Tema 5-UDC y TCP

TCP (Transmission Control Protocol) es un protocolo de nivel de transporte que proporciona un servicio de envío de datagramas fiable y orientado a conexión. Esto significa que antes de que las aplicaciones puedan intercambiar datos, deben establecer una conexión TCP entre ellas. Una vez establecida la conexión, los datos se envían de forma fiable y en orden, y se eliminan duplicados.

Los paquetes TCP se denominan segmentos y TCP es full-duplex, lo que significa que la comunicación es bidireccional y simultánea. TCP gestiona los buffers y ejerce control de flujo de forma eficiente, multiplexa el nivel de aplicación (puertos) e intercambia datos con las aplicaciones. Además, controla errores, retransmite segmentos perdidos o erróneos y efectúa control de congestión.

TCP no admite broadcasting y multicasting, ya que requiere una conexión establecida entre dos aplicaciones. TCP también es independiente del medio de transmisión subyacente, lo que significa que puede utilizarse en cualquier tipo de red.

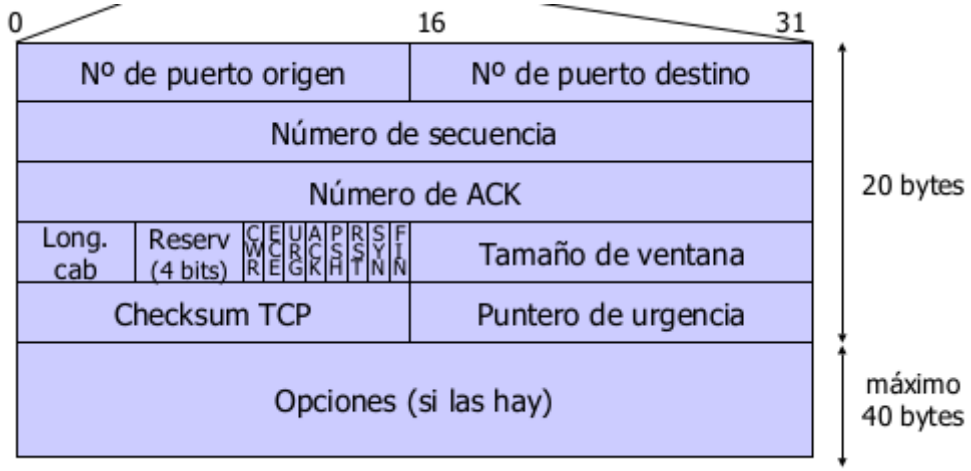
TCP implementa varios mecanismos para garantizar la fiabilidad de los datos transmitidos. Estos mecanismos incluyen:

- **Segmentación de datos**: los datos de la aplicación se dividen en segmentos con la longitud más adecuada para la aplicación. Esto permite que los datos se transmitan en paquetes más pequeños y hace que la transmisión sea más eficiente.
- **Temporizador**: asocia un temporizador con los segmentos que envía. Si no recibe un ACK del destino en un tiempo determinado, TCP retransmite el segmento. Esto asegura que los datos lleguen al destino incluso si algunos paquetes se pierden o se retrasan en la red.
- **Checksum**: utiliza un checksum en la cabecera TCP para comprobar que el segmento recibido es correcto. Si el segmento es incorrecto, TCP no envía un ACK.
- **Reordenamiento de segmentos**: si los segmentos llegan desordenados, TCP los reordena antes de pasarlos a la aplicación. Esto asegura que los datos se entreguen en el orden correcto.
- **Eliminación de segmentos duplicados**: TCP descarta segmentos que se han duplicado para evitar que la aplicación reciba datos redundantes.
- **Control de flujo**: TCP proporciona un mecanismo de control de flujo para evitar el desbordamiento del buffer del receptor. El receptor TCP sólo permite que el otro extremo transmita segmentos que puedan almacenarse en su buffer de entrada sin

provocar un desbordamiento. Esto evita la pérdida de datos debido a la sobrecarga del buffer del receptor.

Cabecera TCP

Un datagrama IP TCP es muy similar al datagrama IP UDP. Lo unico que cambia es la cabecera.



Los número de puerto origen y destino + dir. IP origen y destino de cabecera IP identifican unívocamente la conexión TCP. Este campo especifica los números de puerto (16 bits) de origen y destino, así como las direcciones IP de origen y destino. Juntos, identifican unívocamente la conexión TCP.

El número de secuencia identifica el número de byte en el flujo de bytes TCP entre el emisor y el receptor que supone el primer byte de la sección de datos: Este campo (32 bits) especifica el número de secuencia del primer byte en el segmento TCP. Cuando se llega a 2^{32}-1 se comienza de nuevo por 0. Cuando se establece una conexión, se pone a 1 el flag SYN, y la máquina selecciona un ISN (Initial Sequence Number) para esa conexión.

El número de ACK (acknowledgment) indica el siguiente número de secuencia que el emisor del ACK espera recibir: Este campo (32 bits) especifica el número de secuencia que el receptor espera recibir a continuación. Es el número de secuencia más el tamaño en bytes de los datos recibidos, y representa el siguiente byte esperado. TCP proporciona una comunicación full-duplex al nivel de aplicación, por lo que cada extremo mantiene su propio número de secuencia.

La longitud de cabecera (4 bits) especifica el tamaño de la cabecera TCP en palabras de 32 bits. Su valor máximo es 60 bytes (15×4).

Las flags pueden ser:

- **CWR (Congestion Window Reduced)**: se establece por el emisor para indicar que ha reducido su tamaño de ventana debido a que ha detectado congestión en la red.

- **ECE (ECN Echo)**: se establece por el receptor para indicar que ha recibido un paquete con el indicador ECN (Explicit Congestion Notification) activado en la cabecera IP, lo que significa que se ha detectado congestión en algún punto de la red.

- **URG**: se utiliza para indicar que hay datos urgentes en el segmento TCP. El campo "Puntero de urgencia" indica la posición del último byte de datos urgentes.
- **ACK**: se establece para indicar que el número de ACK es válido. Una vez establecida la conexión, este indicador está siempre activado.
- **PSH**: se utiliza para indicar que el receptor debe pasar los datos al nivel de aplicación lo antes posible, en lugar de acumularlos en un buffer.
- **RST**: se utiliza para reinicializar la conexión TCP, generalmente debido a algún problema en la comunicación.
- **SYN**: se utiliza en el inicio de una conexión para sincronizar los números de secuencia entre el emisor y el receptor.
- **FIN**: se utiliza para indicar que el emisor ha terminado el envío de datos. El receptor debe responder con un ACK para confirmar la recepción de todos los datos.

El tamaño de ventana ndica el número de bytes, a partir del valor del campo de número de reconocimiento, que el receptor puede aceptar. Este campo se utiliza para establecer el control de flujo en la comunicación TCP. Su valor máximo es 65.535, pero existe una opción de factor de escala para aumentar este valor.

El checksum contiene un valor de verificación de errores que se calcula sobre todo el segmento TCP, incluyendo la cabecera y los datos. Es obligatorio, lo que significa que el emisor debe calcularlo y el receptor debe verificarlo para garantizar que no haya errores en la transmisión. El cálculo se realiza utilizando un algoritmo similar al utilizado en UDP.

El puntero de urgencia (solo es válido si el flag URG está activado) indica un desplazamiento que se debe sumar al número de secuencia para indicar la posición de los datos urgentes en el segmento. Este campo se utiliza para transmitir datos urgentes, que deben ser procesados antes que otros datos.

Las opciones se utiliza para incluir información adicional en la cabecera TCP, como la opción de tamaño máximo de segmento (MSS). Esta opción indica el tamaño máximo de los segmentos que el emisor está dispuesto a recibir. La opción Window Scale Factor es otra opción en la cabecera TCP que permite a los dispositivos ampliar el tamaño de la ventana de congestión a un tamaño mayor que los 64KB originales. Esto es importante porque permite a los dispositivos manejar mejor la congestión en la red.

Conexiones TCP

En TCP, para establecer una conexión, antes de que los datos puedan ser enviados entre el cliente y el servidor, se establece una conexión mediante un protocolo de establecimiento de conexión conocido como "Three-Way Handshake". El cliente inicia esta conexión enviando un segmento SYN (Synchronize) al servidor, indicando el número de secuencia inicial que va a utilizar. El servidor responde con un segmento SYN-ACK (Synchronize-Acknowledge) que contiene su propio número de secuencia inicial y también confirma el SYN del cliente agregando 1 al número de secuencia del cliente. Finalmente, el cliente responde con un segmento ACK (Acknowledge) que confirma el SYN-ACK del servidor agregando 1 al número de secuencia del servidor.

El objetivo de este protocolo es establecer una conexión confiable y sincronizada entre el cliente y el servidor. Además, se utiliza un número de secuencia inicial (ISN) generado aleatoriamente por cada extremo de la conexión. Esto evita que segmentos "antiguos" (de una conexión anterior) se confundan con los actuales, lo que se conoce como "reencarnación". De esta manera, se asegura que la conexión se establezca correctamente y que los datos enviados y recibidos sean los correctos.

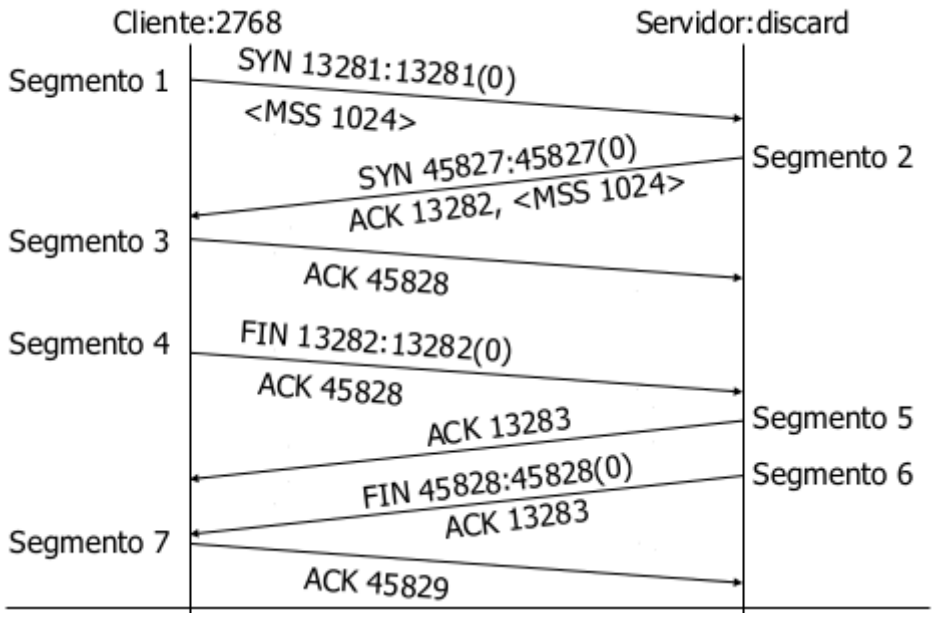
Finalizar una conexión se realiza mediante el intercambio de 4 segmentos para cerrar una conexión. Cada extremo envía un FIN cuando ha finalizado el envío de datos. El extremo que envía el primer FIN realiza el cierre activo, y el otro extremo realiza el cierre pasivo. Cualquiera de los dos extremos puede iniciar el cierre.

Es importante destacar que una conexión TCP es full-duplex y cada dirección se cierra de manera independiente. Además, después de que un extremo haya enviado un FIN, el otro extremo puede continuar enviando datos (half-close).

Los pasos de finalización son los siguientes:

1. El cliente finaliza la aplicación y envía un FIN (segmento 4) con el número de secuencia correspondiente indicando que ha cerrado el flujo de datos desde el cliente al servidor.
2. El servidor responde con un ACK (segmento 5) del número de secuencia +1 (los mensajes FIN consumen un número de secuencia).
3. El servidor envía un FIN (segmento 6) para indicar que ha cerrado el flujo de datos desde el servidor al cliente.
4. El cliente confirma la recepción del FIN con un ACK del número de secuencia recibido + 1 (segmento 7).

Por ejemplo:



MTU y MSS

El Maximum Transmission Unit (MTU) es el número máximo de bytes que un nivel de enlace puede transmitir en un solo paquete. El MTU depende del medio físico de la red y se establece en función del estándar de la tecnología de red utilizada.

Por otro lado, el Maximum Segment Size (MSS) es la cantidad máxima de datos que un host receptor desea recibir en un solo segmento TCP. El MSS se anuncia durante el proceso de establecimiento de conexión TCP y se utiliza para evitar la fragmentación de paquetes en la red.

Es importante mencionar que el MSS no incluye las longitudes de las cabeceras IP y TCP, por lo que el tamaño máximo real de un segmento TCP es MTU - 40 (20 bytes para la cabecera IP y 20 bytes para la cabecera TCP). Por defecto, si no se especifica un valor de MSS, se asume un valor de 536 bytes para el segmento TCP.

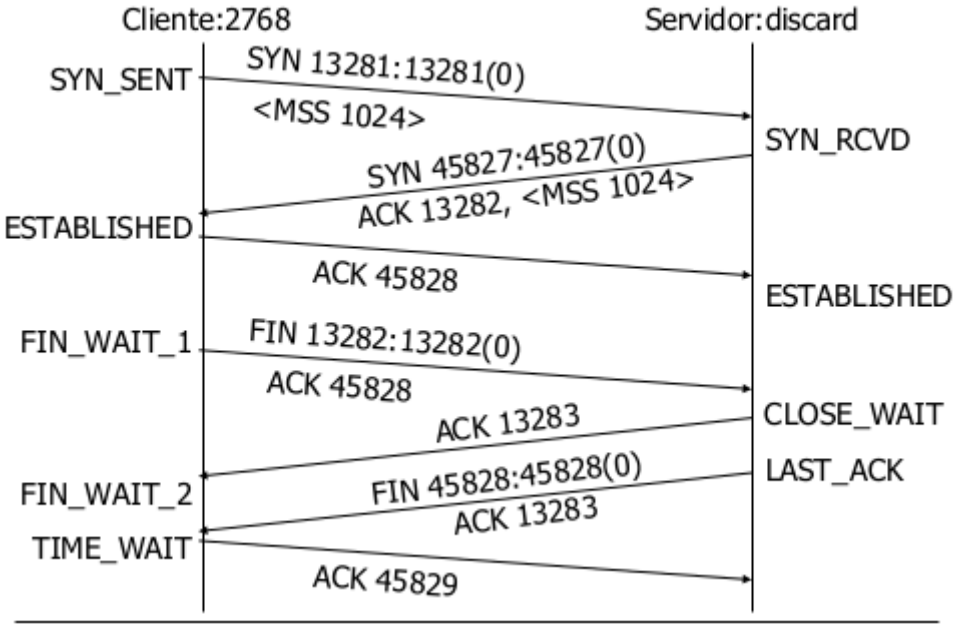
Es recomendable utilizar un MSS grande para amortiguar el costo de las cabeceras y minimizar el tiempo de transmisión. Sin embargo, también es importante tener en cuenta el tamaño del MTU de la red para evitar la fragmentación de paquetes. En caso de que se necesite fragmentar un paquete, el rendimiento de la red puede verse afectado debido al aumento en el tráfico de la red y la posibilidad de pérdida de paquetes.

Estados TCP

El estado TIME_WAIT y el estado FIN_WAIT_2 son estados que pueden tener las conexiones TCP durante el proceso de finalización de la conexión.

El estado TIME_WAIT se produce después de que un extremo haya enviado un FIN y recibido el ACK correspondiente, pero aún necesita esperar para recibir un posible paquete perdido (el ACK del otro extremo) antes de cerrar definitivamente la conexión. TCP espera dos veces el tiempo máximo de vida de un paquete en la red (MSL) antes de salir de este estado, lo que permite a TCP reenviar el ACK en caso de que se haya perdido y garantiza que no aparezcan reencarnaciones de segmentos en futuras conexiones.

El estado FIN_WAIT_2 ocurre cuando un extremo ha enviado su FIN y ha recibido el ACK correspondiente, pero aún está esperando el FIN del otro extremo para confirmar que ambos han terminado de enviar datos. El otro extremo estará en el estado CLOSE_WAIT y debe esperar a que se cierre la aplicación. Si no se recibe el FIN del otro extremo en un tiempo determinado, la conexión pasará al estado CLOSED directamente para evitar una espera infinita.



Segmentos de RESET

Cuando un paquete llega a un extremo de una conexión TCP, se espera que esté relacionado con esa conexión, es decir, que esté destinado al puerto y la dirección IP correspondiente a esa conexión. Sin embargo, en algunos casos, puede llegar un paquete que no está relacionado con ninguna conexión abierta, como por ejemplo un intento de conexión a un puerto que no existe o una respuesta a una conexión semi-abierta. En estos casos, se activa el bit de Reset (RST) en la cabecera TCP del paquete recibido para indicar que no se reconoce la conexión a la que está referido el paquete y que se debe cerrar inmediatamente.

El envío de un segmento de Reset es una forma rápida y efectiva de cerrar una conexión de manera abrupta. Cuando se recibe un paquete con el bit de Reset activado, el extremo

receptor lo interpreta como una señal para cerrar la conexión de forma inmediata, sin realizar ningún tipo de intercambio de datos adicional. Esto puede suceder por ejemplo si se está intentando establecer una conexión con un servidor y se envían paquetes a un puerto que no está abierto para esa conexión. En este caso, el servidor respondería con un segmento de Reset para indicar que no se reconoce la conexión y cerrarla.

- Intento de conexión a un puerto no existente
- Respuesta ante conexiones semi-abiertas