El valor inicial es elegido por el sistema operativo basándose en un contador local y los parámetros de la conexión (puertos y direcciones IP). El estándar recomienda incrementar este contador en una unidad al menos una vez cada 4 μs. Es decir, cada extremo elige su número de secuencia inicial de forma independiente. El contador tarda en dar la vuelta completa y volver a llegar al mismo número de secuencia inicial aproximadamente 4 horas y 55 minutos. Como el tiempo de vida máximo de un segmento en la red es de uno dos minutos aproximadamente, la probabilidad de que dos números de secuencia iniciales coincidan por azar es prácticamente despreciable.

5 Resumen basado en un ejemplo práctico

Para terminar, vamos a recordar los principales contenidos de este artículo aplicándolos a un ejemplo real. En la figura 9 puedes ver una captura de tráfico realizada mediante el programa *wireshark*[1] que muestra un establecimiento de conexión TCP con el servidor web de la UPV (puerto 80). Si te animas puedes instalarte el programa *wireshark* y capturar más establecimientos de conexión desde tu propio ordenador. ¡Es muy fácil!

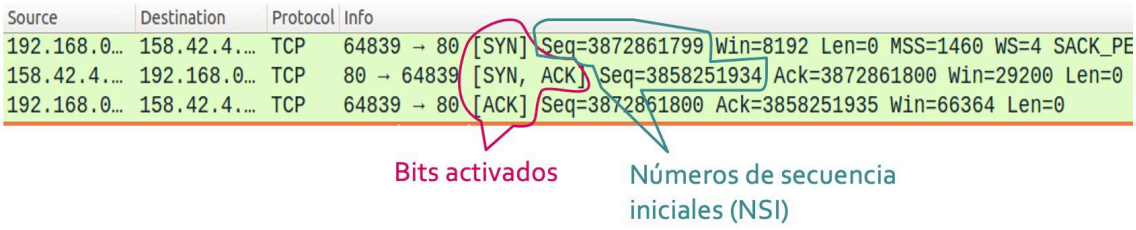

Figura 9. Captura con el programa wireshark de establecimiento de una conexión TCP

Analicemos un poco cada uno de los tres segmentos que aparecen en la figura anterior. El primer segmento lleva el bit SYN activado, pero no el de reconocimiento, puesto que aún no hay nada que reconocer. Además, el cliente ha elegido su número de secuencia inicial, que acaba en 799. Recordemos que es un valor de 32 bits, por lo que no resulta muy amigable de leer. En el artículo lo hemos estado representando con el valor x.

El segundo segmento, la respuesta del servidor, también lleva el bit SYN activado y además el bit de reconocimiento ACK, puesto que reconoce el primer segmento, enviado por el cliente. Observa que el campo de reconocimiento (Ack), cuyo valor representábamos antes

[1] *Software libre disponible para Microsoft Windos, Linux y Mac OS en https://www.wireshark.org/*

como x+1, acaba en 800. Es decir 799+1. Estos dos segmentos no llevan datos, ni pueden llevarlos, puesto que la conexión no está aún establecida en ninguno de los dos extremos. Recordemos que, tras recibir el segmento del servidor, el cliente considera ya establecida la conexión. El estándar TCP admite que el cliente pueda ya enviar datos en el tercer segmento, aunque este segmento aún forma parte del establecimiento de la conexión. En este caso no envía datos todavía, observa que el campo Len ("*Length*" en inglés), indica que el número de bytes de datos enviados es cero.

6 Bibliografía

Eddy, W. "RFC 9293 Transmission Control Protocol (TCP)", Ed. MTI Systems, 2022. Disponible en https://www.rfc-editor.org/rfc/rfc9293.pdf.

Kurose, J.F.; Ross, K.W.: "Redes de computadoras. Un enfoque descendente", en Ed. Pearson, 2017, pág. 209–214.