Transporte TCP

Antunez Joaquin, Gonalez Alejo y Nielsen Maximiliano

Instituto Politécnico Superior Gral. San Martín

2019

Conexiones TCP

Las conexiones del protocolo TCP se podrían resumir en 3 pasos. TCP es orientado a la conexión, es decir, que antes de las dos redes en cuestión se comuniquen y empiece la transferencia de datos se debe establecer una conexión.

1. Establecimiento de la conexión
2. Transferencia de datos
3. Liberación de la conexión

El establecimiento de dicha conexión se realiza con el método three-way handshake y el cierre con el método four-way handshake.

Establecimiento de la conexión

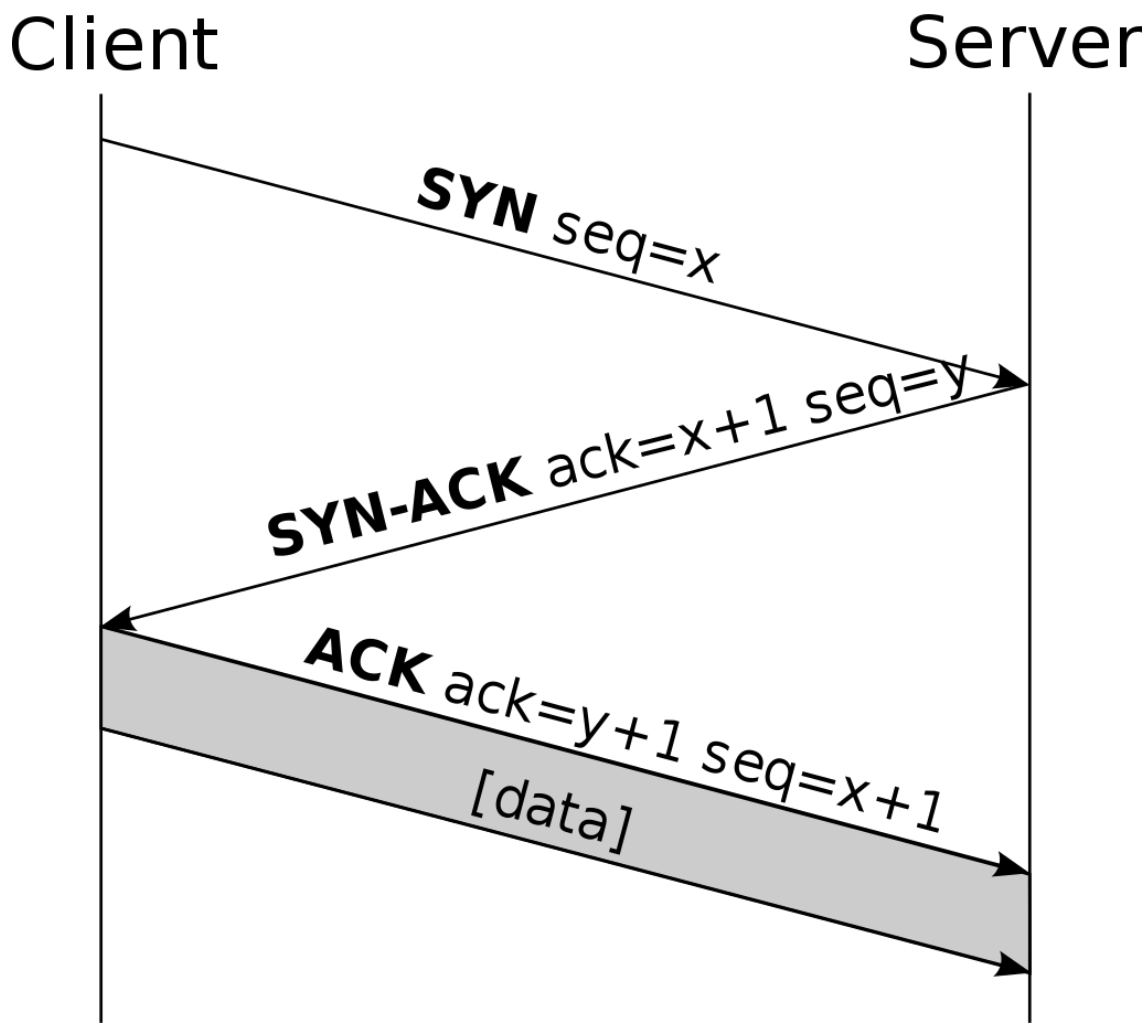
3-way handshake (confirmación de 3 pasos): En este método el cliente y el servidor intercambian paquetes SYN y ACK antes de que la comunicación y transferencia de datos comience.

- SYN: Es utilizado para establecer conexiones. En esencia es usado para indicar REQUERIMIENTO DE CONEXIÓN y CONEXIÓN ACEPTADA.
- ACK: Es igual a 1 para indicar que el número de reconocimiento es válido. Si vale 0, el paquete no contiene un reconocimiento, y entonces el campo número de reconocimiento es ignorado.

Establecimiento de la conexión: 3-way handshake

1. El cliente envía un paquete SYN hacia el servidor para averiguar si este está disponible para entablar nuevas conexiones.
2. El servidor, que tiene abiertos los puertos necesarios para que se puede iniciar una conexión, al recibir el paquete SYN del cliente le responde enviando una confirmación, un paquete SYN/ACK.
3. El cliente responde con un paquete de confirmación ACK.

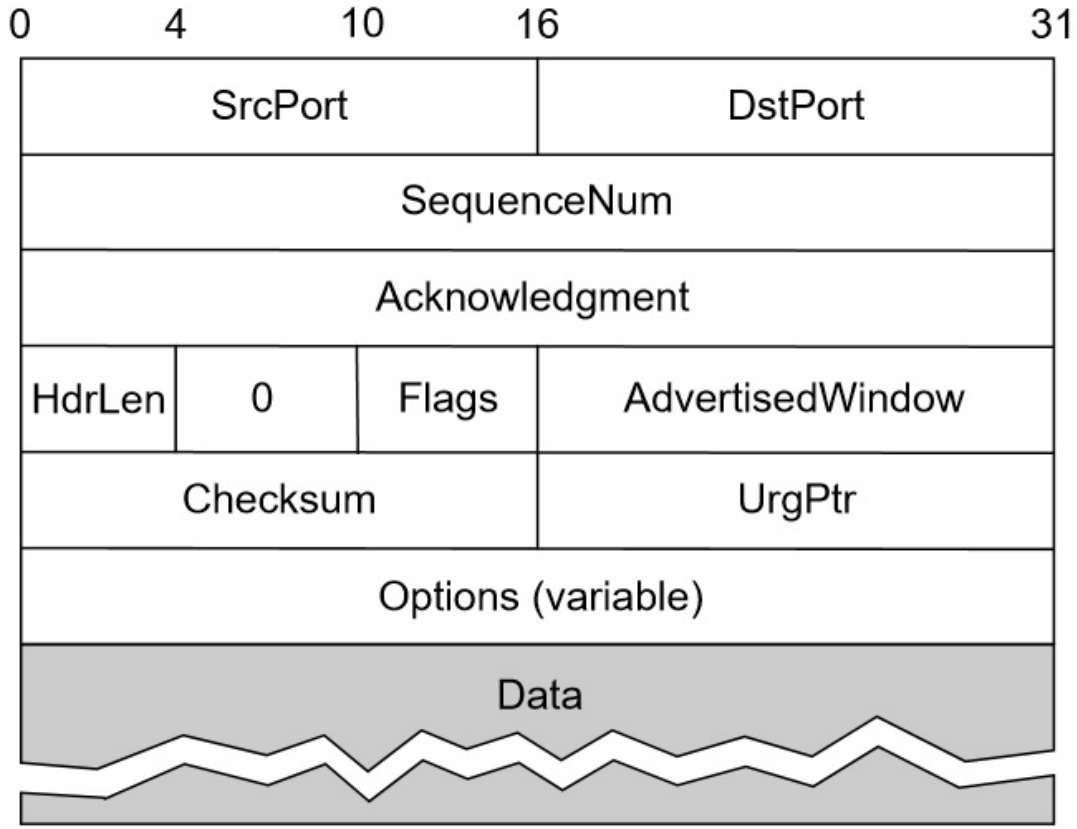
Establecimiento de la conexión: 3-way handshake



Una vez que este proceso concluye, las dos partes ya pueden iniciar la transferencia de datos.

Transferencia de datos

Segmento TCP: En el protocolo TCP (capa transporte) los datos se envían en forma de segmento.



Transferencia de datos: formato de segmento

- Puerto fuente y puerto destino (16 bits) : Identifican los puntos finales locales de la conexión.
- Número de Secuencia (32 bits): El cliente y servidor envían sus números de secuencia (nótese distintos) durante el establecimiento de la conexión, de manera que identifica de forma inequívoca los datos de aplicación que contiene el segmento TCP. Distingue el primer byte de datos.
- Número de reconocimiento (32 bits) (si ACK es establecido) : El indica el siguiente byte que espera el receptor.
- Longitud cabecera (4 bits):Indica cuántas palabras de 32 bits están contenidas en la cabecera de TCP. Es necesario a causa de la longitud variable del campo Opciones.

Transferencia de datos: formato de segmento

- Flags
  - SYN
  - ACK
  - FIN
  - URG: Es igual a 1 si el campo Puntero a urgente está en uso.
  - PSH: "PUSHed data". Indica al receptor que debe entregar los datos a la aplicación inmediatamente después del arribo del paquete. No debe esperar hasta que un buffer total haya sido recibido.
  - RST: Se utiliza para resetear una conexión que se ha vuelto confusa debido a la caída de un host o alguna otra razón.
- Tamaño de la ventana: Indica cuántos bytes pueden ser enviados comenzando desde el último byte reconocido. Para propósitos de control de flujo.
- Suma de verificación (16 bits): Checksum utilizado para la comprobación de errores tanto en la cabecera como en los datos.

Transferencia de datos: formato de segmento

- Código de redundancia: Verifica la cabecera y los datos. Puntero urgente (si URG es establecido): Se utiliza para indicar un desplazamiento en bytes a partir del número de secuencia actual en el que se encuentran los datos urgentes. Esta facilidad se brinda en lugar de los mensajes de interrupción.
- Opciones: Diseñado para proveer una manera de adicionar facilidades extras no cubiertas por la cabecera regular.

Liberación de la conexión

4-way handshake (confirmación de 4 pasos): en este método el cliente y servidor intercambian paquetes FIN y ACK una vez que la transferencia de datos haya finalizado.

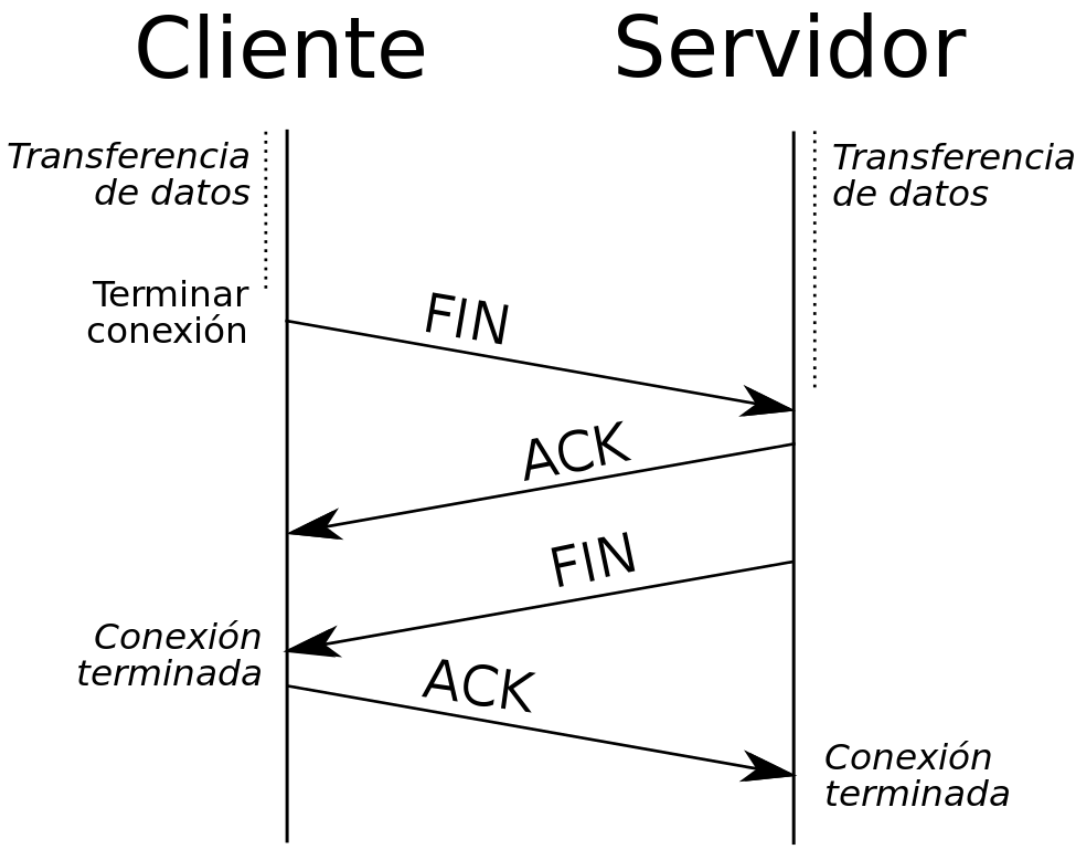
- FIN: Es utilizado para liberar conexiones. Especifica que el emisor no tiene más datos para transmitir.

Liberación de la conexión: 4-way handshake

Pasos:

1. La parte que desee terminar su comunicación (generalmente el cliente) envía un paquete FIN para avisarle a la otra parte que desea concluir su comunicación.
2. La parte que recibe el paquete FIN envía un paquete ACK de confirmación.
3. Esa misma parte que envió el paquete ACK envía un paquete FIN para finiquitar su comunicación.
4. La otra parte le envía un paquete ACK de confirmación.

Liberación de la conexión:4-way handshake

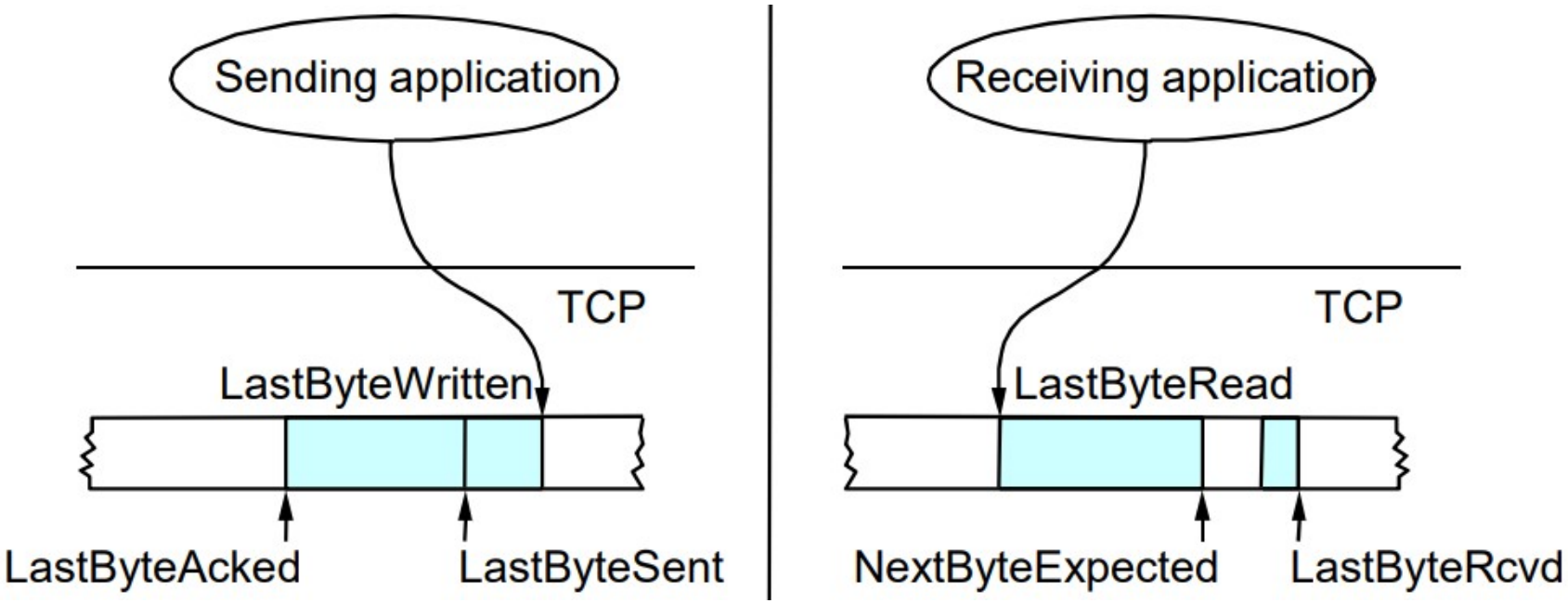


Ventana deslizante

TCP implementa una variante del protocolo de ventana deslizante. Este permite al emisor transmitir múltiples segmentos de información antes de comenzar la espera para que el receptor le confirme la recepción de los segmentos, tal confirmación se llama validación, y consiste en el envío de mensajes denominados ACK del receptor al emisor.
Características del protocolo:

1. Garantiza Confiabilidad
2. La aplicación recibe los datos en orden
3. Fuerza el control de flujo entre receptor y transmisor

Ventana Deslizante comunicacíon



Ventana Deslizante emisor

Lado del transmisor: Se encuentran 3 punteros, que siguen las siguientes relaciones:

$$\textbf{LastByteAcked} <= \textbf{LastByteSent}$$
$$\textbf{LastByteSent} <= \textbf{LastByteWritten}$$

Se bufferean los bytes entre LastByteAcked (los que están a su derecha ya fueron confirmados) y LastByteWritten (los que están a su derecha no fueron generados . . .)

Ventana Deslizante receptor

Las relaciones son menos intuitivas debido al reordenamiento

LastByteRead <NextByteExpected

Un byte no puede ser leído por la aplicación a menos que este se halla recibido y todos su precedentes.

NextByteExpected <= LastByteRcvd +1

Si están en orden se cumple la igualdad . Si están en desorden **NextByteExpected** apunta al espacio vacio antes de **LastByteRead+1**.
Buffereamos los bytes entre **NextByteRead** y **LastByteRcvd**.

Ventana Deslizante Control de Flujo

Tamaño del buffer de envío: **MaxSendBuffer**.
Tamaño del buffer de recepción: **MaxRcvBuffer**.
Lado receptor:

$$\textbf{LastByteRcvd - LastByteRead} <= \textbf{MaxRcvBuffer}$$
$$\textbf{AdvertisedWindow} = \textbf{MaxRcvBuffer - (LastByteRcvd - NextByteRead)}$$

Lado transmisor:

$$\textbf{LastByteSent - LastByteAcked} <= \textbf{AdvertisedWindow}$$
$$\textbf{EffectiveWindow} = \textbf{AdvertisedWindow - (LastByteSent - LastByteAcked)}$$
$$\textbf{LastByteWritten - LastByteAcked} <= \textbf{MaxSendBuffer}$$

Bloquear Tx si (**LastByteWritten - LastByteAcked) + y** <**MaxSenderBuffer**, y bytes que se desean escribir.
Siempre enviar ACK en respuesta a la llegada de segmentos. Tx persiste enviando 1 byte cuando **AdvertisedWindow = 0**