UD1. Capa de Transporte

Índice

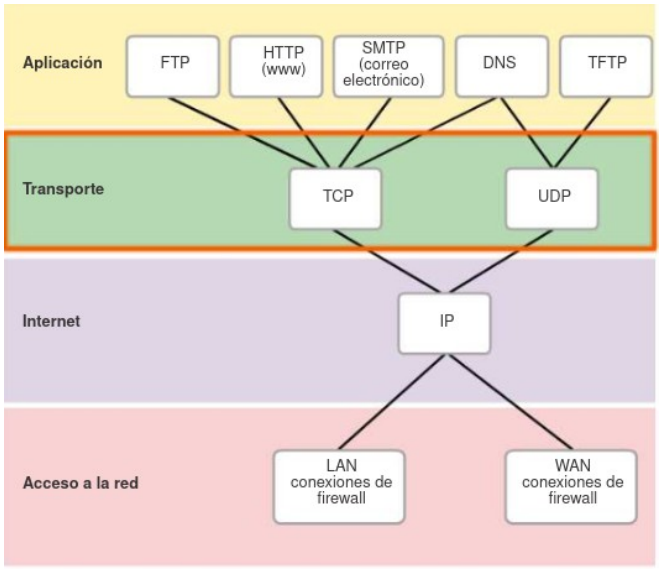
La capa de Transporte

Como sabemos un protocolo de la capa de transporte proporciona una abstracción de la comunicación de manera que podemos operar con dos hosts distantes como si estuvieran conectados directamente.

Estos protocolos están implementados en los hosts terminales, pero no en los routers de la red. La capa de transporte transforma el mensaje en paquetes de capa de transporte conocidos como segmentos. Esto por lo general se hace dividiendo el mensaje original en fragmentos más pequeños y añadiendo una cabecera a cada uno de ellos. Tras esto la capa de transporte pasa a la capa de red, que gestiona con paquetes de capa de red el envío (en un datagrama).

En internet hay dos protocolos que ofrecen distintos tipos de servicios para la capa de transporte, TCP y UDP.

- Servicio TCP: está orientado a conexión (establecer una conexión entre los dos involucrados previo al envío), este transporte es fiable ante pérdidas (control de errores), con control de flujo y de congestión.
- Servicio UDP: no está orientado a conexión, es decir, no se comprueba que ambos estén preparados para realizar la comunicación. El transporte no es fiable, no tiene control de flujo ni de congestión



Funciones

Rastreo de conversaciones individuales

En la capa de transporte, cada conjunto de datos particular que fluye entre una aplicación de origen y una de destino se conoce como “conversación”. Un host puede tener varias aplicaciones que se comunican a través de la red de forma simultánea. Cada una de estas aplicaciones se comunica con una o más aplicaciones en uno o más hosts remotos. Es responsabilidad de la capa de transporte mantener y hacer un seguimiento de todas estas conversaciones.

Segmentación de datos y rearmado de segmentos

Se deben preparar los datos para el envío a través de los medios en partes manejables. La mayoría de las redes tienen un límite de la cantidad de datos que se puede incluir en un solo paquete. Los protocolos de la capa de transporte tienen servicios que segmentan los datos de aplicación en bloques de datos de un tamaño apropiado. En el destino, la capa de transporte debe poder reconstruir las porciones de datos en un stream de datos completo que sea útil para la capa de aplicación.

Identificación de aplicaciones

Puede haber muchas aplicaciones o servicios que se ejecutan en cada host de la red. Para pasar streams de datos a las aplicaciones adecuadas, la capa de transporte debe identificar la aplicación objetivo. Para lograr esto, la capa de transporte asigna un identificador a cada aplicación. Este identificador se denomina “**número de puerto**”. A todos los procesos de software que requieran acceder a la red se les asigna un número de puerto exclusivo en ese host. La capa de transporte utiliza puertos para identificar la aplicación o el servicio.

Multiplexación y demultiplexación:

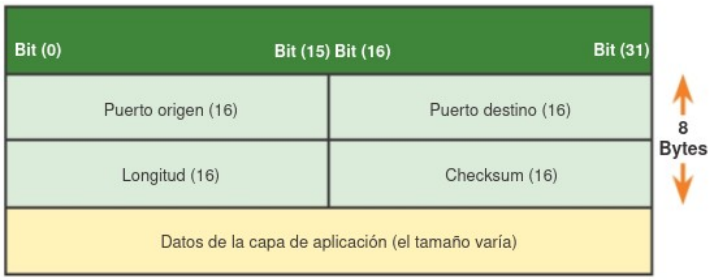
La segmentación de los datos en partes más pequeñas permite que se entrelacen (multiplexen) varias comunicaciones de distintos usuarios en la misma red. La segmentación de los datos según los protocolos de la capa de transporte también proporciona los medios para enviar y recibir datos cuando se ejecutan varias aplicaciones a la vez en un PC.

Sin la segmentación, solo podría recibir datos una aplicación. Por ejemplo, con un streaming video, los medios se consumirían por completo por ese stream de comunicación en lugar de compartirse. No podría recibir correos electrónicos, chatear por mensajería instantánea o visitar páginas Web mientras mira el video.

UDP

Es un servicio de entrega de mejor esfuerzo (best effort), pues no garantiza la correcta entrega de todos los segmentos, pero hará todo lo que pueda para ello.

Entre sus ventajas destacan el control a nivel de aplicación sobre los datos que se envían y cuándo se envían, pues en cuanto lo indica la aplicación, UDP los empaqueta y los envía inmediatamente. Todo esto tiene adicionalmente una menor sobrecarga en la cabecera de los paquetes.



UDP es un protocolo tan simple que, por lo general, se lo describe en términos de lo que no hace en comparación con TCP.

- **Sin conexión:** UDP no establece una conexión entre los hosts antes de que se puedan enviar y recibir datos.
- **Entrega no confiable:** UDP no proporciona servicios para asegurar que los datos se entreguen con confianza. UDP no cuenta con procesos que hagan que el emisor vuelva a transmitir los datos que se pierden o se dañan.
- **Reconstrucción de datos no ordenada:** en ocasiones, los datos se reciben en un orden distinto del de envío. UDP no proporciona ningún mecanismo para rearmar los datos en su secuencia original. Los datos simplemente se entregan a la aplicación en el orden en que llegan.
- **Sin control del flujo:** UDP no cuenta con mecanismos para controlar la cantidad de datos que transmite el dispositivo de origen para evitar la saturación del dispositivo de destino. El origen envía los datos. Si los recursos en el host de destino se sobrecargan, es probable que dicho host descarte los datos enviados hasta que los recursos estén disponibles. A diferencia de TCP, en UDP no hay un mecanismo para la retransmisión automática de datos descartados.

Cuando se envían datagramas múltiples a un destino, pueden tomar diferentes rutas y llegar en el orden equivocado. UDP no realiza un seguimiento de los números de secuencia de la manera en que lo hace TCP. UDP no tiene forma de reordenar datagramas en el orden en que se transmiten, como se muestra en la ilustración.

Por lo tanto, UDP simplemente reensambla los datos en el orden en que se recibieron y los envía a la aplicación. Si la secuencia de datos es importante para la aplicación, esta debe identificar la secuencia adecuada y determinar cómo se deben procesar los datos.

TCP

Es un servicio orientado a conexión, con una entrega ordenada garantizada. Es además full-duplex. TCP garantiza una entrega fiable sobre una capa de red no fiable (IP).

Con TCP, las tres operaciones básicas de confiabilidad son las siguientes:

- Seguimiento de segmentos de datos transmitidos y reconstrucción de datos ordenada
- Conversaciones orientadas a la conexión mediante el establecimiento de sesiones
- Entrega confiable mediante acuse de recibo y retransmisión de datos sin acuse de recibo
- Control del flujo.

La multiplexación y demultiplexación varían frente a UDP, aquí los sockets están identificados por cuatro elementos: Puerto e IP de destino y puerto e IP de origen.

Para entender con propiedad las diferencias entre TCP y UDP, es importante comprender la manera en que cada protocolo implementa las funciones específicas de confiabilidad y la forma en que realizan el seguimiento de las comunicaciones.

Establecimiento de una sesión

TCP es un protocolo orientado a la conexión. Un protocolo orientado a la conexión es uno que negocia y establece una conexión (o sesión) permanente entre los dispositivos de origen y de destino antes de reenviar tráfico. El establecimiento de sesión prepara los dispositivos para que se comuniquen entre sí. Mediante el establecimiento de sesión, los dispositivos negocian la cantidad de tráfico que se puede reenviar en un momento determinado, y los datos que se comunican entre ambos se pueden administrar detenidamente. La sesión se termina solo cuando se completa toda la comunicación.

Entrega confiable

TCP puede implementar un método para garantizar la entrega confiable de los datos. En términos de redes, confiabilidad significa asegurar que cada sección de datos que envía el origen llegue al destino. Por varias razones, es posible que una sección de datos se corrompa o se pierda por completo a medida que se transmite a través de la red. TCP puede asegurar que todas las partes lleguen a destino al hacer que el dispositivo de origen retransmita los datos perdidos o dañados. Tiene un sistema de control de flujo de detección y recuperación de errores (ARQ, Automatic Repeat reQuest). Esto se implementa mediante temporizadores que esperan la confirmación de la recepción (ACK, que es acumulativo, es decir, si se recibe un ack posterior es porque se ha recibido todo lo anterior correctamente). Si se agota el tiempo se reenvía el paquete. Trabaja sobre ventanas adaptables.

Entrega en el mismo orden

Los datos pueden llegar en el orden equivocado, debido a que las redes pueden proporcionar varias rutas que pueden tener diferentes velocidades de transmisión. Al numerar y secuenciar los segmentos, TCP puede asegurar que estos se rearmen en el orden correcto.

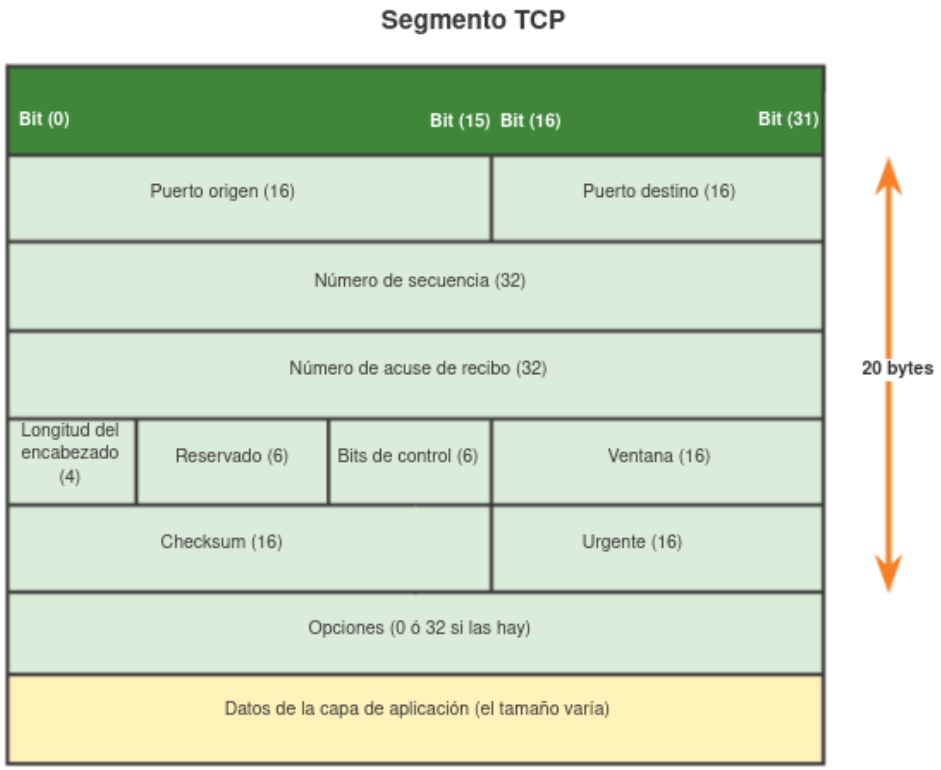
Control de flujo

Los hosts de la red cuentan con recursos limitados, como memoria o ancho de banda. Cuando TCP advierte que estos recursos están sobrecargados, puede solicitar que la aplicación emisora reduzca la velocidad del flujo de datos. Esto lo lleva a cabo TCP, que regula la cantidad de datos que transmite el origen. El control de flujo puede evitar la pérdida de segmentos en la red y evitar la necesitad de la retransmisión.

Utiliza “piggybacking”, es decir, en un paquete que va en un sentido de la comunicación se añade información sobre el otro sentido de la comunicación.

Así aprovechan los paquetes de datos para incluir información de control. Todas estas propiedades nos aseguran que es fiable en el control de congestión y flujo.

Segmentos TCP



La información a enviar por TCP se divide en segmentos TCP. Cada uno de esos segmentos contiene información del **puerto origen y destino**. **Número de secuencia (32 bits):** se utiliza para rearmar datos. **Número de acuse de recibo (32 bits):** indica los datos que se recibieron. **Longitud del encabezado (4 bits):** conocido como "desplazamiento de datos". Indica la longitud del encabezado del segmento TCP. **Reservado (6 bits):** este campo está reservado para el futuro. **Bits de control (6 bits):** incluye códigos de bit, o indicadores, que indican el propósito y la función del segmento TCP. **Tamaño de la ventana (16 bits):** indica la cantidad de segmentos que se puedan aceptar por vez. **Checksum (16 bits):** se utiliza para la verificación de errores en el encabezado y los datos del segmento. **Urgente (16 bits):** indica si la información es urgente.

La cantidad de información de la capa de aplicación que se transmite en cada uno de estos paquetes está limitada por el tamaño máximo del segmento (MSS, Maximum Segment Size), que generalmente está definido por la longitud de la trama más larga de la capa de enlace que el host emisor puede enviar (también conocida como unidad máxima de transmisión, MTU, Maximum Transmission Unit).

Puertos

La capa de transporte debe poder separar y administrar varias comunicaciones con diferentes necesidades de requisitos de transporte. El usuario envía y recibe correo electrónico y mensajes instantáneos, visita sitios Web y realiza una llamada telefónica de voz sobre IP (VoIP) simultáneamente. Cada una de estas aplicaciones envía y recibe datos a través de la red al mismo tiempo, a pesar de los diferentes requisitos de confiabilidad. Además, los datos de la llamada telefónica no están dirigidos al explorador Web y el texto de un mensaje instantáneo no aparece en un correo electrónico.

Por motivos de confiabilidad, los usuarios necesitan que un correo electrónico o una página Web se reciba y presente por completo para que la información se considere útil. Por lo general, se permiten leves retrasos en la carga de correo electrónico o de páginas Web, siempre y cuando el producto final se muestre en su totalidad y de forma correcta.

En cambio, la pérdida ocasional de partes pequeñas de una conversación telefónica se puede considerar aceptable. Incluso si se descartan partes pequeñas de algunas palabras, se puede deducir el audio que falta del contexto de la conversación o solicitar que la otra persona repita lo que dijo. Si la red administrara y reenviara segmentos faltantes, se prefiere lo mencionado anteriormente a los retrasos que se producen. En este ejemplo, es el usuario y no la red quien administra el reenvío o reemplazo de la información que falta.

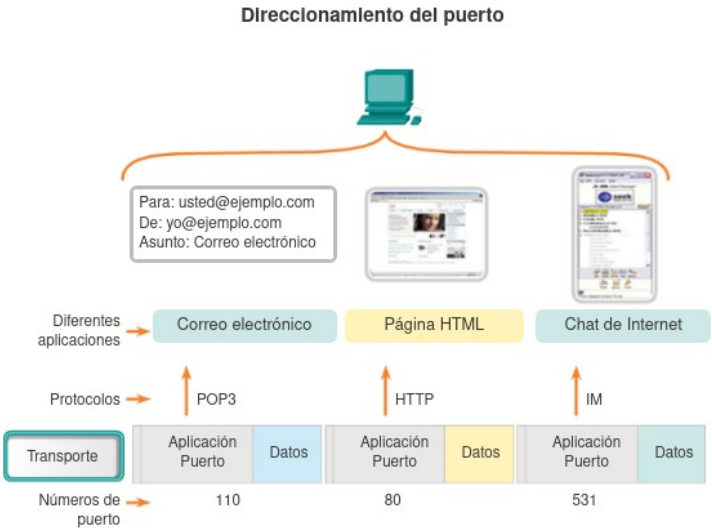
Para que TCP y UDP administren estas conversaciones simultáneas con diversos requisitos, los servicios basados en UDP y TCP deben hacer un seguimiento de las diversas aplicaciones que se comunican. Para diferenciar los segmentos y datagramas para cada aplicación, tanto TCP como UDP cuentan con campos de encabezado que pueden identificar de manera exclusiva estas aplicaciones. Estos identificadores únicos son números de puertos.

En el encabezado de cada segmento o datagrama, hay un puerto origen y uno de destino. Una manera de mejorar la seguridad en un servidor es restringir el acceso al servidor únicamente a aquellos puertos relacionados con los servicios y las aplicaciones a los que deben poder acceder los solicitantes autorizados.

Cuando se envía un mensaje utilizando TCP o UDP, los protocolos y servicios solicitados se identifican con un número de puerto. Un puerto es un identificador numérico de cada segmento, que se utiliza para realizar un seguimiento de conversaciones específicas y de servicios de destino solicitados. Cada mensaje que envía un host contiene un puerto de origen y un puerto de destino.

Puerto de destino

El cliente coloca un número de puerto de destino en el segmento para informar al servidor de destino el servicio solicitado. Por ejemplo: el puerto 80 se refiere a HTTP o al servicio Web. Cuando un cliente especifica el puerto 80 en el puerto de destino, el servidor que recibe el mensaje sabe que se solicitan servicios Web. Un servidor puede ofrecer más de un servicio simultáneamente. Por ejemplo, puede ofrecer servicios Web en el puerto 80 al mismo tiempo que ofrece el establecimiento de una conexión FTP en el puerto 21.



Puerto de origen

El número de puerto de origen es generado de manera aleatoria por el dispositivo emisor para identificar una conversación entre dos dispositivos. Esto permite establecer varias conversaciones simultáneamente. En otras palabras, un dispositivo puede enviar varias solicitudes de servicio HTTP a un servidor Web al mismo tiempo. El seguimiento de las conversaciones por separado se basa en los puertos de origen.

Sockets

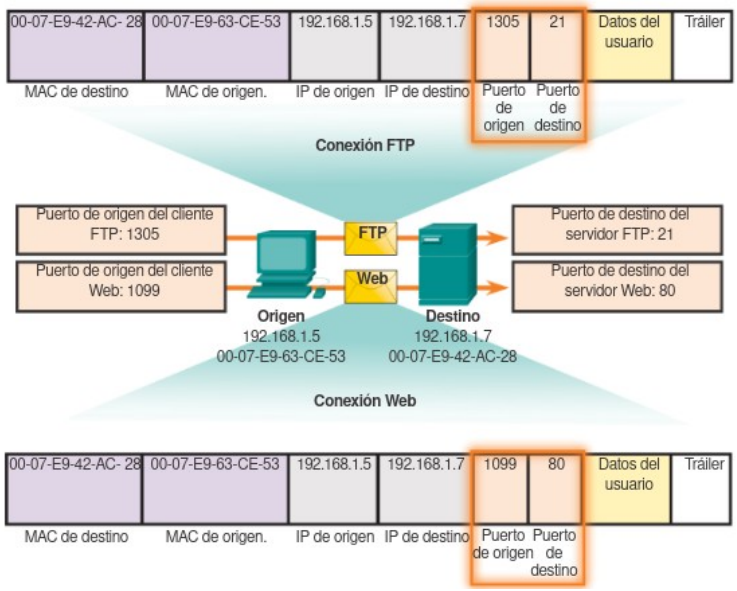
Los puertos de origen y de destino se colocan dentro del segmento. Los segmentos se encapsulan dentro de un paquete IP. El paquete IP contiene la dirección IP de origen y de destino. **La combinación de las direcciones IP de origen y de destino y de los números de puerto de origen y de destino se conoce como "socket"**. El socket se utiliza para identificar el servidor y el servicio que solicita el cliente. Miles de hosts se comunican a diario con millones de servidores diferentes. Los sockets identifican esas comunicaciones.

Un socket de cliente puede ser parecido a esto, donde 1099 representa el número de puerto de origen: 192.168.1.5:1099

El socket en un servidor Web podría ser el siguiente: 192.168.1.7:80

Juntos, estos dos sockets se combinan para formar un par de sockets: 192.168.1.5:1099, 192.168.1.7:80

Con la creación de sockets, se conocen los extremos de la comunicación, de modo que los datos puedan moverse desde una aplicación en un host hacia una aplicación en otro host. Los sockets permiten que los procesos múltiples que se ejecutan en un cliente se distingan entre sí. También permiten la diferenciación de múltiples conexiones a un proceso de servidor.



El puerto de origen de la solicitud de un cliente se genera de manera aleatoria. El número de puerto actúa como dirección de retorno para la aplicación que realiza la solicitud. La capa de transporte hace un seguimiento de este puerto y de la aplicación que generó la solicitud de manera que cuando se devuelva una respuesta, esta se envíe a la aplicación correcta. El número de puerto de la aplicación que realiza la solicitud se utiliza como número de puerto de destino en la respuesta que vuelve del servidor.

Puertos bien conocidos

La Agencia de asignación de números por Internet (IANA) asigna números de puerto. IANA es un organismo normativo responsable de asegurar diferentes estándares de direccionamiento.

Los puertos bien conocidos (números del 0 al 1023) se reservan para servicios y aplicaciones. Se utilizan comúnmente para aplicaciones como HTTP (servidor Web), protocolo de acceso a mensajes de Internet (IMAP) o protocolo simple de transferencia de correo (SMTP) (servidor de correo electrónico) y Telnet. Al definir estos puertos bien conocidos para las aplicaciones de los servidores, las aplicaciones cliente se pueden programar para solicitar una conexión a ese puerto en particular y el servicio relacionado.

Puertos registrados (números del 1024 al 49151): estos números de puerto se asignan a procesos o aplicaciones del usuario. Principalmente, estos procesos son aplicaciones individuales que el usuario elige instalar en lugar de aplicaciones comunes que recibiría un número de puerto bien conocido. Cuando no se utilizan para un recurso del servidor, un cliente puede seleccionar estos puertos de forma dinámica como su puerto de origen.

Puertos dinámicos o privados (números 49152 a 65535): también conocidos como puertos efímeros, generalmente se los asigna de forma dinámica a las aplicaciones cliente cuando el cliente inicia una conexión a un servicio. El puerto dinámico suele utilizarse para identificar la aplicación cliente durante la comunicación, mientras que el cliente utiliza el puerto bien conocido para identificar el servicio que se solicita en el servidor y conectarse a dicho servicio. No es común que un cliente se conecte a un servicio mediante un puerto dinámico o privado (aunque algunos programas de intercambio de archivos punto a punto lo hacen).

PUERTO	PROTOCOLO	APLICACIÓN	USO PRINCIPAL
0	TCP/UDP	Reservado	Puerto origen cuando no está definido.
20	TCP	FTP (datos)	Protocolo de transmisión de ficheros.
21	TCP	FTP (control)	Protocolo de transmisión de ficheros.
22	TCP	SSH	Terminal remoto seguro.
23	TCP	Telnet	Terminal remoto no seguro.
25	TCP	SMTP	Envío de correo electrónico.
53	UDP	DNS	Traducción de nombres de dominio a direcciones IP.
67	UDP	DHCP Server	Establecimiento automático de la configuración de acceso a la red.
68	UDP	DHCP Client	Establecimiento automático de la configuración de acceso a la red.
80	TCP	HTTP	Páginas y servicios web.
110	TCP	POP3	Acceso simple al buzón de correo electrónico.
123	UDP	NTP	Sincronización horaria.
135	TCP/UDP	NetBIOS (ns)	Permite establecer nombre, sesiones, compartir ficheros, etc. en redes NetBIOS. Muy utilizado en las redes Microsoft.
143	TCP	IMAP	Acceso sobre conexiones seguras.
443	TCP	HTTPS	HTTP sobre conexiones seguras.
445	TCP/UDP	Microsoft-ds	Servicios de directorio de MS Windows, similar a NetBIOS.

netstat

A veces es necesario conocer las conexiones TCP activas que están abiertas y en ejecución en el host de red. **Netstat** es una utilidad de red importante que puede usarse para verificar esas conexiones. Netstat indica el protocolo que se está usando, la dirección y el número de puerto locales, la dirección y el número de puerto externos y el estado de la conexión.

Las conexiones TCP desconocidas pueden presentar una amenaza de seguridad grave, ya que pueden indicar que hay algo o alguien conectado al host local. Además, las conexiones TCP innecesarias pueden consumir recursos valiosos del sistema y, por lo tanto, enlentecer el rendimiento del host. Netstat debe utilizarse para examinar las conexiones abiertas de un host cuando el rendimiento parece estar comprometido.

Establecimiento de conexiones TCP

Cuando dos hosts se comunican utilizando TCP, se establece una conexión antes de que puedan intercambiarse los datos. Luego de que se completa la comunicación, se cierran las sesiones y la conexión finaliza. Los mecanismos de conexión y sesión habilitan la función de confiabilidad de TCP.

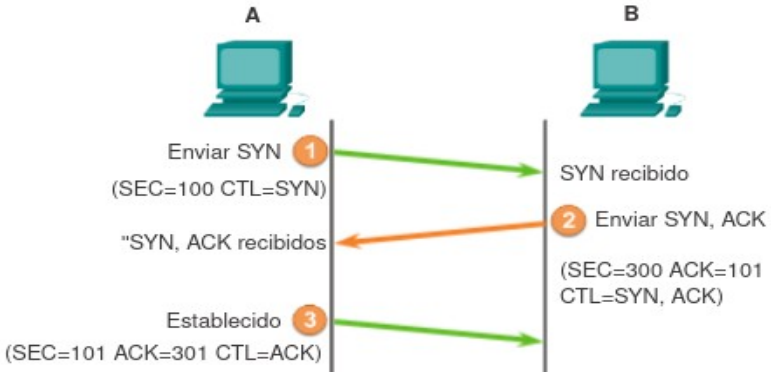
Los hosts hacen un seguimiento de cada segmento de datos dentro de una sesión e intercambian información sobre qué datos se reciben mediante la información del encabezado TCP. TCP es un protocolo full-duplex, en el que cada conexión representa dos streams de comunicación unidireccionales, o sesiones. Para establecer la conexión los hosts realizan un protocolo de enlace de tres vías. Los bits de control en el encabezado TCP indican el progreso y estado de la conexión. Enlace de tres vías:

En las conexiones TCP, el cliente del host establece la conexión con el servidor. Los tres pasos en el establecimiento de una conexión TCP son:

Paso 1. El cliente de origen solicita una sesión de comunicación de cliente a servidor con el servidor.

Paso 2. El servidor acusa recibo de la sesión de comunicación de cliente a servidor y solicita una sesión de comunicación de servidor a cliente.

Paso 3. El cliente de origen acusa recibo de la sesión de comunicación de servidor a cliente.



Para comprender el proceso de enlace de tres vías, observe los diversos valores que intercambian ambos hosts. Dentro del encabezado del segmento TCP, existen seis campos de 1 bit que contienen información de control utilizada para gestionar los procesos de TCP. Estos campos son los siguientes:

- **URG:** campo indicador urgente importante
- **ACK:** campo de acuse de recibo importante
- **PSH:** función de empuje
- **RST:** restablecer la conexión
- **SYN:** sincronizar números de secuencia
- **FIN:** no hay más datos del emisor

ACK o Confirmación de recepción de segmentos

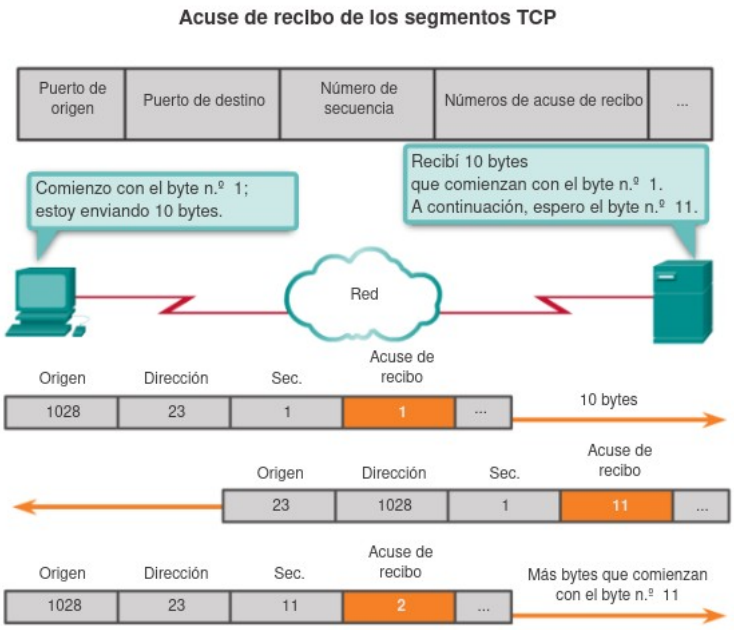
Una vez que se establece una sesión y que comienza la transferencia de datos, el destino envía acuses de recibo al origen por los segmentos que recibe. Estos acuses de recibo forman la base de la confiabilidad dentro de la sesión TCP. Cuando el origen recibe un acuse de recibo, reconoce que los datos se entregaron correctamente y puede dejar de rastrearlos. Si el origen no recibe el acuse de recibo dentro de un tiempo predeterminado, retransmite esos datos al destino.

Parte de la carga adicional que genera el uso de TCP es el tráfico de red generado por los acuses de recibo y las retransmisiones. El establecimiento de las sesiones genera sobrecarga en forma de segmentos adicionales que se intercambian. Hay también sobrecarga en los hosts indivuduales creada por la necesidad de mantener un registro de los segmentos que esperan un acuse de recibo y por el proceso de retransmisión.

Una de las funciones de TCP es garantizar que cada segmento llegue a destino. Los servicios de TCP en el host de destino envían un acuse de recibo de los datos que recibe la aplicación de origen.

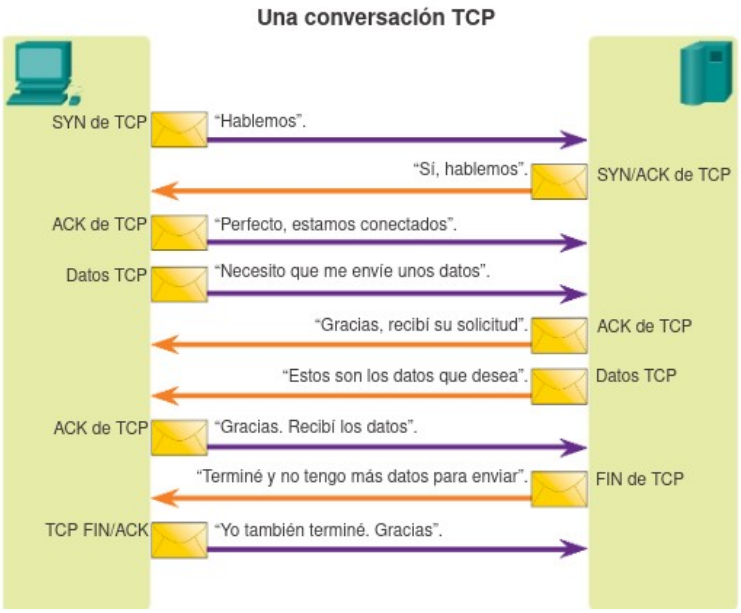
El número de secuencia (**SEQ**) y el número de acuse de recibo (**ACK**) se utilizan juntos para confirmar la recepción de los bytes de datos contenidos en los segmentos transmitidos. El número de SEQ indica la cantidad relativa de bytes que se transmitieron en esta sesión, incluso los bytes en el segmento actual. TCP utiliza el número de ACK reenviado al origen para indicar el próximo byte que el receptor espera recibir. Esto se llama acuse de recibo de expectativa. **Se le informa al origen que el destino recibió todos los bytes de este stream de datos, hasta el byte especificado por el número de ACK, pero sin incluirlo**. Se espera que el host emisor envíe un segmento que utiliza un número de secuencia que es igual al número de ACK.

Recuerda que cada conexión son realmente dos sesiones de una vía. Los números de SEQ y ACK se intercambian en ambas direcciones.



En el ejemplo de la figura, el host de la izquierda envía datos al host de la derecha. Envía un segmento que contiene 10 bytes de datos para esta sesión y un número de secuencia igual a 1 en el encabezado.

El host receptor recibe el segmento en la capa 4 y determina que el número de secuencia es 1 y que tiene 10 bytes de datos. Luego el host envía un segmento de vuelta al host de la izquierda para acusar recibo de estos datos. En este segmento, el host establece el número de ACK en 11 para indicar que el siguiente byte de datos que espera recibir en esta sesión es el byte número 11. Cuando el host emisor recibe este acuse de recibo, puede enviar el próximo segmento que contiene datos para esta sesión a partir del byte 11.



En este ejemplo, si el host emisor tuviera que esperar el acuse de recibo de cada uno de los 10 bytes, la red tendría mucha sobrecarga. Para reducir la sobrecarga de estos acuses de recibo, pueden enviarse varios segmentos de datos y dar acuse de recibo de estos con un único mensaje de TCP en la dirección opuesta. Este acuse de recibo contiene un número de ACK que se basa en la cantidad total de bytes recibidos en la sesión. Por ejemplo, si se comienza con un número de secuencia 2000, si se reciben 10 segmentos de 1000 bytes cada uno, se devolverá al origen un número de ACK igual a 12 001.

La cantidad de datos que un origen

puede transmitir antes de recibir un acuse de recibo se denomina “tamaño de la ventana”, que es un campo en el encabezado TCP que permite administrar datos perdidos y controlar el flujo.

Cuando el TCP en el host de origen no recibe un acuse de recibo después de una cantidad de tiempo predeterminada, este vuelve al último número de ACK recibido y vuelve a transmitir los datos desde ese punto en adelante. La solicitud de comentarios (RFC) no especifica el proceso de retransmisión, pero se deja a criterio de la implementación particular del TCP.

Control del flujo

TCP también proporciona mecanismos para el control del flujo. El control del flujo permite mantener la confiabilidad de la transmisión de TCP mediante el ajuste de la velocidad del flujo de datos entre el origen y el destino para una sesión dada. El control del flujo se logra limitando la cantidad de segmentos de datos que se envían al mismo tiempo y solicitando acuses de recibo antes de enviar más segmentos.

Para lograr el control del flujo, lo primero que determina TCP es la cantidad de segmentos de datos que puede aceptar el dispositivo de destino. El encabezado TCP incluye un campo de 16 bits llamado “**tamaño de la ventana**”. Esta es la cantidad de bytes que el dispositivo de destino de una sesión TCP puede aceptar y procesar al mismo tiempo. El tamaño inicial de la ventana se acuerda durante el inicio de sesión entre el origen y el destino por medio del protocolo de enlace de tres vías. Una vez acordado el tamaño, el dispositivo de origen debe limitar la cantidad de segmentos de datos enviados al dispositivo de destino sobre la base del tamaño de la ventana. El dispositivo de origen puede continuar enviando más datos para la sesión solo cuando obtiene un acuse de recibo de los segmentos de datos recibidos.

Durante el retraso en la recepción del acuse de recibo, el emisor no envía ningún otro segmento. En los períodos en los que la red está congestionada o los recursos del host receptor están exigidos, la demora puede aumentar. A medida que aumenta esta demora, disminuye la tasa de transmisión efectiva de los datos para esta sesión. La disminución de velocidad en la transmisión de datos de cada sesión ayuda a reducir el conflicto de recursos en la red y en el dispositivo de destino cuando se ejecutan varias sesiones.

En este ejemplo, el tamaño de la ventana inicial para una sesión TCP representada se establece en 3000 bytes. Cuando el emisor transmite 3000 bytes, espera por un acuse de recibo de los mismos antes de transmitir más segmentos para esta sesión. Una vez que el emisor obtiene este acuse de recibo del receptor, puede transmitir 3000 bytes adicionales.

TCP utiliza tamaños de ventana para tratar de aumentar la velocidad de transmisión hasta el flujo máximo que la red y el dispositivo de destino pueden admitir y, al mismo tiempo, minimizar las pérdidas y las retransmisiones.

Otra forma de controlar el flujo de datos es utilizar tamaños de ventana dinámicos. Cuando los recursos de la red son limitados el host receptor envía el valor del tamaño de la ventana al host emisor para indicar la cantidad de bytes que puede recibir. Si el destino necesita disminuir la velocidad de comunicación debido, por ejemplo, a una memoria de búfer limitada, puede enviar un valor más pequeño del tamaño de la ventana al origen como parte del acuse de recibo.