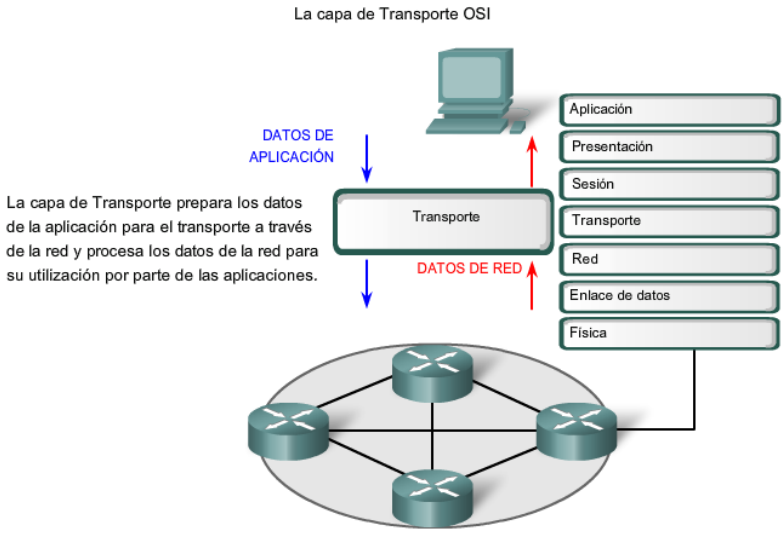
CAPA DE TRANSPORTE - FUNCIÓN DE LA CAPA DE TRANSPORTE



PROPOSITO DE LA CAPA DE TRANSPORTE

La capa de transporte permite la segmentación de datos y brinda el control necesario para reensamblar las partes dentro de los distintos streams de comunicación. Las responsabilidades principales que debe cumplir son:

- Rastreo de comunicación individual entre aplicaciones en los hosts de origen y destino
- Segmentación de datos y manejo de cada parte
- Reensamble de segmentos en streams de datos de aplicación
- Identificación de diferentes aplicaciones

Rastreo de conversaciones individuales: Cualquier host puede tener múltiples aplicaciones que se comunican a través de la red. Cada una de estas aplicaciones se comunicará con una o más aplicaciones en hosts remotos. Es responsabilidad de la capa de transporte mantener los streams de comunicación múltiple entre estas aplicaciones.

Segmentación de datos: Así como cada aplicación crea datos de stream para enviarse a una aplicación remota, estos datos se pueden preparar para enviarse a través de los medios en partes manejables. Los protocolos de la capa de transporte describen los servicios que segmentan estos datos de la capa de aplicación. Esto incluye la encapsulación necesaria en cada sección de datos. Cada sección de datos de aplicación requiere que se agreguen encabezados en la capa de transporte para indicar la comunicación a la cual está asociada.

Reensamble de segmentos: En el host de recepción, cada sección de datos se puede direccionar a la aplicación adecuada. Además, estas secciones de datos individuales también deben reconstruirse para generar un stream completo de datos que sea útil para la capa de aplicación. Los protocolos en la capa de transporte describen cómo se utiliza la información del encabezado de la capa para reensamblar las partes de los datos en streams para pasarlos a la capa de aplicación.

Identificación de aplicaciones: Para pasar streams de datos a las aplicaciones adecuadas, la capa de transporte debe identificar la aplicación meta. Para lograr esto, la capa de

transporte asigna un identificador a la aplicación. Los protocolos TCP/IP denominan a este identificador 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. Este número de puerto se utiliza en el encabezado de la capa de transporte para indicar qué aplicación se asocia a qué parte.

La capa de transporte es el enlace entre la capa de aplicación y la capa inferior que es responsable de la transmisión de la red. Esta capa acepta los datos de diferentes conversaciones y las pasa a las capas inferiores como partes manejables que se pueden multiplexar de forma eventual en la red.

Las aplicaciones no necesitan saber los detalles operativos de la red en uso. Las aplicaciones generan datos que se envían desde una aplicación a otra sin tener en cuenta el tipo de host destino, el tipo de medios sobre los que los datos deben viajar, el paso tomado por los datos, la congestión en un enlace o el tamaño de la red.

Además, las capas inferiores no tienen conocimiento de que existen varias aplicaciones que envían datos en la red. Su responsabilidad es entregar los datos al dispositivo adecuado. La capa de transporte clasifica entonces estas piezas antes de enviarlas a la aplicación adecuada.

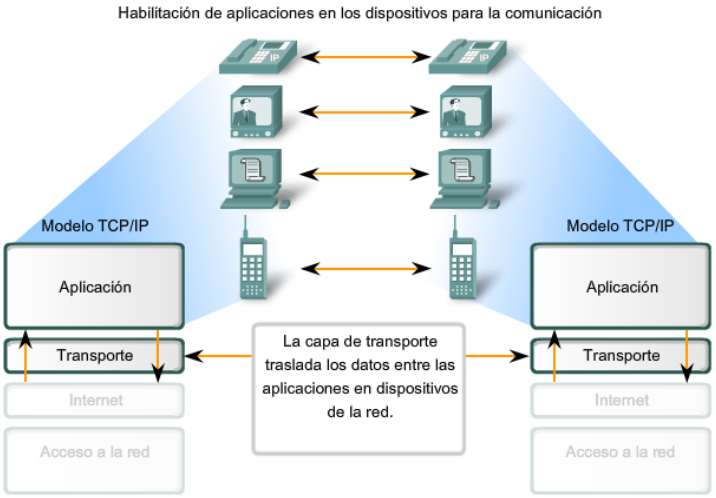
Los requisitos de datos varían

Hay múltiples protocolos de la capa de transporte debido a que las aplicaciones tienen diferentes requisitos. Para algunas aplicaciones, los segmentos deben llegar en una secuencia específica de manera que puedan ser procesados en forma exitosa. En algunos casos, todos los datos deben recibirse para ser utilizados por cualquiera de las mismas. En otros casos, una aplicación puede tolerar cierta pérdida de datos durante la transmisión a través de la red.

En las redes convergentes actuales, las aplicaciones con distintas necesidades de transporte pueden comunicarse en la misma red. Los diferentes protocolos de la capa de transporte poseen distintas reglas para permitir a los dispositivos manejar estos diversos requerimientos de datos.

Algunos protocolos proporcionan sólo las funciones básicas para enviar de forma eficiente partes de datos entre las aplicaicones adecuadas. Estos tipos de protocolos son útiles para aplicaciones cuyos datos son sensibles a retrasos.

Otros protocolos de la capa de transporte describen los procesos que proporcionan características adicionales, como asegurar un envío confiable entre las aplicaciones. Si bien estas funciones adicionales proveen una comunicación más sólida entre aplicaciones de la capa de transporte, representan la necesidad de utilizar recursos adicionales y generan un mayor número de demandas en la red.

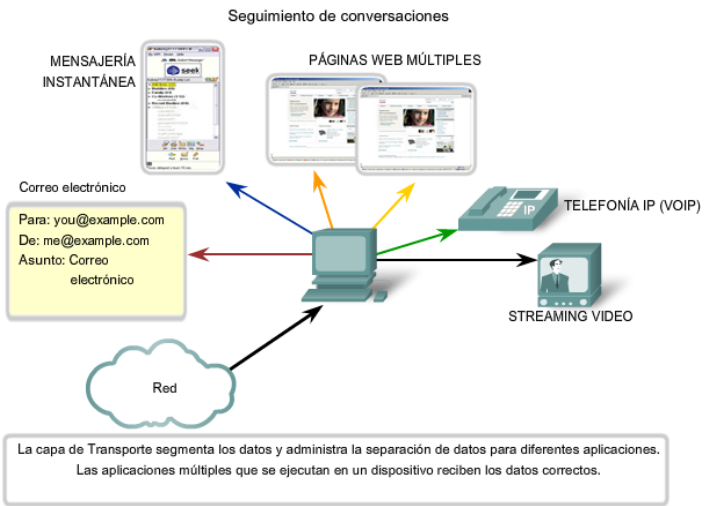


Separación de comunicaciones múltiples

Considere una computadora conectada a una red que recibe y envía correos electrónicos y mensajes instantáneos, explora sitios Web y realiza una llamada telefónica de VoIP de manera simultánea. Cada una de estas aplicaciones envía y recibe datos en la red al mismo tiempo. Sin embargo, los datos de la llamada telefónica no están dirigidos al explorador Web, y el texto de un mensaje instantáneo no aparece en el correo electrónico.

Además, los usuarios necesitan que el correo electrónico o página Web se reciba por completo y se presente para la información que se considere útil. Los retrasos ligeros se consideran aceptables para asegurar que la información se reciba y se presente por compelto.

En cambio, la pérdida ocasional de partes pequeñas de una conversación telefónica se puede considerar aceptable. Se puede inferir la parte de audio perdida del contexto de la conversación o se puede solicitar a la otra persona que repita lo que dijo. Es preferible esto último a las demoras que se producirían si se solicita a la red que gestione y vuelva a enviar los segmentos perdidos. En este ejemplo, el usuario, no la red, gestiona el reenvío o reemplazo de información que falta.



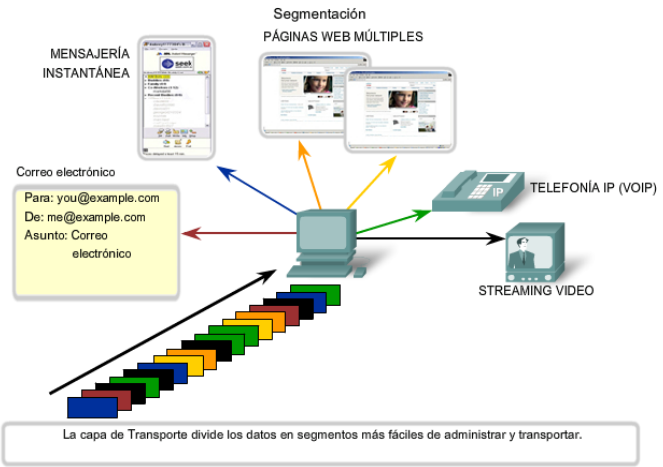
El envío de algunos tipos de datos, un video por ejemplo, a través de la red como un stream de comunicación completa puede impedir que se produzcan otras comunicaciones al mismo tiempo. También dificulta la recuperación de errores y la retransmisión de datos dañados.

Dividir datos en pequeñas partes y enviarlas del origen al destino permite que muchas comunicaciones diferentes se intercalen (multiplexadas) en la misma red.

La segmentación de los datos, que cumple con los protocolos de la capa de transporte, proporciona los medios para enviar y recibir datos cuando se ejecutan varias aplicaciones de manera concurrente en una computadora. Sin segmentación, sólo una aplicación, la corriente de video por ejemplo, podría recibir datos. Puede no recibir correos electrónicos, chatear en mensajería instantánea o ver páginas Web mientras ve un video.

En la capa de transporte, cada conjunto de piezas particular que fluye entre la aplicación de origen y la de destino se conoce como conversación.

Para identificar cada segmento de datos, la capa de transporte añade a la pieza un encabezado que contiene datos binarios. Este encabezado contiene campos de bits. Son los valores de estos campos los que permiten que los distintos protocolos de la capa de transporte lleven a cabo las diversas funciones.



CONTROL DE LAS CONVERSACIONES

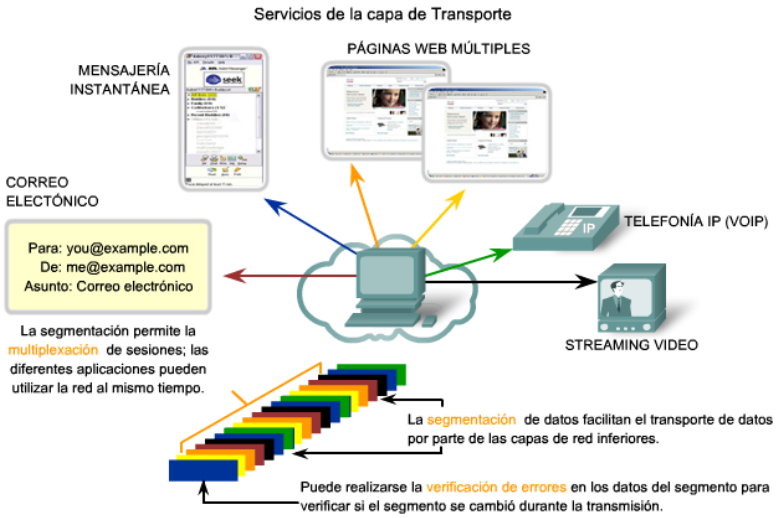
Las funciones principales que especifican los protocolos de la capa de transporte incluyen:

Segmentación y reensamble: la mayoría de las redes tienen una limitación en la cantidad de datos que se pueden incluir en una simple PDU. La capa de transporte divide los datos de aplicación en bloques de datos de un tamaño adecuado. En el destino, la capa de transporte reensambla los datos antes de enviarlos a la aplicación o servicio de destino.

Multiplexación de conversación: puede haber aplicaciones o servicios que se ejecutan en cada host de la red. A cada una de estas aplicaciones o servicios se les asigna una dirección conocida como puerto, de manera que la capa de transporte determina con qué aplicación o servicio se identifican los datos.

Además de utilizar la información contenida en los encabezados, para las funciones básicas de segmentación y reensamble de datos algunos protocolos en la capa de transporte proporcionan:

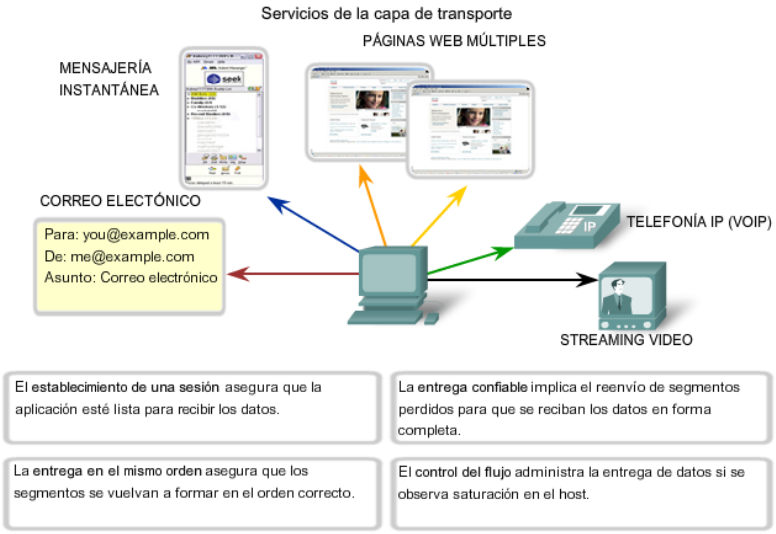
- Conversaciones orientadas a la conexión
- Entrega confiable
- Reconstrucción de datos ordenada
- Control del flujo



Establecimiento de una sesión: La capa de transporte puede brindar esta orientación a la conexión creando una sesión entre las aplicaciones. Estas conexiones preparan las aplicaciones para que se comuniquen entre sí antes de que se transmitan los datos. Dentro de estas sesiones, se pueden gestionar de cerca los datos para la comunicación entre dos aplicaciones.

Entrega confiable: 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. La capa de transporte puede asegurar que todas las partes alcancen su destino haciendo que el dispositivo origen retransmita todos los datos perdidos.

Entrega en el mismo orden: Los datos pueden llegar en el orden equivocado, debido a que las redes pueden proporcionar múltiples rutas que pueden tener diferentes tiempos de transmisión. Al numerar y secuenciar los segmentos, la capa de transporte puede asegurar que los mismos se reensamblen en el orden adecuado.



Control del flujo: Los hosts de la red cuentan con recursos limitados, como memoria o ancho de banda. Cuando la capa de transporte advierte que estos recursos están sobrecargados, algunos protocolos pueden solicitar que la aplicación que envía reduzca la velocidad del flujo de datos. Esto se lleva a cabo en la capa de transporte regulando la cantidad de datos que el origen transmite como grupo. El control de flujo puede evitar la pérdida de segmentos en la red y evitar la necesitad de la retransmisión.

SOPORTE DE COMUNICACIONES CONFIABLES

Cabe recordar que la función principal de la capa de transporte es administrar los datos de aplicación para las conversaciones entre hosts. Sin embargo, cada aplicación tiene determinados requisitos para sus datos y, por lo tanto, se han desarrollado diferentes protocolos de transporte para que cumplan con estos requisitos.

Un protocolo de la capa de transporte puede implementar un método para asegurar el envío confiable de datos. En términos de redes, confiabilidad significa asegurar que cada sección de datos que envía el origen llegue al destino. En la capa de transporte, las tres operaciones básicas de confiabilidad son:

- rastreo de datos transmitidos
- acuse de recibo de datos recibidos
- retransmisión de cualquier dato sin acuse de recibo

Esto requiere que los procesos de la capa de transporte en el origen dé seguimiento a todas las partes de datos de cada conversación y retransmitan cualquier dato del cual el destino no acuso recibo. La capa de transporte del host de recepción también debe rastrear los datos a medida que se reciben y reconocer la recepción de los mismos.

Estos procesos de confiabilidad generan un uso adicional de los recursos de la red debido al reconocimiento, rastreo y retransmisión. Para admitir estas operaciones de confiabilidad se intercambian más datos de control entre los hosts emisores y receptores. Esta información de control se encuentra en el encabezado de la Capa 4.

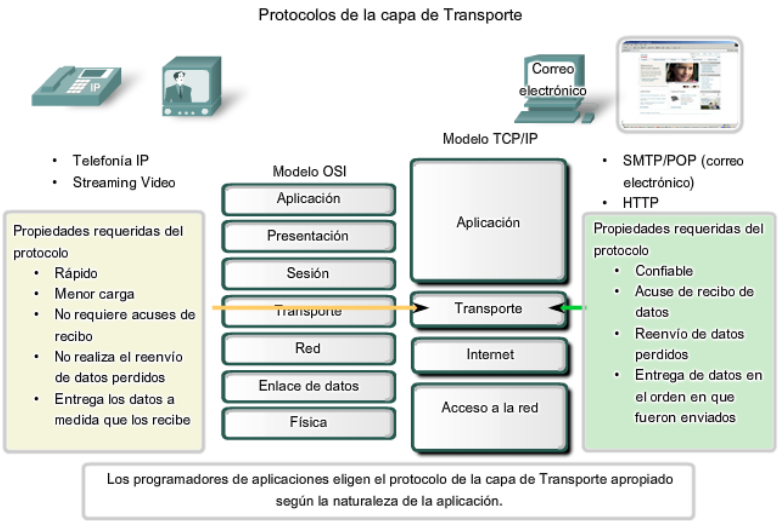
Esto crea una compensación entre el valor de la confiabilidad y la carga que hay en la red. Los desarrolladores de aplicaciones deben elegir qué tipo de protocolo de transporte es adecuado en base a los requerimientos de sus aplicaciones. En la capa de transporte, existen protocolos que especifican métodos para la entrega confiable, garantizada o de

máximo esfuerzo. En el contexto de networking, el envío del mejor esfuerzo se conoce como poco confiable, porque no hay acuse de recibo de que los datos se recibieron en el destino.

Determinación de la necesidad de confiabilidad: Las aplicaciones, tales como bases de datos, páginas Web y correo electrónico, necesitan que todos los datos enviados lleguen al destino en su condición original para que los datos sean útiles. Todos los datos perdidos pueden corromper una comunicación y dejarla incompleta o ilegible. Por lo tanto, estas aplicaciones se diseñan para utilizar un protocolo de capa de transporte que implemente la confiabilidad. Los gastos de red adicionales se consideran necesarios para estas aplicaciones.

Otras aplicaciones son más tolerantes a la pérdida de cantidades pequeñas de datos. Por ejemplo, si uno o dos segmentos de un stream de vídeo no llegan al destino, sólo generará una interrupción momentánea en el stream. Esto puede representar distorsión en la imagen pero quizás ni sea advertido por el usuario.

Imponer el uso de recursos adicionales para asegurar la confiabilidad para esta aplicación puede reducir la utilidad de la misma. La imagen en un streaming video se degradaría en gran medida si el dispositivo de destino tuvo que dar cuenta de los datos perdidos y demorar el stream mientras espera que lleguen. Es conveniente proporcionar la mejor imagen posible al momento en que llegan los segmentos y renunciar a la confiabilidad. Si por algún motivo se requiere confiabilidad, estas aplicaciones pueden proveer verificación de errores y solicitudes de retransmisión.



TCP y UDP

Los dos protocolos más comunes de la capa de transporte del conjunto de protocolos TCP/IP son el Protocolo de control de transmisión (TCP) y el Protocolo de datagramas de usuario (UDP). Ambos protocolos gestionan la comunicación de múltiples aplicaciones. Las diferencias entre ellos son las funciones específicas que cada uno implementa.

Protocolo de datagramas de usuario (UDP)

UDP es un protocolo simple, sin conexión, descrito en la RFC 768. Cuenta con la ventaja de proveer la entrega de datos sin utilizar muchos recursos. Las porciones de comunicación en UDP se llaman datagramas. Este protocolo de la capa de transporte envía estos datagramas como "mejor intento".

Las aplicaciones que utilizan UDP incluyen:
- Sistema de nombres de dominio (DNS)
- Streaming video
- Voz sobre IP (VOIP)

Protocolo de control de transmisión (TCP)
TCP es un protocolo orientado a la conexión descrito en RFC 793. El TCP utiliza recursos adicionales para ganar funciones. Las funciones adicionales especificadas por TCP están en el mismo orden de entrega, son de entrega confiable y de control de flujo. Cada segmento de TCP posee 20 bytes de carga en el encabezado que encapsulan los datos de la capa de aplicación, mientras que cada segmento UDP sólo posee 8 bytes de carga. Vea la figura para hacer una comparación.

Las aplicaciones que utiliza el TCP son:
- Exploradores Web
- Correo electrónico
- Transferencias de archivos

DIRECCIONAMIENTO DE PUERTO
Identificación de conversaciones

Considere el ejemplo anterior de una computadora que recibe y envía correos electrónicos, mensajes instantáneos, páginas Web y llamadas telefónicas VoIP de manera simultánea.

Los servicios basados en TCP y UDP mantienen 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. El número de puerto de origen es el número para esta comunicación asociado con la aplicación que origina la comunicación en el host local. El número de puerto de destino es el número para esta comunicación asociado con la aplicación de destino que origina la comunicación en el host local.

Los números de puerto se asignan de distintas maneras, en virtud de si el mensaje es una solicitud o una respuesta. Mientras que los procesos del servidor tienen números de puerto estáticos asignados, los clientes eligen de forma dinámica un número de puerto para cada conversación.

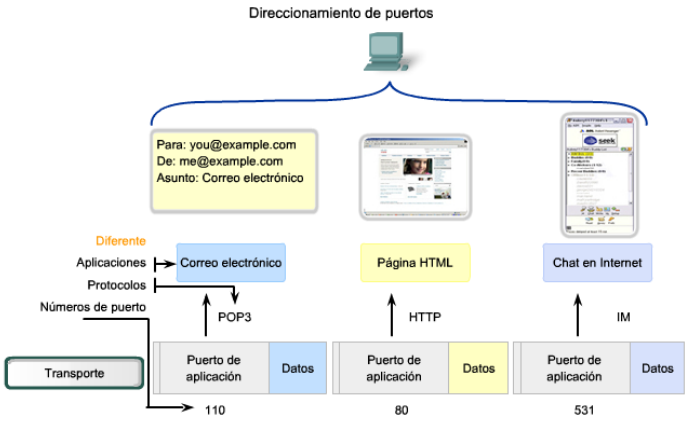
Cuando una aplicación de cliente envía una solicitud a una aplicación de servidor, el puerto de destino contenido en el encabezado es el número de puerto asignado al demonio de servicio se ejecuta en el host remoto. El software del cliente debe conocer el número de puerto asociado con el proceso del servidor en el host remoto. Este número de puerto de destino se puede configurar, ya sea de forma predeterminada o manual. Por ejemplo, cuando una aplicación de explorador Web realiza una solicitud a un servidor Web, el explorador utiliza TCP y el número de puerto 80 a menos que se especifique otro valor. Esto sucede porque el puerto TCP 80 es el puerto predeterminado asignado a aplicaciones de servidores Web. Muchas aplicaciones comunes tienen asignados puertos predeterminados.

El puerto de origen en el encabezado de un segmento o de un datagrama de la solicitud de un cliente se crea de forma aleatoria desde los números de puerto mayores de 1023. Mientras no haya un conflicto con otros puertos en uso en el sistema, el cliente puede elegir cualquier número de puerto del rango de números predeterminados que utiliza el sistema operativo. El número de puerto actúa como dirección de retorno para la aplicación que realiza la solicitud. La capa de transporte mantiene un seguimiento de este puerto y de la aplicación que generó la solicitud, de manera que cuando se devuelva una respuesta, se pueda ser enviar 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.

La combinación del número de puerto de la capa de transporte y de la dirección IP de la capa de red asignada al host identifica de manera exclusiva un proceso en particular que se ejecuta en un dispositivo host específico. Esta combinación se denomina socket. Eventualmente, los términos número de puerto y socket se utilizan en forma indistinta. En el contexto de este curso, el término socket hace referencia sólo a la combinación exclusiva de dirección IP y número de puerto. Un par de sockets, que consiste en las direcciones IP de origen y destino y los números de puertos, también es exclusivo e identifica la conversación entre los dos hosts.

Por ejemplo, una solicitid de página Web HTTP que se envía a un servidor Web (puerto 80) que se ejecuta en un host con una dirección IPv4 de Capa 3 de 192.168.1.20 se destinaría al socket 192.168.1.20:80.

Si el explorador Web que solicita una página Web se ejecuta en el host 192.168.100.48 y el número de puerto dinámico que se asignó al explorador es 49152, el socket para la página Web sería 192.168.100.48:49152.



La Autoridad de números asignados de Internet (IANA) asigna números de puerto. IANA es un organismo normativo responsable de asegurar diferentes estándares de direccionamiento.

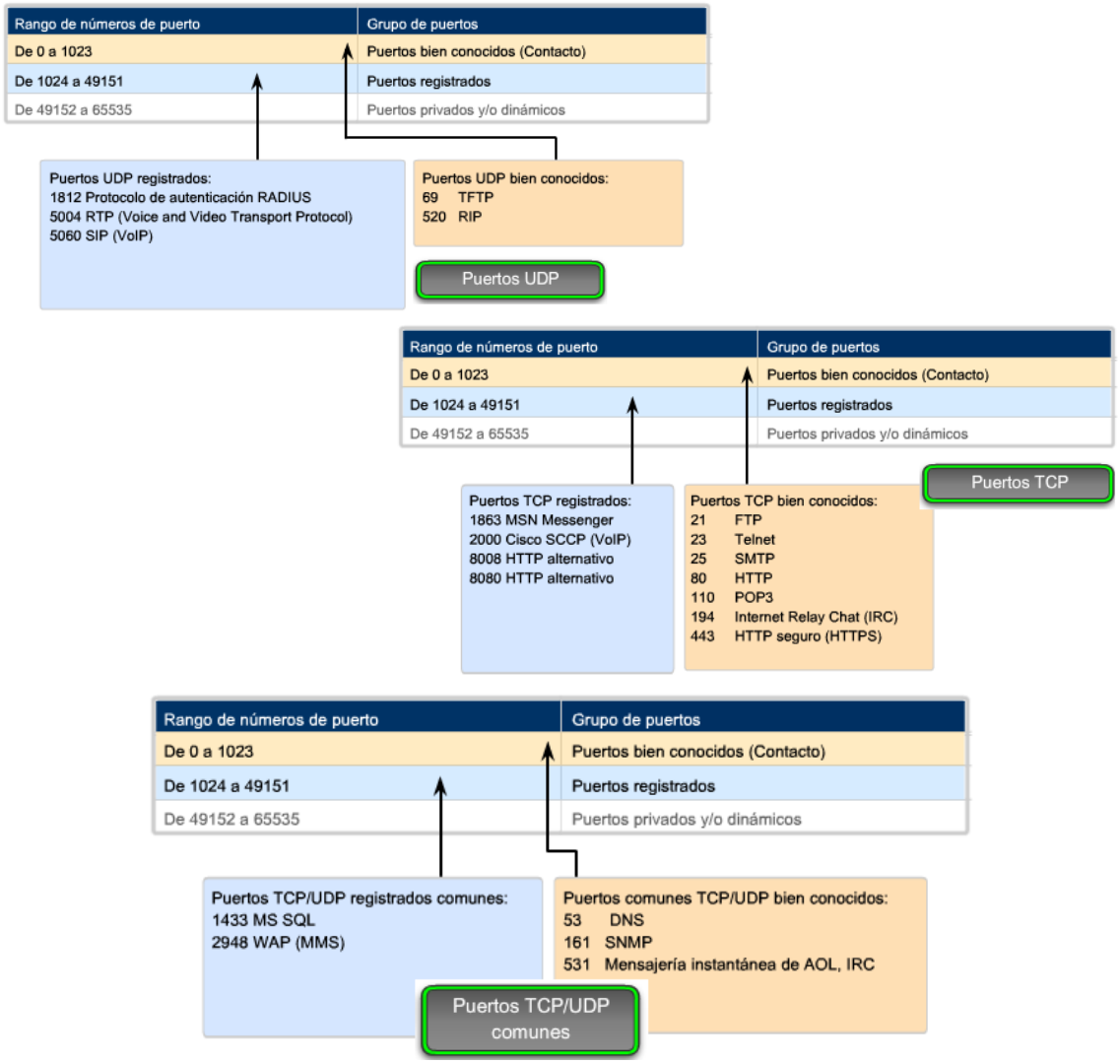
Hay diversos tipos de números de puerto:
Puertos bien conocidos (números del 0 al 1023): estos números se reservan para servicios y aplicaciones. Por lo general, se utilizan para aplicaciones como HTTP (servidor Web), POP3/SMTP (servidor de correo electrónico) y Telnet. Al definir estos puertos bien conocidos para las aplicaciones de los servidores, las aplicaciones cliente se pueden programar para solicitar una conexión a dicho puerto y su servicio asociado.

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

Puertos dinámicos o privados (números 49152 a 65535): también conocidos como puertos efímeros, están usualmente asignados de forma dinámica a las aplicaciones cliente cuando se inicia una conexión. No es muy común que un cliente se conecte a un servicio utilizando un puerto dinámico o privado (aunque algunos programas que comparten archivos punto a punto lo hacen).

Uso de TCP y UDP
Algunas aplicaciones pueden utilizar ambos. Por ejemplo, el bajo gasto de UDP permite que DNS atienda rápidamente varias solicitudes de clientes. Sin embargo, a veces el envío de la información solicitada puede requerir la confiabilidad de TCP. En este caso, el número de puerto bien conocido de 53 lo utilizan ambos protocolos con este servicio.

Enlaces
Una lista actual de números de puerto se puede encontrar en
http://www.iana.org/assignments/port-numbers.

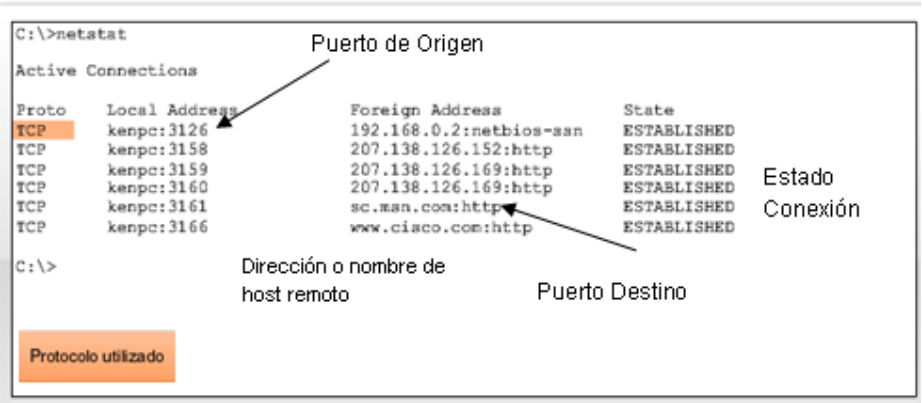


A veces es necesario conocer las conexiones TCP activas que están abiertas y en ejecución en el host de red. Netstat es una utilidad de red importante que puede usarse para verificar esas conexiones. Netstat indica el protocolo en uso, la dirección y el número de puerto locales, la dirección y el número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

Las conexiones TCP no descritas pueden representar una importante amenaza a la seguridad. Esto se debe a que pueden indicar que algo o alguien está conectado al host local. Además, las conexiones TCP innecesarias pueden consumir recursos valiosos del sistema y por lo tanto disminuir el rendimiento del host. Netstat debe utilizarse para determinar las conexiones abiertas de un host cuando el rendimiento parece estar comprometido.

Existen muchas opciones útiles para el comando netstat.

Resultado de netstat



SEGMENTACIÓN Y REENSAMBLAJE: DIVIDE Y VENCERÁS

Algunas aplicaciones transmiten grandes cantidades de datos, en algunos casos muchos gigabytes. Resultaría poco práctico enviar todos estos datos en una sola gran sección. No puede transmitirse ningún otro tráfico de red mientras se envían estos datos. Una gran sección de datos puede tardar minutos y hasta horas en enviarse. Además, si hubiera algún error, el archivo de datos completo se perdería o tendría que ser reenviado. Los dispositivos de red no cuentan con buffers de memoria lo suficientemente grandes como para almacenar esa cantidad de datos durante la transmisión o recepción. El límite varía dependiendo de la tecnología de networking y de medio físico específico a utilizar.

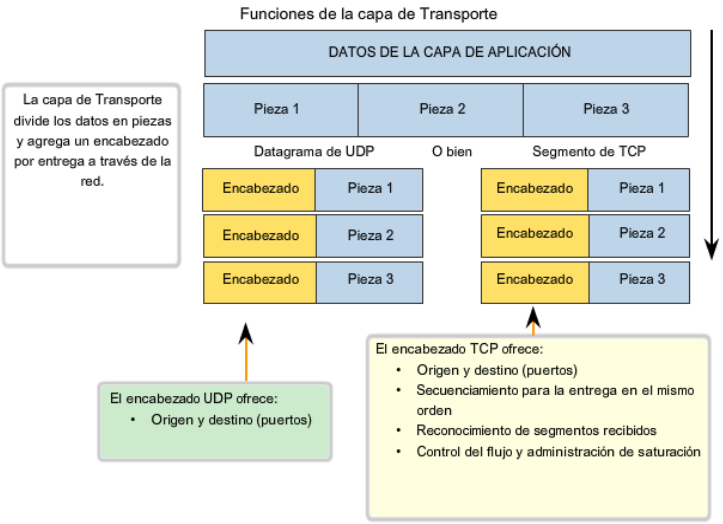
Dividir los datos de la aplicación en partes asegura que éstos se transmitan dentro de los límites de los medios y que se puedan multiplexar en el medio.

Segmentación diferente para el manejo de TCP y UDP.

En TCP, cada encabezado de segmento contiene un número de secuencia. Este número de secuencia permite que las funciones de la capa de transporte del host de destino reensamblen los segmentos en el mismo orden en el cual se transmitieron. Esto asegura que la aplicación de destino tiene los datos en la misma forma que el emisor la planeó.

Aunque los servicios de UDP rastrean también las conversaciones entre las aplicaciones, no están preocupados por el orden en que se transmite la información o por mantener una conexión. No existe número de secuencia en el encabezado UDP. UDP es un diseño simple y genera menos carga que TCP, lo que produce una transferencia de datos más rápida.

La información puede llegar en un orden distinto al que fue transmitida, ya que los paquetes pueden tomar diversas rutas a través de la red. Una aplicación que utiliza UDP debe tolerar el hecho de que los datos no lleguen en el orden en que se enviaron.



PROTOCOLO TCP: COMUNICACIÓN CON CONFIABILIDAD

TCP: Conversación Confiables

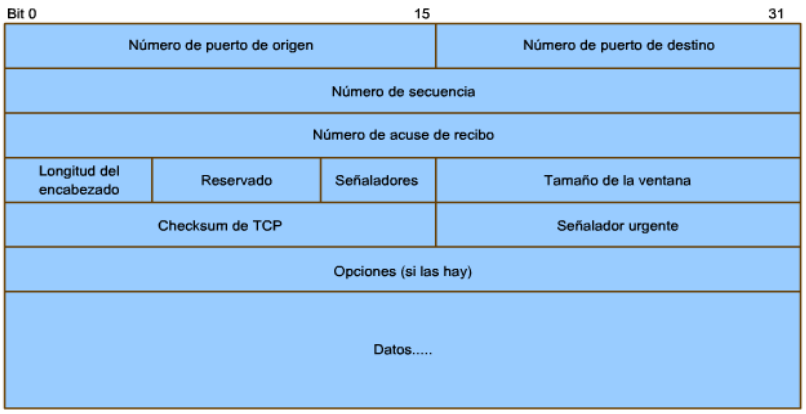
La diferencia clave entre TCP y UDP es la confiabilidad. La confiabilidad de la comunicación TCP se lleva a cabo utilizando sesiones orientadas a la conexión. Antes de que un host que utiliza TCP envíe datos a otro host, la capa de transporte inicia un proceso para crear una conexión con el destino. Esta conexión permite el rastreo de una sesión, o stream de comunicación entre los hosts. Este proceso asegura que cada host tenga conocimiento de la comunicación y se prepare. Una conversación completa de TCP necesita establecer una sesión entre los hosts de ambas direcciones.

Después de establecer una sesión, el destino envía un acuse 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 han entregado con éxito y puede dejar de rastrearlos. Si el origen no recibe el acuse de recibo dentro de un tiempo predeterminado, retransmite esos datos al destino.

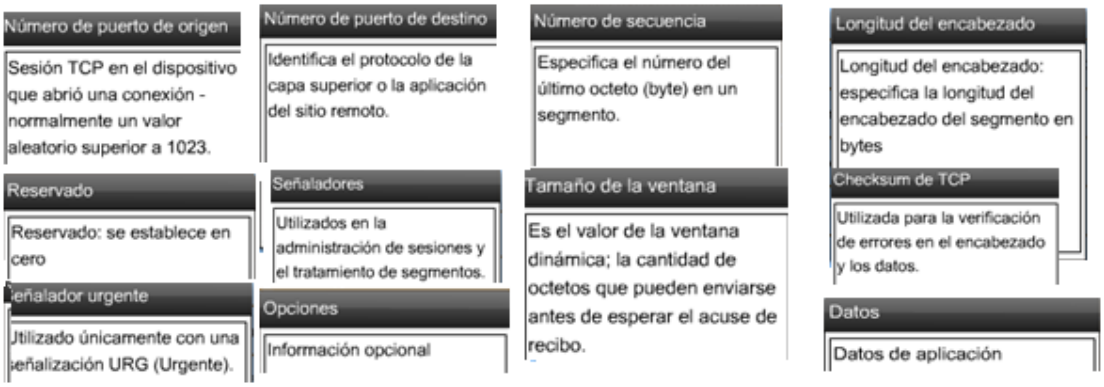
Parte de la carga adicional que genera el uso de TCP es el tráfico de red generado por los acuses de recibo y las retransmisiones. El establecimiento de las sesiones genera cargas en forma de segmentos adicionales intercambiados. Hay también sobrecarga en los hosts indivuduales creada por la necesidad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

Esta confiabilidad se logra al tener archivos en el segmento TCP, cada uno con su función específica, como se muestra en la figura. Estos campos se explicarán más adelante en esta sección.

Campos del encabezado del segmento de TCP



Los campos del encabezado de TCP habilitan TCP para suministrar comunicaciones de datos confiables orientados a la comunicación.



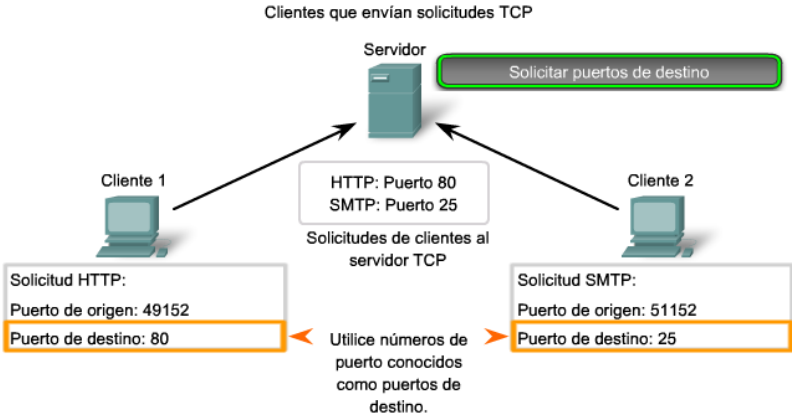
PROCESO DEL SERVIDOR TCP

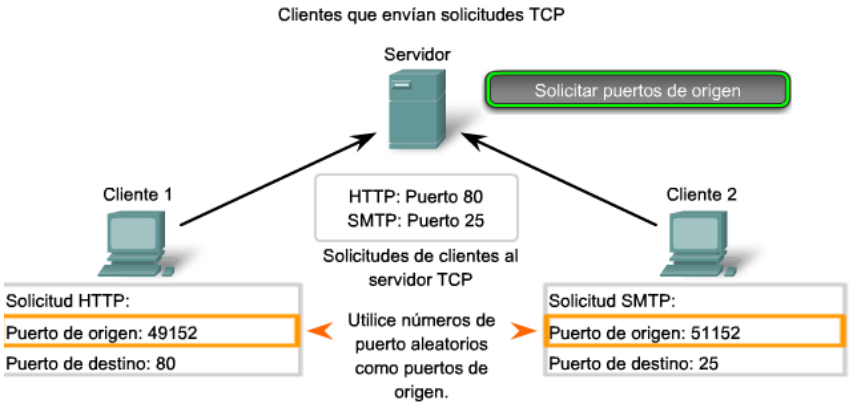
Los procesos de aplicación se ejecutan en servidores. Estos procesos esperan hasta que el cliente inicia comunicación con una solicitud de información u otros servicios.

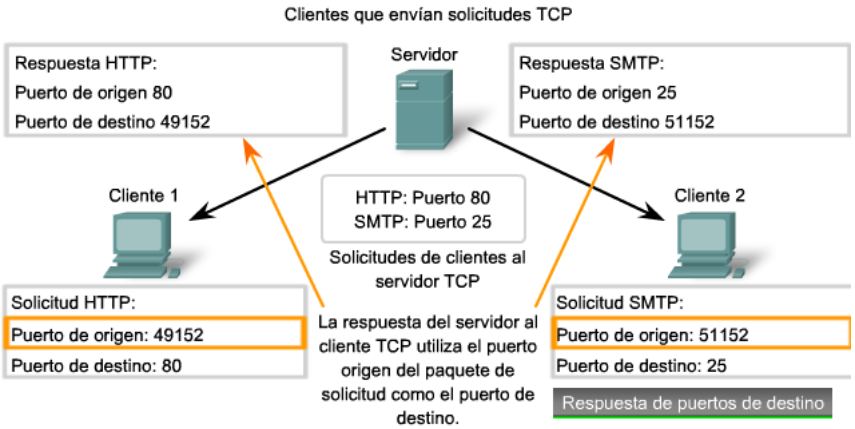
Cada proceso de aplicación que se ejecuta en el servidor se configura para utilizar un número de puerto, ya sea predeterminado o de forma manual por el administrador del sistema. Un servidor individual no puede tener dos servicios asignados al mismo número de puerto dentro de los mismos servicios de la capa de transporte. Un host que ejecuta una aplicación de servidor Web y una de transferencia de archivos no puede configurar ambas para utilizar el mismo puerto (por ejemplo, el puerto TCP 8080). Cuando una aplicación de servidor activa se asigna a un puerto específico, este puerto se considera "abierto" para el servidor. Esto significa que la capa de transporte acepta y procesa segmentos direccionados a ese puerto. Toda solicitud entrante de un cliente direccionada al socket correcto es aceptada y los datos se envían a la aplicación del servidor. Pueden existir varios puertos simultáneos abiertos en un servidor, uno para cada aplicación de servidor activa. Es común para un servidor proporcionar más de un servicio, tal como un servidor Web y un servidor FTP, al mismo tiempo.

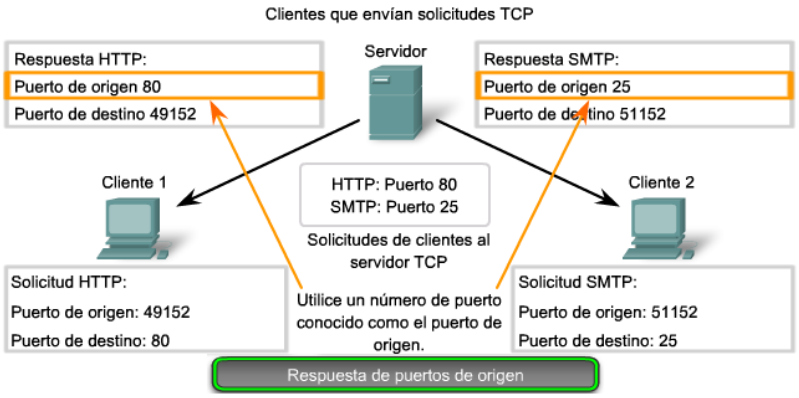
Una manera de mejorar la seguridad en un servidor es restringir el acceso al servidor únicamente a aquellos puertos asociados con los servicios y aplicaciones que deberían estar accesibles para los solicitantes autorizados.

La figura muestra la asignación típica de puertos de origen y destino en operaciones de cliente o servidor TCP.


Clientes que envían solicitudes TCP
Servidor
Solicitar puertos de destino
Cliente 1
HTTP: Puerto 80
SMTP: Puerto 25
Solicitudes de clientes al servidor TCP
Cliente 2
Solicitud HTTP:
Puerto de origen: 49152
Puerto de destino: 80
Utilice números de puerto conocidos como puertos de destino.
Solicitud SMTP:
Puerto de origen: 51152
Puerto de destino: 25


Clientes que envían solicitudes TCP
Servidor
Solicitar puertos de origen
Cliente 1
HTTP: Puerto 80
SMTP: Puerto 25
Solicitudes de clientes al servidor TCP
Cliente 2
Solicitud HTTP:
Puerto de origen: 49152
Puerto de destino: 80
Utilice números de puerto aleatorios como puertos de origen.
Solicitud SMTP:
Puerto de origen: 51152
Puerto de destino: 25


Clientes que envían solicitudes TCP
Respuesta HTTP:
Puerto de origen 80
Puerto de destino 49152
Servidor
Respuesta SMTP:
Puerto de origen 25
Puerto de destino 51152
Cliente 1
HTTP: Puerto 80
SMTP: Puerto 25
Solicitudes de clientes al servidor TCP
Cliente 2
Solicitud HTTP:
Puerto de origen: 49152
Puerto de destino: 80
La respuesta del servidor al cliente TCP utiliza el puerto origen del paquete de solicitud como el puerto de destino.
Solicitud SMTP:
Puerto de origen: 51152
Puerto de destino: 25
Respuesta de puertos de destino



ESTABLECIMIENTO Y FINALIZACIÓN DE LA CONEXIÓN TCP

Cuando dos hosts se comunican mediante 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 del TCP.

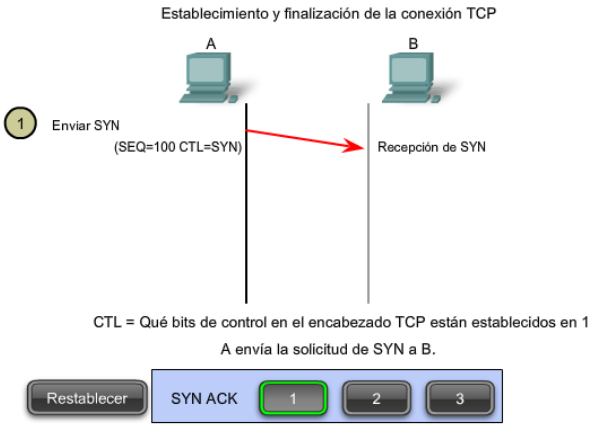
El host rastrea cada segmento de datos dentro de una sesión e intercambia información sobre los datos que recibe cada host mediante información en el encabezado del TCP.

Cada conexión involucra streams de comunicación de una vía, o sesiones para establecer y terminar el proceso del TCP entre dispositivos finales. 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. El enlace de tres vías:

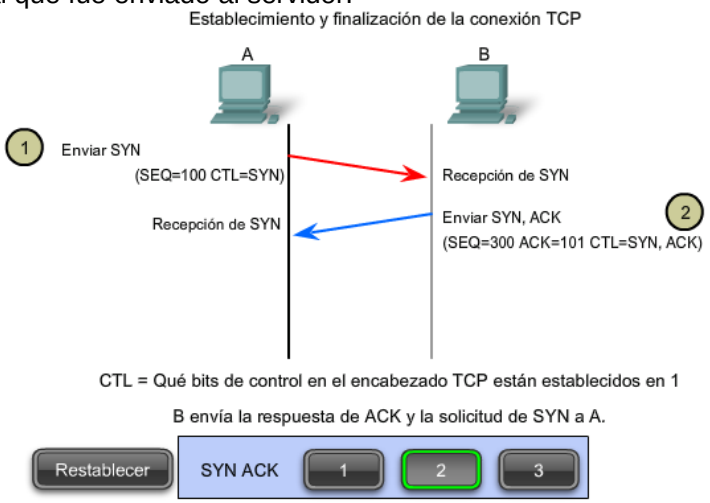
- Establece que el dispositivo de destino se presente en la red
- Verifica que el dispositivo de destino tenga un servicio activo y que acepte solicitudes en el número de puerto de destino que el cliente de origen intenta utilizar para la sesión
- Informa al dispositivo de destino que el cliente de origen intenta establecer una sesión de comunicación en dicho número de puerto

En las conexiones del TCP, el host que sirve como cliente inicia la sesión para el servidor. Para entender cómo funciona el enlace de tres vías que se utiliza en el proceso de conexión del TCP, es importante observar diversos valores que los dos hosts intercambian. Los tres pasos en el establecimiento de una conexión TCP son:

1. El cliente de origen envía un segmento que contiene un valor de secuencia inicial, el cual sirve como solicitud para que el servidor comience una sesión de comunicación.



2. El servidor responde con un segmento que contiene un valor de reconocimiento igual al valor de secuencia recibido más 1, más su propio valor de secuencia de sincronización. El valor es uno mayor que el número de secuencia porque el ACK es siempre el próximo Byte u Octeto esperado. Este valor de reconocimiento permite al cliente unir la respuesta al segmento original que fue enviado al servidor.



3. El cliente que inicia la conexión responde con un valor de reconocimiento igual al valor de secuencia que recibió más uno. Esto completa el proceso de establecimiento de la conexión.

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 reconocimiento importante

PSH: función de pulsación

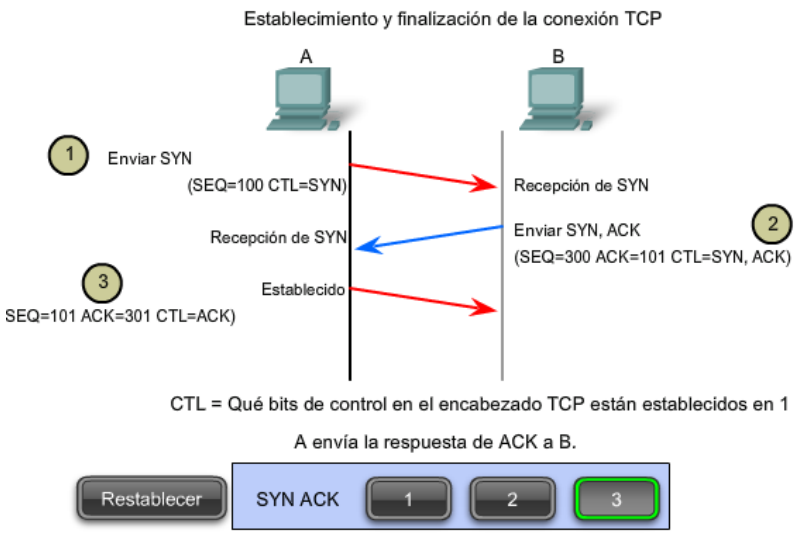
RST: restablecer la conexión

SYN: sincronizar números de secuencia

FIN: no hay más datos del emisor

Se hace referencia a estos campos por medio de señaladores, porque el valor de uno de estos campos es sólo 1 bit y, por lo tanto, sólo tiene dos valores: 1 o 0. Cuando el valor de un bit se establece en 1, indica qué información de control se incluye en el segmento.

Los señaladores se intercambian para terminar una conexión del TCP mediante un proceso de cuatro pasos.



PROTOCOLO TCP DE ENLACE DE TRES VÍAS

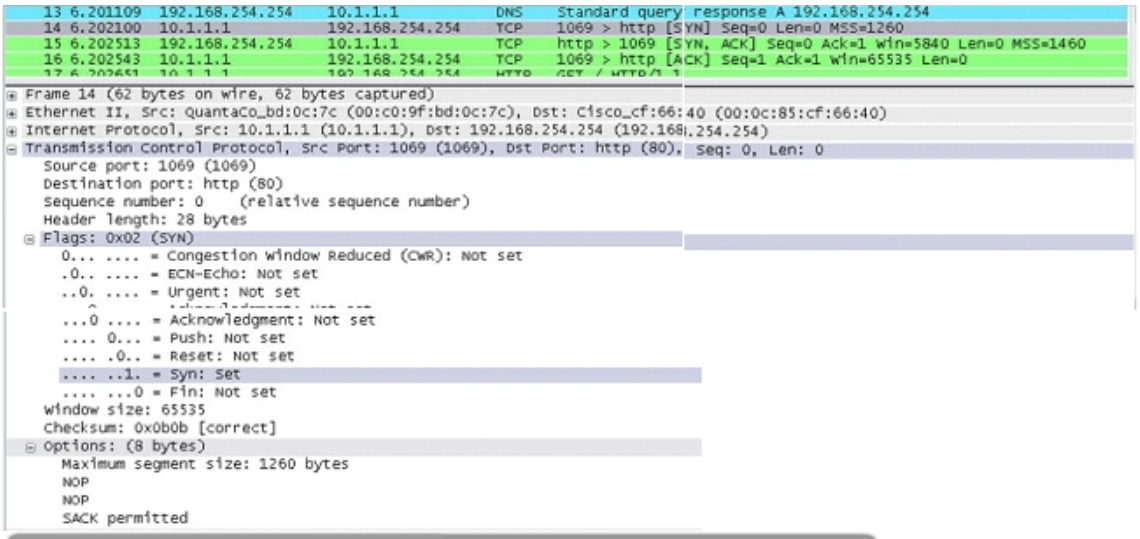
Puede revisar la operación del enlace de tres vías del TCP por medio de los resultados del Wireshark:

Paso 1: Un cliente del TCP inicia un enlace de tres vías al enviar un segmento con el señalador de control de SYN activo, lo que indica que un valor inicial en el campo de número de secuencia en el encabezado. Este valor inicial para el número de secuencia, conocido como número de secuencia inicial (ISN), se elige de manera aleatoria y se utiliza para comenzar a rastrear el flujo de datos desde el cliente al servidor para esta sesión. El ISN en el encabezado de cada segmento se incrementa en uno por cada byte de datos enviados desde el cliente hacia el servidor mientras continúa la conversación de datos.

Como se muestra en la figura, el resultado de un analizador de protocolos muestra el señalizador de control SYN y el número de secuencia relativa.

Se establece el señalizador de control SYN y el número de secuencia relativa en 0. A pesar de que el analizador de protocolos en el gráfico indica los valores relativos para los números de secuencia y de acuse de recibo, los valores reales son números binarios de 32 bits. Se pueden determinar los números reales enviados en los encabezados de los segmentos examinando el panel de bytes del paquete. Aquí se pueden ver los cuatro bytes representados en formato hexadecimal.

Protocolo TCP de enlace de tres vías (SYN)



El analizador de protocolo muestra la solicitud del cliente inicial para la sesión en la trama 14

El segmento TCP en esta trama muestra:

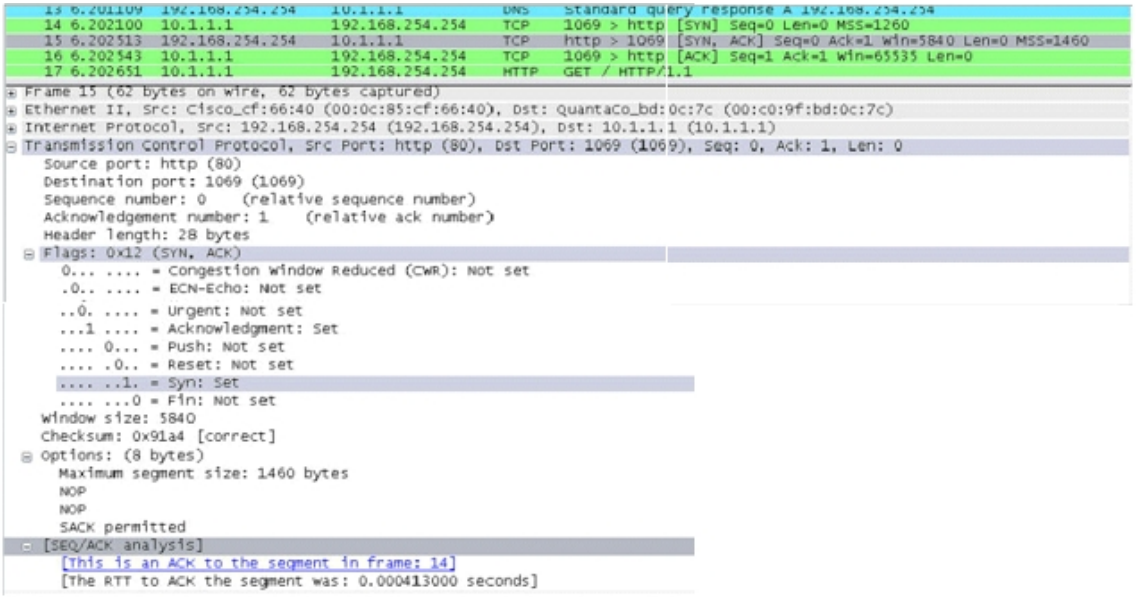
- El señalizador SYN establecido para validar un número de secuencia inicial
- Número de secuencia aleatorio válido (el valor relativo es 0)
- Puerto de origen aleatorio 1069
- El puerto de destino conocido es 80 (puerto HTTP) según indica el servidor Web (httpd)

Paso 2: El servidor TCP necesita dar acuse de recibo del segmento SYN del cliente para establecer la sesión de cliente a servidor. Para hacerlo, el servidor envía un segmento al cliente con el señalizador ACK establecido indicando que el número de acuse de recibo es significativo. Con este señalizador establecido en el segmento, el cliente interpreta esto como acuse de recibo de que el servidor ha recibido el SYN del cliente TCP.

El valor del campo de número del acuse de recibo es igual al número de secuencia inicial del cliente más 1. Esto establece una sesión desde el cliente al servidor. El señalizador ACK permanecerá establecido para mantener el equilibrio de la sesión. Cabe recordar que la conversación entre el cliente y el servidor está compuesta en realidad por dos sesiones de una vía: una del cliente al servidor y la otra del servidor al cliente. En este segundo paso del enlace de tres vías, el servidor debe iniciar la respuesta del servidor al cliente. Para comenzar esta sesión, el servidor utiliza el señalizador SYN de la misma manera en que lo hizo el cliente. Establece el señalizador de control SYN en el encabezado para establecer una sesión del servidor al cliente. El señalizador SYN indica que el valor inicial del campo de número de secuencia se encuentra en el encabezado. Este valor se utiliza para rastrear el flujo de datos en esta sesión desde el servidor y de regreso al cliente.

Como se muestra en la figura, el resultado del analizador de protocolos muestra que los señaladores de control ACK y SYN se establecieron y los números de secuencia y de acuse de recibo se muestran.

Protocolo TCP de enlace de tres vías (SYN, ACK)



Un analizador de protocolos muestra la respuesta del servidor en la trama 15

- El señalizador ACK está establecido para indicar un número de acuse de recibo válido
- Respuesta del número de acuse de recibo al número de secuencia inicial como valor relativo de 1
- Señalizador SYN establecido para indicar el número de secuencia inicial para el servidor a la sesión del cliente
- Número de puerto de destino de 1069 para la correspondencia con los puertos de origen de clientes
- Número de puerto de origen de 80 (HTTP) que indica el servicio del servidor Web (httpd)

Paso 3: Por último, el cliente TCP responde con un segmento que contiene un ACK que actúa como respuesta al SYN de TCP enviado por el servidor. No existen datos de usuario en este segmento. El valor del campo número de acuse de recibo contiene uno más que el número de secuencia inicial recibido del servidor. Una vez que se establecen ambas sesiones entre cliente y servidor, todos los segmentos adicionales que se intercambian en esta comunicación tendrán establecido el señalador ACK.

Como se muestra en la figura, el resultado del analizador de protocolos muestra el señalador de control ACK establecido y los números de secuencia relativa y de acuse de recibo se muestran.

Se puede añadir seguridad a la red de datos de la siguiente manera:

- Denegar el establecimiento de sesiones del TCP
- Permitir sólo sesiones que se establezcan para servicios específicos
- Permitir sólo tráfico como parte de sesiones ya establecidas

Esta seguridad se puede implementar para todas las sesiones del TCP o sólo para las sesiones seleccionadas.

Protocolo TCP de enlace de tres vías (ACK)

```
   13 6.201109  192.168.254.254   10.1.1.1          DNS    Standard query response A 192.168.254.254
   14 6.202100  10.1.1.1          192.168.254.254   TCP    1069 > http [SYN] Seq=0 Len=0 MSS=1260
   15 6.202513  192.168.254.254   10.1.1.1          TCP    http > 1069 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
   16 6.202543  10.1.1.1          192.168.254.254   TCP    1069 > http [ACK] Seq=1 Ack=1 Win=65535 Len=0
   17 6.202651  10.1.1.1          192.168.254.254   HTTP   GET / HTTP/1.1
⊞ Frame 16 (54 bytes on wire, 54 bytes captured)
⊞ Ethernet II, Src: QuantaCo_bd:0c:7c (00:c0:9f:bd:0c:7c), Dst: Cisco_cf:66:40 (00:0c:85:cf:66:40)
⊞ Internet Protocol, Src: 10.1.1.1 (10.1.1.1), Dst: 192.168.254.254 (192.168.254.254)
⊟ Transmission Control Protocol, Src Port: 1069 (1069), Dst Port: http (80), Seq: 1, Ack: 1, Len: 0
    Source port: 1069 (1069)
    Destination port: http (80)
    Sequence number: 1    (relative sequence number)
    Acknowledgement number: 1    (relative ack number)
    Header length: 20 bytes
  ⊟ Flags: 0x10 (ACK)
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN-Echo: Not set
      ..0. .... = Urgent: Not set
      ...1 .... = Acknowledgment: Set
      .... 0... = Push: Not set
      .... .0.. = Reset: Not set
      .... ..0. = Syn: Not set
      .... ...0 = Fin: Not set
    Window size: 65535
    Checksum: 0xd538 [correct]
  ⊟ [SEQ/ACK analysis]
      [This is an ACK to the segment in frame: 15]
      [The RTT to ACK the segment was: 0.000030000 seconds]
```

El analizador de protocolo muestra la respuesta del cliente inicial para la sesión en la trama 16

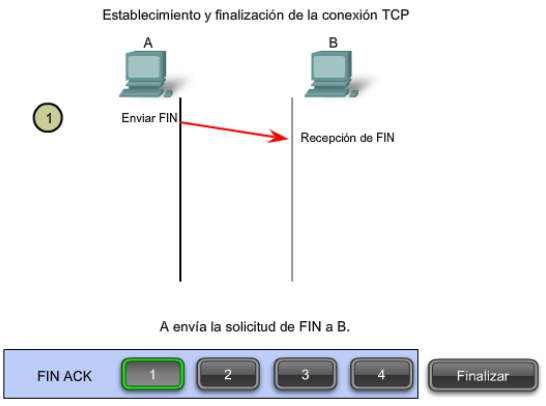
El segmento TCP en esta trama muestra:

- El señalizador ACK está establecido para indicar un número de acuse de recibo válido
- Respuesta del número de acuse de recibo al número de secuencia inicial como valor relativo de 1
- Número de puerto de origen de 1069 para la correspondencia
- Número de puerto de destino de 80 (HTTP) que indica el servicio del servidor Web (httpd)

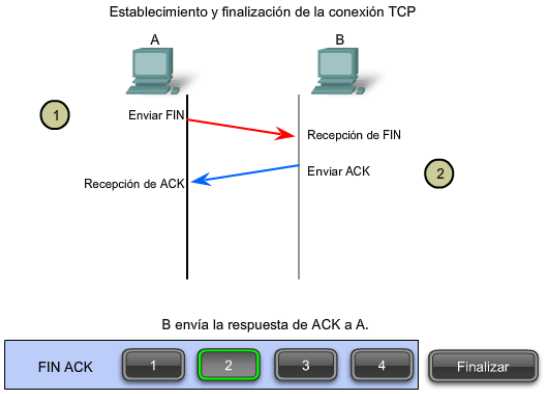
FINALIZACIÓN DE LA SECIÓN TCP

Para cerrar la conexión se debe establecer el señalador de control FIN (Finalizar) en el encabezado del segmento. Para finalizar todas las sesiones TCP de una vía, se utiliza un enlace de dos vías, que consta de un segmento FIN y un segmento ACK. Por lo tanto, para terminar una conversación simple admitida por TCP, se requieren cuatro intercambios para finalizar ambas sesiones. Nota: En esta explicación, los términos cliente y servidor se utilizan como referencia por facilidad, pero el proceso de finalización lo pueden iniciar dos hosts cualquiera que completen la sesión:

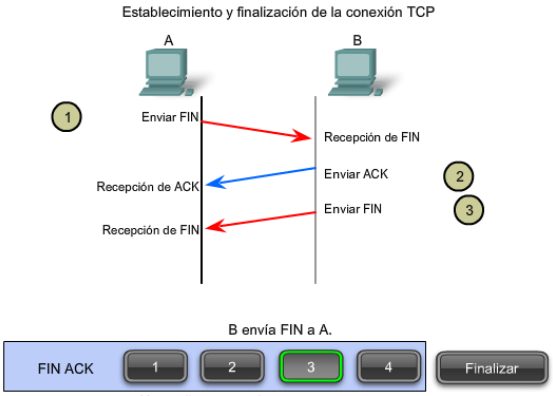
1. Cuando el cliente ni tiene más datos para enviar en el stream, envía un segmento con el señalador FIN establecido.



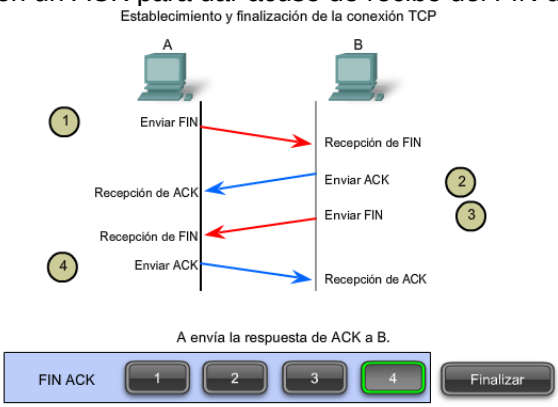
2. El servidor envía un ACK para acusar de recibo el FIN para terminar la sesión de cliente a servidor.



3. El servidor envía un FIN al cliente para terminar la sesión de servidor a cliente.



4. El cliente responde con un ACK para dar acuse de recibo del FIN desde el servidor.



Cuando el cliente que finaliza la sesión no tiene más datos que transferir, establece el señalador FIN en el encabezado de un segmento. Luego, el servidor finaliza la conexión y envía un segmento normal que contiene datos con el señalizador ACK establecido utilizando el número de acuse de recibo, confirmando así que se han recibido todos los bytes de datos. Cuando se dio acuse de recibo de todos los segmentos, la sesión se cierra.

La sesión en la otra dirección se cierra con el mismo proceso. El receptor indica que no existen más datos para enviar estableciendo el señalizador FIN en el encabezado del segmento enviado al origen. Un acuse de recibo devuelto confirma que todos los bytes de datos se recibieron y que la sesión, a su vez, finalizó.

Como se muestra en la figura, los señaladores **FIN** y **ACK** se establecen en el encabezado del segmento, de modo que se cierra una sesión de HTTP.

También es posible terminar la conexión por medio de un enlace de tres vías. Cuando el cliente no posee más datos para enviar, envía un señalizador FIN al servidor. Si el servidor tampoco tiene más datos para enviar, puede responder con los señalizadores FIN y ACK, combinando dos pasos en uno. El cliente responde con un ACK.

Finalización de la sesión TCP (FIN)

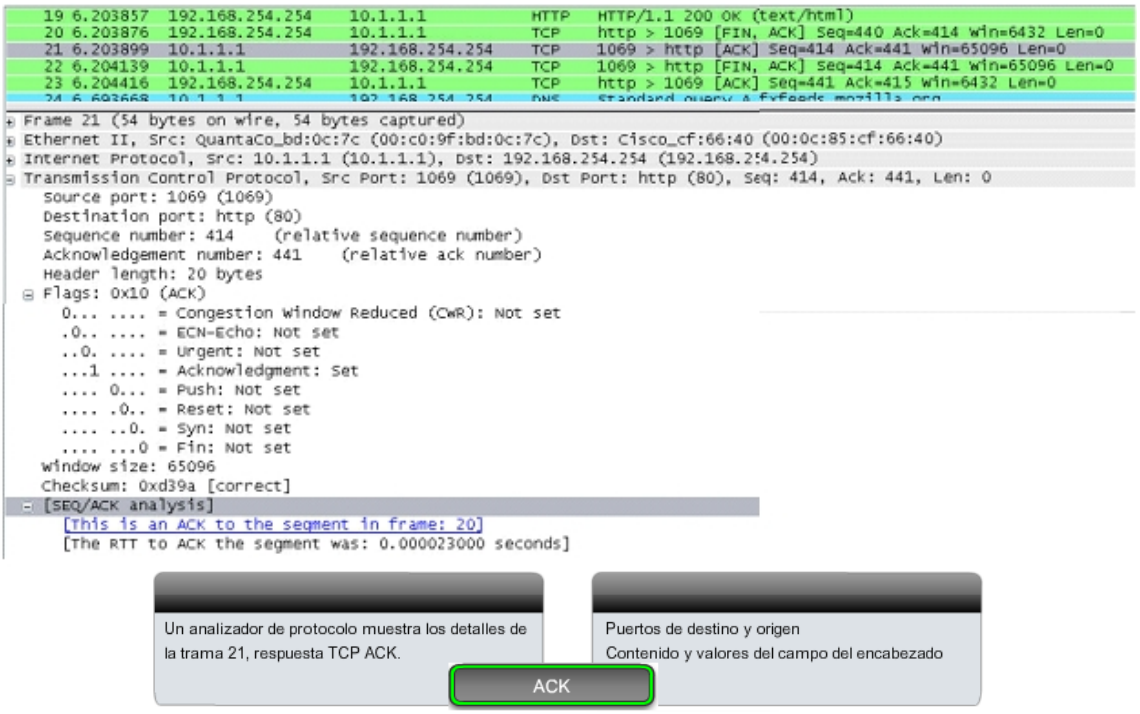
```
   19 6.203857  192.168.254.254    10.1.1.1          HTTP   HTTP/1.1 200 OK (text/html)
   20 6.203876  192.168.254.254    10.1.1.1          TCP    http > 1069 [FIN, ACK] Seq=440 Ack=414 Win=6432 Len=0
   21 6.203899  10.1.1.1           192.168.254.254   TCP    1069 > http [ACK] Seq=414 Ack=441 Win=65096 Len=0
   22 6.204139  10.1.1.1           192.168.254.254   TCP    1069 > http [FIN, ACK] Seq=414 Ack=441 Win=65096 Len=0
   23 6.204416  192.168.254.254    10.1.1.1          TCP    http > 1069 [ACK] Seq=441 Ack=415 Win=6432 Len=0
   24 6.693668  10.1.1.1           192.168.254.254   DNS    Standard query A forfeeds.mozilla.org

⊞ Frame 20 (60 bytes on wire, 60 bytes captured)
⊞ Ethernet II, Src: Cisco_cf:66:40 (00:0c:85:cf:66:40), Dst: Quantaco_bd:0c:7c (00:c0:9f:bd:0c:7c)
⊞ Internet Protocol, Src: 192.168.254.254 (192.168.254.254), Dst: 10.1.1.1 (10.1.1.1)
⊟ Transmission Control Protocol, Src Port: http (80), Dst Port: 1069 (1069), Seq: 440, Ack: 414, Len: 0
    Source port: http (80)
    Destination port: 1069 (1069)
    Sequence number: 440    (relative sequence number)
    Acknowledgement number: 414    (relative ack number)
    Header length: 20 bytes
  ⊟ Flags: 0x11 (FIN, ACK)
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN-Echo: Not set
      ..0. .... = Urgent: Not set
      ...1 .... = Acknowledgment: Set
      .... 0... = Push: Not set
      .... .0.. = Reset: Not set
      .... ..0. = Syn: Not set
      .... ...1 = Fin: Set
    Window size: 6432
    Checksum: 0xb8c3 [correct]
```

Un analizador de protocolo muestra los detalles de la trama 20, solicitud TCP FIN.

Puertos de destino y origen
Contenido y valores del campo del encabezado

FIN

Finalización de la sesión TCP (ACK)



ADMINISTRACIÓN DE SECIONES TCP

REENSAMBLAJE DE SEGMENTOS TCP

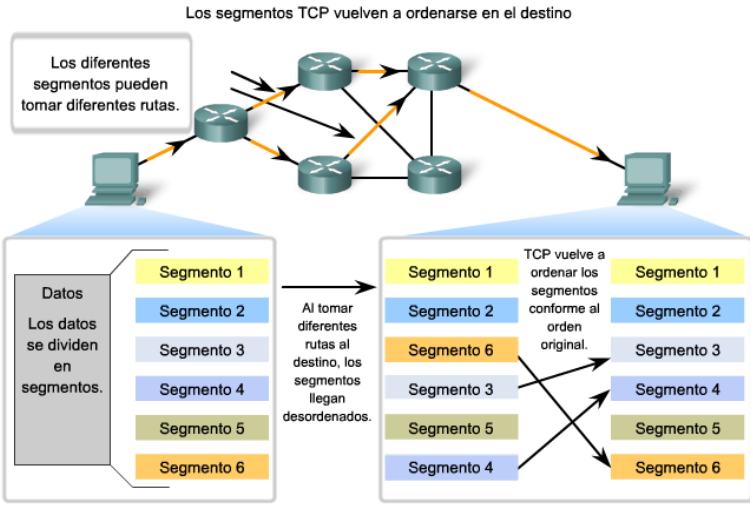
Resecuenciamiento de segmentos para transmitir en orden

Cuando los servicios envían datos mediante el TCP, los segmentos pueden llegar a su destino en desorden. Para que el receptor comprenda el mensaje original, los datos en estos segmentos se reensamblan en el orden original. Para lograr esto, se asignan números de secuencia en el encabezado de cada paquete.

Durante la configuración de la sesión, se establece un número de secuencia inicial (ISN). Este número de secuencia inicial representa el valor de inicio para los bytes de esta sesión que se transmitirán a la aplicación receptora. A medida que se transmiten los datos durante la sesión, el número de secuencia se incrementa en el número de bytes que se han transmitido. Este rastreo de bytes de datos permite que cada segmento se identifique y se envíe acuse de recibo de manera exclusiva. Se pueden identificar segmentos perdidos.

Los números de secuencia de segmento permiten la confiabilidad indicando cómo reensamblar y reordenar los segmentos recibidos, como se muestra en la figura.

El proceso de recepción del TCP coloca los datos del segmento en un búfer de recepción. Los segmentos se colocan en el orden de número de secuencia adecuado y se pasa a la capa de aplicación cuando se reensamblan. Todos los segmentos que llegan con números de secuencia no contiguos se mantienen para su procesamiento posterior. Luego, cuando llegan con los segmentos con bytes perdidos, se procesan.



ACUSE DE RECIBO DE TCP CON EL USO DE VENTANAS

Confirmación de recepción de segmentos

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

El número de secuencia y el número de acuse de recibo del encabezado del segmento se utilizan para confirmar la recepción de los bytes de datos contenidos en los segmentos. El número de secuencia es el número relativo de bytes que ha sido transmitido en esta sesión más 1 (que es el número del primer byte de datos en el segmento actual). TCP utiliza el número de acuse de recibo en segmentos que se vuelven a enviar al origen para indicar el próximo byte de esta sesión que espera el receptor. Esto se llama acuse de recibo de expectativa.

Se le informa al origen que el destino ha recibido todos los bytes de este stream de datos, pero sin incluir el byte que se especifica por el número de acuse de recibo. Se espera que el host emisor envíe un segmento que utiliza un número de secuencia que es igual al número de acuse de recibo.

Recuerde que cada conexión son realmente dos sesiones de una vía. Los números de secuencia y los números de acuse de recibo 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.

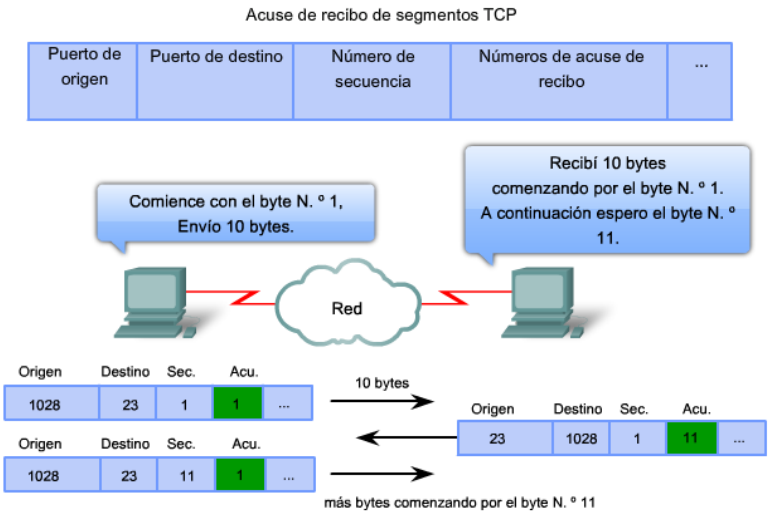
El host receptor de la derecha recibe el segmento en la Capa 4 y determina que el número de secuencia es 1 y que posee 10 bytes de datos. Luego el host envía un segmento de vuelta al host de la izquierda para acusar recibo de estos datos. En este segmento, el host establece el número de acuse de recibo en 11 para indicar que el próximo byte de datos que espera recibir en esta sesión es el byte número 11. Nota, el valor de Ack. en el host de origen permanece en 1 para indicar que el segmento es parte de una conversación continua y que el número en el campo de número de acuse de recibo es válido.

Cuando el host emisor de la izquierda recibe este acuse de recibo, puede enviar el próximo segmento que contiene datos para esta sesión a partir del byte 11.

Observando este ejemplo, si el host emisor tuviera que esperar el acuse de recibo por la recepción de cada uno de los 10 bytes, la red estaría demasiado sobrecargada. Para reducir la sobrecarga de estos acuses de recibo, los segmentos de datos múltiples pueden enviarse previamente y ser reconocidos con un mensaje TCP simple en la dirección opuesta. Este reconocimiento contiene un número de acuse de recibo en base al número 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 acuse de recibo igual a 12000.

La cantidad de datos que un origen puede transmitir antes de que se deba recibir un acuse de recibo se denomina tamaño de la ventana. El tamaño de la ventana es un campo en el encabezado del TCP que permite la administración de datos perdidos y el control del flujo.



RETRANSMICIÓN DE TCP

Manejo de segmentos perdidos

Por más óptimo que sea el diseño de una red, siempre se producirán pérdidas ocasionales de datos. Por lo tanto, TCP cuenta con métodos para gestionar dichas pérdidas de segmentos. Entre estos está un mecanismo para retransmitir segmentos con datos sin acuse de recibo.

Un servicio de host de destino que utiliza TCP generalmente sólo da acuse de recibo de datos para bytes de secuencia continuos. Si uno o más segmentos se pierden, sólo se acusa recibo de los datos de los segmentos que completan el stream.

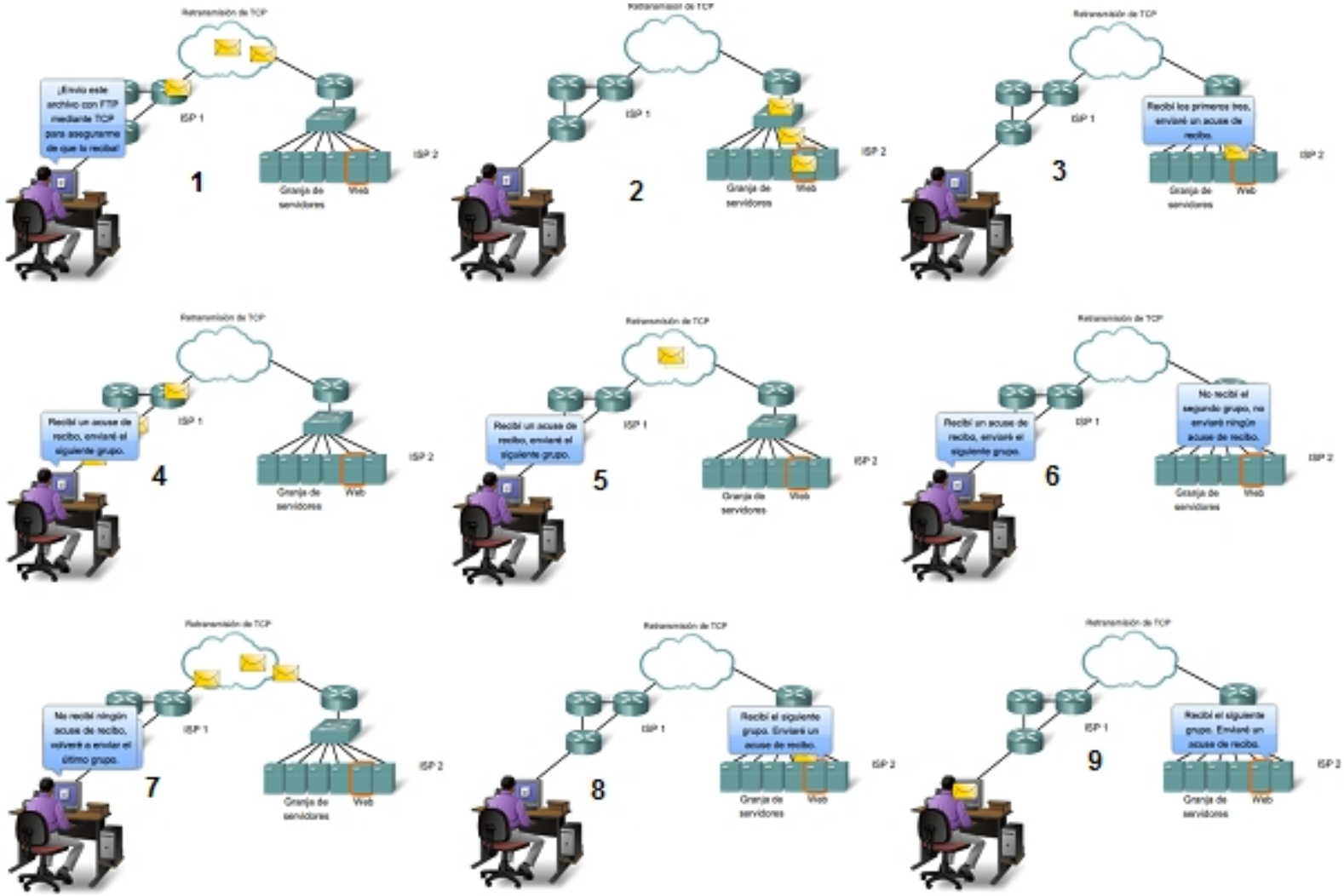
Por ejemplo, si se recibieron los segmentos con números de secuencia de 1500 a 3000 y de 3400 a 3500, el número de acuse de recibo sería 3001. Esto es porque hay segmentos con números de secuencia del 3001 al 3399 que no se han recibido.

Cuando el TCP en el host de origen no recibe un acuse de recibo luego de un determinado período de tiempo, éste regresará al último número de acuse de recibo que recibió y volverá a transmitir los datos desde dicho punto.

El proceso de retransmisión no lo especifica el RFC, pero se deja al criterio de la implementación particular del TCP.

Para una implementación de TCP típica, un host puede transmitir un segmento, colocar una copia en una cola de retransmisión e iniciar un temporizador. Cuando se recibe el acuse de recibo de los datos, se elimina el segmento de la cola. Si no se recibe el acuse de recibo antes de que el temporizador venza, el segmento es retransmitido.

Los hosts pueden emplear también una función opcional llamada Acuses de recibo selectivos. Si ambos hosts admiten el Acuse de recibo selectivo, es posible que el destino reconozca los bytes de segmentos discontinuos y el host sólo necesitará retransmitir los datos perdidos.


Retransmisión de TCP
¡Envío este archivo con FTP mediante TCP para asegurarme de que la reciba!
ISP 1
ISP 2
1
Granja de servidores
Web
Retransmisión de TCP
ISP 1
ISP 2
2
Granja de servidores
Web
Retransmisión de TCP
Recibí los primeros tres, enviaré un acuse de recibo.
ISP 1
ISP 2
3
Granja de servidores
Web
Retransmisión de TCP
Recibí un acuse de recibo, enviaré el siguiente grupo.
ISP 1
ISP 2
4
Granja de servidores
Web
Retransmisión de TCP
Recibí un acuse de recibo, enviaré el siguiente grupo.
ISP 1
ISP 2
5
Granja de servidores
Web
Retransmisión de TCP
Recibí un acuse de recibo, enviaré el siguiente grupo.
No recibí el segundo grupo, no enviaré ningún acuse de recibo.
ISP 1
ISP 2
6
Granja de servidores
Web
Retransmisión de TCP
No recibí ningún acuse de recibo, volveré a enviar el último grupo.
ISP 1
ISP 2
7
Granja de servidores
Web
Retransmisión de TCP
Recibí el siguiente grupo. Enviaré un acuse de recibo.
ISP 1
ISP 2
8
Granja de servidores
Web
Retransmisión de TCP
Recibí el siguiente grupo. Enviaré un acuse de recibo.
ISP 1
ISP 2
9
Granja de servidores
Web

CONTROL DE CONGESTIÓN DE TCP: MINIMIZAR LA PÉRDIDA DE SEGMENTOS

Control de flujo

TCP también proporciona mecanismos para el control del flujo. El control del flujo contribuye con la confiabilidad de la transmisión TCP ajustando la tasa efectiva de flujo de datos entre los dos servicios de la sesión. Cuando se le informa al origen que se recibió una cantidad específica de datos en los segmentos, puede seguir enviando más datos para esta sesión.

El campo de Tamaño de ventana en el encabezado del TCP especifica la cantidad de datos que se pueden transmitir antes de que se deba recibir un acuse de recibo. El tamaño inicial de la ventana se determina durante el arranque de sesión por medio del enlace de tres vías.

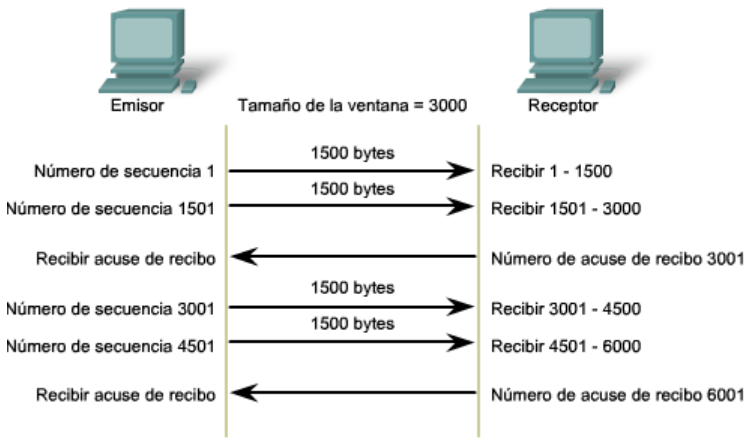
El mecanismo de retroalimentación del TCP ajusta la velocidad eficaz de transmisión de datos al flujo máximo que la red y el dispositivo de destino pueden admitir sin pérdidas. TCP intenta gestionar la tasa de transmisión de manera que todos los datos se reciban y se reduzcan las retransmisiones.

En la figura se observa una representación simplificada del tamaño de la ventana y los acuses de recibo. 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 tiene este acuse de recibo del receptor, ya puede transmitir 3000 bytes adicionales.

Durante el retraso en la recepción del acuse de recibo, el emisor no enviará ningún segmento adicional para esta sesión. 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 la velocidad de los datos ayuda a reducir la contención de recursos.

Acuse de recibo de segmentos TCP y tamaño de la ventana



El tamaño de la ventana determina la cantidad de bytes enviados antes de esperar un acuse de recibo.
El número de acuse de recibo es el número del próximo byte esperado.

Reducción del tamaño de la ventana

Otra forma de controlar el flujo de datos es utilizar tamaños de ventana dinámicos. Cuando los recursos de la red son limitados, TCP puede reducir el tamaño de la ventana para lograr que los segmentos recibidos sean reconocidos con mayor frecuencia. Esto reduce de forma

efectiva la velocidad de transmisión porque el origen espera que se de acuse de recibo de los datos con más frecuencia.

El host receptor del TCP envía el valor del tamaño de la ventana al TCP emisor para indicar el número de bytes que está preparado para recibir como parte de la sesión. Si el destino necesita disminuir la velocidad de comunicación debido a su memoria de búfer limitada, puede enviar un valor más pequeño del tamaño de la ventana al origen como parte del acuse de recibo.

Como se muestra en la figura, si un host receptor está congestionado, puede responder al host emisor con un segmento con tamaño reducido de la ventana. En este gráfico, se produjo la pérdida de uno de los segmentos. El receptor cambió el campo de la ventana en el encabezado del TCP de los segmentos devueltos en esta conversación de 3000 a 1500. Esto hizo que el emisor redujera el tamaño de la ventana a 1500.

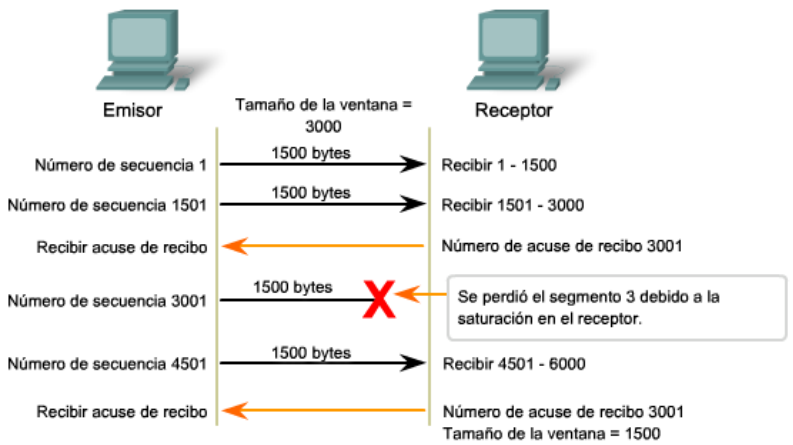
Después de períodos de transmisión sin pérdidas de datos o recursos limitados, el receptor comenzará a aumentar el tamaño de la ventana. Esto reduce la sobrecarga de la red, ya que se requiere enviar menos acuses de recibo. El tamaño de la ventana continuará aumentando hasta que haya pérdida de datos, lo que producirá una disminución del tamaño de la misma.

Estas disminuciones y aumentos dinámicos del tamaño de la ventana representan un proceso continuo en TCP que determina el tamaño óptimo de la ventana para cada sesión del TCP. En redes altamente eficientes, los tamaños de la ventana pueden ser muy grandes porque no se pierden datos. En redes donde se tensiona la infraestructura subyacente, el tamaño de la ventana probablemente permanecerá pequeño.

Enlaces

Los detalles de las diferentes características para administrar la congestión del TCP se encuentran en RFC 2581.
http://www.ietf.org/rfc/rfc2581.txt

Saturación de TCP y control del flujo



Si se pierden segmentos debido a la saturación, el receptor acusará recibo del último segmento secuencial recibido y responderá con un tamaño de ventana reducido.

PROTOCOLO UDP

UDP: BAJA SOBRECARGA vs. CONFIABILIDAD

UDP es un protocolo simple que provee las funciones básicas de la capa de transporte. Tiene una sobrecarga mucho menor que el TCP, ya que no está orientado a la conexión y no proporciona mecanismos sofisticados de retransmisión, secuenciamiento y flujo de control.

Esto no significa que las aplicaciones que utilizan UDP no son siempre poco confiables. Sólo quiere decir que estas funciones no las contempla el protocolo de la capa de transporte y se deben implementar aparte, si fuera necesario.

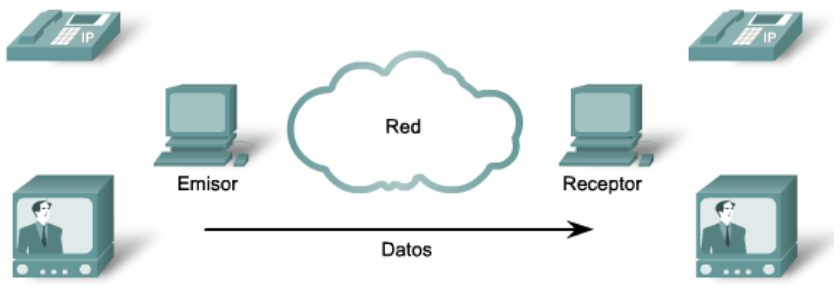
Pese a que es relativamente baja la cantidad total de tráfico UDP que puede encontrarse en una red típica, los protocolos clave de la capa de aplicación que utiliza UDP incluyen:

- Sistema de nombres de dominio (DNS)
- Protocolo simple de administración de red (SNMP, Simple Network Management Protocol)
- Protocolo de configuración dinámica de host (DHCP)
- Protocolo de información de enrutamiento (RIP)
- Protocolo de transferencia de archivos trivial (TFTP)
- Juegos en línea

Algunas aplicaciones, tales como los juegos en línea o VoIP, pueden tolerar la pérdida de algunos datos. Si estas aplicaciones utilizaran TCP, experimentarían largas demoras, ya que TCP detecta la pérdida de datos y los retransmite. Estas demoras serían más perjudiciales para la aplicación que las pequeñas pérdidas de datos. Algunas aplicaciones, como DNS, simplemente vuelven a intentar la solicitud si no reciben una respuesta y, por lo tanto, no necesitan el TCP para garantizar la entrega del mensaje.

La baja sobrecarga del UDP es deseada por dichas aplicaciones.

Transporte de datos con baja sobrecarga de UDP



UDP no establece ninguna conexión antes de enviar datos.

UDP suministra transporte de datos con baja sobrecarga debido a que posee un encabezado de datagrama pequeño sin tráfico de administración de red.

REENSAMBLAJE DE DATAGRAMAS DE UDP

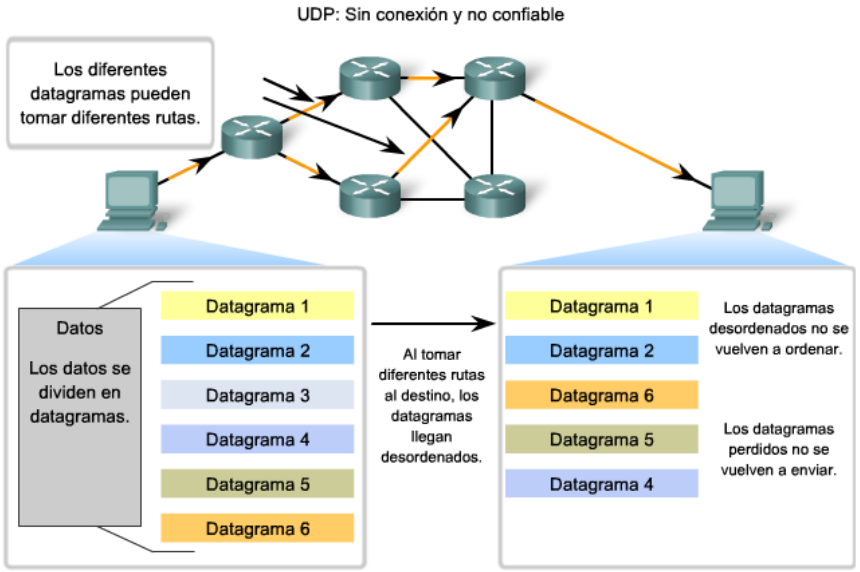
Ya que UDP opera sin conexión, las sesiones no se establecen antes de que se lleve a cabo la comunicación, como sucede con TCP. Se dice que UDP es basado en

transacciones. En otras palabras, cuando una aplicación tiene datos que enviar, sólo los envía.

Muchas aplicaciones que utilizan UDP envían pequeñas cantidades de datos que pueden ajustarse en un segmento. Sin embargo, algunas aplicaciones envían cantidades más grandes que deben dividirse en varios segmentos. La PDU del UDP se conoce como un datagrama, aunque los términos segmento y datagrama se utilizan algunas veces de forma intercambiable para describir una PDU de la capa de transporte.

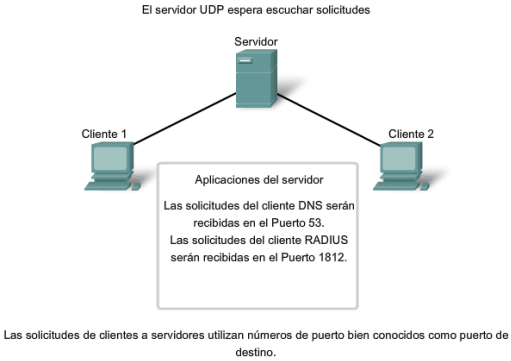
Cuando se envían datagramas múltiples a un destino, pueden tomar diferentes rutas y llegar en el orden equivocado. UDP no mantiene un seguimiento de los números de secuencia de la manera en que lo hace TCP. UDP no puede reordenar los datagramas en el orden de la transmisión. Observe la figura.

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 los datos es importante para la aplicación, la misma deberá identificar la secuencia adecuada y determinar cómo procesarlos.



PROCESOS Y SOLICITUDES DEL SERVIDOR UDP

Al igual que las aplicaciones basadas en TCP, a las aplicaciones de servidor con base en UDP se les asignan números de puerto bien conocidos o registrados. Cuando se ejecutan estas aplicaciones o procesos, aceptan los datos que coincidan con el número de puerto asignado. Cuando UDP recibe un datagrama destinado a uno de esos puertos, envía los datos de aplicación a la aplicación adecuada en base a su número de puerto.



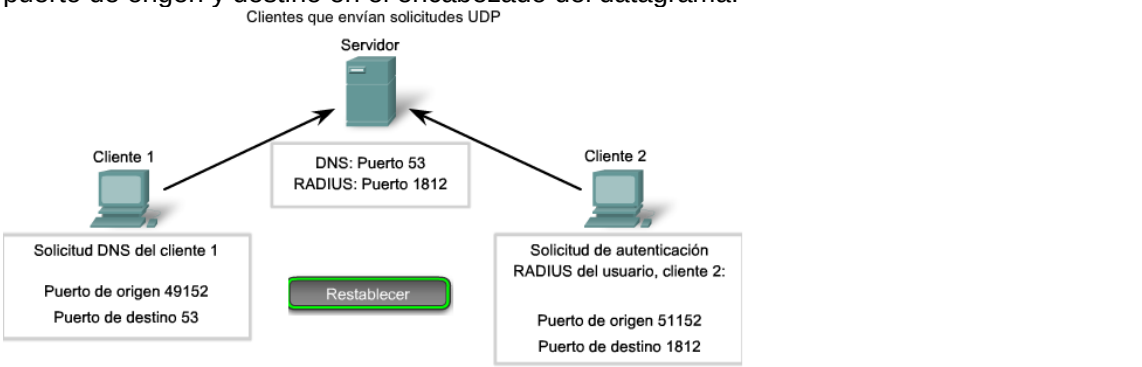
PROCESOS DE CLIENTES UDP

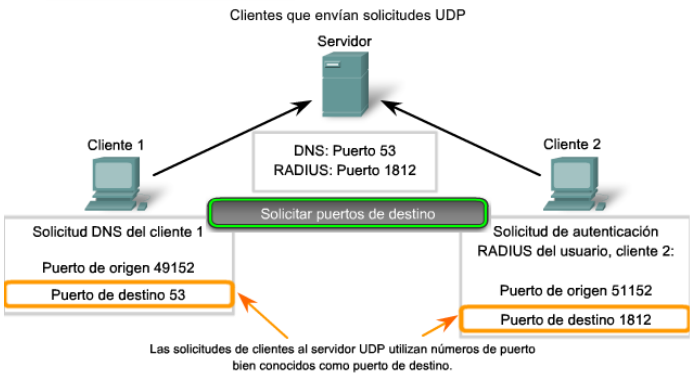
Como en TCP, la comunicación cliente-servidor la inicia una aplicación cliente que solicita datos de un proceso del servidor. El proceso de cliente UDP selecciona al azar un número de puerto del rango dinámico de números de puerto y lo utiliza como puerto de origen para la conversación. El puerto de destino por lo general será el número de puerto bien conocido o registrado asignado al proceso del servidor.

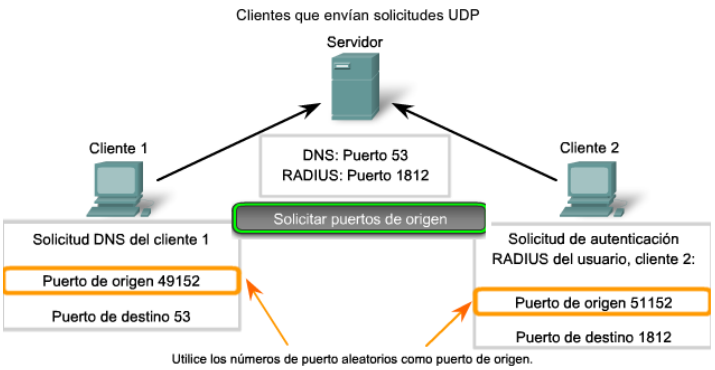
Los números de puerto de origen seleccionados al azar colaboran con la seguridad. Si existe un patrón predecible para la selección del puerto de destino, un intruso puede simular el acceso a un cliente de manera más sencilla intentando conectarse al número de puerto que tenga mayor posibilidad de estar abierto.

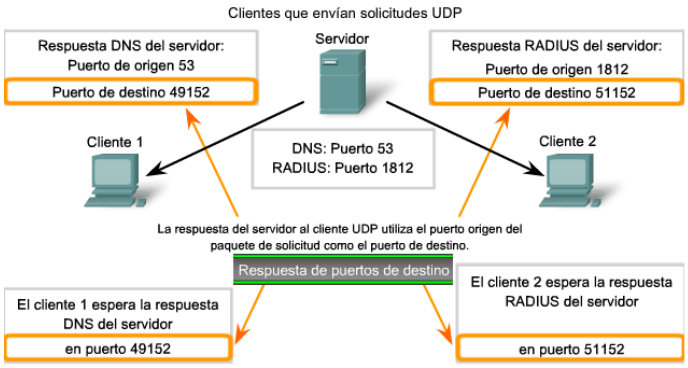
Ya que no se crean sesiones con UDP, tan pronto como los datos están listos para enviarse y los puertos estén identificados, UDP puede formar el datagrama y pasarlo a la capa de red para direccionarlo y enviarlo a la red.

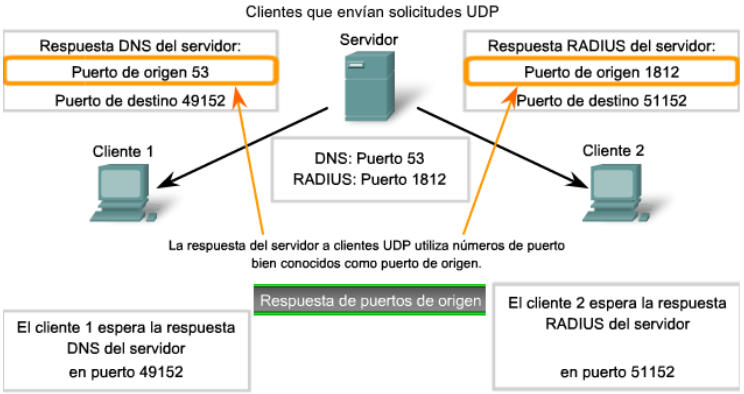
Cabe recordar que una vez que el cliente ha elegido los puertos de origen y destino, estos mismos puertos se utilizarán en el encabezado de todos los datagramas que se utilicen en la transacción. Para la devolución de datos del servidor al cliente, se invierten los números de puerto de origen y destino en el encabezado del datagrama.








Clientes que envían solicitudes UDP
Respuesta DNS del servidor:
Puerto de origen 53
Puerto de destino 49152
Servidor
Respuesta RADIUS del servidor:
Puerto de origen 1812
Puerto de destino 51152
Cliente 1
DNS: Puerto 53
RADIUS: Puerto 1812
Cliente 2
La respuesta del servidor al cliente UDP utiliza el puerto origen del paquete de solicitud como el puerto de destino.
Respuesta de puertos de destino
El cliente 1 espera la respuesta DNS del servidor
en puerto 49152
El cliente 2 espera la respuesta RADIUS del servidor
en puerto 51152


Clientes que envían solicitudes UDP
Respuesta DNS del servidor:
Puerto de origen 53
Puerto de destino 49152
Servidor
Respuesta RADIUS del servidor:
Puerto de origen 1812
Puerto de destino 51152
Cliente 1
DNS: Puerto 53
RADIUS: Puerto 1812
Cliente 2
La respuesta del servidor a clientes UDP utiliza números de puerto bien conocidos como puerto de origen.
Respuesta de puertos de origen
El cliente 1 espera la respuesta DNS del servidor
en puerto 49152
El cliente 2 espera la respuesta RADIUS del servidor
en puerto 51152