

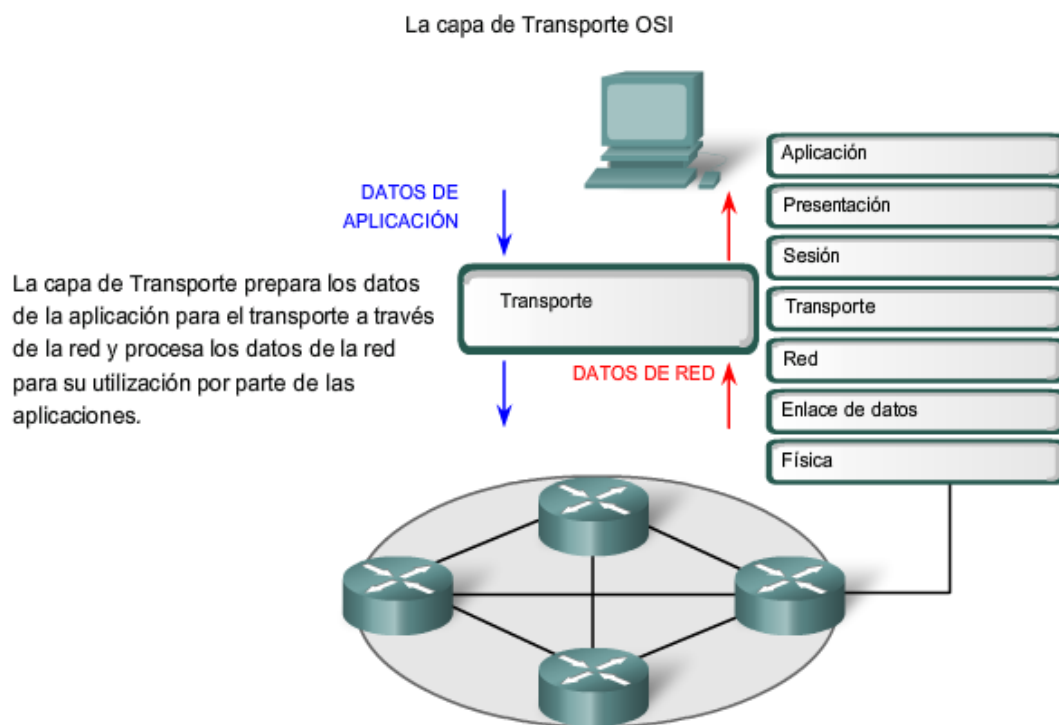
4 Capa de Transporte del modelo OSI

4.0 Introducción del capítulo

4.0.1 Introducción del capítulo

Las redes de datos e Internet brindan soporte a la red humana al proporcionar la comunicación continua y confiable entre las personas, tanto de manera local como alrededor del mundo. En un único dispositivo, las personas pueden utilizar varios servicios como e-mails, la Web y la mensajería instantánea para enviar mensajes o recuperar información. Las aplicaciones como clientes de correo electrónico, exploradores Web y clientes de mensajería instantánea permiten que las personas utilicen las computadoras y las redes para enviar mensajes y buscar información.

Los datos de cada una de estas aplicaciones se empaquetan, transportan y entregan al daemon de servidor o aplicación adecuados en el dispositivo de destino. Los procesos descritos en la capa de Transporte del modelo OSI aceptan los datos de la capa de Aplicación y los preparan para el direccionamiento en la capa de Red. La capa de Transporte es responsable de la transferencia de extremo a extremo general de los datos de aplicación.



4.1 Funciones de la capa de Transporte

4.1.1 Propósito 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:

- seguimiento de la comunicación individual entre aplicaciones en los hosts origen y destino,
- segmentación de datos y gestión de cada porción,
- reensamble de segmentos en flujos de datos de aplicación, e
- identificación de las diferentes aplicaciones.

Seguimiento de Conversaciones individuales

Cualquier host puede tener múltiples aplicaciones que se están comunicando 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 diversos streams de comunicación entre estas aplicaciones.

Segmentación de datos

Debido a que cada aplicación genera un stream de datos para enviar a una aplicación remota, estos datos deben prepararse para ser enviados por 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 puede ser direccionada 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 de la capa de Transporte describen cómo se utiliza la información de encabezado de dicha capa para reensamblar las secciones de datos en streams y enviarlas a la capa de Aplicación.

Identificación de las aplicaciones

Para poder transferir los streams de datos a las aplicaciones adecuadas, la capa de Transporte debe identificar la aplicación de destino. 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 con qué aplicación está asociada esa sección de datos.

La capa de Transporte es el enlace entre la capa de Aplicación y las capas inferiores, que son responsables de la transmisión en la red. Esta capa acepta datos de distintas conversaciones y los transfiere a las capas inferiores como secciones manejables que puedan ser eventualmente multiplexadas a través del medio.

Las aplicaciones no necesitan conocer los detalles de operación 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. Luego la capa de Transporte ordena estas secciones antes de entregarlas a la aplicación adecuada.

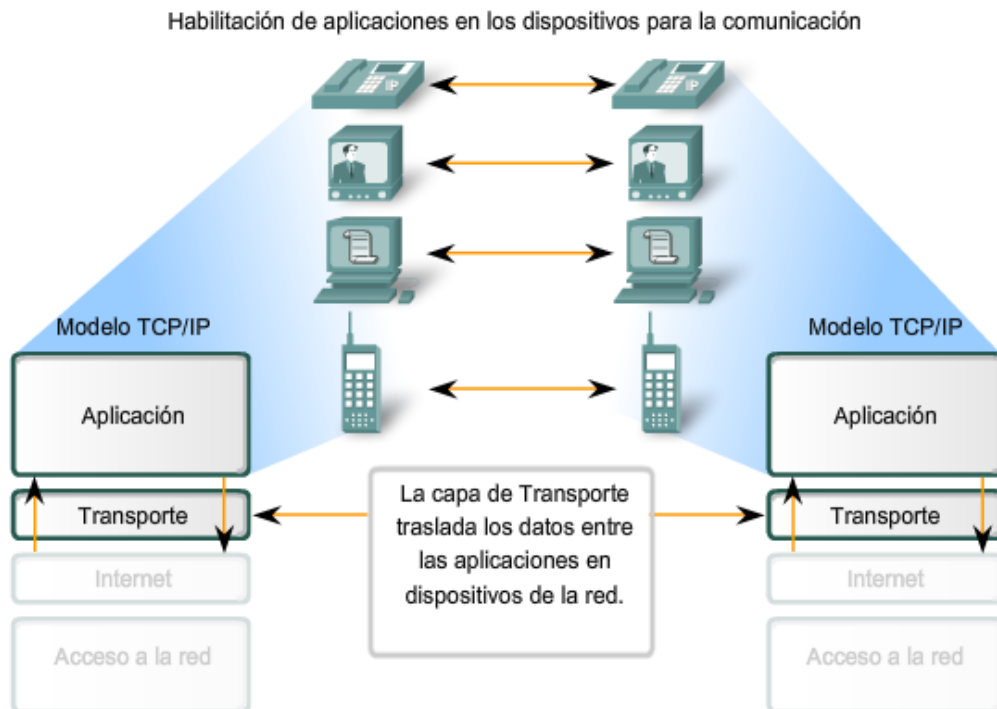
Los requerimientos de datos varían

Debido a que las distintas aplicaciones poseen distintos requerimientos, existen varios protocolos de la capa de Transporte. 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 distintos protocolos de la capa de Transporte poseen distintas reglas que permiten que los dispositivos gestionen los diversos requerimientos de datos.

Algunos protocolos proporcionan sólo las funciones básicas para la entrega eficiente de las secciones de datos entre las aplicaciones adecuadas. Estos tipos de protocolos son útiles para aquellas aplicaciones cuyos datos son sensibles a las demoras.

Otros protocolos de la capa de Transporte describen procesos que brindan funciones adicionales, como asegurar la entrega 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.

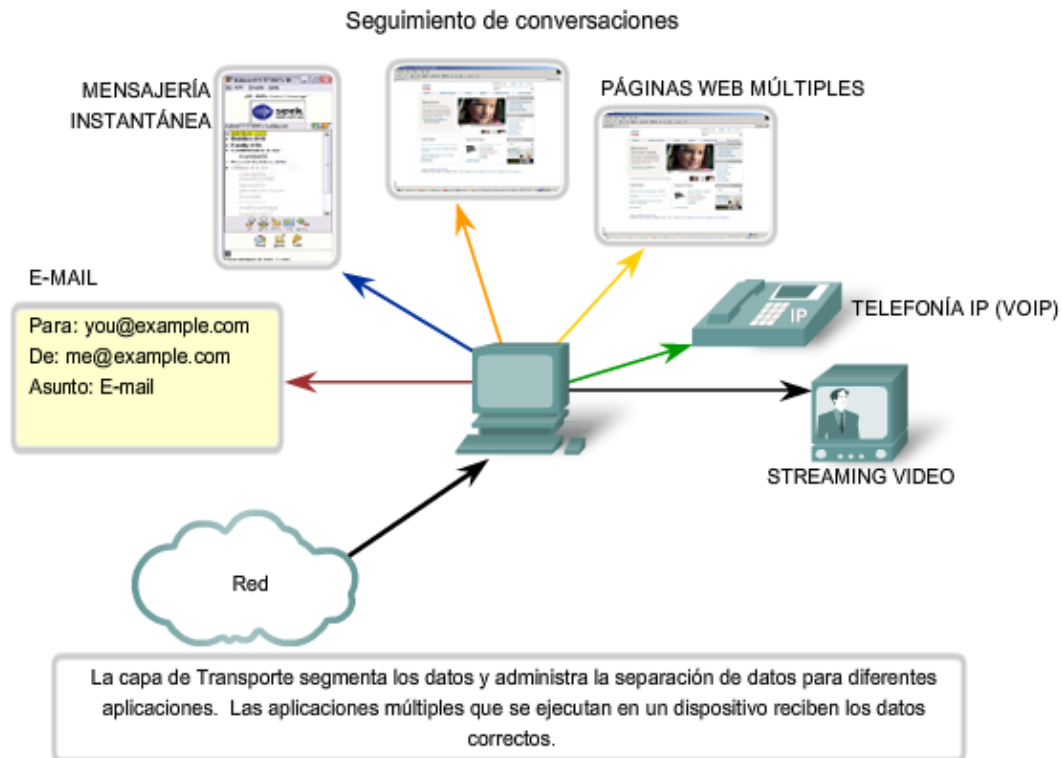


Separación de comunicaciones múltiples

Considere una computadora conectada a una red que recibe y envía e-mails 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 se direccionan al explorador Web y el texto de un mensaje instantáneo no aparece en el e-mail.

Además, los usuarios precisan que un e-mail o una página Web sean recibidos y presentados de manera completa para que la información sea considerada útil. Las demoras leves se consideran aceptables para asegurar que se reciba y presente la información completa.

Por el contrario, la pérdida ocasional de pequeñas partes de una conversación telefónica puede considerarse 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.



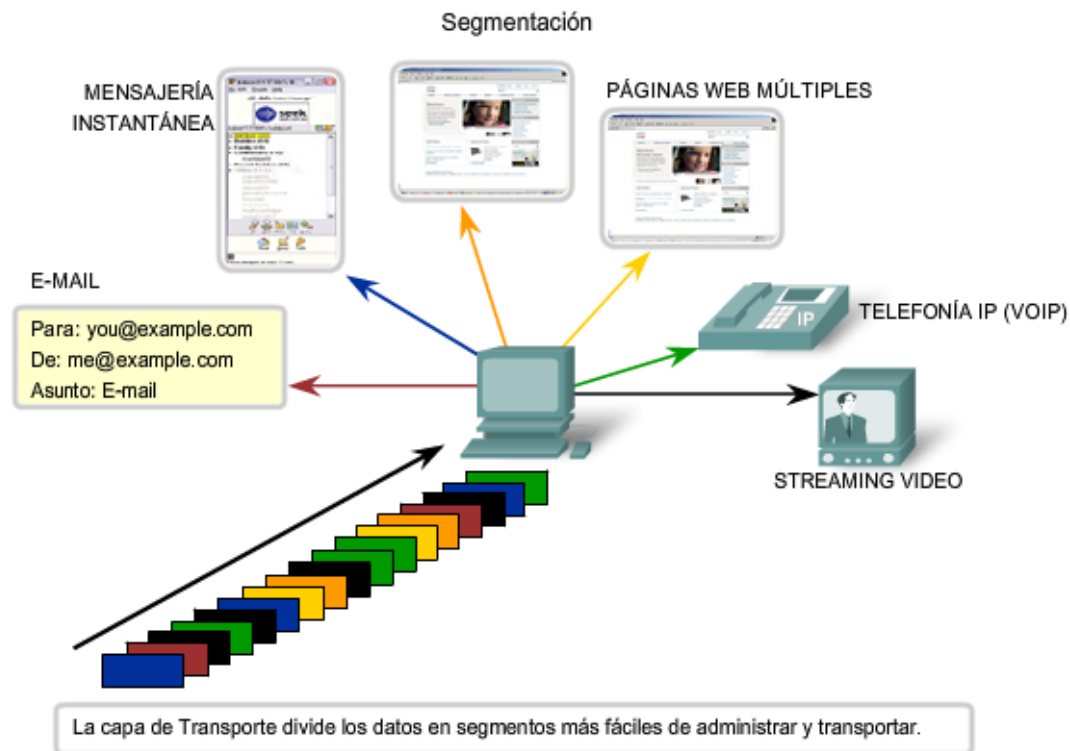
Como se explicó en un capítulo anterior, el envío de algunos tipos de datos, un vídeo por ejemplo, a través de la red como un stream de comunicación completa puede impedir que se produzcan otras comunicaciones al mismo tiempo. También hace dificultosa la recuperación de errores y la retransmisión de datos dañados.

La división de los datos en partes pequeñas y el envío de estas partes desde el origen hacia el destino permiten que se puedan entrelazar (multiplexar) distintas comunicaciones 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 stream de vídeo por ejemplo, podría recibir datos. No se podrían recibir correos electrónicos, chats ni mensajes instantáneos ni visualizar páginas Web y ver un vídeo al mismo tiempo.

En la capa de Transporte, cada conjunto de secciones en particular que fluyen desde una aplicación de origen a una de destino se conoce como conversación.

Para identificar todos los segmentos de datos, la capa de Transporte agrega un encabezado a la sección 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.



4.1.2 Control de las conversaciones

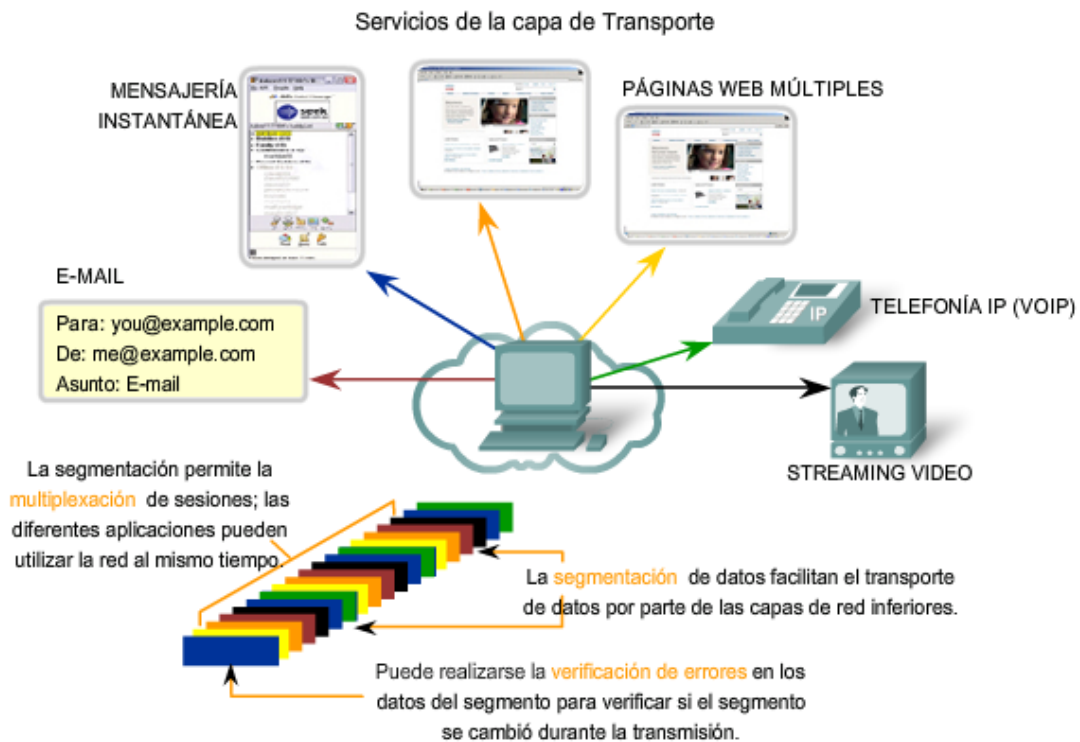
Las funciones principales especificadas por todos los protocolos de la capa de Transporte incluyen:

Segmentación y reensamblaje: La mayoría de las redes poseen una limitación en cuanto a la cantidad de datos que pueden incluirse en una única PDU (Unidad de datos del protocolo). 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 conversaciones: Pueden existir varias aplicaciones o servicios ejecutándose en cada host de la red. A cada una de estas aplicaciones o servicios se les asigna una dirección conocida como puerto para que la capa de Transporte pueda determinar 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 reensamblaje de datos, algunos protocolos de la capa de Transporte proveen:

- conversaciones orientadas a la conexión,
- entrega confiable,
- reconstrucción ordenada de datos, y
- 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 secciones lleguen a destino al contar con el dispositivo de origen para volver a transmitir los datos que se hayan perdido.

Entrega en el mismo orden

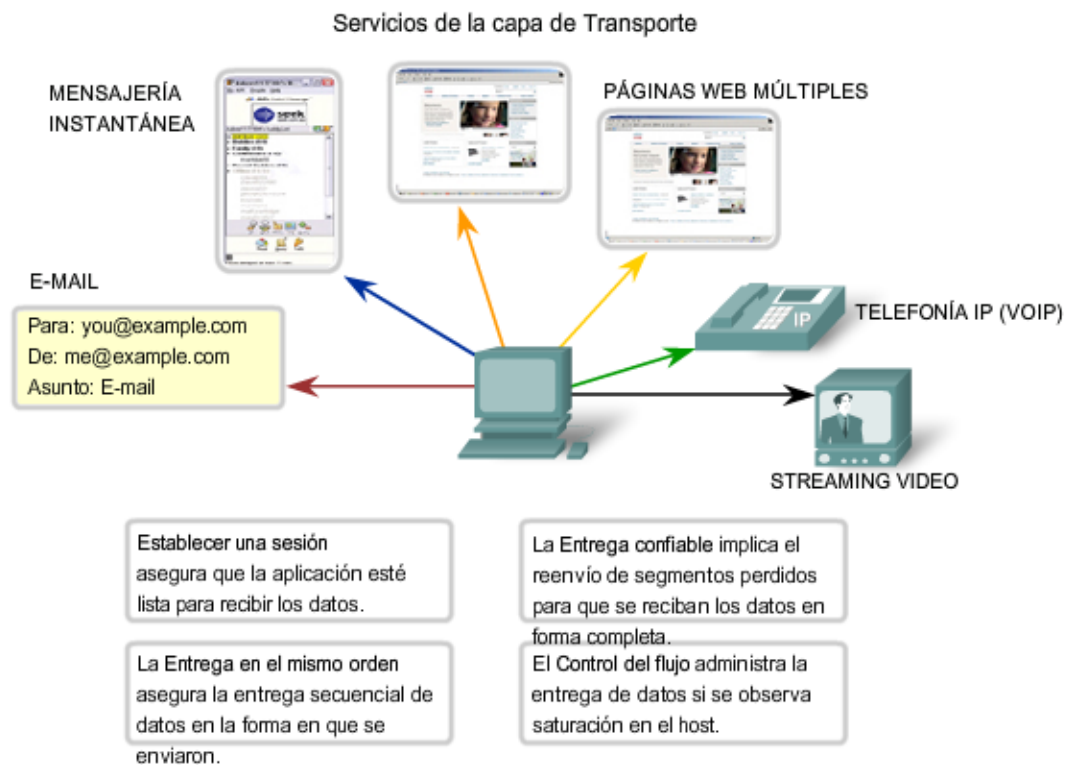
Ya que las redes proveen rutas múltiples que pueden poseer distintos tiempos de transmisión, los datos pueden llegar en el orden incorrecto. Al numerar y secuenciar los segmentos, la capa de Transporte puede asegurar que los mismos se reensamblen en el orden adecuado.

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 del flujo puede prevenir la pérdida de segmentos en la red y evitar la necesidad de retransmisión.

Estos servicios se describirán con más detalle a medida que se expliquen los protocolos en este capítulo.



4.1.3 Soporte de comunicación confiable

Cabe recordar que la función principal de la capa de Transporte es administrar los datos de aplicación para las conversaciones entre hosts. Sin embargo, las diferentes aplicaciones tienen diferentes requerimientos para sus datos y, por lo tanto, se han desarrollado diferentes protocolos de Transporte para satisfacer estos requerimientos.

Un protocolo de la capa de Transporte puede implementar un método para asegurar 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. En la capa de Transporte, las tres operaciones básicas de confiabilidad son:

- seguimiento de datos transmitidos,
- acuse de recibo de los datos recibidos, y
- retransmisión de cualquier dato sin acuse de recibo.

Esto requiere que los procesos de la capa de Transporte de origen mantengan el seguimiento de todas las porciones de datos de cada conversación y retransmitan cualquiera de los datos que no dieron acuse de recibo por el destino. La capa de Transporte del host de recepción también debe rastrear los datos a medida que se reciben y reconocer la recepción de los datos.

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 está contenida en el encabezado de la Capa 4.

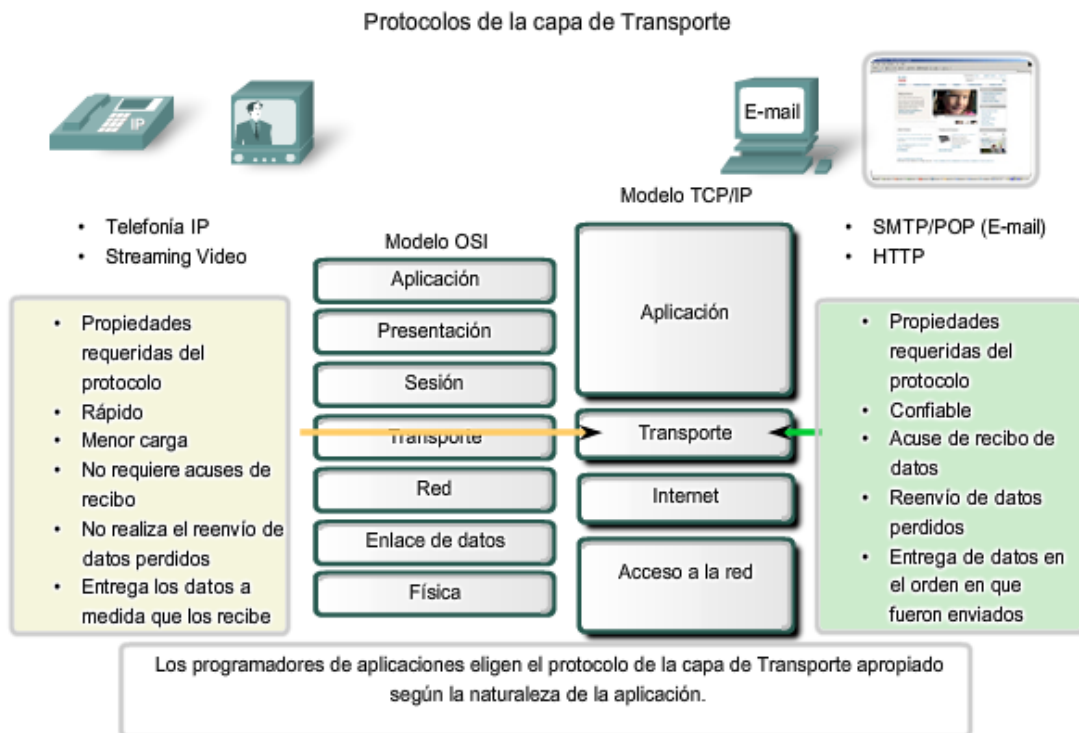
Esto genera un equilibrio ("trade-off") entre el valor de confiabilidad y la carga que representa para la red. Los desarrolladores de aplicaciones deben elegir qué tipo de protocolo de transporte es adecuado en base a los requerimientos de sus aplicaciones. En la capa de Transporte, existen protocolos que especifican métodos para entrega confiable, garantizada o de máximo esfuerzo. En el contexto de las redes, la entrega de máximo esfuerzo se considera no confiable, ya que no existe acuse de recibo de que los datos hayan llegado al destino.

Determinación de la necesidad de confiabilidad

Las aplicaciones, como bases de datos, las páginas Web y los e-mails, requieren que todos los datos enviados lleguen al destino en su condición original, de manera que los mismos 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. El uso de recursos de red adicionales se considera necesario para estas aplicaciones.

Otras aplicaciones son más tolerantes en lo que se refiere a la pérdida de pequeñas cantidades 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 vídeo 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.



4.1.4 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 Protocolos 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 "maximo esfuerzo".

Entre las aplicaciones que utilizan UDP se incluyen:

sistema de nombres de dominios (DNS),

streaming de vídeo, y

Voz sobre IP (VoIP).

Protocolo de control de transmisión (TCP)

TCP es un protocolo orientado a la conexión, descrito en la RFC 793. TCP incurre en el uso adicional de recursos para agregar 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. Ver la figura para obtener una comparación.

Las aplicaciones que utilizan TCP son:

exploradores Web,

e-mail, y

transferencia de archivos

Encabezados TCP y UDP



4.1.5 Direccionamiento del puerto

Identificación de las conversaciones

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

Los servicios basados en TCP y UDP mantienen un seguimiento de las varias 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 los números de los puertos.

En el encabezado de cada segmento o datagrama hay un puerto de origen y 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 en el host remoto.

Los números de puerto se asignan de varias maneras, en función de si el mensaje es una solicitud o una respuesta. Mientras que los procesos en el servidor poseen números de puertos estáticos asignados a ellos, los clientes eligen un número de puerto de forma dinámica para cada conversación.

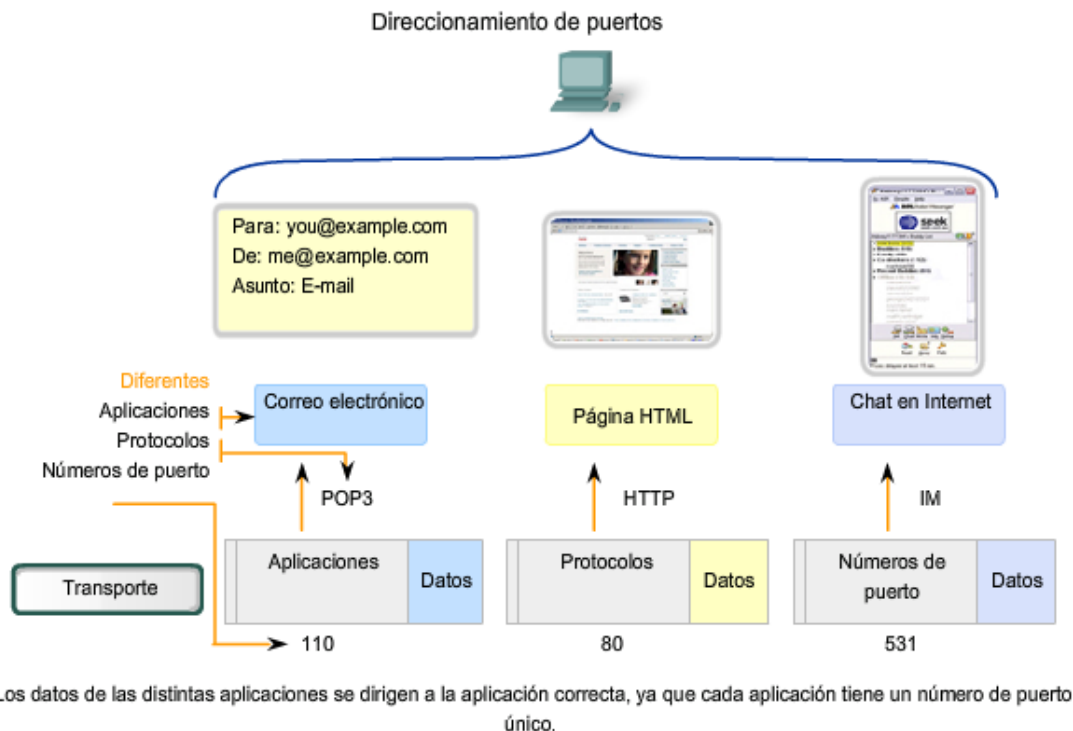
Cuando una aplicación de cliente envía una solicitud a una aplicación de servidor, el puerto de destino contenido en el encabezado es el número de puerto que se asigna al daemon de servicio que se ejecuta en el host remoto. El software del cliente debe conocer el número de puerto asociado con el proceso 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 del encabezado de un segmento o datagrama de un cliente se genera de manera aleatoria. Siempre y cuando no entre en conflicto con otros puertos en uso en el sistema, el cliente puede elegir cualquier número de puerto. 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, pueda ser enviada 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 y los números de puerto de origen y de destino, también es exclusivo e identifica la conversación entre dos hosts.

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

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



La [Autoridad de números asignados de Internet \(IANA\)](#) asigna números de puerto. IANA es un organismo de estándares responsable de la asignación de varias normas de direccionamiento.

Existen distintos tipos de números de puerto:

Puertos bien conocidos (Números del 0 al 1 023): estos números se reservan para servicios y aplicaciones. Por lo general, se utilizan para aplicaciones como HTTP (servidor Web), POP3/SMTP (servidor de e-mail) y Telnet. Al definir estos [puertos conocidos](#) para las aplicaciones del servidor, las aplicaciones del cliente pueden ser programadas para solicitar una conexión a un puerto específico y su servicio asociado.

Puertos Registrados (Números 1024 al 49151): estos números de puertos están asignados 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 también pueden utilizarse si un usuario los selecciona de manera dinámica como puerto de origen.

Puertos dinámicos o privados (Números del 49 152 al 65 535): también conocidos como puertos efímeros, suelen asignarse de manera dinámica a aplicaciones de 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).

Utilización de los dos protocolos TCP y UDP

Algunas aplicaciones pueden utilizar los dos protocolos: TCP y UDP. 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 53 de puerto conocido es utilizado por ambos protocolos con este servicio.

Enlaces

Se puede encontrar una lista actual de números de puertos en <http://www.iana.org/assignments/port-numbers>.

Números de puerto

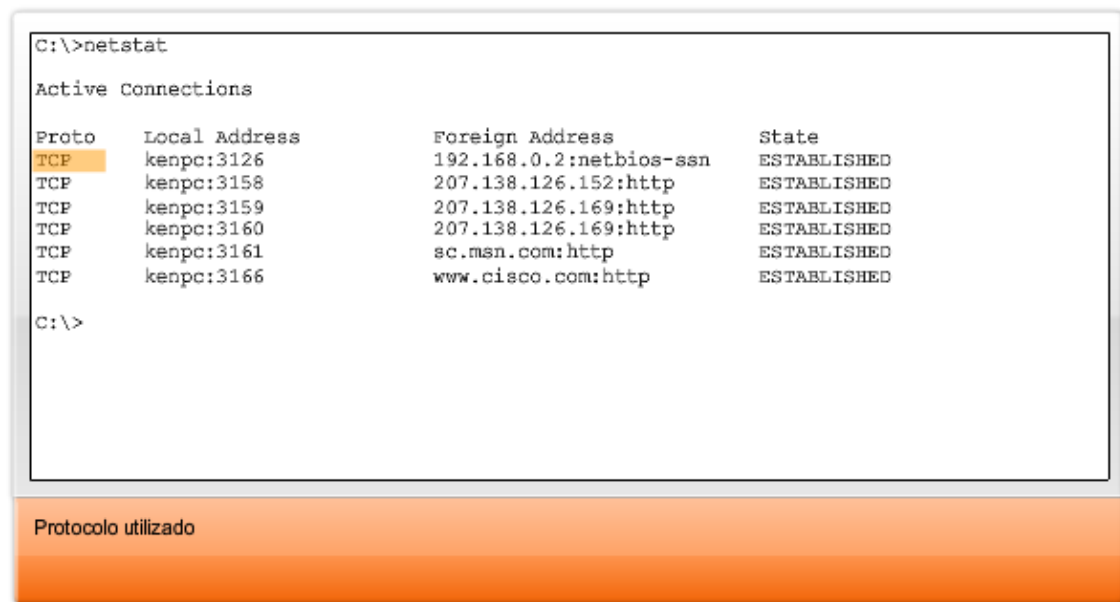
Rango de números de puerto	Grupo de puertos
De 0 a 1023	Puertos bien conocidos (Contacto)
De 1024 a 49151	Puertos registrados
De 49152 a 65535	Puertos privados y/o dinámicos

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 ajenos 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



```
C:\>netstat

Active Connections

Proto Local Address Foreign Address State
TCP kenpc:3126 192.168.0.2:netbios-ssn ESTABLISHED
TCP kenpc:3158 207.138.126.152:http ESTABLISHED
TCP kenpc:3159 207.138.126.169:http ESTABLISHED
TCP kenpc:3160 207.138.126.169:http ESTABLISHED
TCP kenpc:3161 sc.msn.com:http ESTABLISHED
TCP kenpc:3166 www.cisco.com:http ESTABLISHED

C:\>
```

Protocolo utilizado

4.1.6 Segmentación y reensamblaje: Divide y vencerás

Un capítulo anterior explicaba cómo se construyen las PDU enviando datos de una aplicación a través de los varios protocolos para crear una PDU que luego se transmite en el medio. En el host de destino, este proceso se invierte hasta que los datos puedan enviarse a la aplicación.

Algunas aplicaciones transmiten grandes cantidades de datos; en algunos casos, varios 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 en función de la tecnología de la red y del medio físico específico que se utiliza.

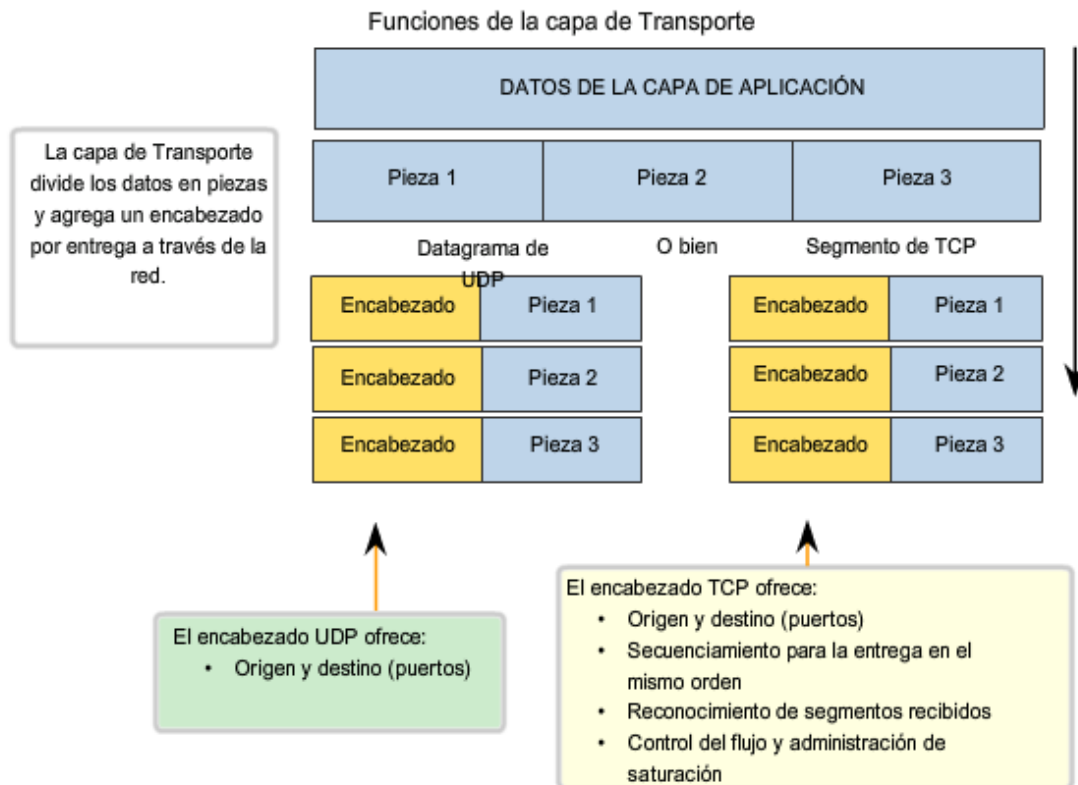
Dividir los datos de aplicación en secciones garantiza que los datos se transmitan dentro de los límites del medio y que los datos de distintas aplicaciones puedan ser multiplexados en el medio.

TCP y UDP gestionan la segmentación de forma distinta.

Con 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 que fueron transmitidos. Esto asegura que la aplicación de destino cuente con los datos en la forma exacta en la que se enviaron.

A pesar de que los servicios que utilizan UDP también rastrean las conversaciones entre aplicaciones, no tienen en cuenta el orden en el que se transmitió la información ni el mantenimiento de la 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 el que fueron enviados.



4.2 Protocolo TCP: Comunicación con confiabilidad

4.2.1 TCP: Cómo generar conversaciones 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 TCP completa requiere el establecimiento de una sesión entre los hosts en ambas direcciones.

Luego de establecida la sesión, 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 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. También existen cargas adicionales en los hosts individuales, generadas por la necesidad de mantener un seguimiento de los segmentos que esperan acuse de recibo y por el proceso de retransmisión.

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

Campos del encabezado del segmento de TCP

Bit 0	15			31
Número de puerto de origen			Número de puerto de destino	
Número de secuencia				
Número de acuse de recibo				
Longitud del encabezado	(Reservado)	Señalizadores	Tamaño de la ventana	
Checksum de TCP			Señalador urgente	
Opciones (si las hay)				
Datos.....				

4.2.2 Procesos del servidor TCP

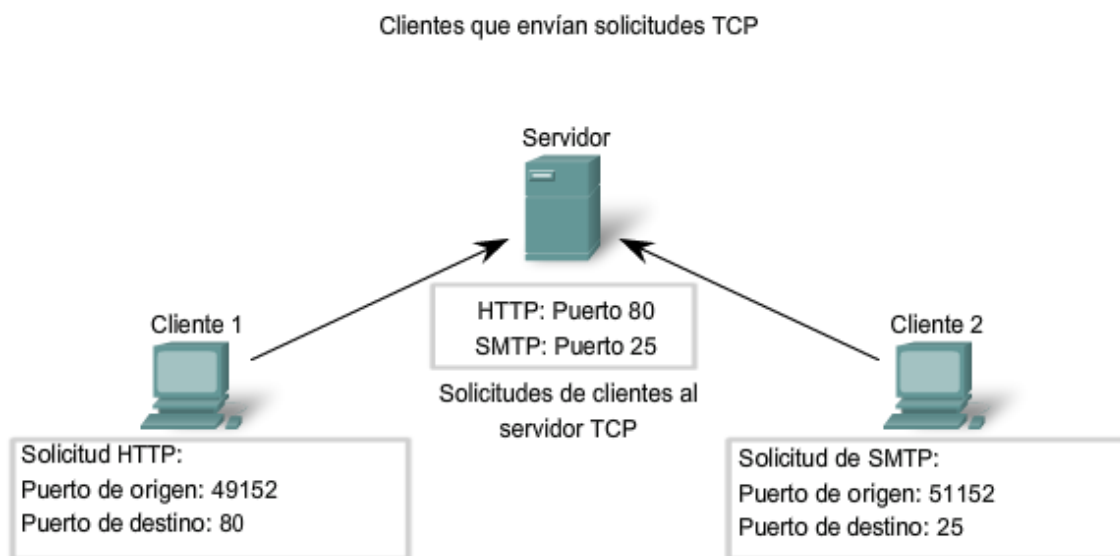
Como se explicó en el capítulo anterior, los procesos de aplicación se ejecutan en servidores. Estos procesos esperan hasta que un cliente inicie comunicación con una solicitud de información o de otros servicios.

Cada proceso de aplicación que se ejecuta en el servidor es configurado por el administrador del sistema para utilizar un número de puerto, de forma predeterminada o manual. **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 8.080). 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 que un servidor provea más de un servicio, 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 a sólo aquellos puertos asociados con los servicios y aplicaciones accesibles a solicitantes autorizados.

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



4.2.3 Establecimiento y finalización de la conexión 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 de sesión habilitan la función de confiabilidad de TCP.

Ver la figura para observar los pasos para establecer y finalizar una conexión TCP.

El host rastrea cada segmento de datos dentro de una sesión e intercambia información sobre los datos recibidos por cada host a través de la información del encabezado TCP.

Cada conexión representa dos streams de comunicación de una vía o sesiones. Para establecer la conexión los hosts realizan un [intercambio de señales de tres vías](#). Los bits de control en el encabezado TCP indican el progreso y estado de la conexión. Enlace de tres vías:

- Establece que el dispositivo de destino esté presente en la red.

- Verifica que el dispositivo de destino tenga un servicio activo y esté aceptando las peticiones en el número de puerto de destino que el cliente que lo inicia intente usar para la sesión.
- Informa al dispositivo de destino que el cliente de origen intenta establecer una sesión de comunicación en ese número de puerto.

En conexiones TCP, el host que brinde el servicio como cliente inicia la sesión al servidor. Los tres pasos para el establecimiento de una conexión TCP son:

1. El cliente que inicia la conexión envía un segmento que contiene un valor de secuencia inicial, que actúa como solicitud para el servidor para comenzar 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, además de 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.

Para entender el proceso de enlace de tres vías, es importante observar los distintos valores que intercambian los dos 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: Urgente campo de señalizador significativo,

ACK: Campo significativo de acuse de recibo,

PSH: Función de empuje,

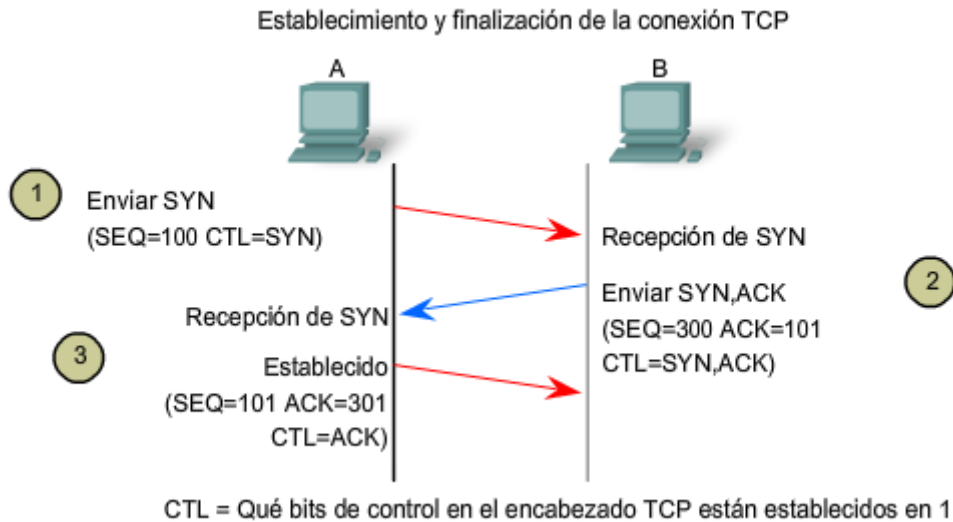
RST: Reconfiguración de la conexión,

SYN: Sincronizar [números de secuencia](#),

FIN: No hay más datos desde el emisor.

A estos campos se los denomina señaladores porque el valor de uno de estos campos es sólo de 1 bit, entonces tiene sólo dos valores: 1 ó 0. Si el valor del bit se establece en 1, indica la información de control que contiene el segmento.

Si se utiliza un proceso de cuatro pasos, los señaladores se intercambian para finalizar la conexión TCP.



4.2.4 Protocolo TCP de enlace de tres vías

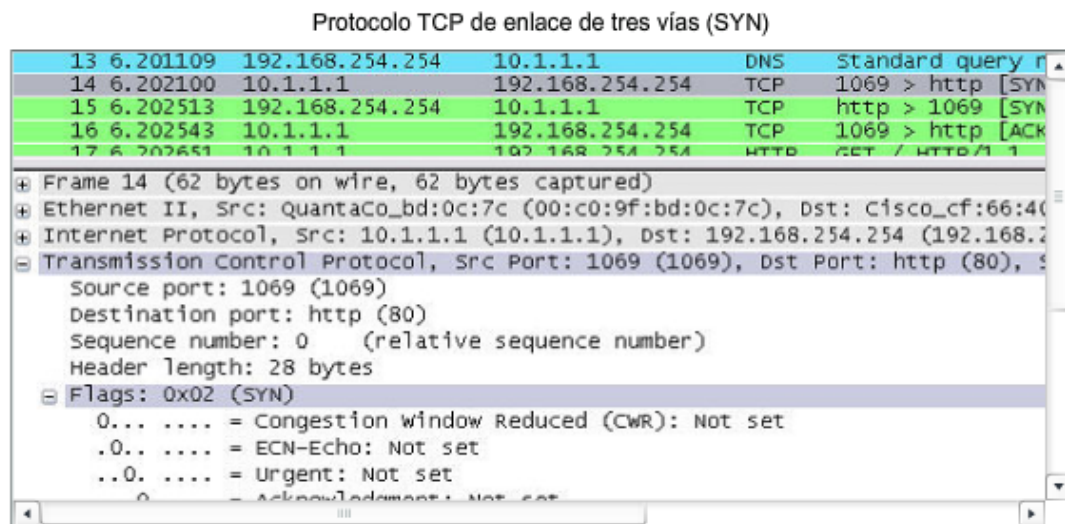
Con los resultados Wireshark, podrá examinar el funcionamiento del protocolo TCP de enlace de tres vías:

Paso 1

Un cliente TCP comienza el enlace de tres vías enviando un segmento con el señalizador de control SYN (Sincronizar números de secuencia) establecido, indicando un valor inicial en el campo de número de secuencia del 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 [hexadecimal](#).



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

El segmento TCP en esta trama muestra:

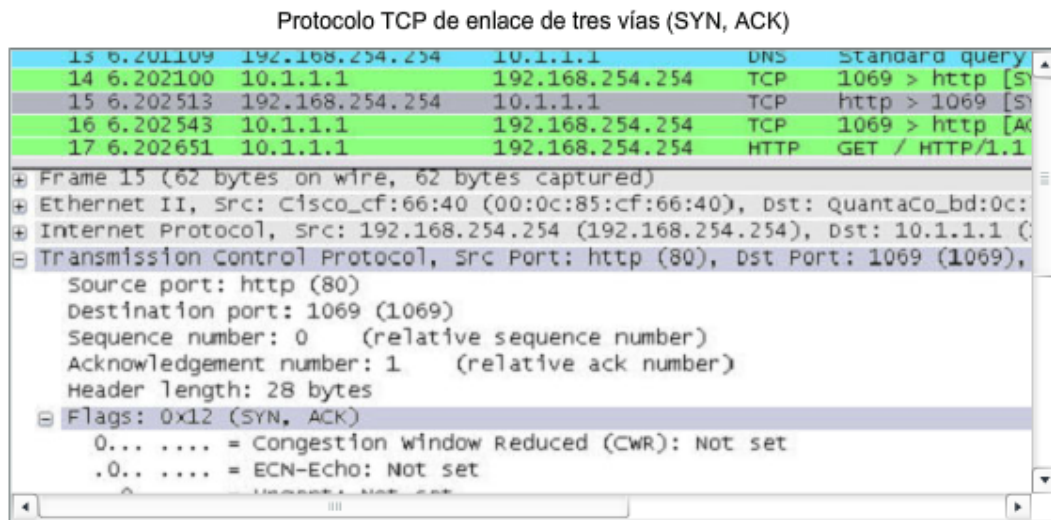
- El señalizador SYN establecido para validar un número de secuencia inicial
- Número de secuencia aleatorio válido (el valor relativo es 0)
- Puerto de origen aleatorio 1069
- El puerto de destino conocido es 80 (puerto HTTP) según indica el servidor Web (httpd)

Paso 2

El servidor TCP necesita reconocer la recepción 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 número de campo 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 utilizará para rastrear el flujo de datos en esta sesión del servidor al cliente.

Como se muestra en la figura, el resultado del analizador de protocolos muestra que están establecidos los señalizadores de control ACK y SYN y se muestran los números relativos de secuencia y reconocimiento.



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 establecidas ambas sesiones entre el cliente y el servidor, todos los segmentos adicionales que se intercambien en la comunicación tendrán establecido el señalizador ACK.

Como se muestra en la figura, el resultado del analizador de protocolos muestra el señalizador de control ACK establecido y se muestran los números relativos de secuencia y reconocimiento.

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

- denegar el establecimiento de sesiones TCP,
- sólo permitir sesiones para ser establecidas por servicios específicos, o
- sólo permitir tráfico como parte de sesiones ya establecidas.

Esta seguridad puede implementarse para todas las sesiones 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 re
14	6.202100	10.1.1.1	192.168.254.254	TCP	1069 > http [SYN]
15	6.202513	192.168.254.254	10.1.1.1	TCP	http > 1069 [SYN,
16	6.202543	10.1.1.1	192.168.254.254	TCP	1069 > http [ACK]
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
 + 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: 1069, Win: 0, 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

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)

4.2.5 Terminación de la sesión TCP

Para cerrar la conexión se debe establecer el señalizador 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 se usan los términos cliente y servidor como referencia por simplicidad pero la finalización del proceso puede ser iniciada por cualquiera de los dos hosts que completan la sesión:

1. Cuando el cliente no tiene más datos para enviar al stream, envía un segmento con el señalizador FIN establecido.
2. El servidor envía un ACK para acusar recibo de Fin y terminar la sesión del cliente al servidor.
3. El servidor envía un FIN al cliente para finalizar la sesión del servidor al cliente.
4. El cliente responde con un ACK para dar acuse de recibo de FIN desde el servidor.

Cuando la finalización de sesión del cliente no tiene más datos para transferir, establece el señalizador 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 produce el acuse de recibo de todos los segmentos, se cierra la sesión.

La sesión en la otra dirección se cierra mediante 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 de retorno confirma que todos los bytes de datos han sido recibidos y, por lo tanto, se ha cerrado la sesión.

Como se muestra en la figura, los señalizadores de control FIN y ACK se establecen en el encabezado del segmento, cerrando por lo tanto la sesión HTTP.

También es posible terminar la conexión mediante 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.

Terminación de la sesión TCP (FIN)

19	6.203857	192.168.254.254	10.1.1.1	HTTP	HTTP/1.1 200 OK (C...
20	6.203876	192.168.254.254	10.1.1.1	TCP	http > 1069 [FIN, ...]
21	6.203899	10.1.1.1	192.168.254.254	TCP	1069 > http [ACK]
22	6.204139	10.1.1.1	192.168.254.254	TCP	1069 > http [FIN, ...]
23	6.204416	192.168.254.254	10.1.1.1	TCP	http > 1069 [ACK]
24	6.602668	10.1.1.1	192.168.254.254	DNS	Standard query A...

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: (00:0c:85:cf:66:40)	
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, Win: 0, 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	

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

Terminación de la sesión TCP (ACK)

19	6.203857	192.168.254.254	10.1.1.1	HTTP	HTTP/1.1 200 OK (C...
20	6.203876	192.168.254.254	10.1.1.1	TCP	http > 1069 [FIN, ...]
21	6.203899	10.1.1.1	192.168.254.254	TCP	1069 > http [ACK]
22	6.204139	10.1.1.1	192.168.254.254	TCP	1069 > http [FIN, ...]
23	6.204416	192.168.254.254	10.1.1.1	TCP	http > 1069 [ACK]
24	6.602668	10.1.1.1	192.168.254.254	DNS	Standard query A...

Frame 21 (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: 414, Win: 0, Len: 0	
Source port:	1069 (1069)
Destination port:	http (80)
Sequence number:	414 (relative sequence number)
Acknowledgement number:	441 (relative ack number)
Header length:	20 bytes
Flags:	0x10 (ACK)
0... .. = Congestion Window Reduced (CWR): Not set	

Un analizador de protocolo muestra los detalles de la trama 21, respuesta TCP ACK.

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

4.3 Administración de sesiones TCP

4.3.1 Reensamblaje de segmentos TCP

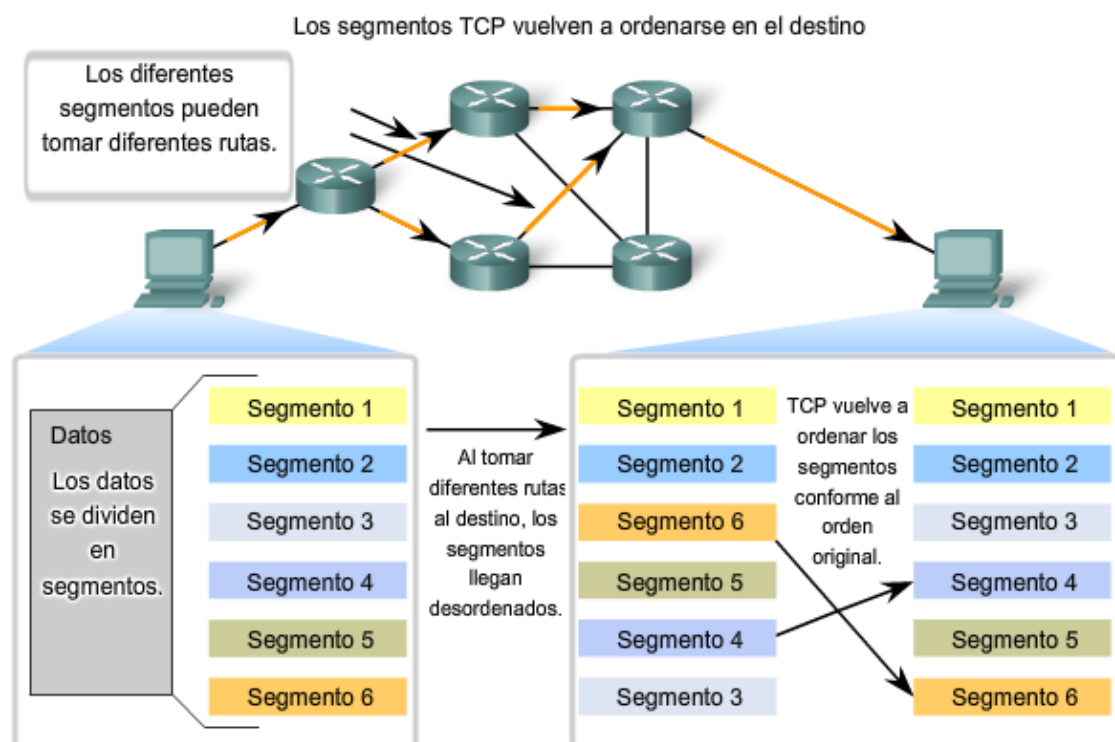
Resecuenciamiento de segmentos al orden transmitido

Cuando los servicios envían datos utilizando TCP, los segmentos pueden llegar a destinos desordenados. 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 TCP receptor 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 son reensamblados. Todos los segmentos que llegan con números de secuencia no contiguos se mantienen para su procesamiento posterior. Luego, se procesan los segmentos cuando llegan con los bytes perdidos.



4.3.2 Acuse de recibo de TCP con uso de ventanas

Confirmación de recepción de segmentos

Una de las funciones de 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 reconocimiento 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 especificado por el número de acuse de recibo. Se espera que el host emisor envíe un segmento que utiliza un número de secuencia igual al número de acuse de recibo.

Recuerde que cada conexión se representa en realidad por dos sesiones de una vía. Los números de secuencia y de acuse de recibo se intercambian en ambas direcciones.

En el ejemplo de la figura, el host en 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.

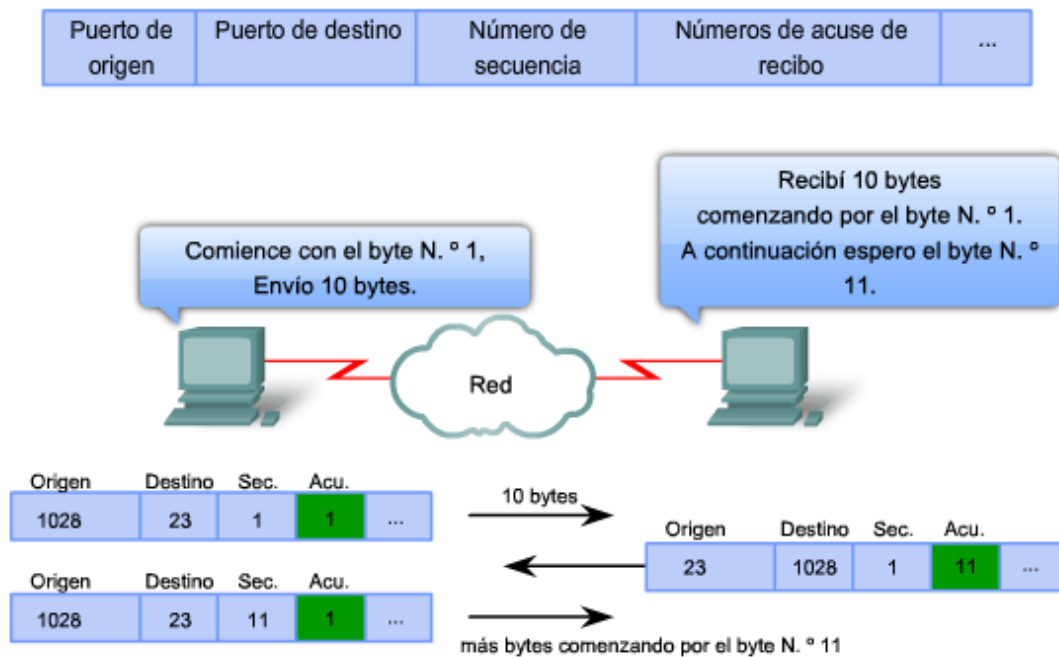
Cuando el host emisor de la izquierda recibe este acuse de recibo, puede enviar el próximo segmento que contiene datos para esta sesión a partir del byte 11.

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 reconocimiento igual a 12001.

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

Acuse de recibo de segmentos TCP



4.3.3 Retransmisión de TCP

Manejo de la pérdida de segmentos

Por 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 los mismos existe un mecanismo para retransmitir segmentos con datos no reconocidos.

Un servicio de host de destino que utiliza TCP, por lo general sólo reconoce datos para secuencias de bytes contiguas. 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 reciben los segmentos con números de secuencia de 1500 a 3000 y de 3400 a 3500, el número de acuse de recibo será 3001. Esto sucede porque existen segmentos con números de secuencia de 3001 a 3399 que no se recibieron.

Cuando TCP en el host de origen no recibe un acuse de recibo pasado un tiempo predeterminado, volverá al último número de acuse de recibo que recibió y retransmitirá los datos a partir de éste.

El proceso de retransmisión no es especificado por RFC, sino que depende de la implementación de TCP en particular.

Para una implementación de TCP típica, un host puede transmitir un segmento, colocar una copia del segmento 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 actuales también suelen emplear 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.

4.3.4 Control de congestión de TCP: Cómo minimizar la pérdida de segmentos

Control del flujo

TCP también provee 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 el origen advierte que se recibió la cantidad de datos especificados en los segmentos, puede continuar enviando más datos para esta sesión.

El campo Tamaño de la ventana en el encabezado TCP especifica la cantidad de datos que puede transmitirse antes de que se reciba el acuse de recibo. El tamaño de la ventana inicial se determina durante el comienzo de la sesión a través del enlace de tres vías.

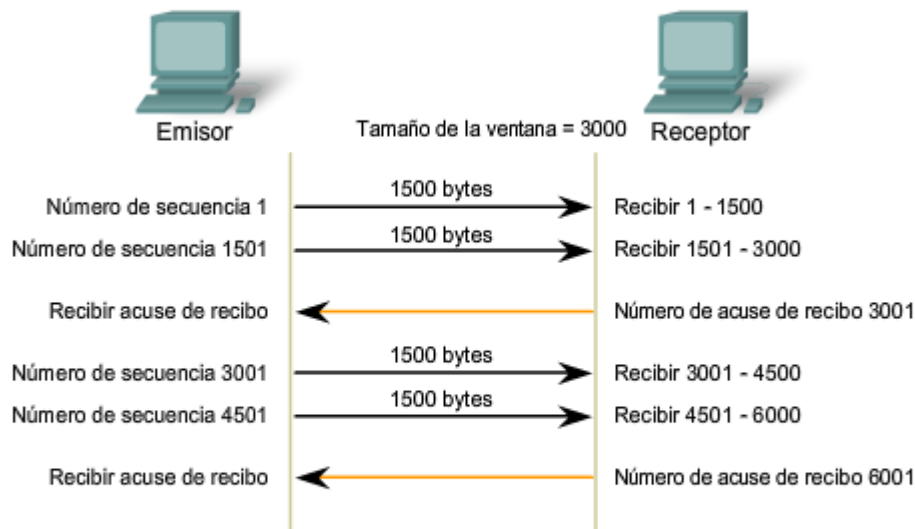
El mecanismo de retroalimentación de TCP ajusta la tasa de transmisión de datos efectiva al flujo máximo que la red y el dispositivo de destino pueden soportar sin sufrir pérdidas. TCP intenta gestionar la tasa de transmisión de manera que todos los datos se reciban y se reduzcan las retransmisiones.

Ver la figura para obtener 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 ha recibido este acuse de recibo del receptor, ya puede transmitir 3000 bytes adicionales.

Durante la demora 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 tasa de 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 dinámicos de ventana. 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 disminuye de manera efectiva la tasa de transmisión, ya que el origen espera que los datos sean recibidos con más frecuencia.

El host receptor 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 tasa de comunicación debido a limitaciones de memoria del búfer, puede enviar un valor de tamaño de la ventana menor al origen como parte de un acuse de recibo.

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

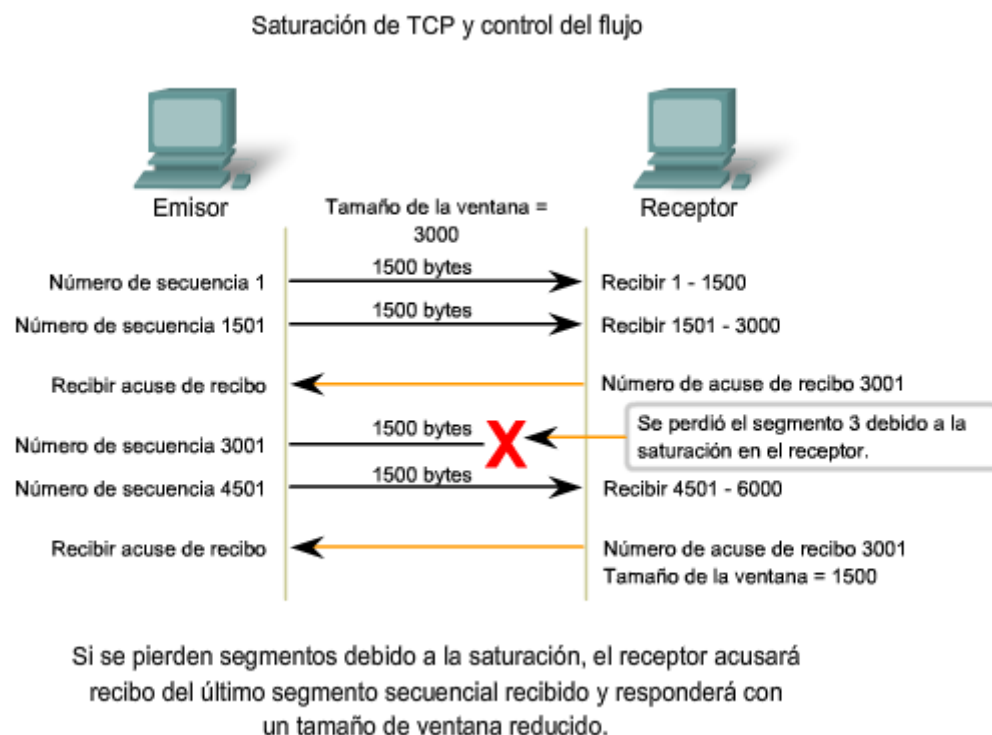
Después de períodos de transmisión sin pérdidas de datos o recursos limitados, el receptor comenzará a aumentar el tamaño de la ventana. Esto reduce la sobrecarga de la red, ya que se requiere enviar menos acuses de recibo. El tamaño de la ventana continuará aumentando hasta que haya pérdida de datos,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

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

Enlaces

Detalles de las varias características de administración de la congestión de TCP se pueden encontrar en RFC 2581.

<http://www.ietf.org/rfc/rfc2581.txt>



4.4 Protocolo UDP: Comunicación con baja sobrecarga

4.4.1 UDP: Baja sobrecarga vs. Confiabilidad

UDP es un protocolo simple que provee las funciones básicas de la capa de Transporte. Genera mucho menos sobrecarga que TCP, ya que no es orientado a la conexión y no cuenta con los sofisticados mecanismos de retransmisión, secuenciación y control del flujo.

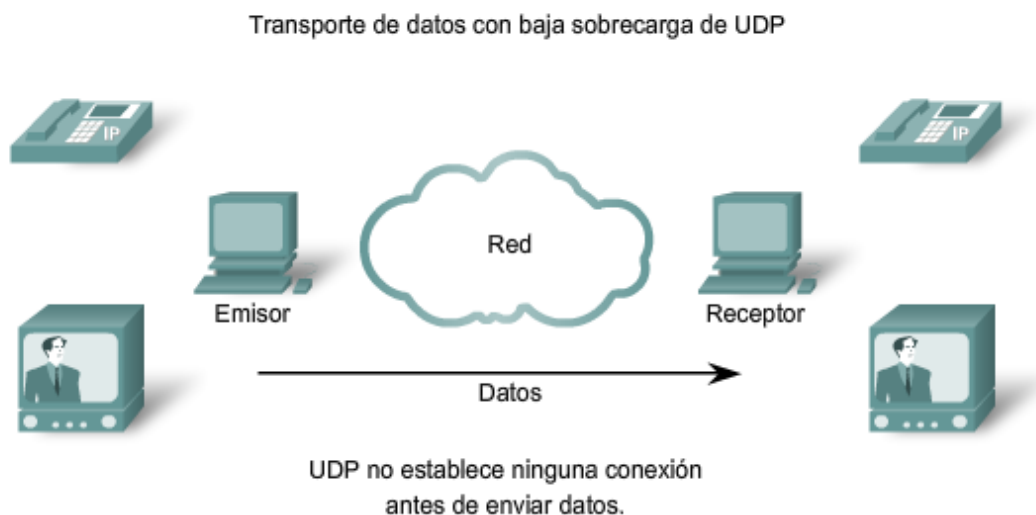
Esto no significa que las aplicaciones que utilizan UDP no sean confiables. Sólo quiere decir que estas funciones no son contempladas por el protocolo de la capa de Transporte y deben implementarse aparte, si fuera necesario.

Pese a que es relativamente baja la cantidad total de tráfico UDP que puede encontrarse en una red típica, entre los protocolos principales de la capa de Aplicación que utilizan UDP se incluyen:

- sistema de denominación de dominio (DNS),
- protocolo simple de administración de red (SNMP),
- protocolo de configuración dinámica de host (DHCP),
- protocolo de información de enrutamiento (RIP),
- protocolo trivial de transferencia de archivos (TFTP), y
- juegos en línea.

Algunas aplicaciones como los juegos en línea o VoIP pueden tolerar algunas pérdida de 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 reintentan enviar la solicitud si no obtienen respuesta y, por lo tanto, no necesitan TCP para garantizar la entrega del mensaje.

La baja sobrecarga de UDP lo hacen deseable para dichas aplicaciones.



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

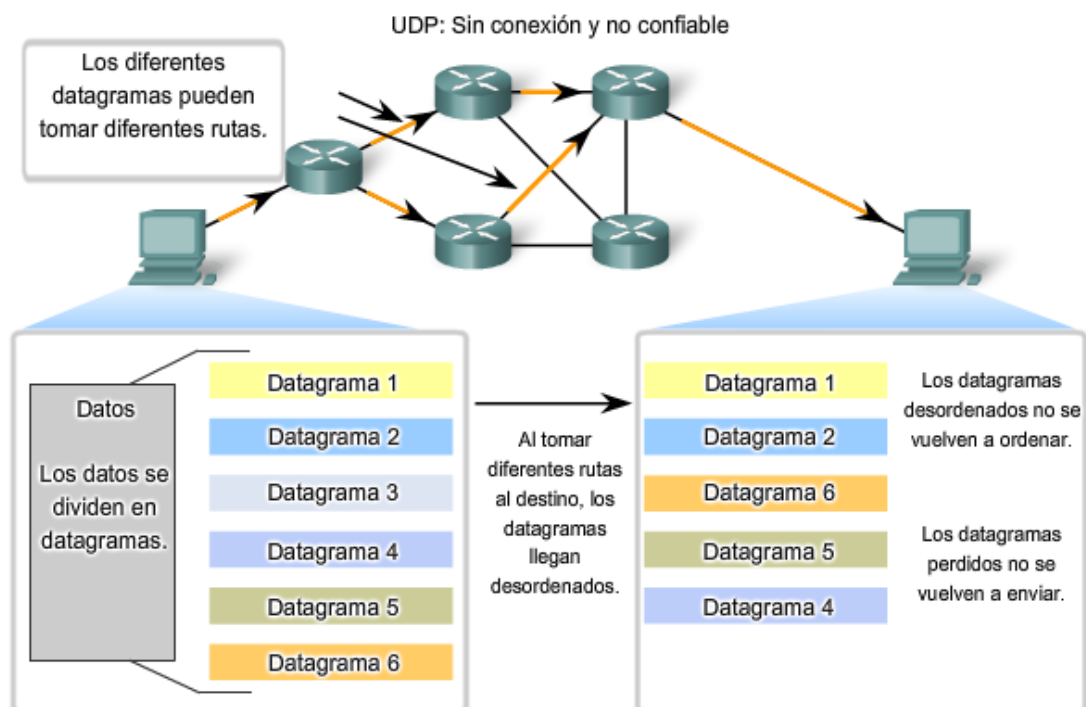
4.4.2 Reensamblaje de datagramas de UDP

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 posee datos para enviar, simplemente los envía.

Muchas aplicaciones que utilizan UDP envían pequeñas cantidades de datos que pueden ocupar un segmento. Sin embargo, algunas aplicaciones enviarán cantidades mayores de datos que deben dividirse en varios segmentos. La PDU de UDP se conoce como *datagrama*, pese a que los términos *segmento* y *datagrama* a veces se utilizan de manera indistinta para describir una PDU de la capa de Transporte.

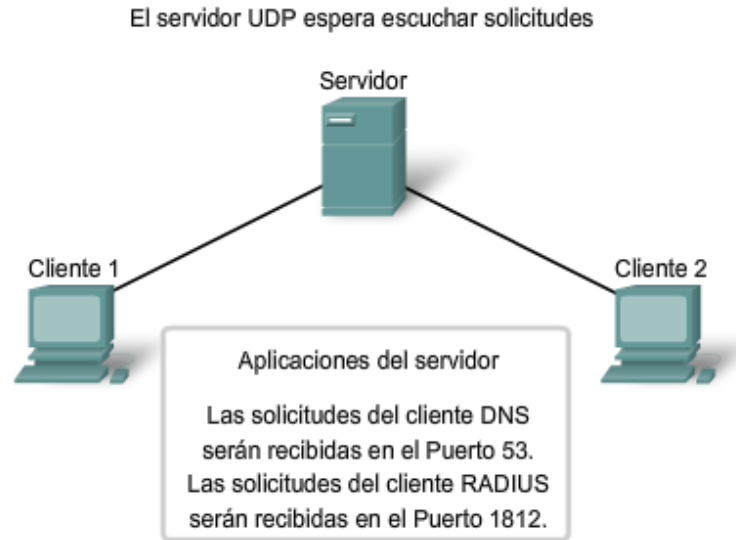
Cuando se envían múltiples datagramas a un destino, los mismos pueden tomar rutas distintas y llegar en el orden incorrecto. UDP no mantiene un seguimiento de los números de secuencia de la manera en que lo hace TCP. UDP no puede reordenar los datagramas en el orden de la transmisión. Ver la figura.

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 de datos y determinar cómo procesarlos.



4.4.3 Procesos y solicitudes del servidor UDP

Al igual que las aplicaciones basadas en TCP, a las aplicaciones de servidor basadas en UDP se les asigna 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.



Las solicitudes de clientes a servidores utilizan números de puerto bien conocidos como puerto de destino.

4.4.4 Procesos de cliente UDP

Como en TCP, la comunicación cliente/servidor se inicia por 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 ser enviados y los puertos estén identificados, UDP puede formar el datagrama y enviarlo a la capa de Red para direccionamiento y envío 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.