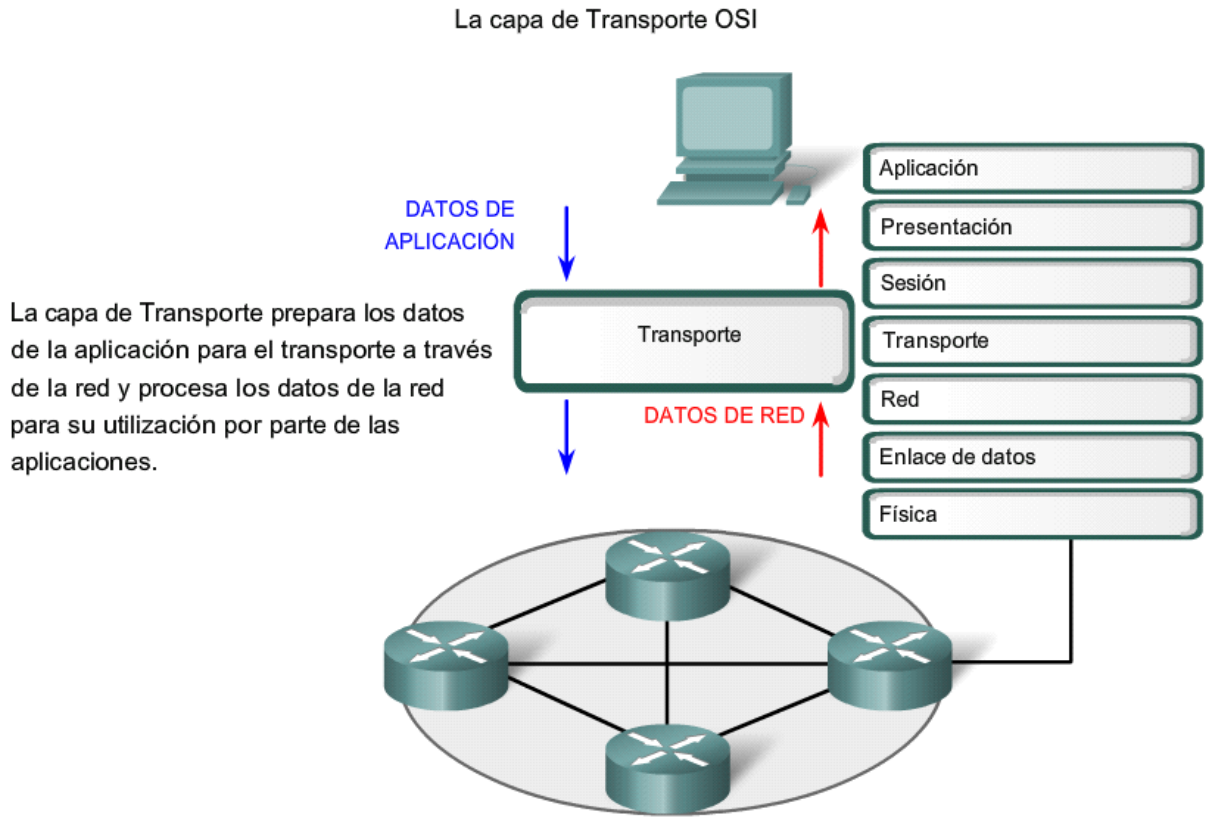
Capitulo IV - 4.0.1 La capa de transporte del modelo OSI

Los procesos descriptos en la capa de Transporte aceptan los datos procedentes de las aplicaciones y los prepara su direccionamiento en la capa IP.
La capa de Transporte es responsable de la transferencia de extremo a extremo de los archivos de la capa de Aplicación.
Funciones de la capa de transporte:

- Permite aplicaciones múltiples para comunicarse en la red al mismo tiempo que en un dispositivo sencillo
- Asegura que, si es necesario, la aplicación correcta reciba todos los datos de forma confiable y en orden
- Emplea mecanismos de manejo de errores

Objetivos del estudio de la capa de transporte:

- Transferencia de datos de extremo a extremo entre las aplicaciones.
- Descripción de las funciones de los protocolos TCP/IP de la capa de transporte: TCP y UDP.
- Explicación de las funciones confiabilidad, direccionamiento de puerto y segmentación.
- Indicar cuando es adecuado el uso del protocolo TCP o UDP.



4.1.1.1 Propósitos de la capa de transporte

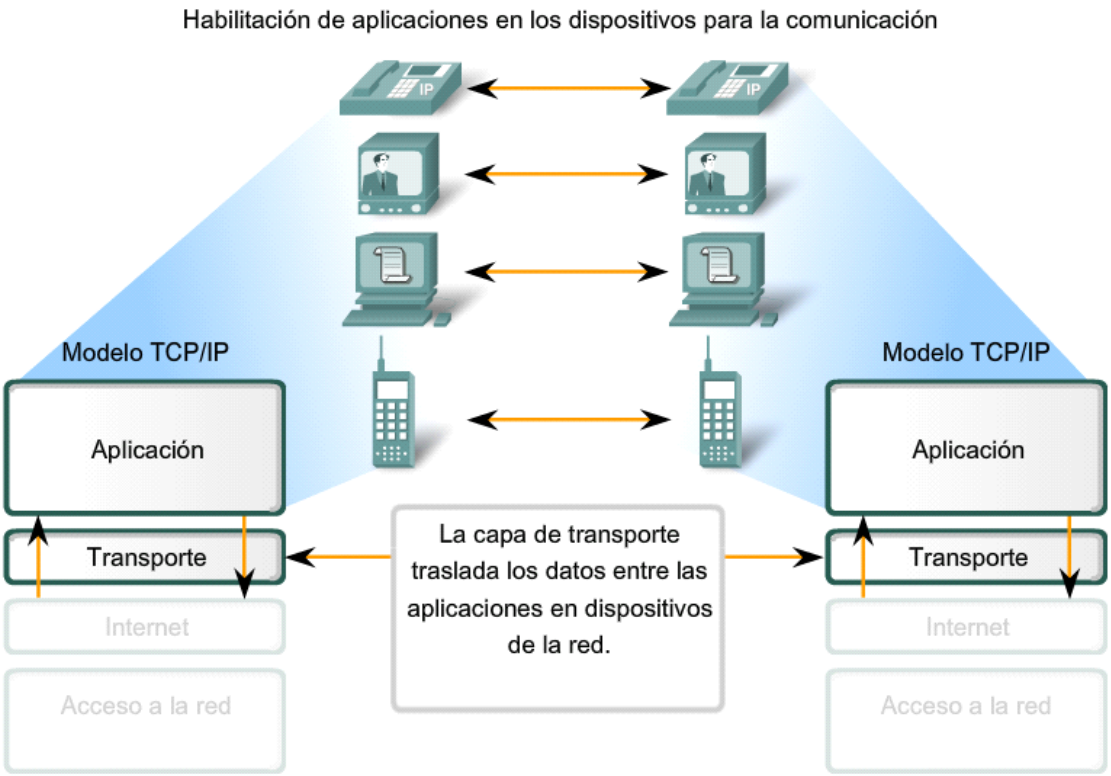
Permite la segmentación de los datos y brinda el control necesario para el re-ensamblaje de los distintos flujos de comunicaciones. Las responsabilidades que debe cumplir esta capa son:

1. 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 flujos de comunicación múltiple entre estas aplicaciones.

2. Segmentación de datos

Así como cada aplicación crea flujos de datos 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.



4.1.1.2 Propósitos de la capa de transporte

3. Re-ensamble 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 flujo 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 re-ensamblar las partes de los datos en flujos para pasarlos a la capa de aplicación.

4. Identificación de las aplicaciones

La capa de Transporte asigna un identificador a cada una de las aplicaciones. Este identificador se denomina número de puerto.

La capa de transporte es el enlace entre la capa de Aplicación y las capas inferiores para la transmisión de cada uno de los segmentos o datagramas. Los datos son multiplexados para su transmisión en la red. Las aplicaciones no necesitan conocer los detalles operativos de la red sobre la cual va a ser enviada la información como por ejemplo: la clase de host de destino, el tipo de medio de transmisión de la red, la congestión en la red ect.

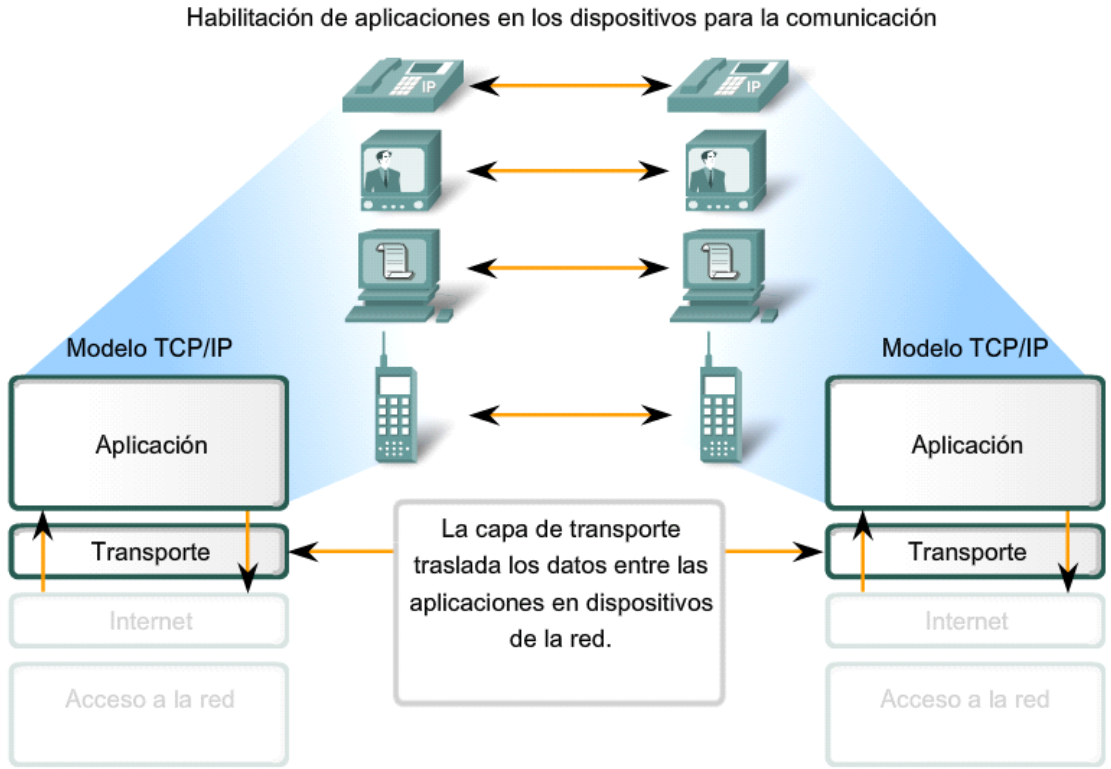
Las capas inferiores no tienen conocimiento que existen varias aplicaciones que envían información a la red.

La capa de transporte clasifica los fragmentos de los datos para su envío a la red conforme a las prioridades establecidas para el envío del flujo de datos a través de los medios.

Los requisitos de los datos varían:

Cada una de las aplicaciones tiene diferentes requisitos. Algunas aplicaciones permiten la pérdida de datos.

Los protocolos tienen distintas reglas para el intercambio de información entre el originador y el destino.



4.1.1.3 Propósitos de la capa de transporte

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 aplicaciones 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.

4.1.1.4 Propósitos de la capa de transporte

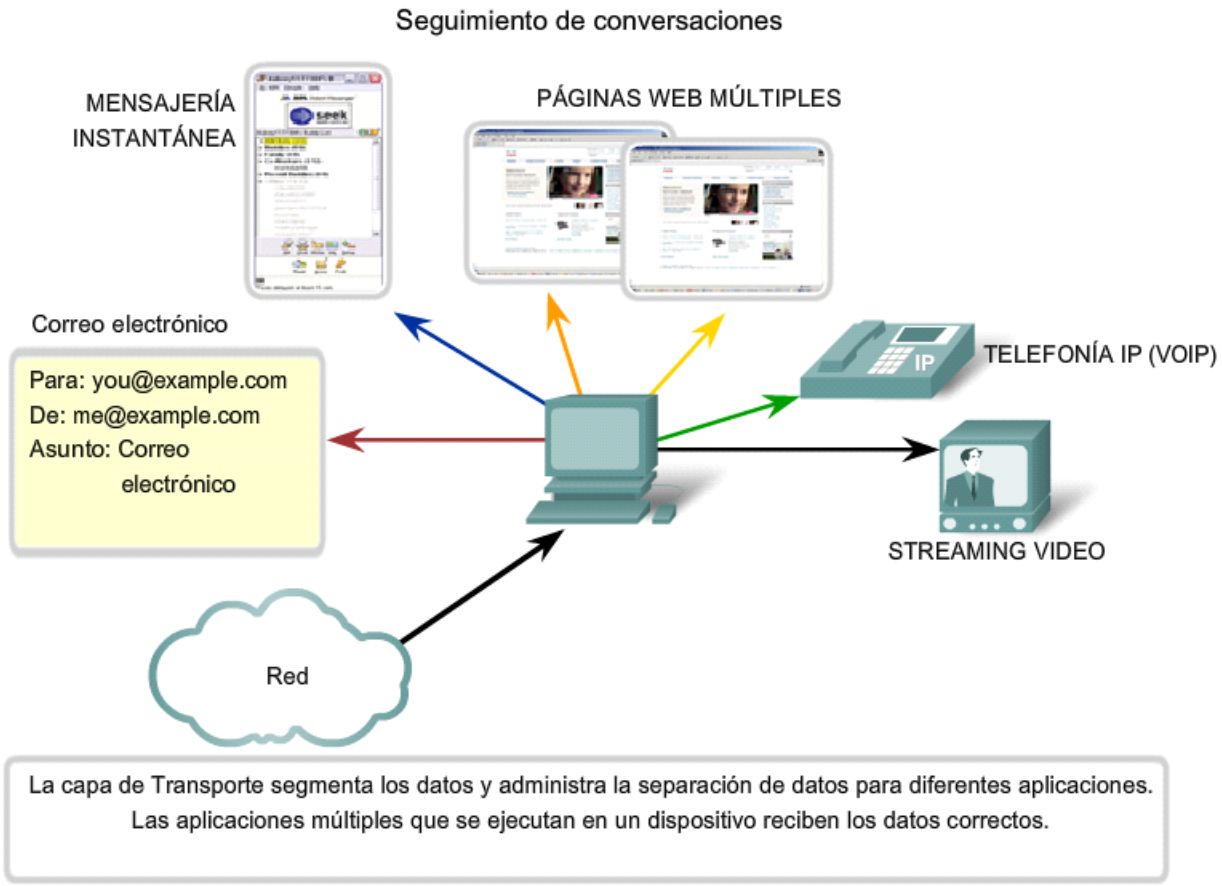
Separación de las comunicaciones múltiples

Cada una de las aplicaciones envía o recibe sus datos independientemente de las otras aplicaciones y de manera simultánea.

Los datos correspondientes a las aplicaciones de correo electrónico y el acceso a los portales pueden funcionar con retardos debido a la congestión de la red o a la repetición de los contenidos.

Los datos correspondientes a las aplicaciones de voz y video son susceptibles a los retardos. La red no gestiona la retransmisión de los contenidos.

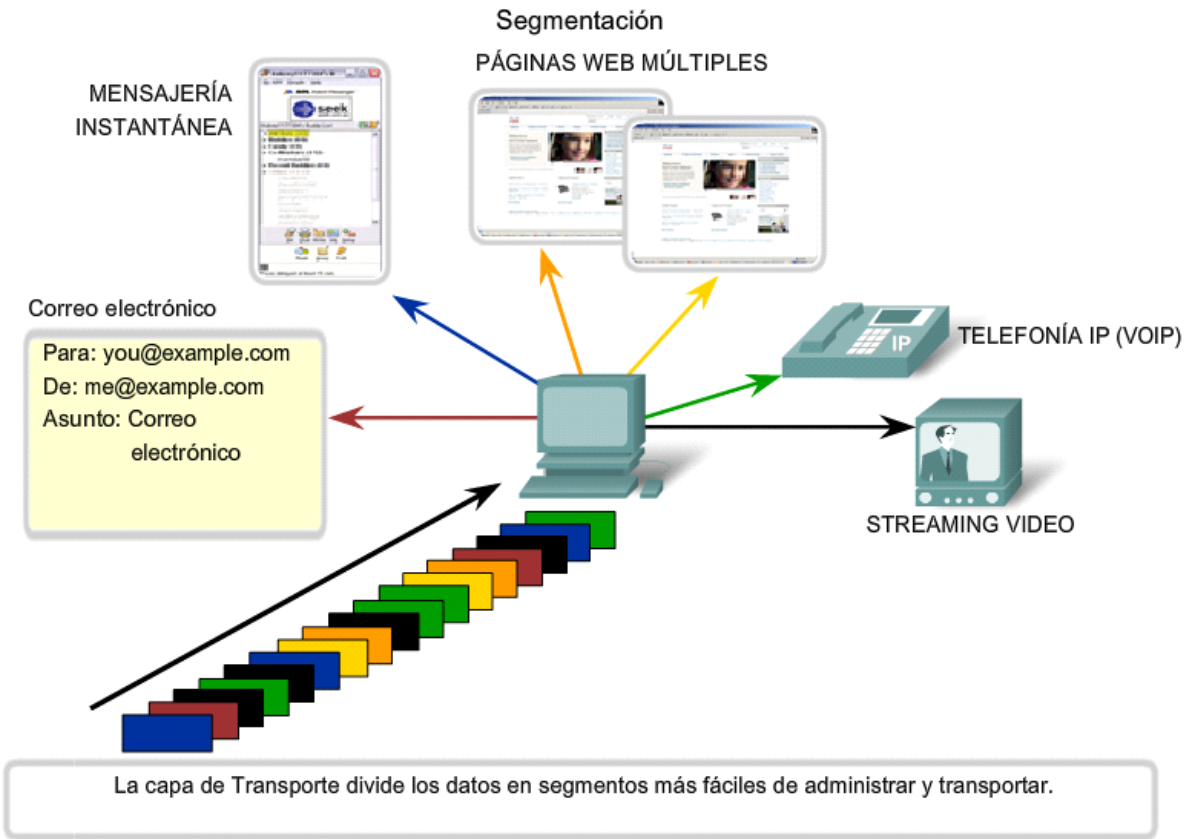
Para el caso de algún retardo o contenido faltante en un mensaje de voz, el interlocutor interpreta el contenido por el contexto del dialogo o solicita la repetición del contenido. La red no realiza gestión alguna tendiente a la recuperación de los contenidos faltantes.



4.1.1.5 Propósito de la capa de transporte

sábado, 07 de abril de 2012 5:08 p. m.

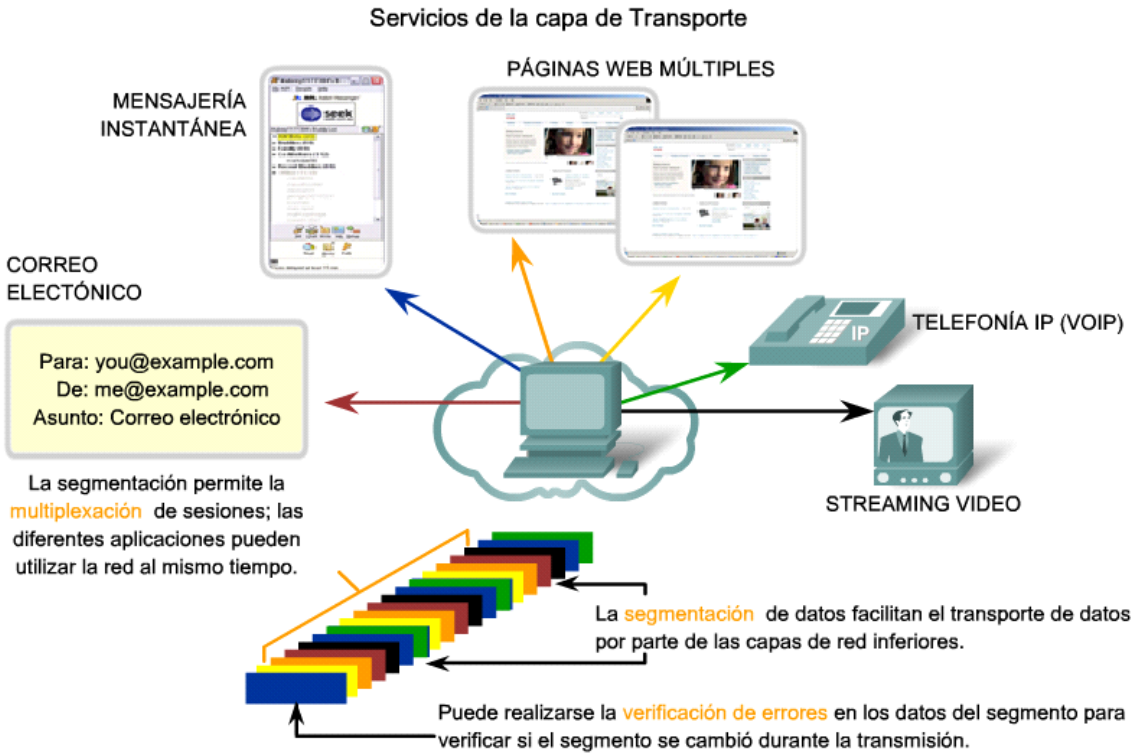
Es necesario evitar que una aplicación use toda red sin permitirle el acceso a las otras aplicaciones aun cuando le pertenezcan mismo usuario.
La capa de transporte realiza la segmentación de los datos procedentes de cada una de las aplicaciones.
La multiplexación permite que los datos de cada uno de los usuarios puedan ser intercalados para la transmisión.
Esto permite que las aplicaciones puedan funcionar de manera concurrente.
En la capa de transporte, cada conjunto de piezas que fluye entre una misma aplicación desde el origen hasta el destino se denomina conversación.
La multiplexación permite que un usuario pueda leer sus archivos mientras tiene un dialogo interactivo a través de una comunicación de voz y video IP.
Para identificar cada pieza fraccionada de información, la capa de transporte le agrega unos campos de cabecera con un objetivo definido.



4.1.2.1 Control de las conversaciones

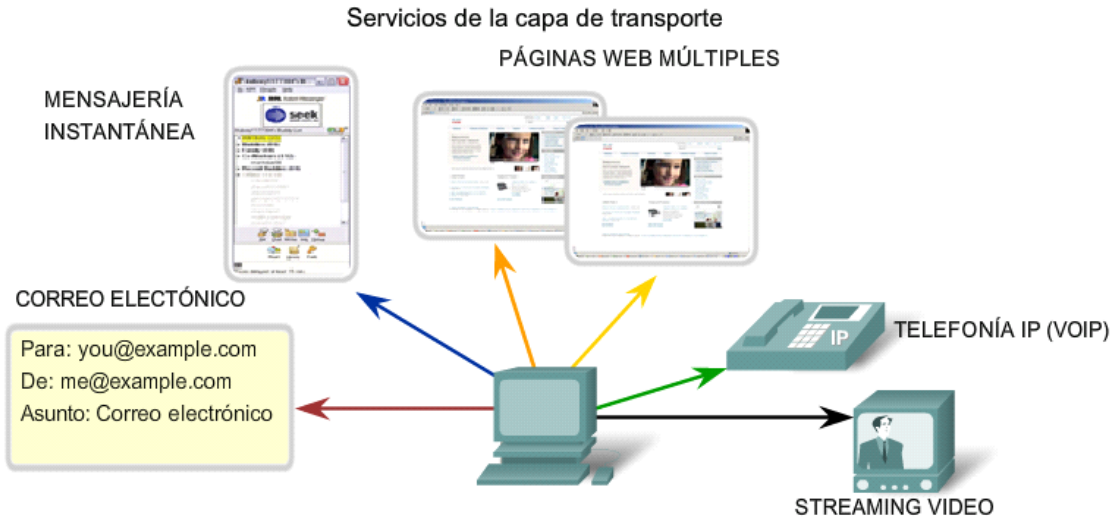
Segmentación y re-ensamble: la mayoría de las redes tienen una limitación en la cantidad de datos que se pueden incluir en una simple PDU. La capa de transporte divide los datos de aplicación en bloques de datos de un tamaño adecuado. En el destino, la capa de transporte re-ensambla los datos antes de 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 re-ensamble de datos algunos protocolos en la capa de transporte proporcionan:

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



4.1.2.2 Control de las conversaciones

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.



El **establecimiento de una sesión** asegura que la aplicación esté lista para recibir los datos.

La **entrega confiable** implica el reenvío de segmentos perdidos para que se reciban los datos en forma completa.

La **entrega en el mismo orden** asegura que los segmentos se vuelvan a formar en el orden correcto.

El **control del flujo** administra la entrega de datos si se observa saturación en el host.

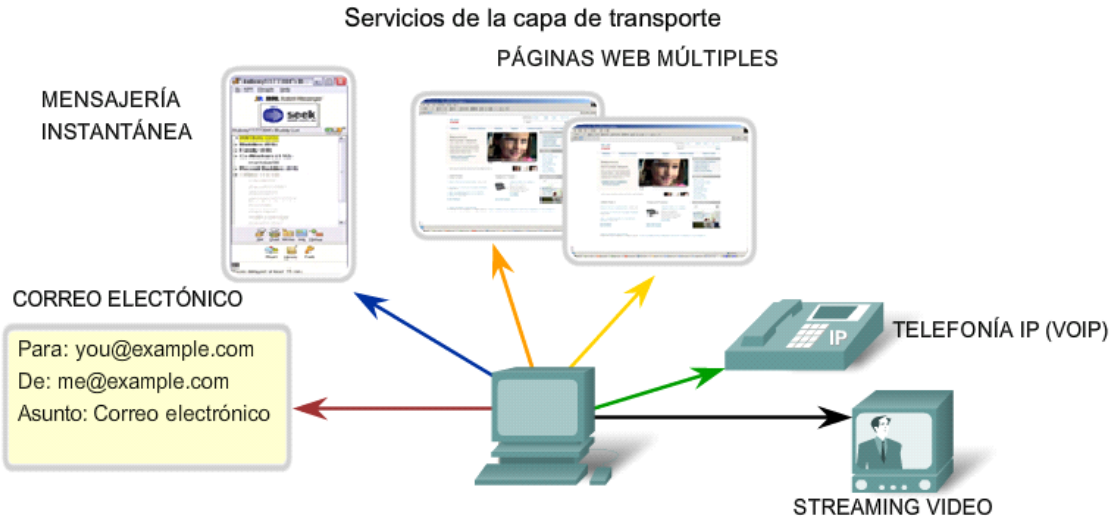
4.1.2.3 Control de las conversaciones

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 re-ensamblen 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.



El **establecimiento de una sesión** asegura que la aplicación esté lista para recibir los datos.

La **entrega confiable** implica el reenvío de segmentos perdidos para que se reciban los datos en forma completa.

La **entrega en el mismo orden** asegura que los segmentos se vuelvan a formar en el orden correcto.

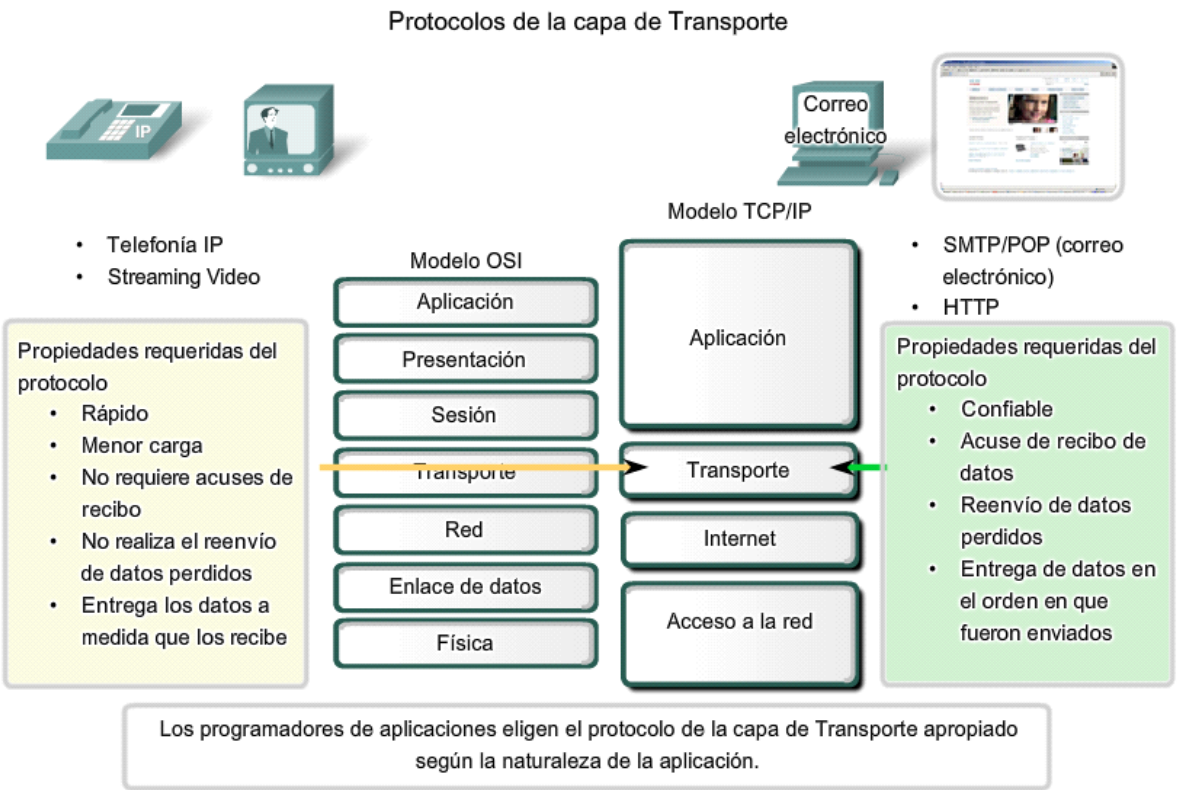
El **control del flujo** administra la entrega de datos si se observa saturación en el host.

4.1.3.1 Soporte de una comunicación confiable

4.1.3.1 Soporte de una comunicación confiable
Cada una de las aplicaciones tiene sus propios requisitos para la transmisión y recepción de sus datos. Se han desarrollado diferentes clases de protocolos que cumplan con los requisitos de cada una de las aplicaciones.
El protocolo debe permitir el envío confiable de los datos de extremo a extremo de la comunicación.
Las tres operaciones básicas para un envío confiable son:

- Rastreo de los datos transmitidos.
- Acuse de recibo de los datos recibidos.
- Retransmisión de cualquier dato sin acuse de recibo.

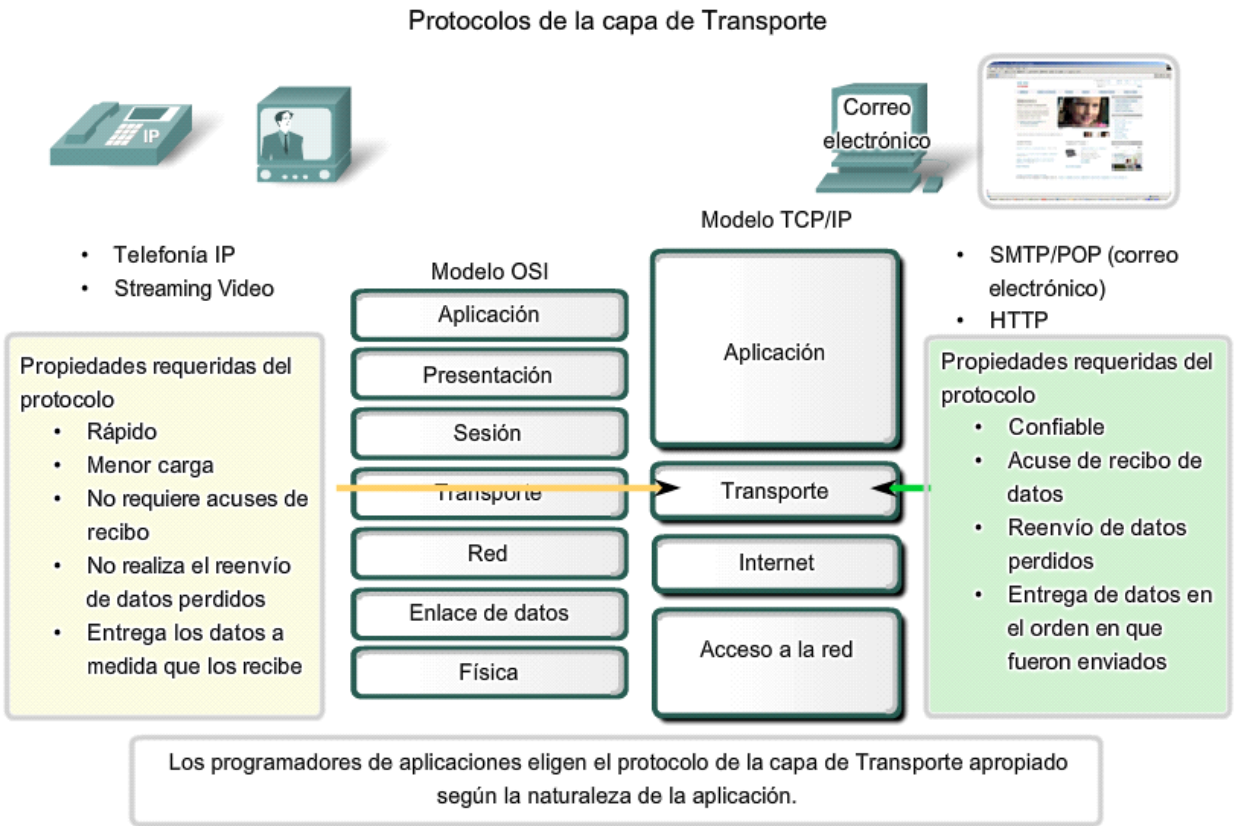
La confiabilidad requiere que el host emisor realice un rastreo de todas las partes de datos de una conversación y proceda a la retransmisión de cualquier dato que el host de destino no acusó recibo.
La confiabilidad también requiere que el host de destino realice un rastreo de todas las partes de datos de una conversación y reconozca la recepción de cada parte por medio de un acuse de recibo.
Estos procesos de confiabilidad generan un uso adicional de los recursos de la red para el rastreo y retransmisión de los datos. Es necesario un incremento en el volumen de los datos de control de extremo a extremo. Esta información de control se encuentra en la cabecera de cada parte de un mensaje.
La confiabilidad crea una sobrecarga en la red. Los desarrolladores de aplicaciones deben determinar el tipo de protocolo requerido para cada tipo de aplicación.
Existen protocolos de entrega confiable y protocolos de entrega no confiable de los datos.
El envío de la información con el mejor esfuerzo está clasificado como no confiable debido a que no hay un acuse de recibo de la información enviada hacia el destino.



4.1.3.2 Soporte de una comunicación confiable

Determinación de la necesidad de la confiabilidad

Las aplicaciones tales como correo electrónico, páginas web y en general bases de datos necesitan que todos los datos enviados lleguen a destino para que sean útiles.
Si la información se recibe incompleta, algunas partes de una aplicación dejan de funcionar adecuadamente.
En este caso se debe utilizar un protocolo que garantice la entrega integral de la información sin error alguno.
Existen otras aplicaciones como la transmisión de voz y video en la cual no tiene transcendencia la pérdida de algunos fragmentos del contenido transmitido. El usuario final lo puede interpretar del contexto o no detecta esta situación.
En este caso se utiliza un protocolo que no sobrecargue la red.



4.1.4 TCP y UDP

Los dos protocolos más comunes en la capa de transporte son el Protocolo de Datagrama del Usuario UDP y el Protocolo de Control de Transmisión. Ambos protocolos gestionan la comunicación de las aplicaciones en la capa de Transporte.

1) Protocolo de datagrama del usuario UDP

Es un protocolo sin conexión.
No garantiza la entrega de la información en el destino: mejor esfuerzo o mejor intento.
Las porciones de información se denominan datagramas. Ejemplos de aplicaciones UDP:

- Sistemas de nombre de dominio: DNS
- Streaming de video.
- Voz sobre IP.

La cabecera utiliza ocho octetos.

2) Protocolo de control de transmisión TCP

Es un protocolo orientado a la conexión.
Garantiza la entrega de la información: secuenciamiento, control de flujo, retransmisión ect.
Las porciones de información se denominan segmentos. Ejemplos de aplicaciones TCP.

- Exploradores Web.
- Correo electrónico.
- Transferencia de archivos.

La cabecera utiliza 20 octetos.

Encabezados TCP y UDP

Segmento de TCP

Bit (0)	Bit (15)	Bit (16)	Bit (31)
Puerto de origen (16)		Puerto de destino (16)	
Número de secuencia (32)			
Número de acuse de recibo (32)			
Longitud del encabezado (4) Reservado (6) Bits de código (6)		Ventana (16)	
Checksum (16)		Urgente (16)	
Opciones (0 ó 32 si las hay)			
DATOS DE LA CAPA DE APLICACIÓN (el tamaño varía)			

20 Bytes

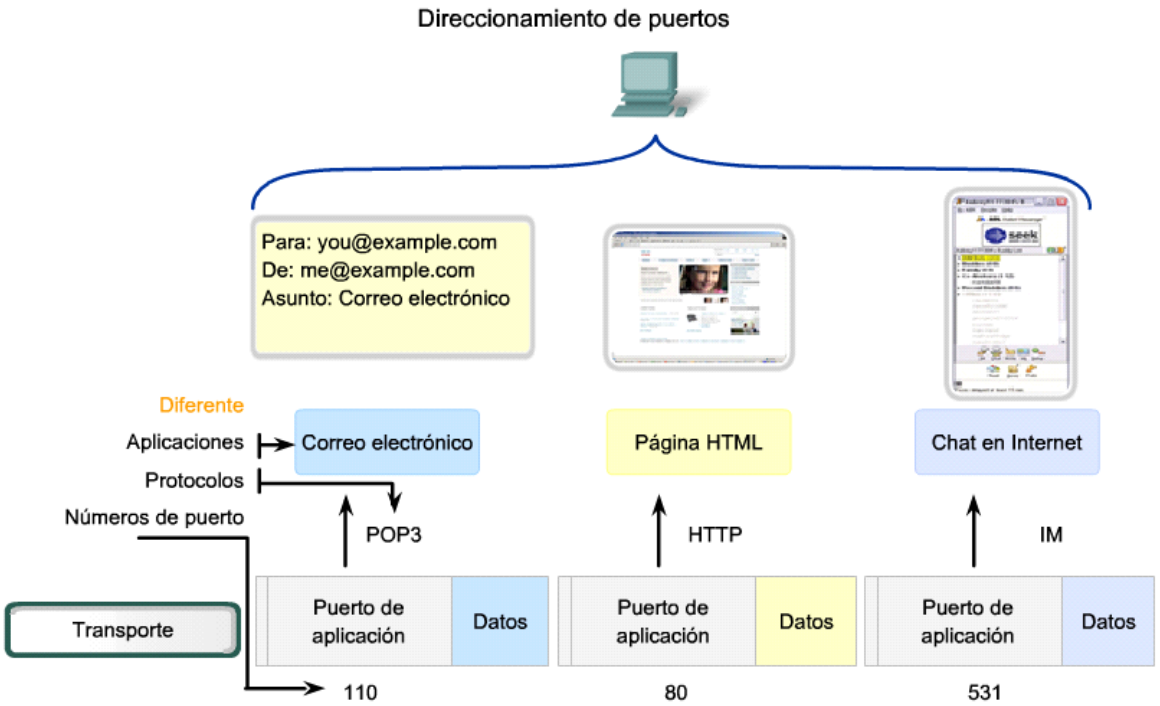
Datagrama de UDP

Bit (0)	Bit (15)	Bit (16)	Bit (31)
Puerto de origen (16)		Puerto de destino (16)	
Longitud (16)		Checksum (16)	
DATOS DE LA CAPA DE APLICACIÓN (el tamaño varía)			

8 Bytes

4.1.5.1 Direccionamiento del puerto

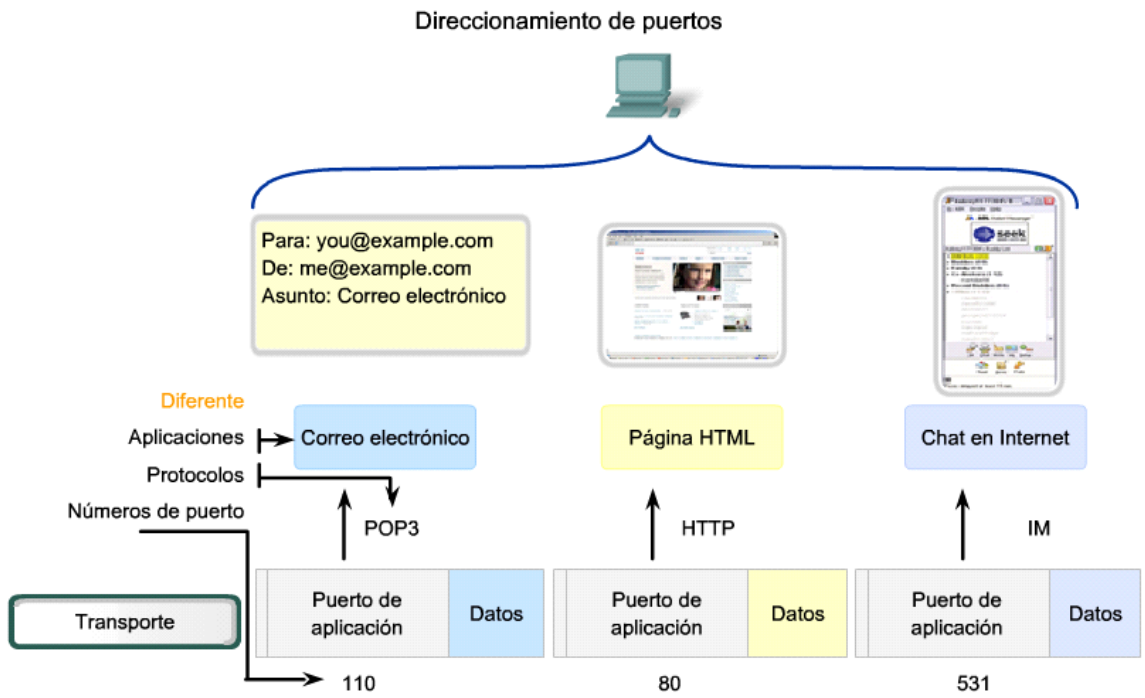
Los servicios basados en los protocolos TCP y UDP mantienen un seguimiento de las diversas comunicaciones que realizan.
Para diferenciar los segmentos y datagramas para cada aplicación, tanto TCP como UDP cuentan con campos de encabezado que pueden identificar de manera exclusiva estas aplicaciones. Estos identificadores únicos son los números de puertos.
En el encabezado de cada segmento o datagrama hay un puerto de origen y un puerto de destino.
El número de puerto de origen es el número para esta comunicación que está asociado con la aplicación que origina la comunicación. El número de puerto de destino es el número para esta comunicación que está asociado con la aplicación que recibe la comunicación.
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.
Los números de puerto se asignan dependiendo si el mensaje es una solicitud o una respuesta. Los procesos del servidor tienen números estáticos asignados. Los clientes eligen de forma aleatoria un número de puerto asignado para una conversación.



Los datos de las distintas aplicaciones se dirigen a la aplicación correcta, ya que cada aplicación tiene un número de puerto único.

4.1.5.2 Direccionamiento del puerto

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 ejecuta en el servidor remoto.
El software del cliente debe conocer el número de puerto del servidor remoto correspondiente a dicha aplicación.
Los números de puertos para los servidores están estandarizados.
El número de puerto de origen en el encabezado de un segmento o un datagrama de la solicitud de un cliente, se crea de forma aleatoria. Es un número mayor que 1023. El sistema operativo tiene reservado un rango de números de puertos que utiliza aleatoriamente.
El número de puerto de origen creado aleatoriamente en la aplicación del cliente se usa como dirección de retorno de la respuesta del servidor.
La capa de transporte mantiene un seguimiento a la aplicación y al número de puerto de origen para la devolución de la respuesta de la aplicación procedente del servidor.
Un proceso en particular se identifica en una aplicación por medio del protocolo, la dirección IP y el número de puerto.
Un par de sockets identifican una conversación entre los dos hosts.
Ejemplo: un explorador web que solicita a un servidor puede tener asignado el socket {tcp,192.168.1.1:1025}.
Ejemplo: un servidor web que debe entregar una respuesta a un cliente puede tener el socket {tcp,192.168.1.2:80}.



Los datos de las distintas aplicaciones se dirigen a la aplicación correcta, ya que cada aplicación tiene un número de puerto único.

4.1.5.3 Direccionamiento del puerto

Asignados por la IANA, organismo responsable de garantizar el direccionamiento.
Existen diversos tipos de números de puertos.

Puertos bien conocidos

Números de puerto desde el 0 hasta el 1023. Se utilizan en los servidores para los protocolos HTTP, SMTP/POP3, Telnet entre otros.
Las aplicaciones de cliente pueden ser diseñadas para que soliciten una conexión con estos puertos.

Puertos registrados

Números de puertos desde el 1024 hasta el 49151. El usuario puede seleccionar uno de estos números para una determinada aplicación de cliente. Estos números también se usan seleccionados aleatoriamente en el cliente.

Puertos dinámicos o privados

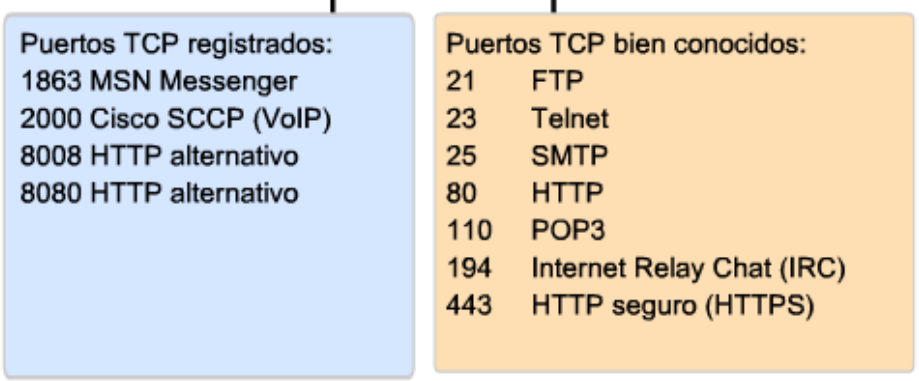
Números de puerto desde el 49152 hasta el 65535. Usados en el cliente. Conocidos como puertos efímeros y asignados de forma dinámica cuando se establece una conexión temporal con otro proceso. Su uso es poco común.

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

4.1.5.4 Direccionamiento del puerto

Puertos TCP

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



4.1.5.4 Direccionamiento del puerto

Puertos UDP

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

Puertos UDP registrados:
1812 Protocolo de autenticación RADIUS
5004 RTP (Voice and Video Transport Protocol)
5060 SIP (VoIP)

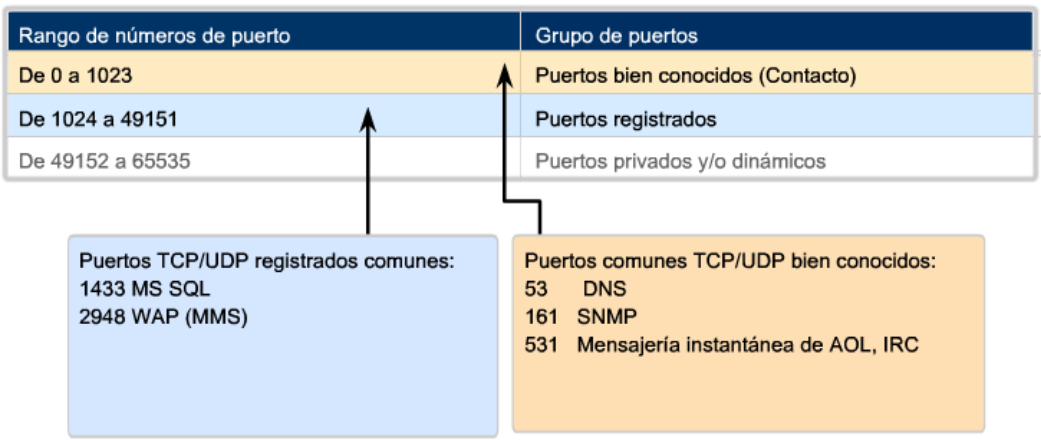
Puertos UDP bien conocidos:
69 TFTP
520 RIP

4.1.5.6 Direccionamiento de puerto

Uso de TCP y UDP

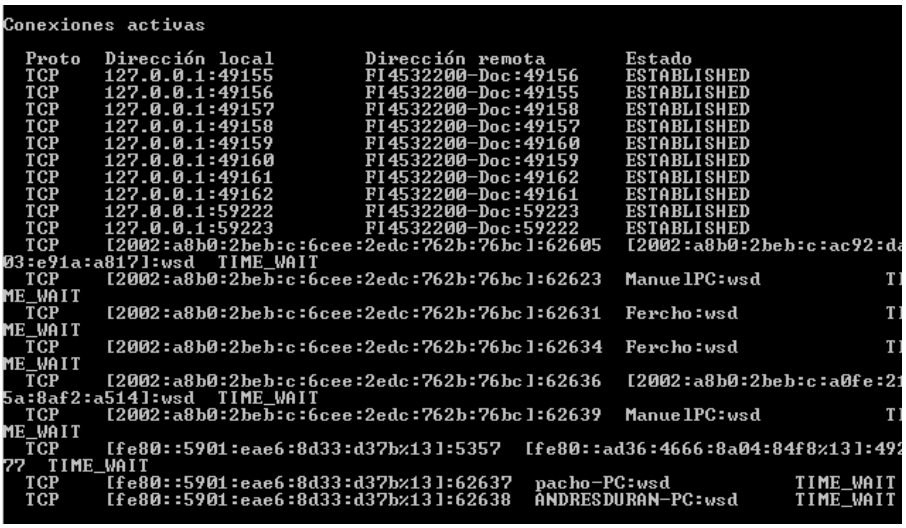
Algunas aplicaciones pueden usar ambos protocolos. Por ejemplo la poca información que transporta una aplicación de DNS y el poco recorrido en una red sea adecuado para el uso de un protocolo con baja sobrecarga con el UDP.

En otros casos el largo recorrido en una red extensa de una aplicación DNS puede ser protegido con un protocolo TCP.



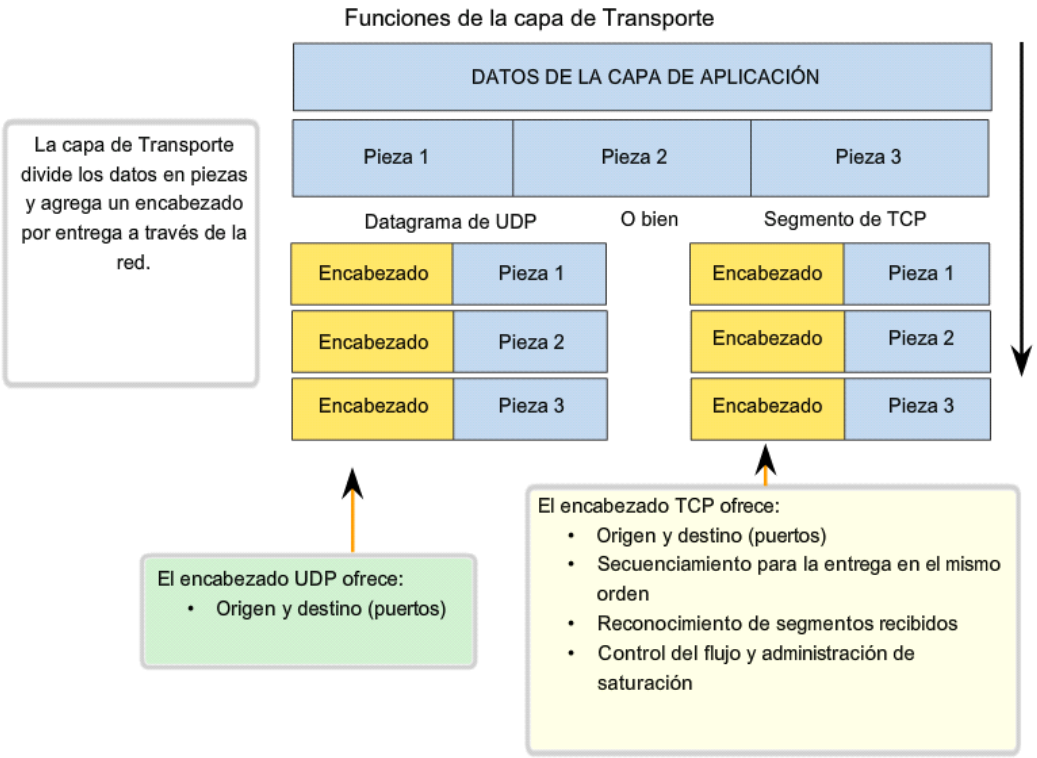
4.1.5.7 Direccionamiento del puerto

El comando Netstat permite conocer las direcciones ip y los números de puertos asociados con cada proceso en un dispositivo terminal.
Es importante para la seguridad, considerando que es posible la localización de procesos asociados a un número de puerto que podrían estar funcionando en el equipo sin autorización.
Es posible la localización y eliminación de procesos sin importancia que consumen el ancho de banda de la red



4.1.6 Segmentación y re-ensamblaje

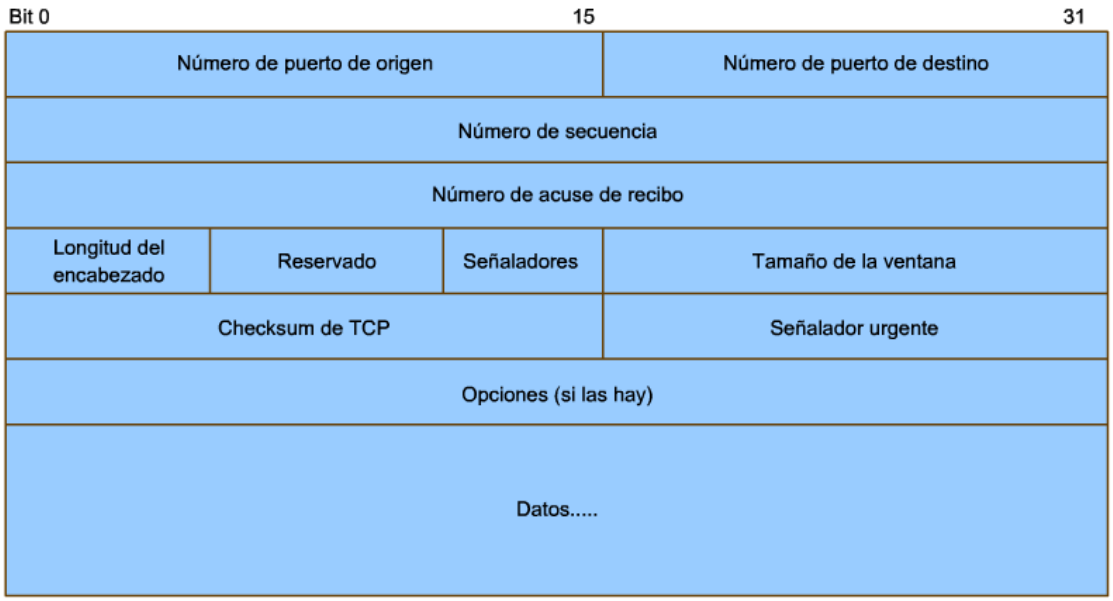
Los datos de la capa de aplicación deben ser segmentados y multiplexados para su transmisión.
Esto evita que un archivo con una gran cantidad de octetos ocupe la red por un tiempo largo, sin compartirlo con otros usuarios.
En caso de algún error en la transmisión del archivo no segmentado, debe de retransmitirse desde el principio.
Los dispositivos de red no están equipados con una gran cantidad de memoria para el almacenamiento temporal de un gran volumen de datos que ingrese a la red.
La división de los datos de aplicación en partes permite la multiplexación.
Segmentación diferente para TCP y UDP
En el protocolo TCP cada encabezado de un segmento contiene un número de secuencia que identifica al segmento transmitido. Los segmentos pueden llegar desordenados al host de destino. El número de secuencia permite el re-ensamblaje de los segmentos desordenados y la recuperación de los segmentos faltantes. La cabecera consta de 20 octetos.
Las cabeceras de los datagramas de un protocolo UDP no contiene un número de secuencia que permita el re-ensamblaje de los datagramas desordenados o faltantes. La cabecera consta de ocho octetos.



4.2.1.1 TCP: Generación de 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 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 individuales 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.

Campos del encabezado del segmento de TCP



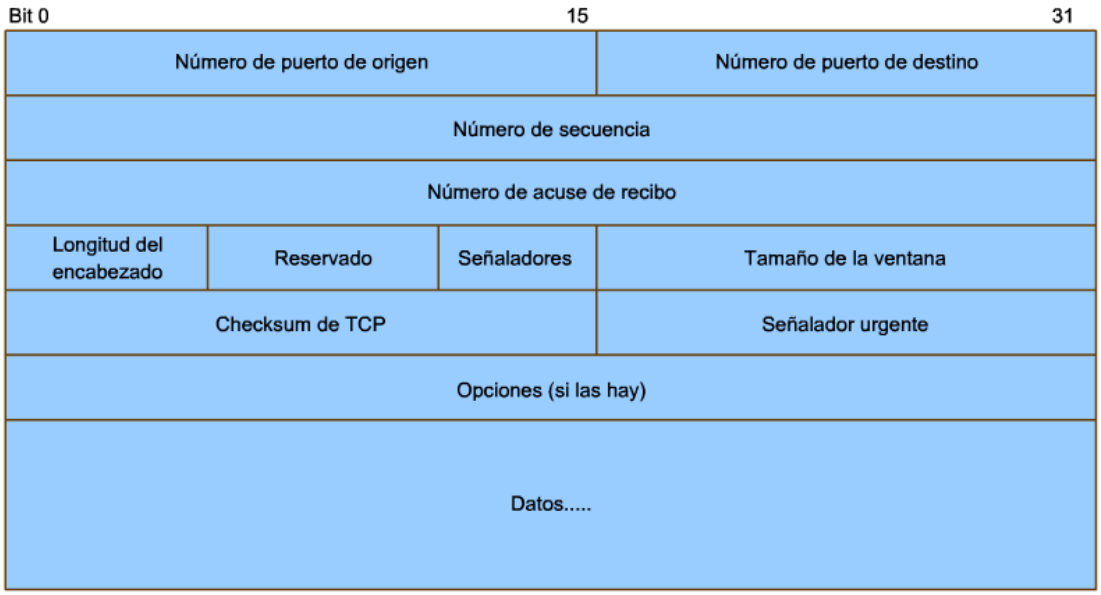
Los campos del encabezado de TCP habilitan TCP para suministrar comunicaciones de datos confiables orientados a la comunicación.

4.2.1.2 TCP: Generación de conversaciones confiables

Significado de los campos

Número de puerto de origen: inicio de la sesión en un host, valor aleatorio mayor que 1023.
Número de puerto de destino: identifica el protocolo o la aplicación de la capa superior del sitio remoto.
Número de secuencia: especifica la posición que ocupa el primer octeto del campo de datos del segmento en relación con el comienzo de flujos de octetos del segmento.
Número de acuse de recibo: especifica el próximo octeto esperado por el receptor.
Longitud del encabezado: longitud del encabezado del segmento en octetos.
Reservado: se establece en cero.
Señaladores: administración de sesiones y el tratamiento de segmentos.
Tamaño de la ventana: cantidad de octetos que deben enviarse antes de esperar el acuse de recibo.
Checksum de TCP: verificación de errores en el encabezado y en los datos.
Señalador urgente: utilizado con una señalización urgente.
Opciones: información opcional.
Datos: datos de aplicación.

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.

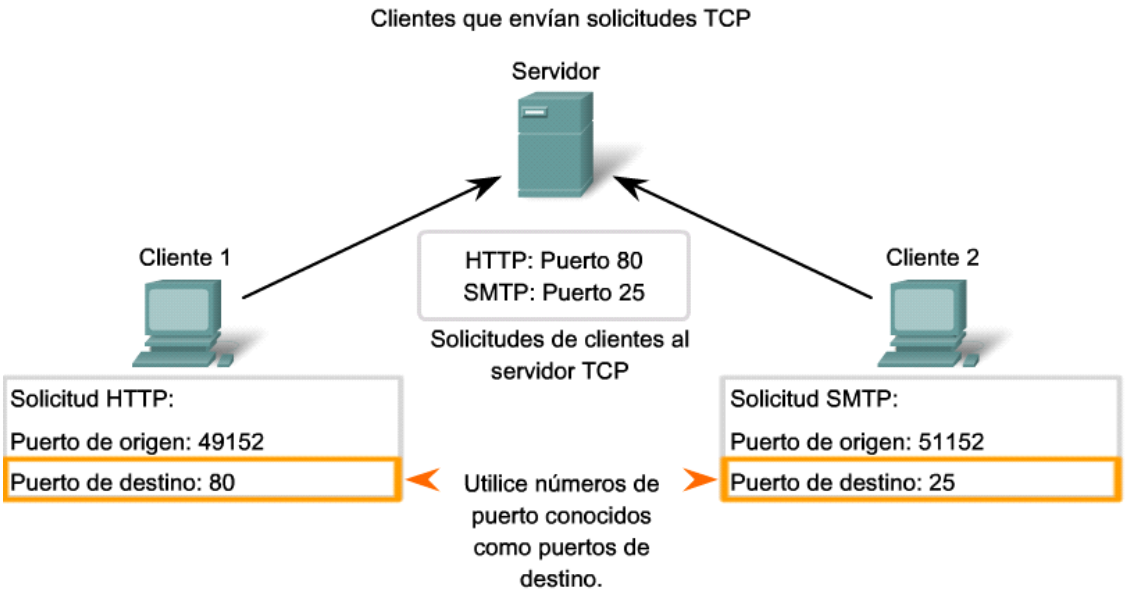
4.2.1.3 TCP: Generación de conversaciones confiables

Los seis bits del campo de control son usados para los siguientes propósitos:
URG: cuando este bit está activo, el campo de segmento urgente es válido.
ACK: cuando este bit está activo, el reconocimiento de un evento TCP es válido.
PSH: cuando este bit está activo, el receptor es obligado a entregar este segmento a la aplicación de recepción en la brevedad posible.
RST: cuando este bit está activo, el receptor es informado que el emisor está cortando la conexión y que todos los datos almacenados y pendientes en una fila para esta conexión pueden ser liberados o abandonados.
SYN: Cuando este bit está activo, se le informa al receptor que el emisor está intentando la sincronización con un número de secuencia.
FIN: cuando este bit está activo, el emisor informa al receptor que alcanzó el final del flujo de bits para la conexión TCP actual y que el receptor no debe esperar más datos.

Urg	Ack	Psh	Rst	Syn	Fin

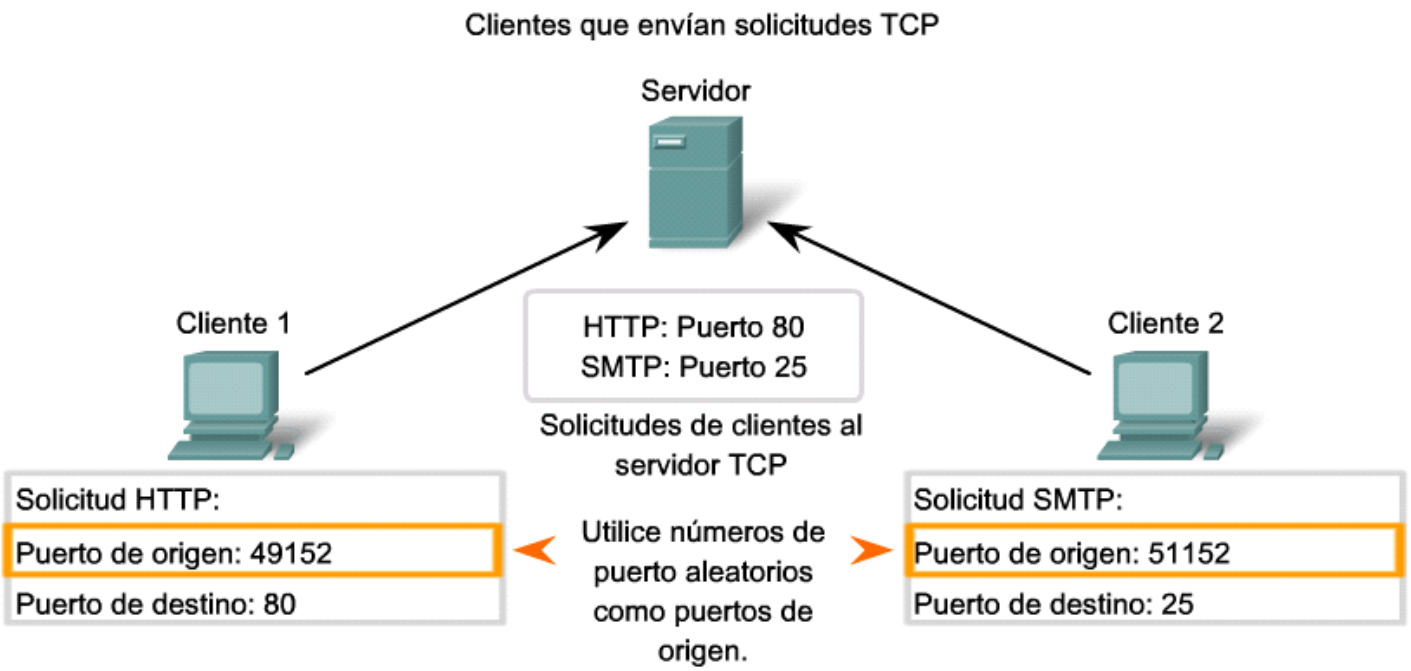
4.2.2.1 Procesos del servidor TCP : solicitud de puertos de destino

Los procesos de aplicación se ejecutan en los servidores.
Un servidor no puede tener dos servicios diferentes asignados a un mismo número de puerto. Ejemplo: HTTP y SMTP al puerto 80. Cada una de estas aplicaciones tiene asignado un número de puerto diferente.
Un servidor puede atender diferentes servicios simultáneamente. Las solicitudes ingresan a la aplicación del servidor con la información del correspondiente socket.
Una manera de mejorar la seguridad de las aplicaciones es restringir el acceso por medio de la habilitación de los números de puertos de usuarios con una identificación establecida en la dirección IP.
Envío de una solicitud TCP



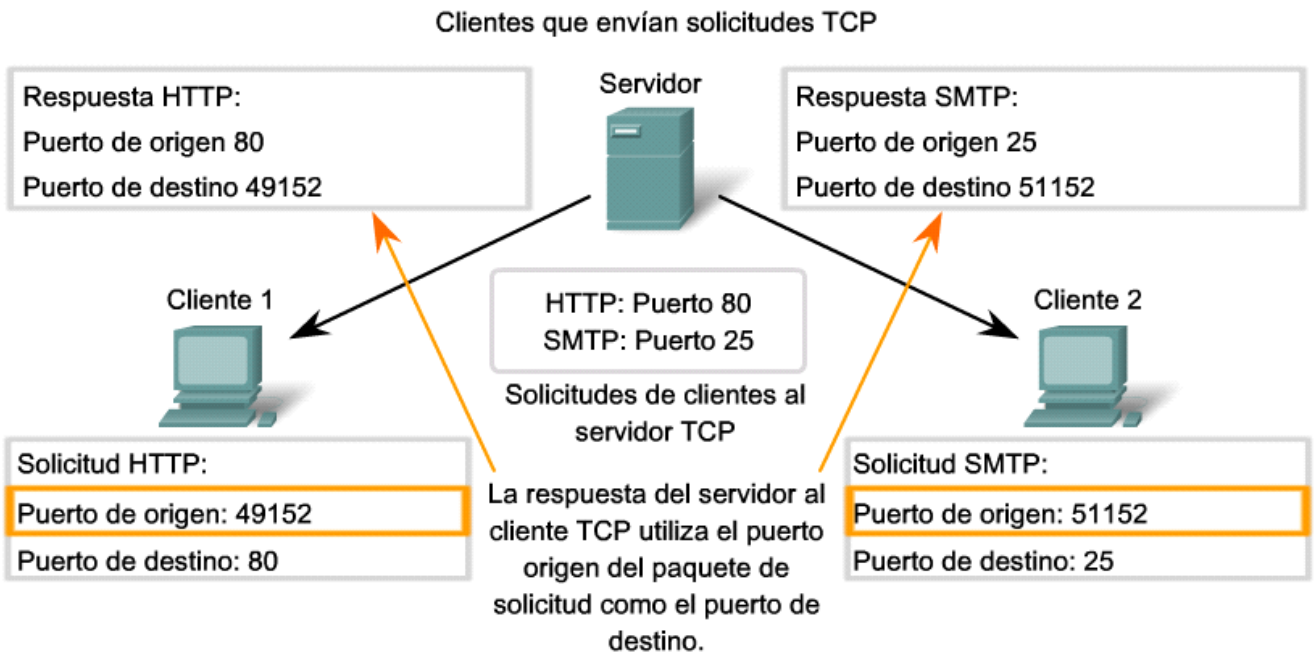
4.2.2.2 Procesos del servidor TCP

Asignación de puertos en el origen en forma aleatoria en cada uno de los clientes.



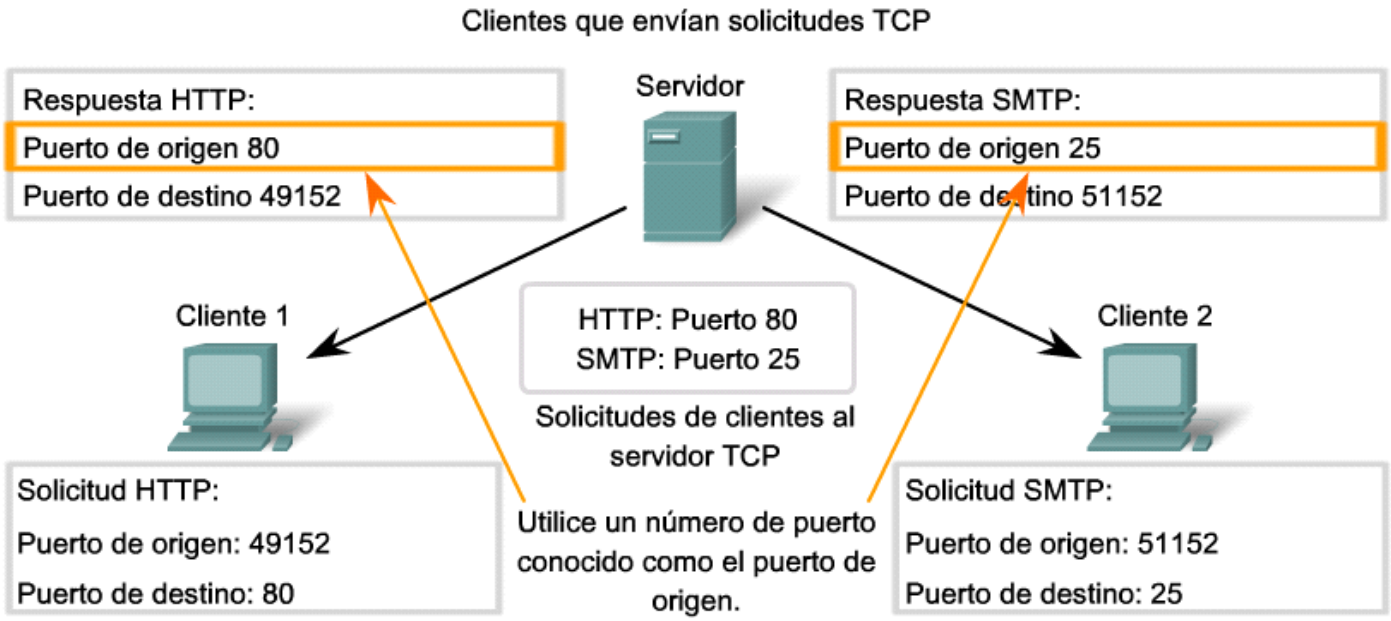
4.2.2.3 Procesos del servidor TCP: Respuesta de puertos de

Respuestas del servidor al cliente. Usa los puertos de origen usados en las transacciones de consulta de los clientes.



4.2.2.4 Procesos del servidor TCP: Respuesta de puertos de origen.

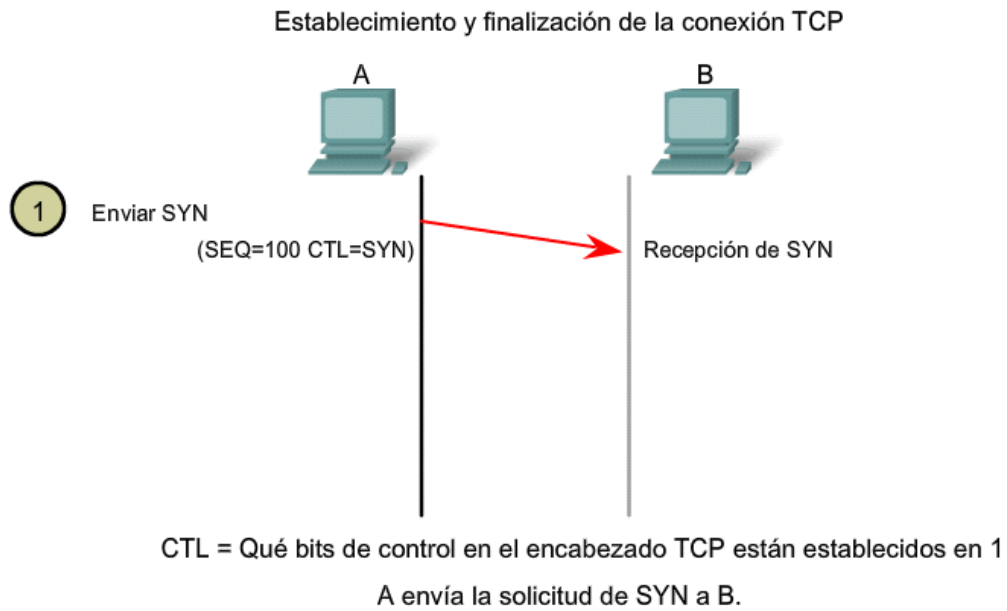
Identificación del puerto del servidor en la respuesta



4.2.3.1 Establecimiento de la conexión TCP

Antes del inicio de la transferencia de la información se establece la conexión entre el cliente y el servidor. Este es un proceso de tres vías.
Dos sockets intercambian información aun cuando el socket no forma parte del segmento TCP.
El número de secuencia se inicializa aleatoriamente en ambos extremos.
Se consideran campos implícitos las direcciones IP de origen y destino, sin embargo estos campos no forman parte de un segmento.
Son asignados los campos de puerto de origen y puerto de destino en el cliente y en el servidor.
En este proceso el cliente activa solamente el bit de SYN=1, para el establecimiento de una comunicación desde el cliente hasta el servidor.
Se considera que cada uno de los flujos de inicialización tiene solamente un octeto de datos.
Campos de un bit:

Urg	Ack	Psh	Rst	Syn	Fin
0	0	0	0	1	0



4.2.3.2 Establecimiento de una conexión TCP

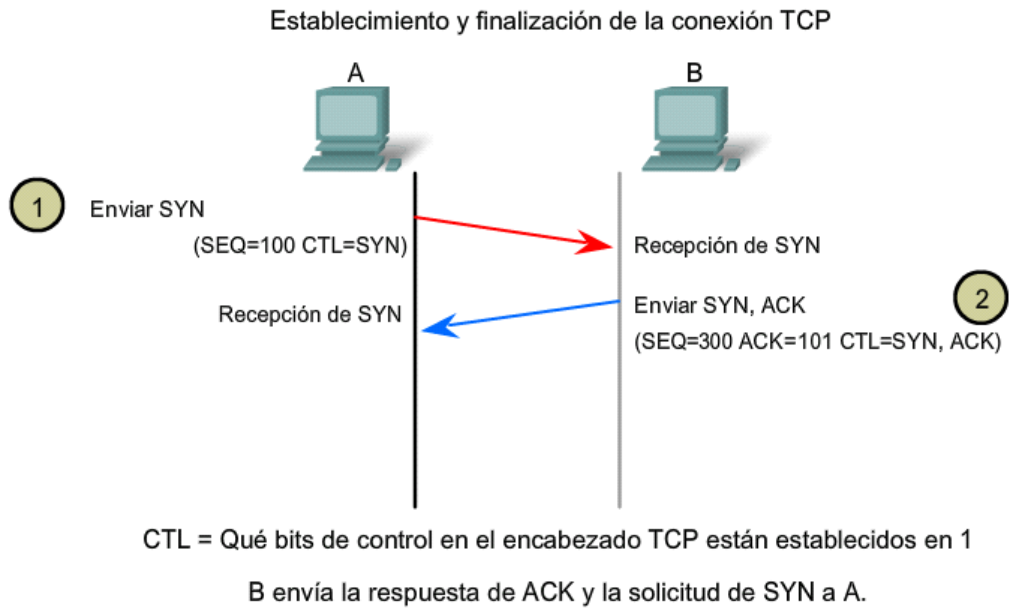
Si el servidor no tiene activado el correspondiente puerto, envía un mensaje de rechazo de la conexión por medio de RST=1.
Si el receptor tiene activado el número de puerto:

- Inicializa el número de secuencia con un valor arbitrario.
- Activa los bits de SYN=1 y ACK=1. El bit de SYN se utiliza para el establecimiento de una comunicación entre el servidor y el cliente. El bit de ACK se usa para la indicación del asentimiento o confirmación del número de octetos recibidos y hace referencia al último octeto más uno.

El valor del campo del número de secuencia es un número que excede en un byte al número de bytes recibidos. Esto número de ACK + 1 le corresponde al número del siguiente octeto esperado.
El señalizador SYN=1 del receptor indica que se tiene un número de secuencia inicial asignado en el servidor.
Este valor del número de secuencia se utiliza para el rastreo de la cantidad de datos que salen del servidor e ingresan al cliente.

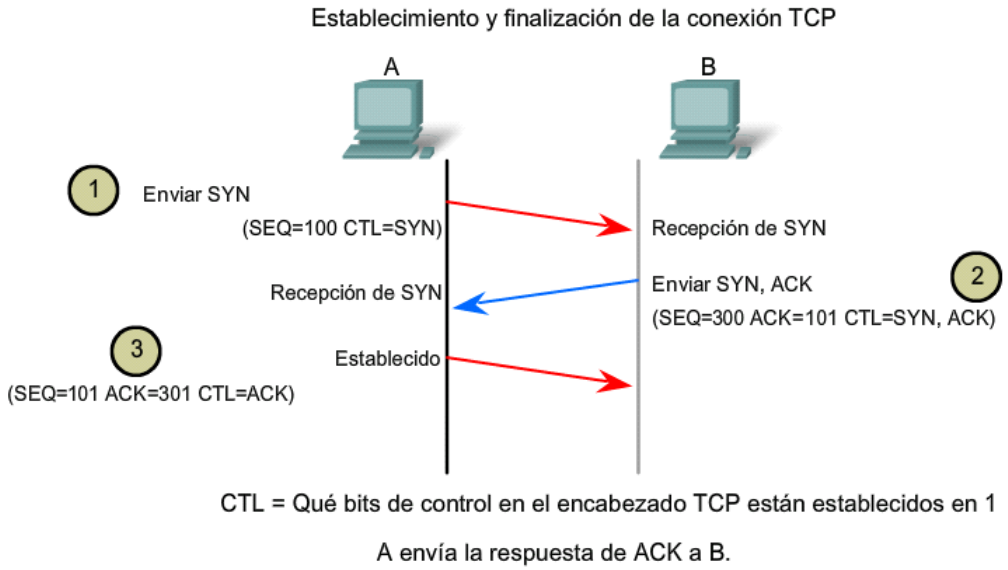
Urg	Ack	Psh	Rst	Syn	Fin
0	0	0	0	1	0

Urg	Ack	Psh	Rst	Syn	Fin
0	1	0	0	1	0



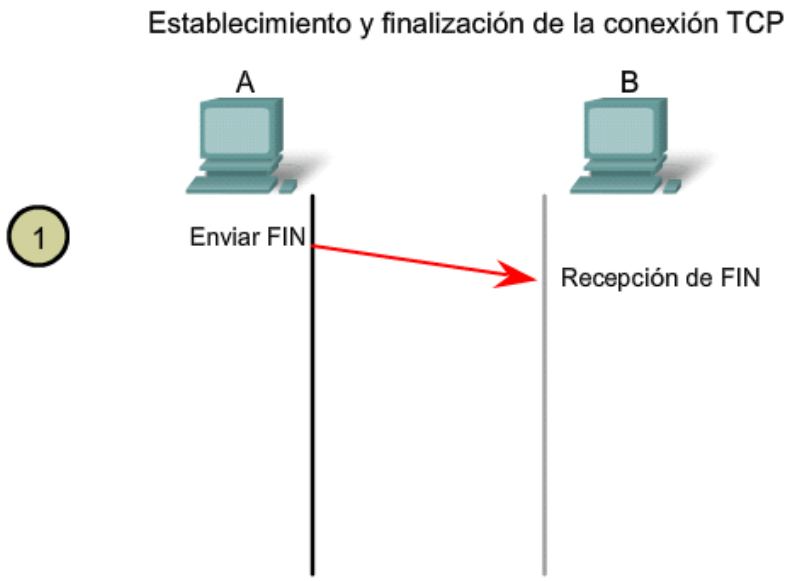
4.2.3.3 Establecimiento de una conexión TCP

El cliente contesta un segmento ACK cuyo valor es igual al número de secuencia inicial más uno procedente del servidor.
La etapa de conexión se denomina SYN,SYN/ACK,ACK.
Después de haberse cumplido el proceso de tres vías, se inicia la transferencia de información entre el cliente y el servidor.
Después de finalizar el proceso de establecimiento de la conexión de tres vías, al valor de ACK+1 que se reciba hace referencia al número de secuencia inicial más uno establecido en cada equipo de transmisión de datos.
Se considera que cada segmento de SYN transporta un octeto de datos.



4.2.3.4 Finalización de una comunicación TCP

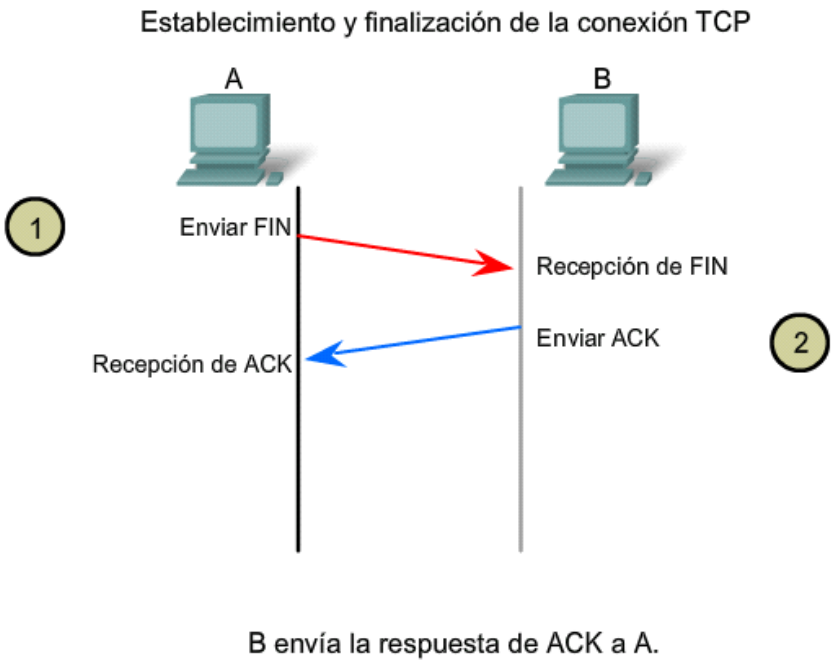
La desconexión es un proceso de cuatro vías. Uno de los extremos inicia el proceso de finalización por medio del bit FIN=1.
El proceso de finalización de la comunicación lo puede iniciar cualquiera de los equipos terminales de datos.



A envía la solicitud de FIN a B.

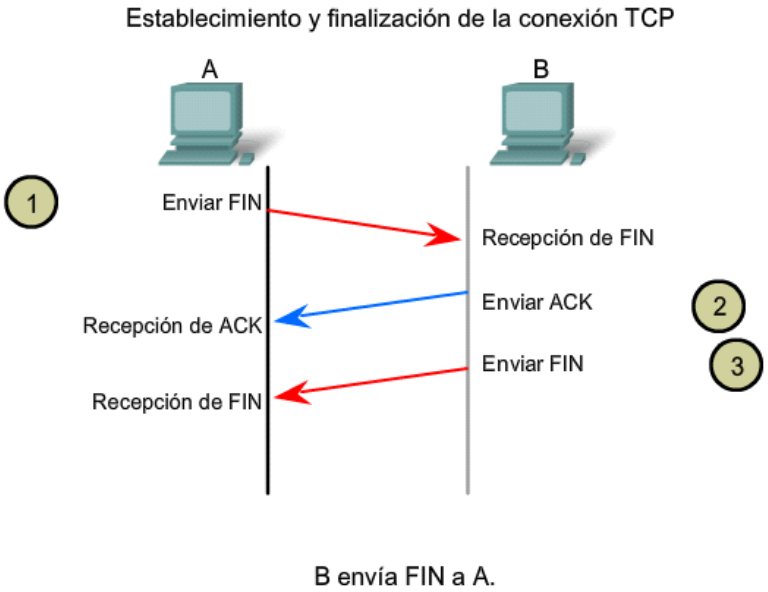
4.2.3.5 Finalización de la comunicación

El receptor le informa a la aplicación activa la solicitud de finalización enviada por el emisor.
El receptor contesta con un mensaje de asentimiento.



4.2.3.6 Finalización de la comunicación

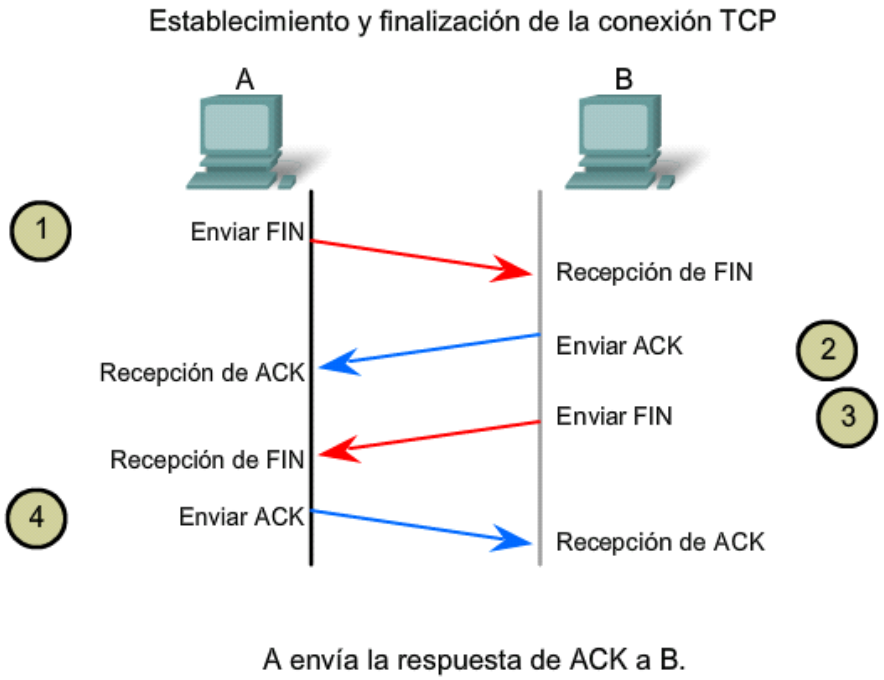
El receptor también inicia el proceso de desconexión, para que la conexión no quede abierta en uno de los sentidos del flujo de tráfico.
Se hace referencia a estos campos por medio de señaladores, porque el valor de uno de estos campos es sólo 1 bit y, por lo tanto, sólo tiene dos valores: 1 o 0. Cuando el valor de un bit se establece en 1, indica qué información de control se incluye en el segmento.



B envía FIN a A.

4.2.3.7 Finalización de la comunicación

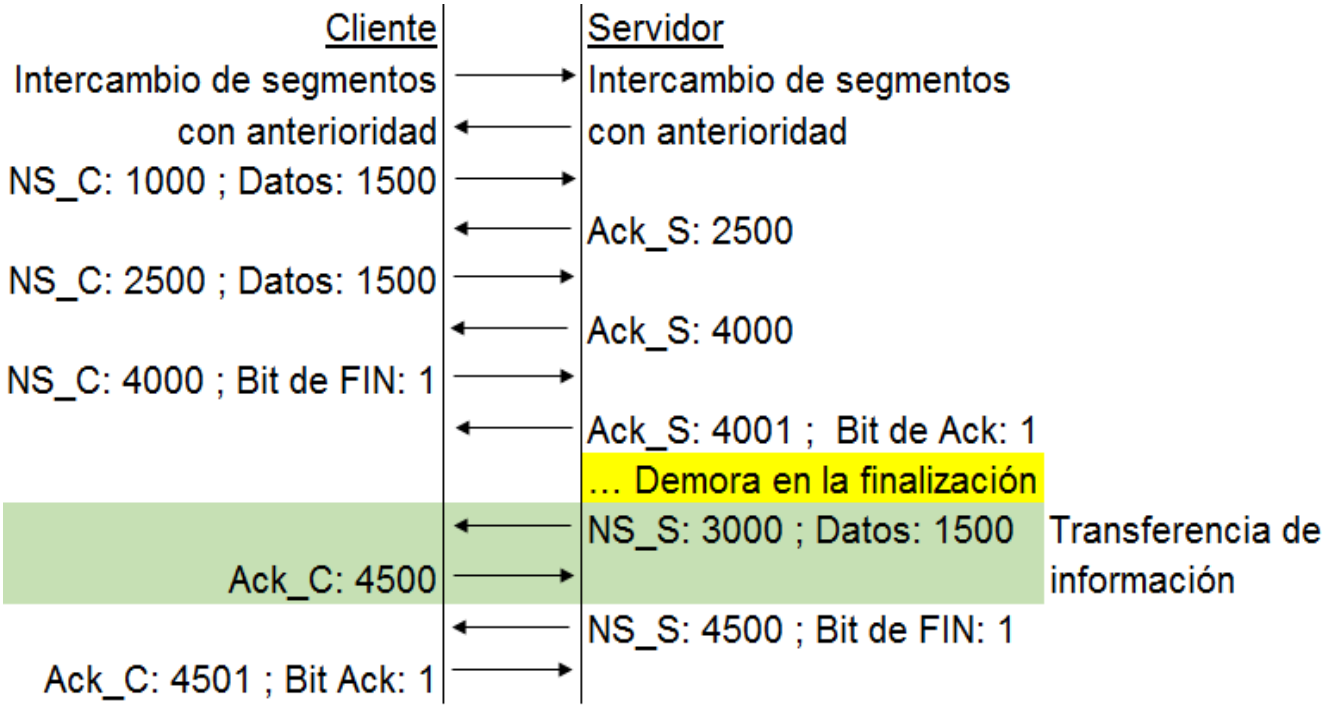
Cada uno de los extremos debe finalizar la comunicación.
El extremo que ha finalizado la comunicación no puede continuar enviado datos.
Si uno de los extremos no ha finalizado la comunicación, puede continuar enviado datos.



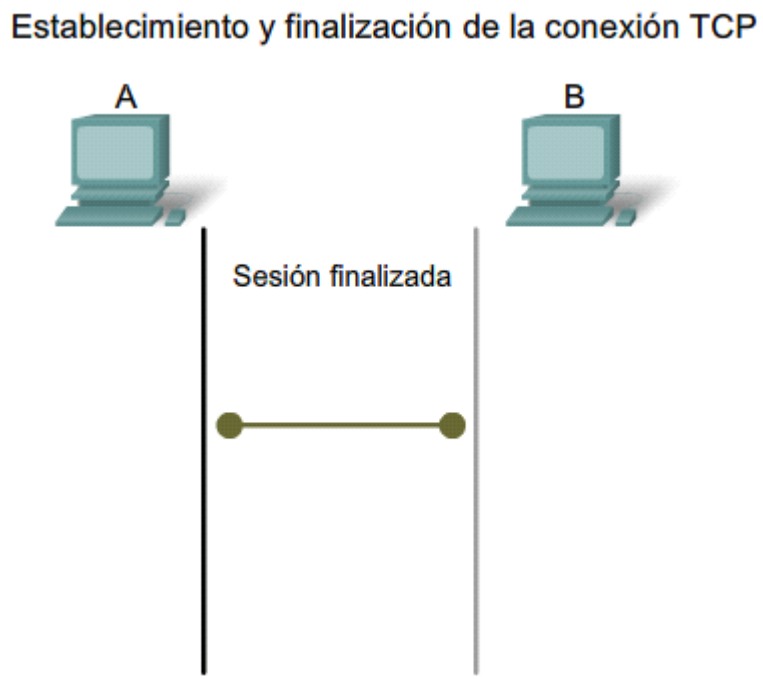
A envía la respuesta de ACK a B.

4.2.3.8 Transferencia de datos durante la finalización de una comunicación.

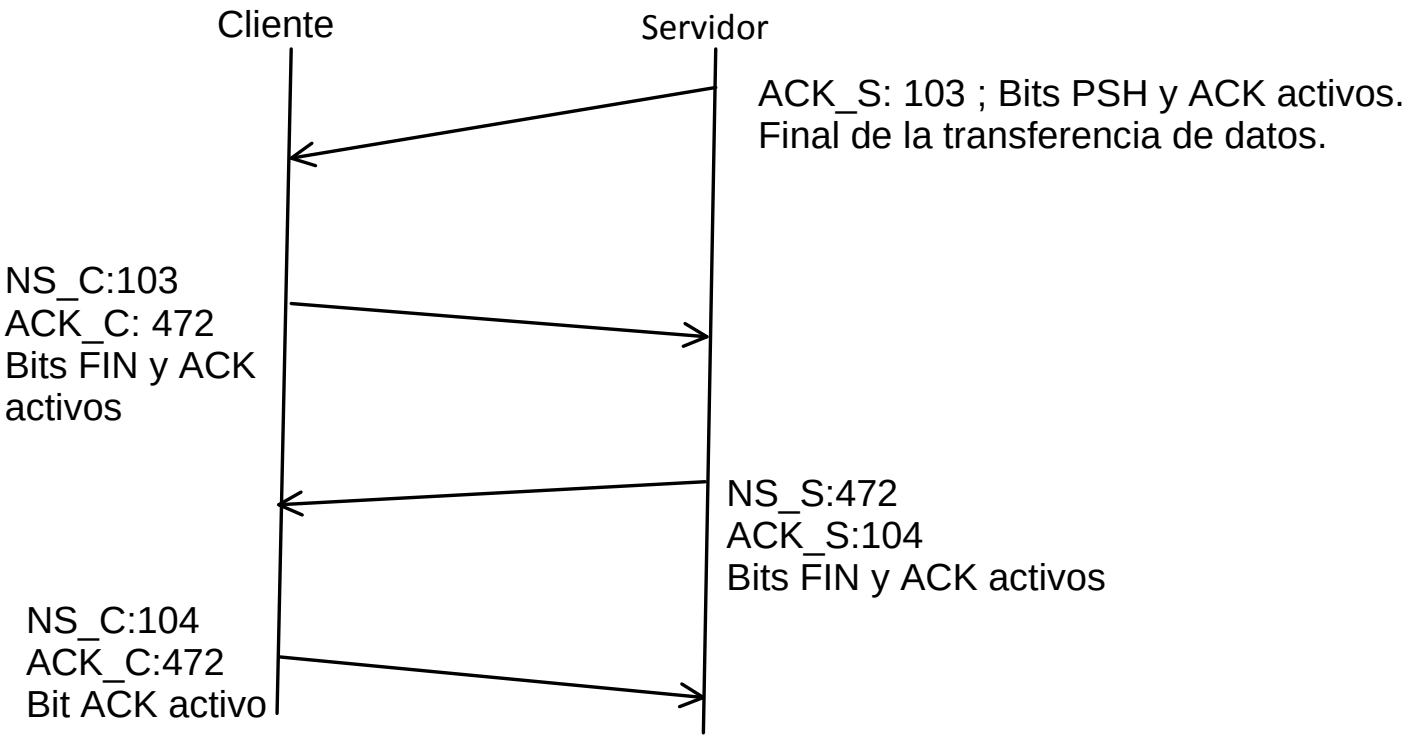
En caso de retardo en el proceso de finalización de una comunicación, es posible la transferencia de información durante el proceso de finalización de una comunicación. El NS_S: 3000 es debido a los números de segmentos anteriores en el servidor.



4.2.3.9 Finalización de la comunicación



4.2.3.10 Finalización de tres vías

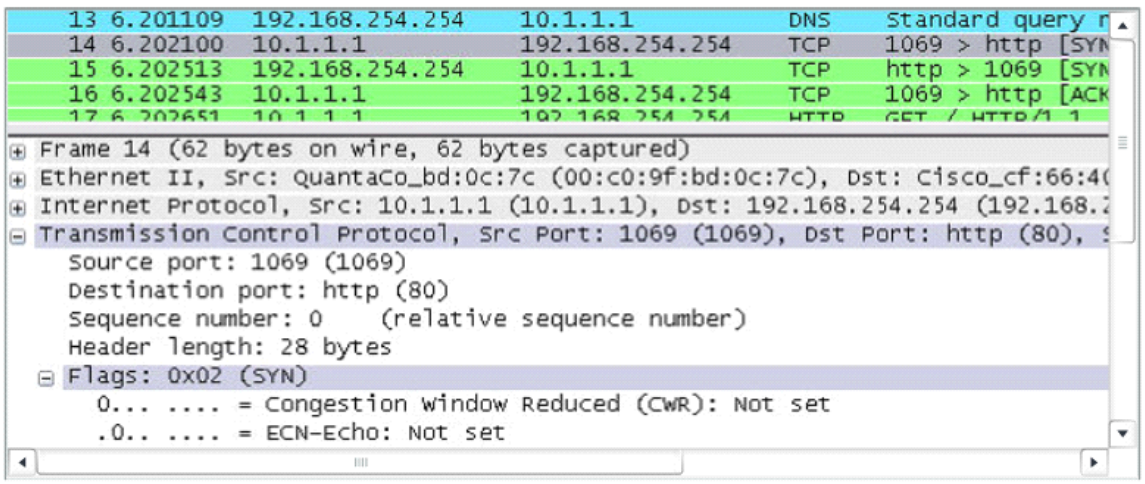


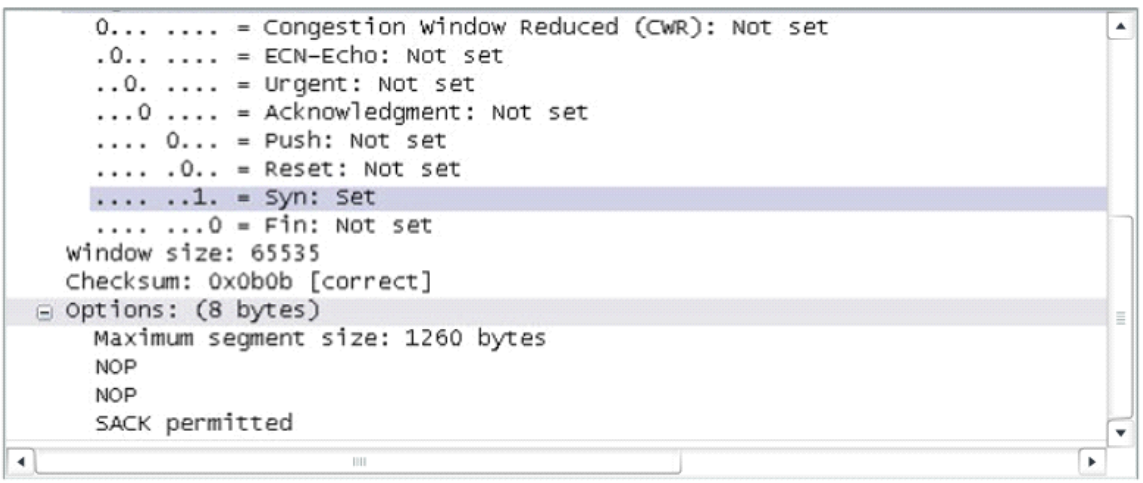
4.2.4.1 Protocolo TCP de enlace de tres vías

Paso 1.

Dirección de origen: 10.1.1.1 , dirección de destino 192.168.254.254
Activación del control de SYN=1 e inicialización del número de secuencia con un valor aleatorio ISN=0.
Puerto de origen: 1069 , puerto de destino: 80, HTTP.

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:

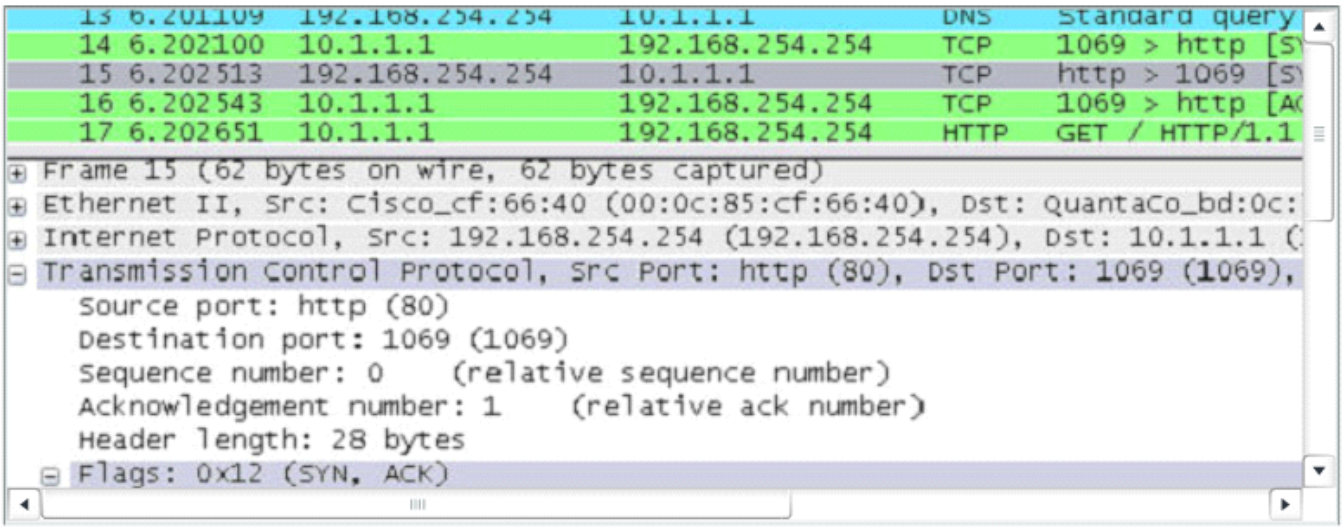
- El señalizador SYN establecido para validar un número de secuencia inicial
- Número de secuencia aleatorio válido (el valor relativo es 0)
- Puerto de origen aleatorio 1069
- El puerto de destino conocido es 80 (puerto HTTP) según indica el servidor Web (httpd)

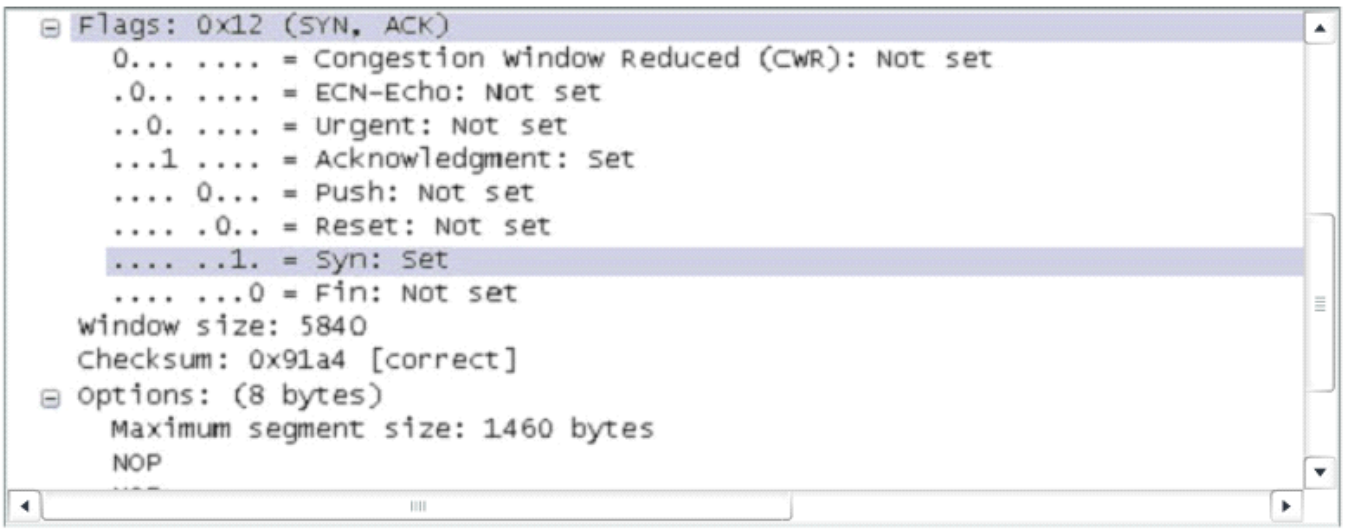
4.2.4.2 Protocolo TCP de enlace de tres vías

Paso 2.

En el servidor activa el bit de ACK=1 para indicar un asentimiento recibido.
El servidor activa el bit de SYN=1 para indicar una conexión con un puerto del cliente.
El valor de ACK es igual al número de secuencia inicial recibido más uno.
El número de secuencia inicial del servidor es aleatorio.

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





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

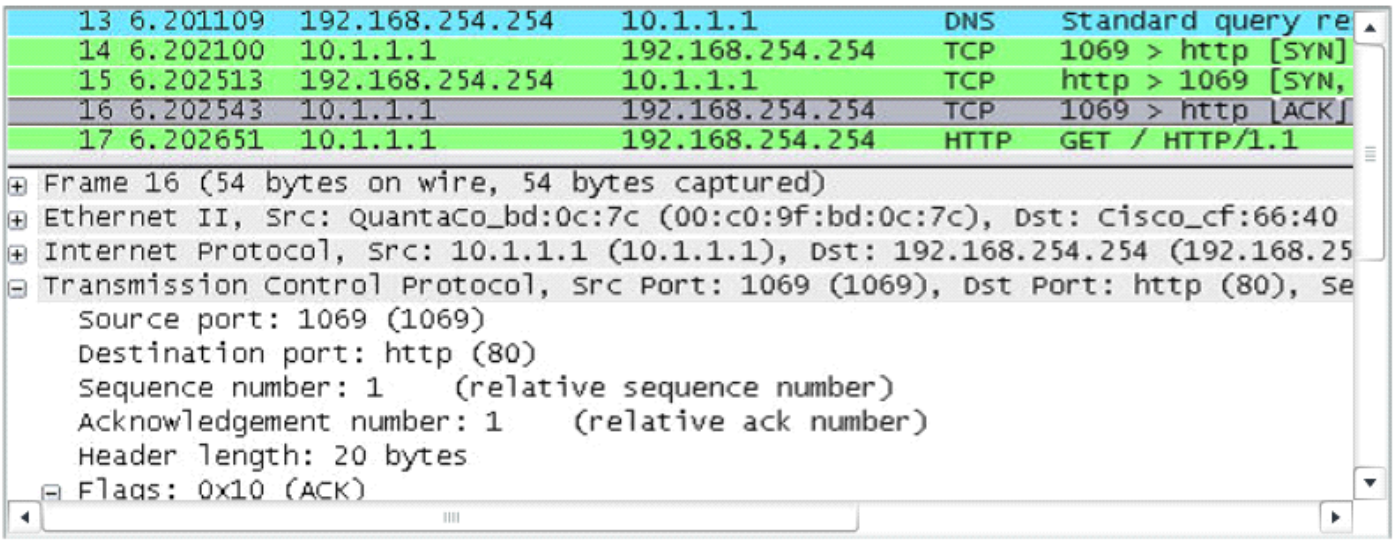
- El señalizador ACK está establecido para indicar un número de acuse de recibo válido
- Respuesta del número de acuse de recibo al número de secuencia inicial como valor relativo de 1
- Señalizador SYN establecido para indicar el número de secuencia inicial para el servidor a la sesión del cliente
- 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)

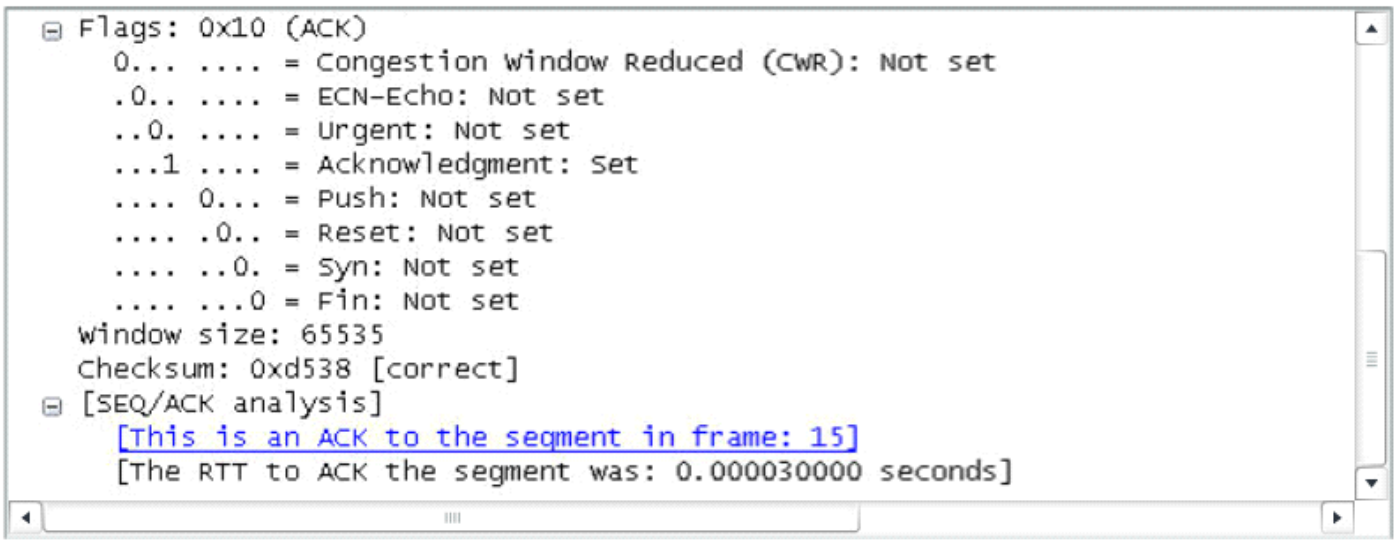
4.2.4.3 Protocolo TCP de enlace de tres vías

Paso 3.

Respuesta del cliente. Activación únicamente del bit ACK=1.

Protocolo TCP de enlace de tres vías (ACK)





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

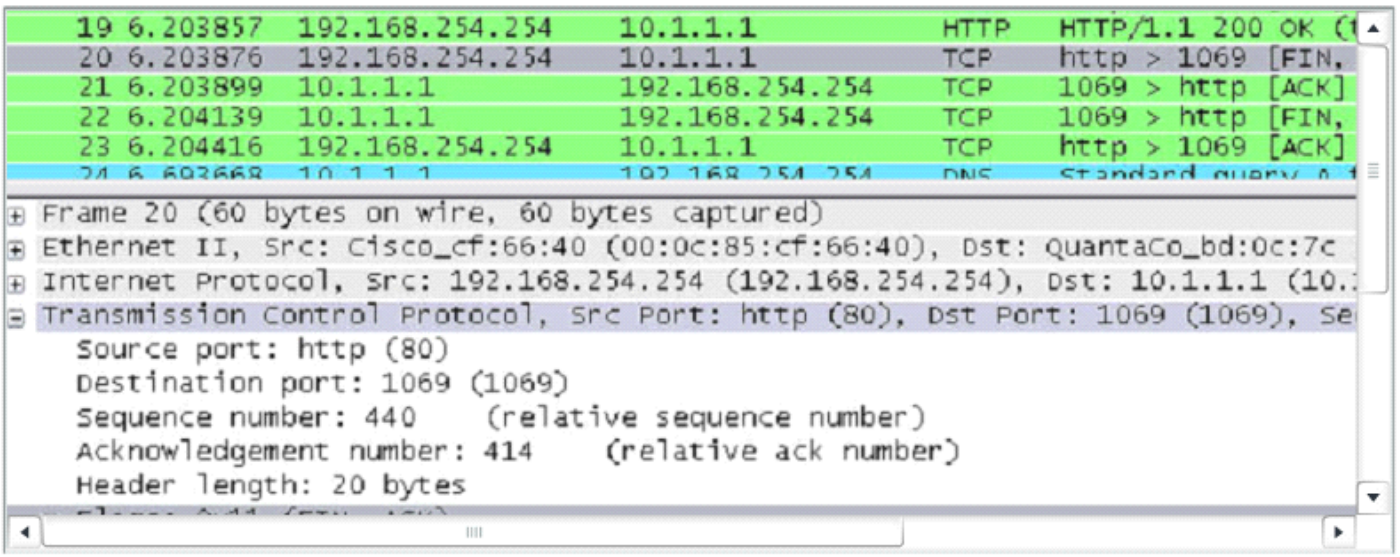
El segmento TCP en esta trama muestra:

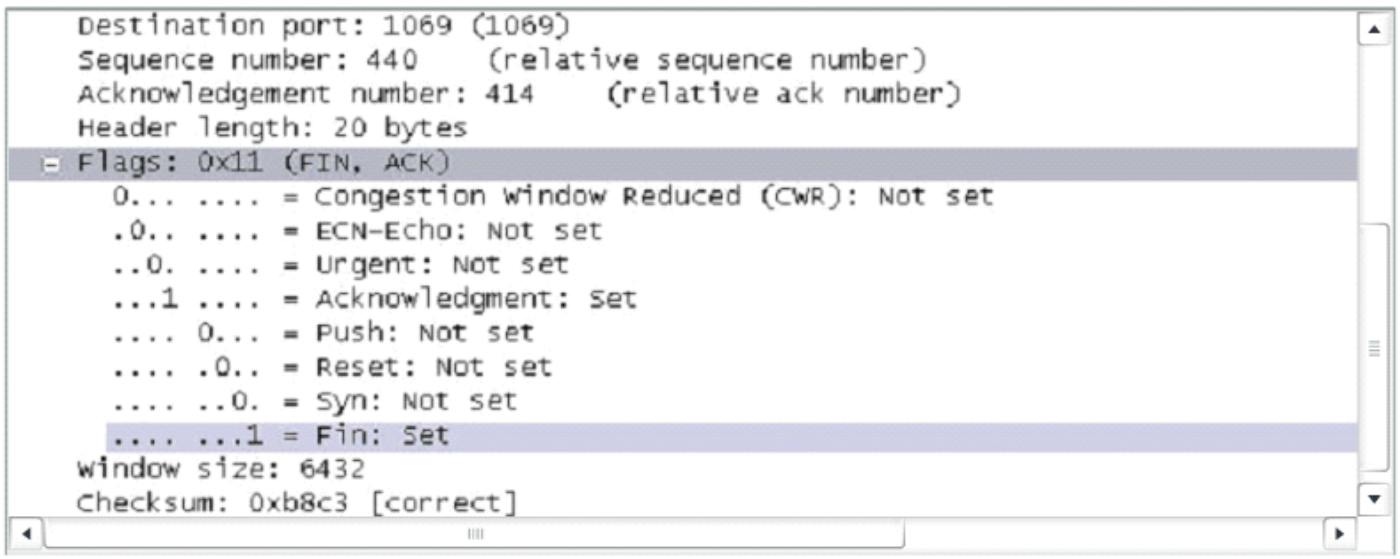
- El señalizador ACK está establecido para indicar un número de acuse de recibo válido
- Respuesta del número de acuse de recibo al número de secuencia inicial como valor relativo de 1
- Número de puerto de origen de 1069 para la correspondencia
- Número de puerto de destino de 80 (HTTP) que indica el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

4.2.5.1 Finalización de la sesión TCP

Se utilizan dos procesos de dos vías.
El proceso lo puede iniciar cualquiera de los dos dispositivos, cliente o servidor.
Uno de los extremos envía una señal de FIN=1 y existe un valor de ACK.

Finalización de la sesión TCP (FIN)





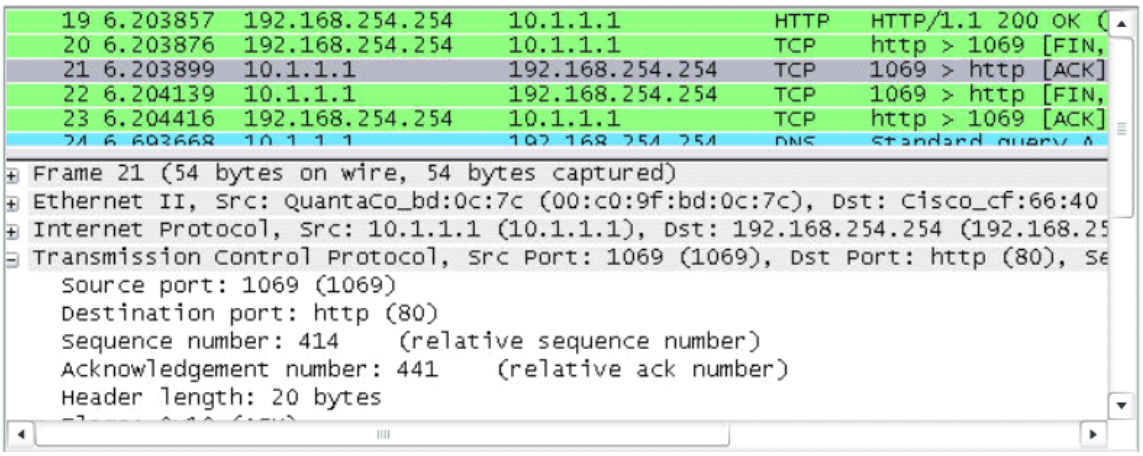
Un analizador de protocolo muestra los detalles de la trama 20, solicitud TCP FIN.

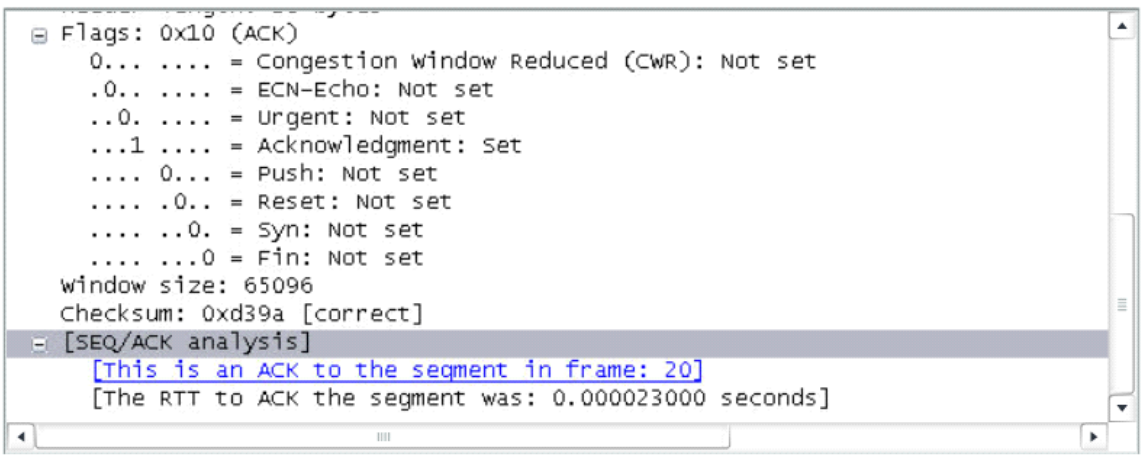
Puertos de destino y origen
Contenido y valores del campo del encabezado

4.2.5.2 Finalización de la sesión TCP

El receptor envía un mensaje de ACK, con el bit ACK=1.
Este proceso se repite desde el receptor hacia el emisor, en la cual el receptor cierra el proceso con un bit de FIN=1.
Finalmente el emisor contesta con el bit ACK=1.
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 mensaje ACK.

Finalización de la sesión TCP (ACK)



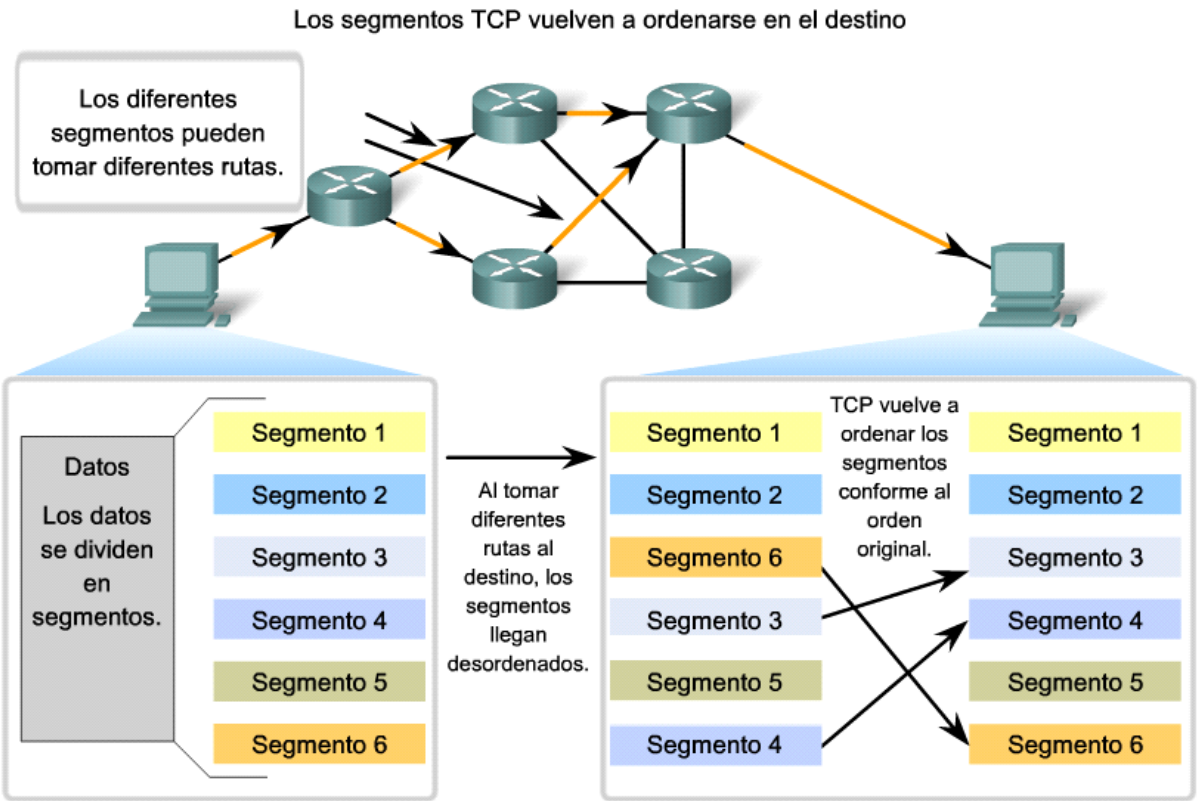


Un analizador de protocolo muestra los detalles de la trama 21, respuesta TCP ACK.

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

4.3.1 Re-ensamblaje de segmentos TCP

En la sesión, se establece un número secuencial inicial en la máquina del cliente. Una vez establecida la conexión se procede al intercambio de información.
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.
El número de secuencia se incrementa conforme al número de octetos transmitidos.
Durante la transmisión de la información, algunos segmentos pudieron haber sido recibidos desordenados conforme al número secuencial de los mismos.
El receptor confirma la recepción de la cantidad de bytes recibidos, tomando como referencia los valores que han tenido asentimiento.
El re-ensamblaje se efectúa conforme al número secuencial de cada segmento transmitido.
Los números de secuencia de segmento permiten la confiabilidad indicando cómo re-ensamblar y reordenar los segmentos recibidos.
Uso de memorias de almacenamiento y procedimientos de clasificación u ordenamiento de los contenidos en las memorias, antes de la entrega de los segmentos a la capa de aplicación.



4.3.2 Secuencias para entrega fuera de orden

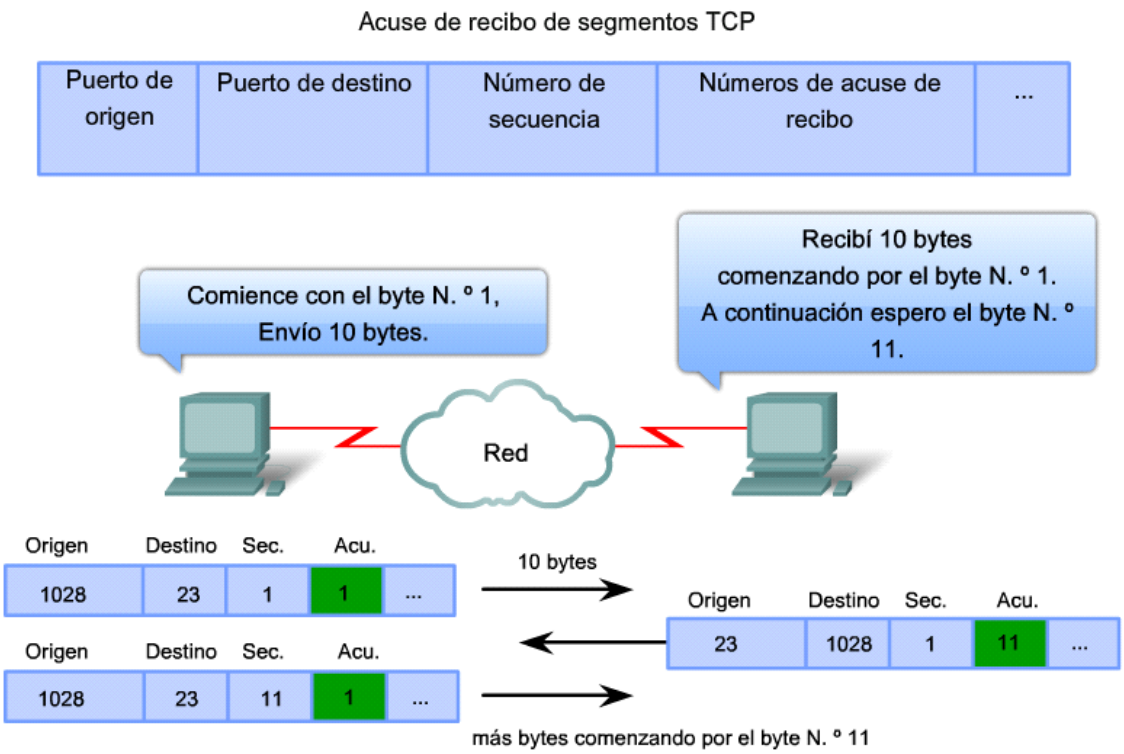
Los segmentos pudieron haber llegado fuera de orden debido al recorrido de los mismos por redes de diferentes velocidades y por diferentes números de saltos.
El transmisor agrega un número de secuencia a cada paquete transmitido y el receptor almacena el número de secuencia del último paquete recibido en orden, así como la lista de los paquetes que llegaron fuera de orden.
Si el segmento recibido es el segmento esperado, el protocolo lo entrega a la capa superior y busca en su lista los paquetes adicionales que también puede entregar.
Si algún segmento ha llegado fuera de secuencia, permanece en la lista de segmentos almacenados para su posterior entrega a la aplicación.

4.3.3 Secuenciación para eliminar segmentos duplicados

La duplicación de segmentos puede presentarse si un transmisor defectuoso detecta una colisión en la trama transmitida, en consecuencia procede retransmitir dicho segmento con el mismo número de secuencia.
El receptor pudo haber detectado a ambos segmentos como válidos, con el mismo número de secuencia.
El receptor examina los números de secuencia recibidos y detecta si el segmento ya ha sido entregado a la aplicación con anterioridad o si es un segmento esperado.
Esto permite la eliminación del último segmento cuyo número de secuencia esta duplicado.
La retransmisión de segmentos puede ocasionar la aparición de segmentos duplicados en el receptor.
El transmisor no distingue entre los segmentos perdidos y los segmentos que tienen un gran retardo en su llegada al receptor.

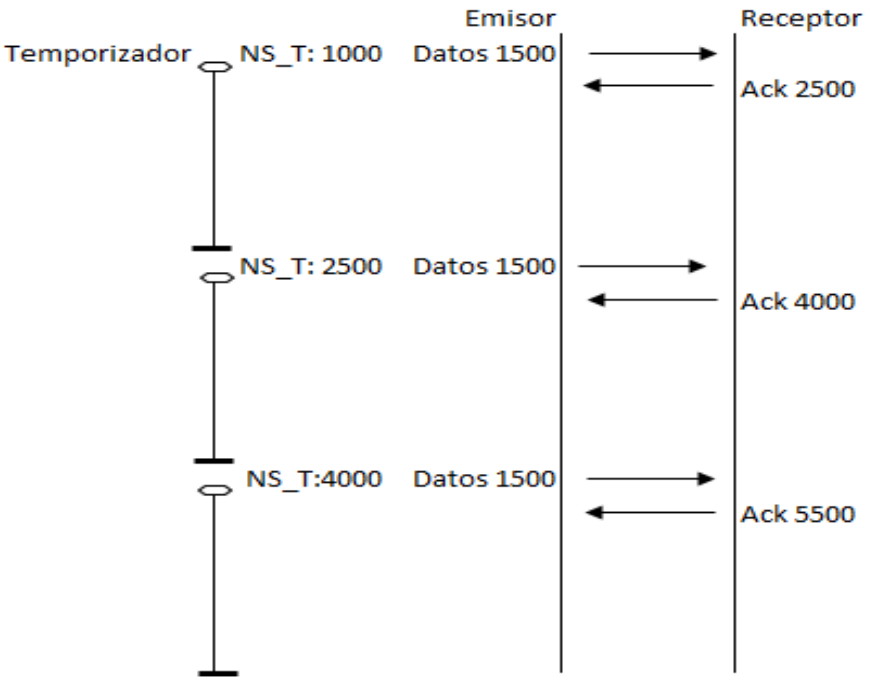
4.3.4 Acuse de recibo TCP con uso de ventanas

Al principio el emisor establece un número de secuencia inicial para el establecimiento de la comunicación entre el transmisor y el receptor.
El receptor determina la cantidad de octetos de datos recibidos y contesta con un asentimiento ACK igual a la cantidad de octetos recibidos más uno, el cual constituye el número del siguiente octeto esperado.
Se espera que el host originador envíe el siguiente segmento con un número secuencial igual al número de ACK+1 reportado por el receptor.
Este es un proceso de dos vías, se espera que el transmisor y el receptor adecuen los valores de los números de secuencia enviados y números de ACK+1 en ambos extremos del enlace.
En este ejemplo el emisor envía diez octetos de datos, el primer octeto tiene un número de secuencia igual a uno.
Recibe una respuesta de ACK=11, el número del siguiente octeto esperado.
El emisor continua transmitiendo el siguiente bloque comenzando con un número de secuencia igual a once.
Con este procedimiento de envío de un asentimiento por cada octeto se tendría una red con bastante sobrecarga.
El tamaño de la ventana permite el establecimiento de la cantidad de segmentos que pueden ser recibidos para el envío de un solo mensaje de asentimiento. Por ejemplo, un solo asentimiento por los diez octetos enviados.
Si el número de secuencia inicial es de 2000, cada segmento consta de 1000 octetos y la ventana es de 10000, se devolverá al origen un acuse de recibo con el número 12000, el cual es el siguiente segmento esperado.
El tamaño de una ventana está especificado por la cantidad de octetos que un emisor puede enviar y recibir una sola confirmación por todos los octetos.
El incremento en el valor de la ventana permite el descongestionamiento de la red.



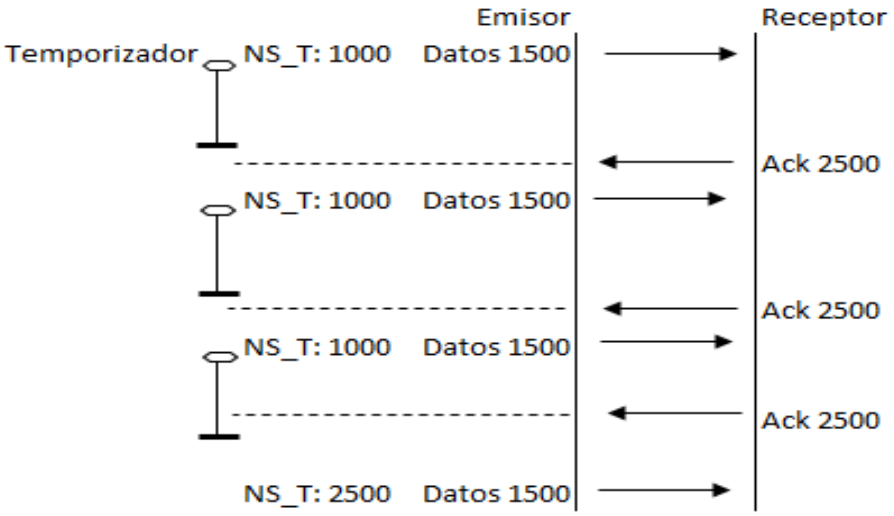
4.3.5.1 Temporización larga uniforme

Las primeras redes punto a punto tenían un retardo casi constante entre una trama transmitida y una respuesta de confirmación Ack.
En dichas redes, el número de usuarios y de aplicaciones era reducida.
Los diseñadores de red y los operadores de red fijaban los temporizadores en unos valores un poco mayores que el tiempo normal de retardo.
Esto implicaba que las tramas posteriores eran transmitidas con suficiente retardo después de haber recibido la confirmación Ack.
La red quedaba inactiva durante algunos segundos, el rendimiento no se maximiza.



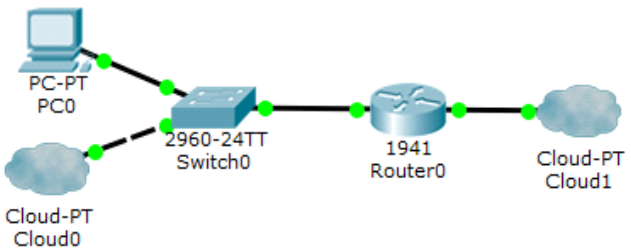
4.3.5.2 Temporización corta uniforme

Si el vencimiento del temporizador ocurría en muy corto tiempo, numerosas respuestas válidas Ack no eran recibidas dentro del intervalo de tiempo establecido y el emisor volvía a retransmitir la misma trama. El rendimiento de la red tampoco se maximiza.



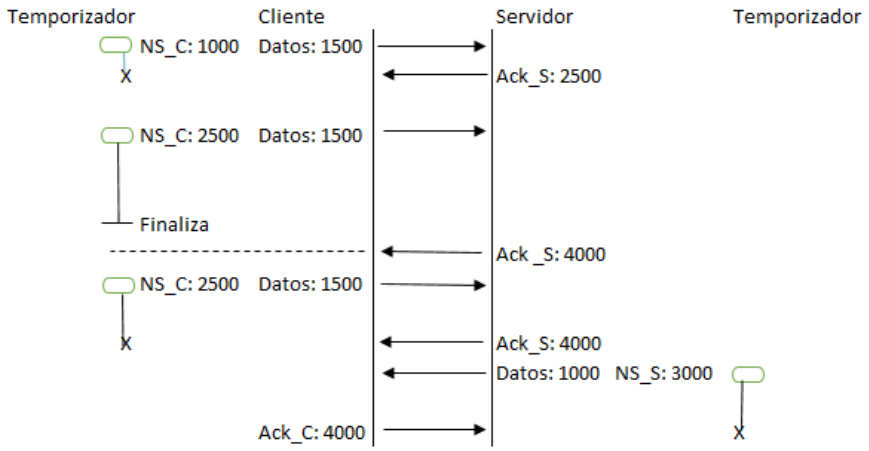
4.3.5.3 Temporización adaptable

Un host puede consultar a un servidor localizado en su misma red local o a un servidor ubicado en una red muy lejana.
El retardo de la respuesta Ack de la red local es despreciable. El retardo de la respuesta Ack de la red lejana es de varios milisegundos.
El protocolo TCP supervisa los tiempos de retardo de las respuestas por medio de relojes que supervisan el instante de la transmisión de una trama y la recepción de la respuesta Ack procedente del receptor.
Esta diferencia de tiempo le permite conocer el retardo para cada enlace. El emisor establece un promedio ponderado para la fijación del temporizador.
El protocolo TCP reacciona con rapidez ante los cambios de ráfagas de los mensajes enviados y las respuestas recibidas, lo cual le permite el ajuste del cronometro logrando una temporización adaptable.
Cuando el retardo es constante, el TCP ajusta el temporizador con un valor un poco superior que el retardo medio del mensaje en su recorrido de emisión y confirmación de recepción.



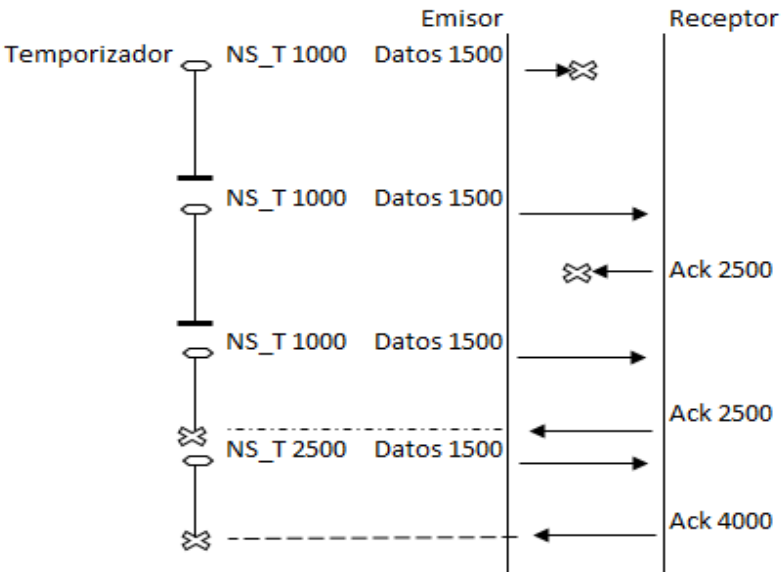
4.3.5.4 Temporización adaptable y retransmisión

Cuando el emisor transmite un paquete, inicia un temporizador con el propósito de retransmitir el mismo paquete en caso que no se reciba una confirmación Ack dentro de un tiempo establecido. Si la confirmación Ack se recibe antes del vencimiento del tiempo, el temporizador se detiene y el emisor puede continuar con la transmisión del segmento siguiente. En este caso el protocolo funciona en el modo de parada y espera.



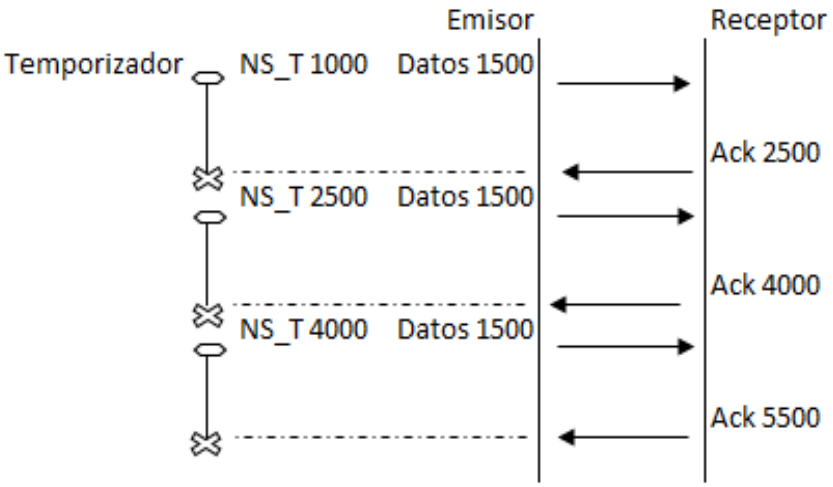
4.3.5.5 Retransmisión de segmentos no confirmados

La retransmisión es selectiva. Cada nuevo segmento se envía después de la recepción de la confirmación del segmento anterior. La transferencia de la información funciona en el modo de parada y espera.
En este caso pueden llegar numerosos segmentos duplicados al receptor



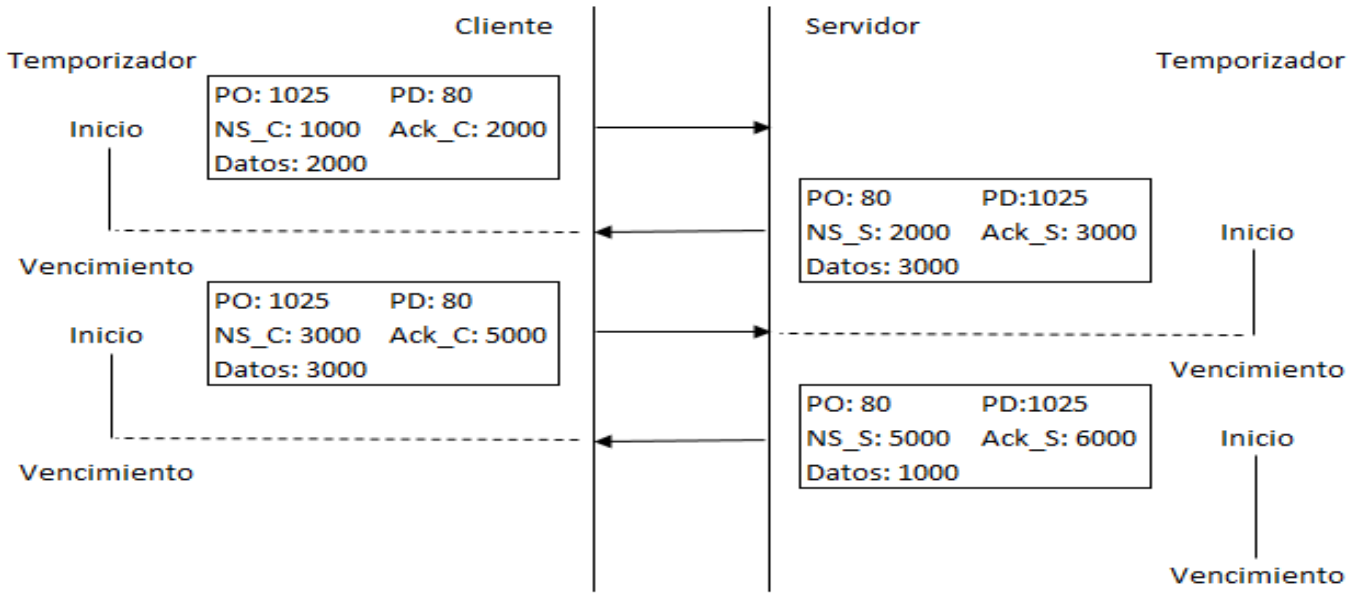
4.3.5.6 Transmisión simple: confirmación positiva

Cada segmento se transmite después de la recepción de la confirmación del segmento anterior. El protocolo funciona en el modo de parada y espera. La transferencia de información es ineficiente. Solamente se confirman Ack los segmentos recibidos sin error. No existe confirmación ni notificación cuando el segmento enviado está perdido o si el segmento es recibido con algún error.



4.3.5.7 Transmisión simple: confirmación positiva

Un cliente y un servidor han intercambiado numerosos segmentos, previo al monitoreo donde se observan las siguientes tramas. El servidor es HTTP. No existen segmentos perdidos en este ejemplo.



4.3.5.8 Buffers, control de flujo y ventanas

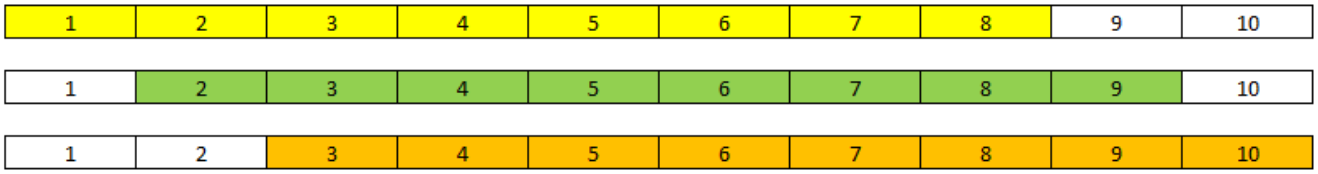
El protocolo TCP controla el flujo de datos en una red por medio del mecanismo de ventana.
Una vez establecida la conexión entre dos dispositivos, cada lado asigna el tamaño del buffer de almacenamiento de los datos y transmite este contenido al host opuesto.
A medida que se reciben los datos, el receptor regresa los acuses de recibo Ack y el tamaño de la ventana disponible.
Si la aplicación receptora puede leer los datos a la misma velocidad a la cual llegan, transmitirá un anuncio de ventana positiva con cada acuse de recibo Ack.

Control de flujo

Si el transmisor opera a una velocidad mayor que el receptor, los datos de entrada llenarán el buffer del receptor, el cual enviará un anuncio de ventana cero.
El emisor debe interrumpir la transmisión de información hasta que el receptor anuncie una ventana con un valor positivo.

4.3.5.9 Fundamentos de la ventana deslizante

La ventana deslizante le permite al emisor el envío de numerosos segmentos sin esperar la confirmación de la recepción de cada segmento en el dispositivo receptor.
En la gráfica se tiene una ventana deslizante compuesta por ocho segmentos, los cuales pueden ser transmitidos sin esperar la confirmación de recepción de cada uno de los segmentos.
Cuando se recibe la confirmación de recepción del primer segmento transmitido, la ventana puede deslizarse un segmento y transmitir el noveno segmento.
Cuando se recibe la confirmación del segundo segmento se envía el décimo segmento.



Si han sido transmitido los primeros ocho segmentos, sin haber recibido la confirmación de ningún segmento desde el receptor, la ventana no avanza ninguna posición hacia otros segmentos que deben ser transmitidos.
Una ventana esta descripta por

- Ancho de la ventana: número de octetos que comprende la ventana.
- Indicador de la posición inicial del primer octeto de la ventana.
- Indicador de la posición final del último octeto de la ventana.
- Indicador de los segmentos cuya recepción está confirmada por el receptor.

4.3.5.10 Fundamentos de la ventana deslizante

Cuando el emisor transmite varias tramas consecutivamente conforme al valor de la ventana deslizante, cada trama enviada tiene asignado un temporizador.
Si no se recibe una respuesta de confirmación del receptor antes de la expiración del temporizador, el emisor vuelve a retransmitir la trama faltante en cualquiera de las formas posibles

- Retransmisión selectiva: solamente se retransmite la trama faltante.
- Retransmisión simple: se retransmiten todas las tramas a partir de la trama faltante, incluso las tramas que ya han sido confirmadas.

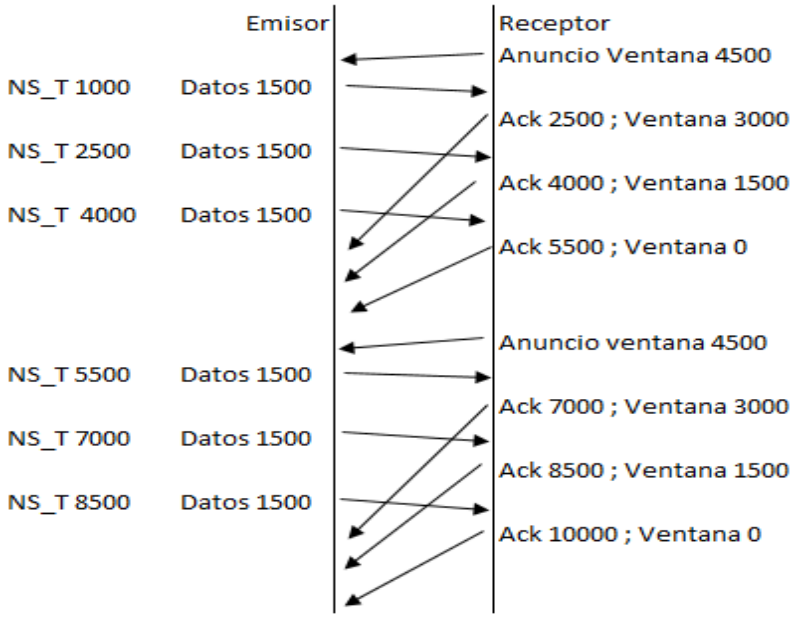
El intercambio de información entre dos hosts es bidireccional, en consecuencia los datos asignados a los segmentos pueden presentarse en ambos extremos.

4.3.5.11 Fundamentos de la ventana deslizante

El emisor y el receptor determinan el número de octetos destinados a la transmisión y recepción de tramas.
Los segmentos recibidos en el receptor ocupan una cantidad de memoria.
La disponibilidad de memoria en el receptor se le reporta al emisor en cada respuesta del receptor.
Si la cantidad de memoria disponible en el receptor es cero, el emisor no continua transmitiendo mensajes.
Una vez que el receptor entregue los segmentos recibidos a la aplicación se libera espacio en la memoria para continuar recibiendo más segmentos.
En el emisor, los segmentos enviados y no confirmados son retenidas en una memoria temporal para su retransmisión en caso necesario.
Los mensajes de confirmación procedentes del receptor liberan espacio de memoria en el emisor para el envío de nuevos segmentos.

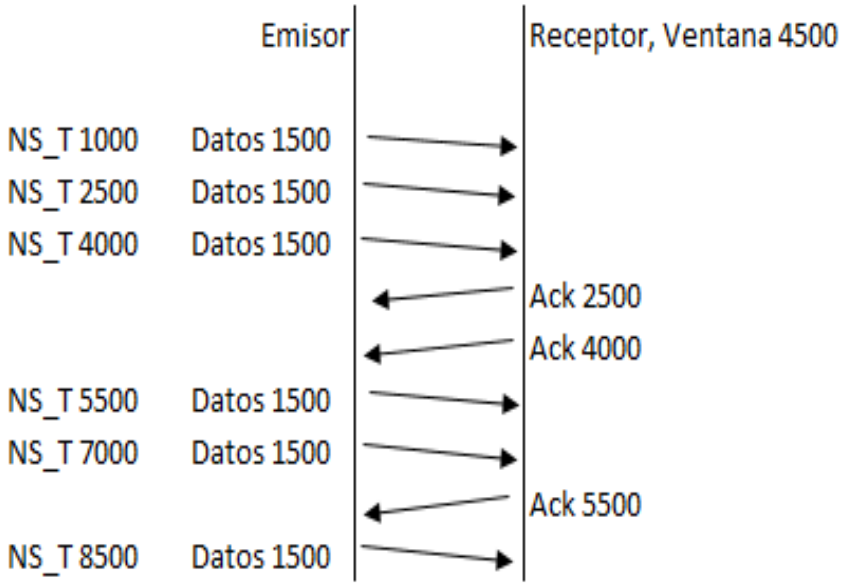
4.3.5.12 Anuncios de ventana

Durante la transferencia de información el receptor y el emisor intercambian los valores de las ventanas que representan el valor del buffer de almacenamiento. El receptor indica la cantidad de memoria disponible y su utilización durante la recepción de los segmentos.
Cuando el valor de la ventana tiene el valor de cero, el emisor no continua con la transmisión hasta que exista disponibilidad de memoria. La ventana contiene tres segmentos.



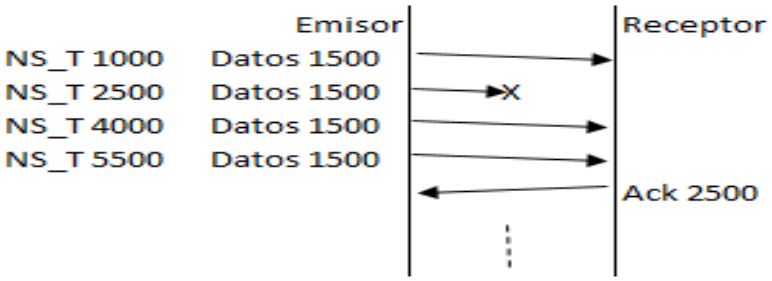
4.3.5.13 Ejemplo de ventana deslizante

El emisor no espera la confirmación de recepción de cada segmento transmitido. La ventana contiene tres segmentos. Este es un ejemplo sin errores en la trama enviada.



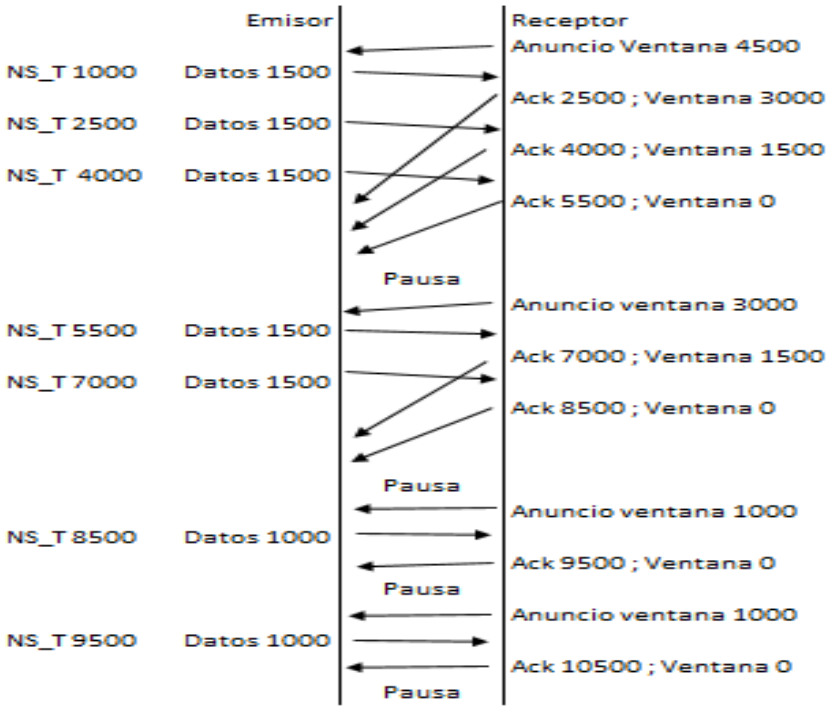
4.3.5.14 Error con rechazo simple - segmento no recibido

Existen varios procedimientos de recuperación de segmentos faltantes en un proceso de intercambio de información por medio de la venta deslizante.



4.3.5.15 Control de flujo

Se implementa para evitar que un host receptor sea inundado por un número de segmentos que supere su capacidad de almacenamiento.
La implementación de la ventana deslizante conlleva a un dimensionamiento de la cantidad de memoria que está asignada en el receptor para el almacenamiento de los octetos recibidos.
Una manera de reducir la cantidad de segmentos transmitidos desde el emisor hacia el receptor se realiza con una disminución de la ventana deslizante, en la cual se reporta la cantidad de memoria disponible para la recepción de los datos del emisor.
El ventana toma valores dinámicos crecientes o decrecientes en función de la cantidad de memoria disponible en el receptor.



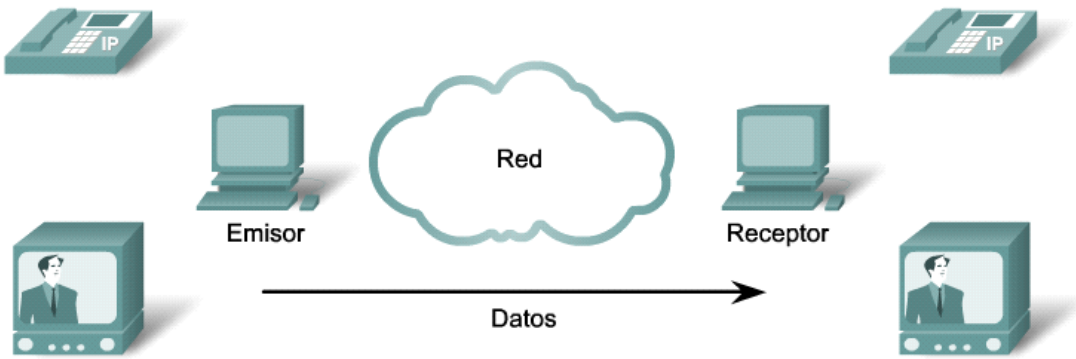
4.4.1 UDP: baja sobrecarga vs confiabilidad

El protocolo UDP es más simple que el protocolo TCP y proporciona mucho menos sobrecarga que el protocolo TCP.
El protocolo UDP no proporciona el secuenciamiento de los datagramas transmitidos, no retransmite los datagramas perdidos no realiza el control de flujo, tampoco el re-ensamblaje de los datagramas en el receptor.
Numerosas redes presentan alta confiabilidad en la transferencia de datos con el protocolo UDP.
Protocolos que utilizan UDP

- Sistemas de nombres de dominio DNS, puerto 53.
- Protocolo simple de administración de red SNMP, puerto 161.
- Protocolo de configuración dinámica de host DHCP, servidor de inicio puerto 67 ; cliente de inicio puerto 68.
- Protocolo de información de enrutamiento RIP.
- Protocolo de transferencia de archivo trivial TFTP, puerto 69.
- Voz sobre IP VoIP.
- Juegos en línea.

En el caso de la transferencia de voz e imagen es preferible que se presente pérdidas de contenidos antes que la retransmisión y demoras en el procesamiento de las tramas.
Los servidores de DNS y DHCP usualmente se encuentran en redes cercanas al usuario, el contenido de información de estos servidores no es extenso. El usuario puede volver a solicitar el acceso a estos recursos en caso necesario.

Transporte de datos con baja sobrecarga de UDP

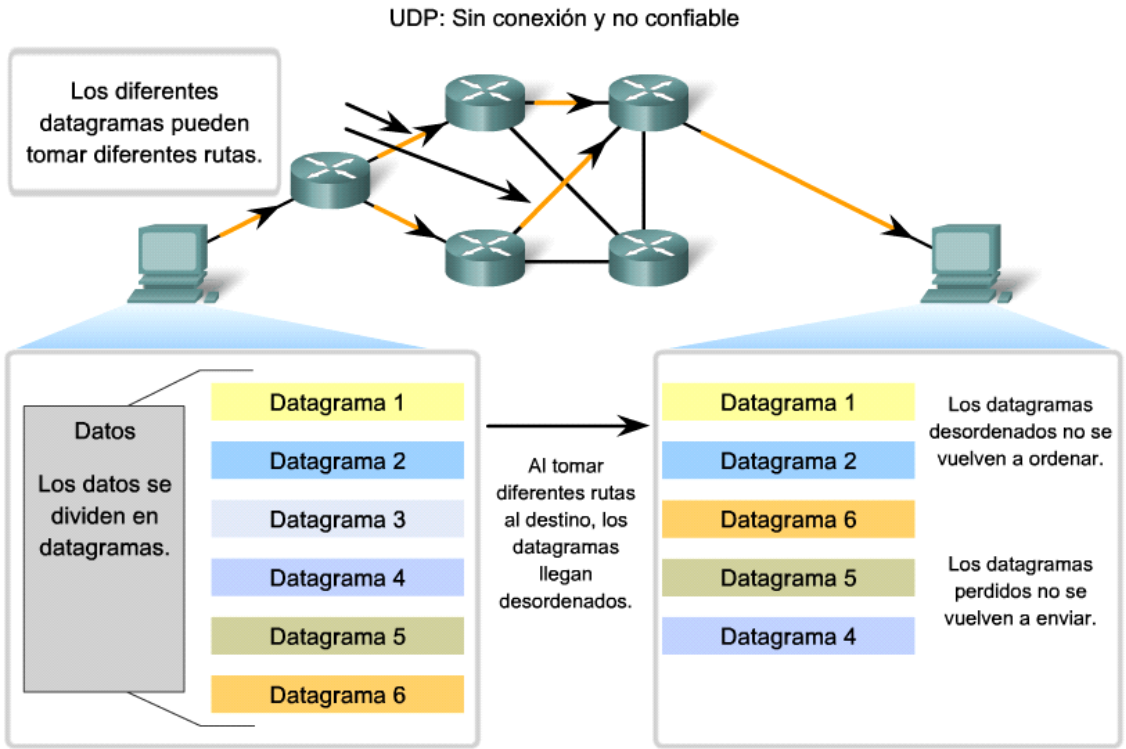


UDP no establece ninguna conexión antes de enviar datos.

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

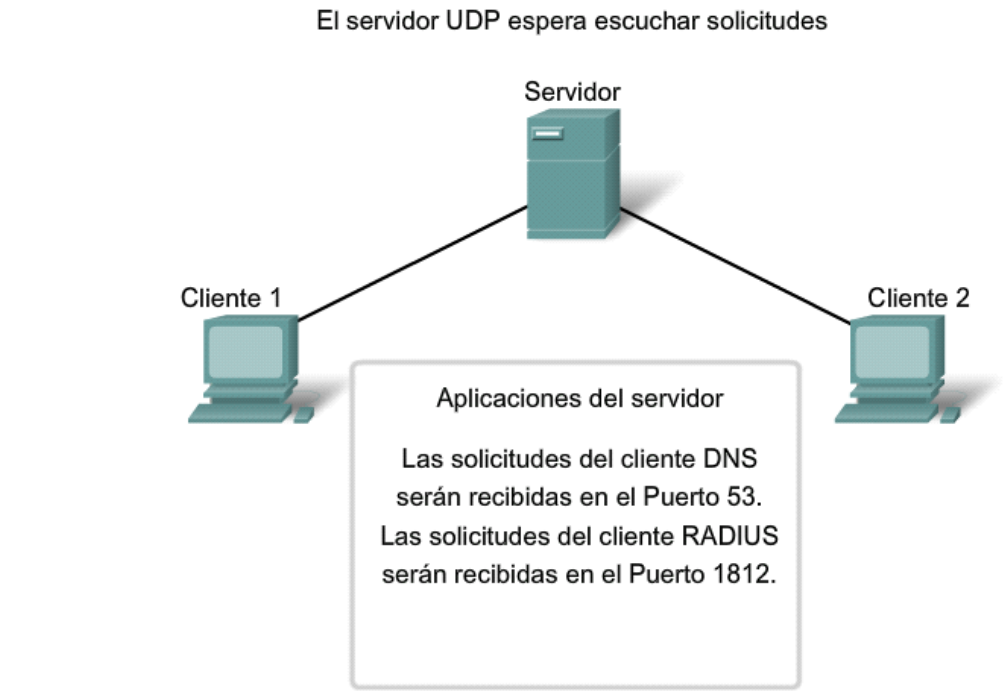
4.4.2 Re-ensamblaje de datagramas de UDP

Los datos de la capa de aplicación son fragmentados en datagramas.
UDP no establece una sesión de conexión ni finalización para la transferencia de información.
UDP no numera secuencialmente los datagramas, ni realiza acuses de recibos, ni retransmite datagramas perdidos, ni control de flujo de los datos transmitidos.
UDP no ordena los datagramas recibidos. Envía al puerto correspondiente a la aplicación en la secuencia que fueron recibidos.
En caso necesario cuando se tratan de transacciones importantes, la aplicación debe proveer algún procedimiento de seguridad en la transferencia de la información.



4.4.3 Procesos y solicitudes de UDP

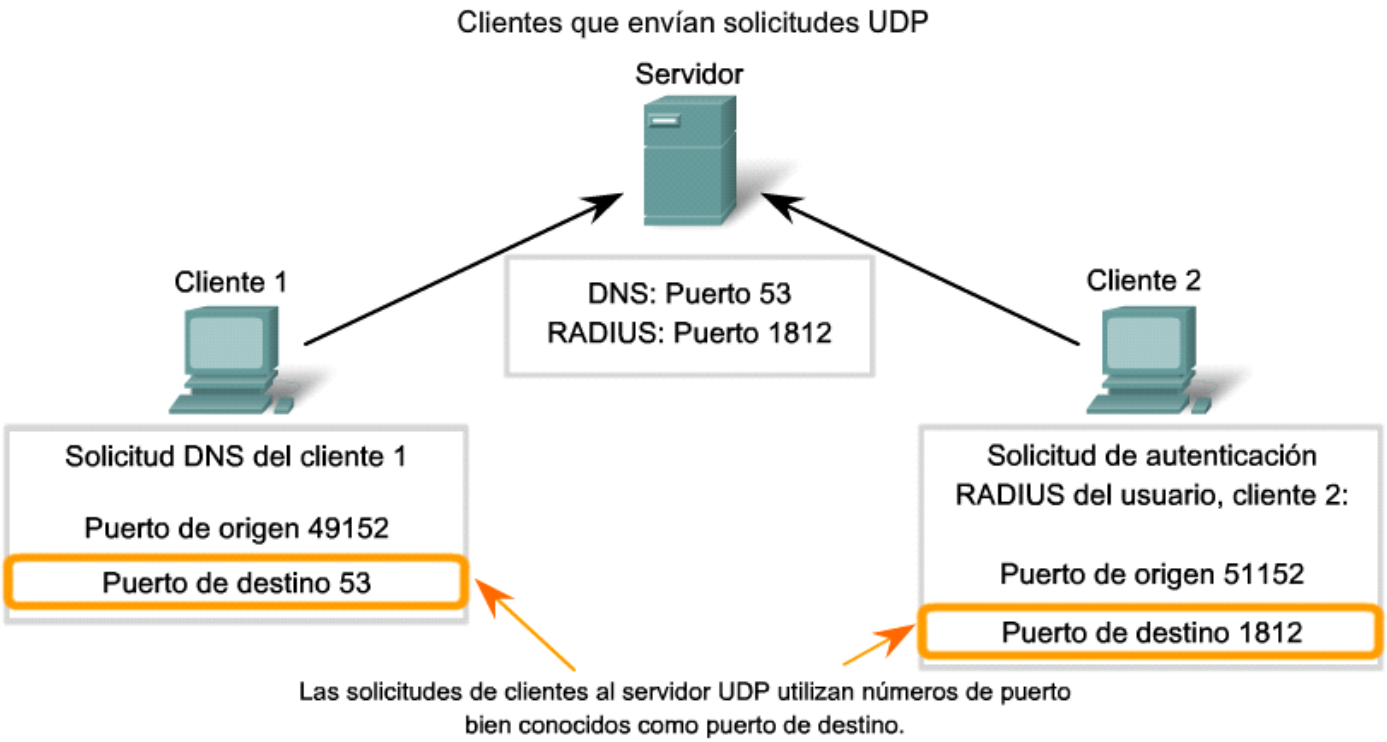
UDP maneja números de puertos para las aplicaciones, de manera similar al protocolo TCP. Ejemplos de puertos con protocolo UDP: DNS tiene asignado el puerto 53, SNMP tiene asignado el puerto 161.



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

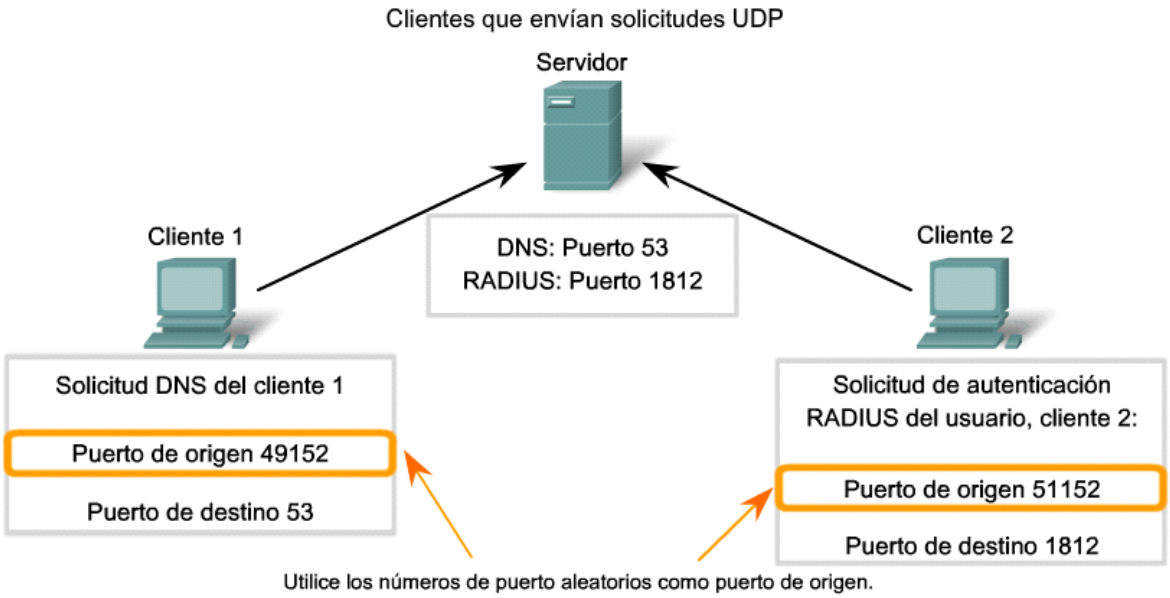
4.4.4.1 Proceso de cliente UDP

Conforme a la aplicación, existen números de puerto de destino bien conocidos. Solicitud de puerto de destino.



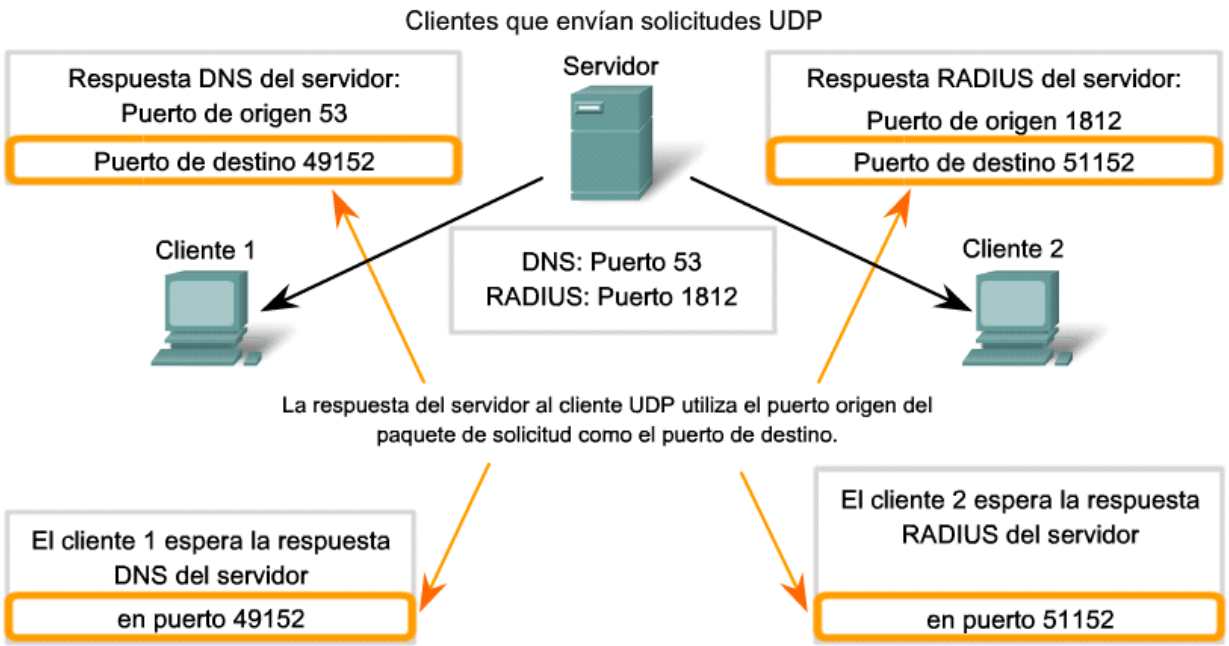
4.4.4.2 Proceso de cliente UDP

Los números de puerto de origen son asignados aleatoriamente. La selección al azar colabora con la seguridad, esto impide que algunos usuarios traten de reemplazar o sustituir a otros usuarios escribiendo intencionalmente el probable número de puerto asignado.



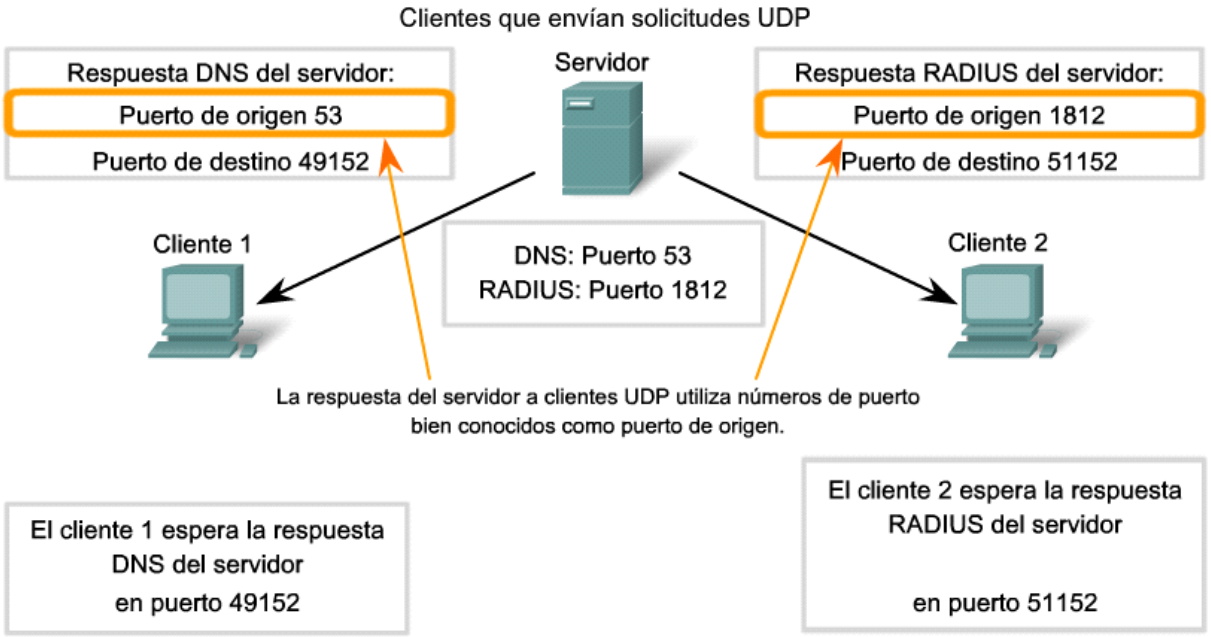
4.4.4.3 Proceso de cliente UDP

El servidor contesta con los números de puerto de origen y destino intercambiados con relación a la asignación original.



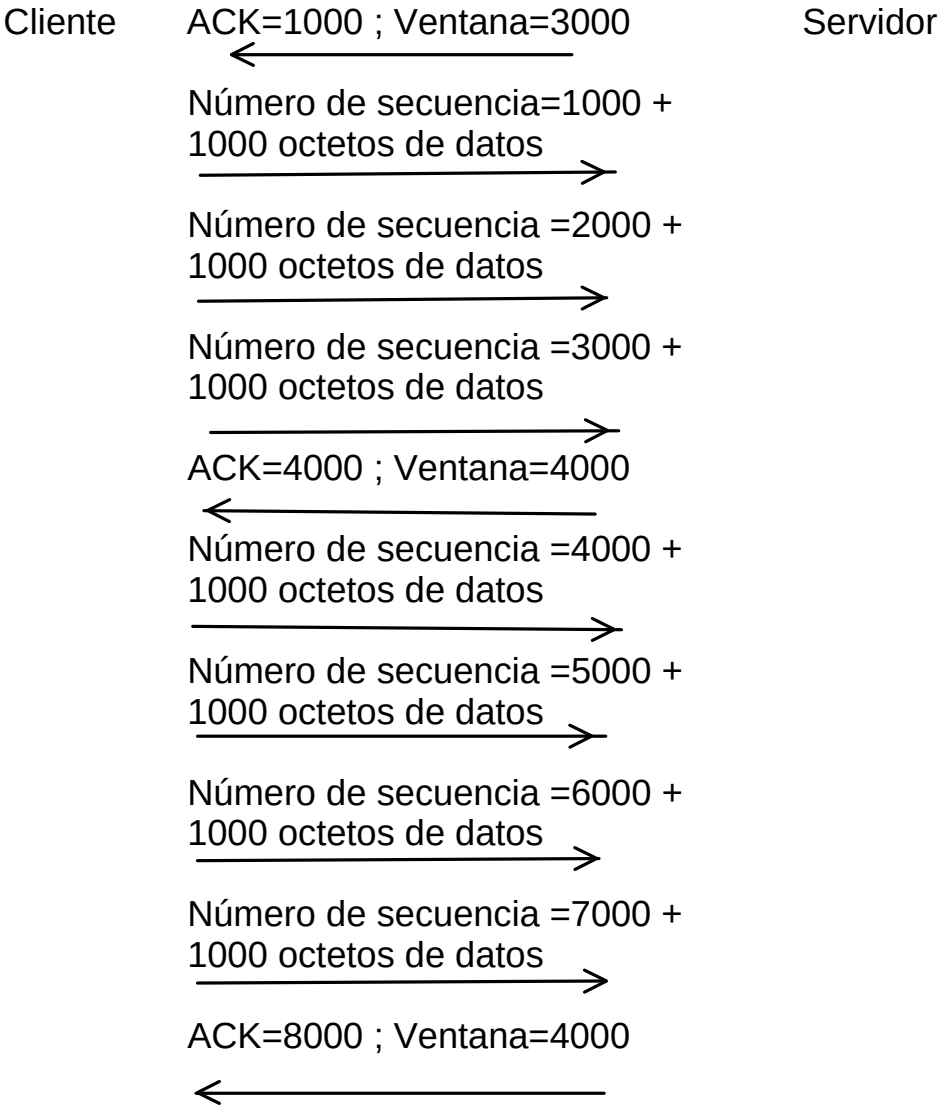
4.4.4.4 Proceso de cliente UDP

Respuesta de puertos de origen. Los números de puerto de origen y destino permanecen inalterables cuando se accede a una aplicación.



4.5.3 Ejemplo: Control de flujo con el mecanismo de Ventana

Ventana dinámica, ventana deslizante.
En este ejemplo cada segmento tiene 1000 octetos, la ventana inicial es de 3000 octetos.
El emisor debe esperar la recepción de un acuse de recibo para continuar con el envío de los siguientes segmentos.
La comunicación no ha tenido error alguno por un largo tiempo, en consecuencia el valor de la ventana cambia a 4000 octetos.



El siguiente grupo de octetos funcionará con un valor de ventana = 4000.

4.5.4 Ejemplo: Establecimiento y finalización de la conexión

Establecimiento de la conexión HTTP

Se considera que los segmentos de conexión y desconexión transportan un octeto de datos en cada paquete.

Cliente Servidor

Número de sec_C = 200 ; Bit SYN=1
Puerto_O = 1027 ; Puerto_D = 80

Número de sec_S=1450; Ack_S=201; Bit SYN=1
Bit ACK=1; Puerto_O=80 ; Puerto_D=1027

Número de sec_C=201; Ack_C=1451; Bit ACK=1
Puerto_O=1027 ; Puerto_D=80

Finalización de la conexión HTTP

Cliente Servidor

Número de sec_C=1000 ; Bit FIN=1
Ack_C= n+1 , el cliente recibió n octetos
con anterioridad

Bit ACK=1; Ack_S=1001

Número de sec_S=1470 ; Bit FIN=1

Bit ACK=1; Ack_C=1471

4.5.5 Ejemplo

Complete los números de secuencia o asentimiento en la siguiente gráfica.