CAPÍTULO 20

Protocolos de transporte

◆ ◆

CUESTIONES BÁSICAS

- El protocolo de transporte proporciona un servicio de transferencia de datos extremo a extremo que aísla las capas superiores de los detalles de la red o redes intermedios. Un protocolo de transporte puede ser orientado a conexión, como en el caso de TCP, o no orientado a conexión, como ocurre con UDP.
- Si el servicio de la red o interconexión subyacente no es fiable, como ocurre en el caso de IP, un protocolo fiable de transporte orientado a conexión resulta ser muy complejo. La causa básica de esta complejidad reside en la necesidad de tratar con los retardos variables y relativamente altos que se experimentan entre sistemas finales. Estos retardos complican las técnicas de control de flujo y de control de errores.
- TCP emplea una técnica de control de flujo basada en créditos que es, en cierta forma, diferente del control de flujo con ventana deslizante que encontramos en X.25 y HDLC. Básicamente, TCP separa las confirmaciones y la gestión del tamaño de la ventana deslizante.
- Aunque el mecanismo basado en créditos de TCP se diseñó para el control de flujo extremo a extremo, también se utiliza para ayudar en el control de congestión en las interconexiones de redes. Cuando una entidad TCP detecta la presencia de congestión en Internet, reduce el flujo de datos que envía por Internet hasta que detecta alivio en la congestión.

◆ ◆

En una arquitectura de protocolos, el protocolo de transporte se sitúa sobre la capa de red o de interconexión, que proporciona los servicios relacionados con la red, y justo debajo de las capas de aplicación y de otros protocolos de capas superiores. El protocolo de transporte proporciona servicios a los usuarios del servicio de transporte (TS, *Transport Service*), como FTP, SMTP y TELNET. La entidad local de transporte se comunica con alguna otra entidad de transporte remota utilizando los servicios de alguna capa inferior, como puede ser el protocolo Internet (IP). El servicio general proporcionado por un protocolo de transporte es el transporte de datos extremo a extremo, de forma que aísle al usuario TS de los detalles de los sistemas de comunicaciones subyacentes.

Comenzaremos este capítulo analizando los mecanismos de protocolo que se requieren para proporcionar los servicios anteriormente citados. Encontraremos que la mayor parte de la complejidad se refiere a los servicios fiables orientados a conexión. Como cabría esperar, cuanto menos ofrece el servicio de red, más tareas debe realizar el protocolo de transporte. El resto del capítulo examina los dos protocolos de transporte más utilizados: el protocolo de control de transmisión (TCP, *Transmission Control Protocol*) y el protocolo de datagrama de usuario (UDP, *User Datagram Protocol*).

La Figura 2.15 destaca la posición de estos protocolos dentro del conjunto de protocolos TCP/IP.

20.1. MECANISMOS DE LOS PROTOCOLOS DE TRANSPORTE ORIENTADOS A CONEXIÓN

Son posibles dos tipos básicos de servicio de transporte: orientado a conexión y no orientado a conexión o servicio de datagramas. Un servicio orientado a conexión proporciona el establecimiento, mantenimiento y cierre de una conexión lógica entre usuarios de TS. Éste ha sido hasta ahora el

tipo de servicio de protocolo más utilizado y tiene una gran variedad de aplicaciones. El servicio orientado a conexión implica generalmente que el servicio es fiable. Esta sección examina los mecanismos del protocolo de transporte necesarios para ofrecer un servicio orientado a conexión.

Un protocolo de transporte orientado a conexión completo, como TCP, es muy complejo. Por motivos de claridad, presentamos los mecanismos del protocolo de transporte de forma incremental. Empezaremos con un servicio de red que facilite el funcionamiento del protocolo de transporte garantizando la entrega en orden de todas las unidades de datos de transporte y definiendo los mecanismos requeridos. Después examinaremos los mecanismos de protocolo de transporte requeridos para hacer frente a un servicio de red no fiable. Toda esta discusión se aplica en general a los protocolos de la capa de transporte. En la Sección 20.2, aplicaremos los conceptos desarrollados en esta sección para describir TCP.

SERVICIO DE RED DE ENTREGA ORDENADA FIABLE

Supongamos que el servicio de red acepta mensajes de tamaño arbitrario y que, con prácticamente una fiabilidad del 100 por cien, los entrega en secuencia al destino. Algunos ejemplos de estas redes son:

- Una red de conmutación de paquetes altamente fiable con una interfaz X.25.
- Una red de retransmisión de tramas que utilice el protocolo de control LAPF.
- Una LAN IEEE 802.3 que utilice el servicio LLC orientado a conexión.

En todos esos casos, el protocolo de transporte se utiliza como un protocolo extremo a extremo entre dos sistemas finales conectados a la misma red, en lugar de hacerlo a través de una interconexión de red.

La suposición de servicios de red con entrega en orden fiable permite el uso de un protocolo de transporte bastante sencillo. Hay cuatro cuestiones a considerar:

- Direccionamiento.
- Multiplexación.
- Control de flujo.
- Establecimiento/cierre de la conexión.

Direccionamiento

La cuestión sobre el direccionamiento es simplemente ésta: un usuario de una entidad de transporte dada desea, bien establecer una conexión, o bien realizar una transferencia de datos con un usuario de alguna otra entidad de transporte que utilice el mismo protocolo de transporte. Es necesario que se identifique al usuario destino mediante toda la información siguiente:

- Identificación del usuario. ----> estación+puerto
- Identificación de la entidad de transporte.
- Dirección de la estación.
- Número de la red.

El protocolo de transporte debe ser capaz de extraer de la dirección del usuario TS la información indicada anteriormente. Normalmente, la dirección de usuario se especifica como (*estación*, *puerto*). La variable *puerto* representa un usuario TS particular en la estación especificada. En general,

Un usuario de TS, sería una aplicación en un equipo, por ejemplo
la pestaña de un navegador web. Otro usuario de TS puede ser un proceso.

Que diferencias interpreta el alumno entre porceso y aplicación?

socket : combinación de puerto y estación (ip)

existirá una sola entidad de transporte en cada estación, por lo que no se necesita una identificación de la entidad. Si más de una entidad de transporte está presente, normalmente hay una de cada tipo. En este último caso, la dirección debe incluir una indicación del tipo de protocolo de transporte (por ejemplo, TCP o UDP). En el caso de una única red, la *estación* identifica a un dispositivo de red conectado a la misma. En el caso de un conjunto de redes interconectadas, *estación* es una dirección global de red. En TCP, la combinación del puerto y la estación se denomina *socket*.

Ya que el encaminamiento no es una cuestión de la capa de transporte, ésta simplemente pasa la parte de la dirección *estación* al servicio de red. El campo *puerto* se incluye en una cabecera de transporte para que la entidad del protocolo de transporte destino la utilice. =>puerto existe en capa transporte solamente!!

Queda todavía una cuestión por abordar: ¿cómo puede el usuario TS que inicia la comunicación conocer la dirección del usuario TS destino? Existen dos estrategias estáticas y dos dinámicas que se explican por sí mismas:

1. El usuario TS previamente conoce la dirección que desea utilizar. Ésta es básicamente una función de configuración del sistema. Por ejemplo, puede estar ejecutándose un proceso que sólo interese a un número limitado de usuarios TS, como por ejemplo un proceso que recoja estadísticas sobre prestaciones. De vez en cuando, una rutina de gestión de red central se conecta al proceso para obtener las estadísticas. En general, estos procesos no son, y no deben ser, bien conocidos ni accesibles por todos.

procesos dirección conocida

2. A algunos servicios usados comúnmente se le asignan direcciones bien conocidas. Por ejemplo, servidores de FTP, SMTP y algunos otros protocolos estándares.

servicios dirección conocido

si no conocemos donde están no podemos usar el servicio!

3. Proporcionando un servidor de nombres. El usuario TS solicita un servicio mediante algún nombre genérico o global. Esta petición se envía a un servidor de nombres, que realiza una búsqueda en un directorio y devuelve una dirección. La entidad de transporte procede entonces con la conexión. Este servicio es útil para aplicaciones comunes que cambien su localización de vez en cuando. Por ejemplo, un proceso de entrada de datos se podría cambiar de una estación a otra en una red local para balancear la carga.

servicios resolución de nombre

4. En algunos casos, el usuario destino es un proceso creado en el momento de la solicitud. El usuario que inicia la comunicación puede enviar una solicitud a una dirección bien conocida. El usuario en esa dirección es un proceso del sistema con privilegios que genera al nuevo proceso y devuelve una dirección. Por ejemplo, un programador ha desarrollado una aplicación privada (por ejemplo, un programa de simulación) que se ejecutará en un servidor remoto, pero que será invocado desde una estación de trabajo local. Se puede mandar una petición a un proceso de gestión de trabajos remoto para que lance el proceso de simulación.

proceso cambia a una nueva dirección

Multiplexación

El concepto de multiplexación se discutió en términos generales en la Sección 18.1. Con respecto a la interfaz entre el protocolo de transporte y los protocolos de capas superiores, el protocolo de transporte lleva a cabo una función de multiplexado/demultiplexado. Es decir, múltiples usuarios emplean el mismo protocolo de transporte, distinguiéndose unos de otros mediante números de puerto o puntos de acceso al servicio.

La entidad de transporte también puede llevar a cabo una función de multiplexación con respecto a los servicios de red que usa. Recuerde que definimos multiplexación hacia arriba como la multiplexación de múltiples conexiones sobre una única conexión de la capa inferior y multiplexa-

ción hacia abajo como la división de una única conexión entre múltiples conexiones de capas inferiores.

Por ejemplo, considere una entidad de transporte que haga uso de un servicio de X.25. ¿Por qué debería una entidad de transporte emplear multiplexación hacia arriba? Después de todo, hay 4.095 circuitos virtuales disponibles. En el caso típico, esto es más que suficiente para gestionar todos los usuarios TS activos. Sin embargo, la mayoría de las redes X.25 basan parte de su tarificación en el tiempo de conexión del circuito virtual, ya que cada circuito virtual consume algunos recursos de memoria temporal del nodo. Por ello, si un solo circuito virtual proporciona el rendimiento suficiente para varios usuarios TS, es indicado utilizar multiplexación hacia arriba.

Por otra parte, la división o multiplexación hacia abajo se podría emplear para mejorar el rendimiento. Por ejemplo, cada circuito virtual X.25 está restringido a un número de secuencia de 3 o 7 bits. Para redes de alta velocidad y gran retardo podría requerirse un mayor espacio de números de secuencia. Por supuesto, el rendimiento sólo se puede incrementar hasta este nivel. Si sólo existe un único enlace estación-nodo sobre el cual se multiplexan todos los circuitos virtuales, el rendimiento de una conexión de transporte no puede exceder la velocidad de transmisión de datos de ese enlace.

Control de flujo

Mientras que el control de flujo es un mecanismo relativamente sencillo en la capa de enlace, en la capa de transporte constituye un mecanismo bastante complejo por dos razones principales:

- El retardo de transmisión entre entidades de transporte es generalmente grande comparado con el tiempo de transmisión real. Esto significa que existe un retardo considerable en la comunicación de la información de control de flujo.
- Ya que la capa de transporte opera sobre una red o una interconexión de redes, la cantidad de retardo en la transmisión puede ser muy variable. Esto hace difícil utilizar de forma efectiva un mecanismo de tiempos de expiración para la retransmisión de datos perdidos.

En general existen dos razones por las que una entidad de transporte querría moderar la tasa de transmisiones de segmentos[1] sobre una conexión de otra entidad de transporte:

- Que el usuario de la entidad de transporte receptora no pueda mantener la tasa del flujo de datos que recibe.
- Que la propia entidad de transporte receptora no pueda mantener la tasa del flujo de segmentos.

¿Cómo se manifiestan los problemas mencionados? Presumiblemente, una entidad de transporte tiene una cierta capacidad de memoria temporal. Los segmentos que se reciben se almacenan en esa memoria. Cada segmento almacenado es procesado (es decir, se examina su cabecera de transporte) y los datos se envían al usuario de TS. Cualquiera de los dos problemas mencionados antes causará que la memoria temporal se llene. Por ello, la entidad de transporte necesita tomar medidas para detener o disminuir el flujo de segmentos con objeto de evitar el desbordamiento de la memoria temporal. Este requisito es difícil de cumplir a causa del molesto intervalo de tiempo que hay entre el emisor y el receptor. Volveremos a este punto más adelante. Primero, vamos a presentar cuatro formas de hacer frente al requisito de control de flujo. La entidad de transporte receptora puede:

[1] Recuerde del Capítulo 2 que los bloques de datos (unidades de datos del protocolo) intercambiados por las entidades TCP se denominan segmentos TCP.

1. No hacer nada.
2. Rechazar la aceptación de más segmentos del servicio de red.
3. Usar un protocolo de ventana deslizante fija.
4. Usar un esquema de créditos.

La alternativa 1 significa que los segmentos que desborden la memoria temporal serán descartados. La entidad de transporte emisora, al no obtener una confirmación, los retransmitirá. Esto es inaceptable, ya que la ventaja de una red fiable es que uno nunca tiene que retransmitir. Es más, el efecto de esta maniobra es que se agrava el problema: el emisor incrementa sus envíos para incluir nuevos segmentos además de los antiguos.

La segunda alternativa es un mecanismo de contrapresión que se basa en el servicio de red para hacer el trabajo. Cuando una memoria temporal de una entidad de transporte está llena, esta entidad rechaza datos adicionales del servicio de red. Esto dispara los procedimientos de control de flujo dentro de la red que regulan el servicio de red en el extremo del emisor. En respuesta, este servicio rechaza segmentos adicionales de su entidad de transporte. Debe quedar claro que este mecanismo es poco preciso y tosco. Por ejemplo, si varias conexiones de transporte se multiplexan sobre una única conexión de red (circuito virtual), el control de flujo se ejerce sólo sobre el agregado de todas las conexiones de transporte.

La tercera estrategia resultará familiar al lector del análisis sobre los protocolos de la capa de enlace del Capítulo 7. Recuerde que los ingredientes clave son:

- El empleo de números de secuencia en las unidades de datos.
- El empleo de una ventana de tamaño fijo.
- El empleo de confirmaciones para avanzar la ventana.

Con un servicio de red fiable, la técnica de ventana deslizante funcionaría realmente bien. Por ejemplo, considere un protocolo con un tamaño de ventana de 7. Cuando el emisor recibe una confirmación de un segmento particular, se le autoriza automáticamente a enviar los siete segmentos siguientes (por supuesto, algunos pueden haber sido ya enviados). Cuando la capacidad de la memoria temporal del receptor disminuya a 7 segmentos, el receptor puede retener las confirmaciones de los segmentos que reciba para evitar el desbordamiento. La entidad de transporte emisora puede enviar como mucho siete segmentos adicionales y después debe dejar de enviar. Como el servicio de red subyacente es fiable, los temporizadores del emisor no expirarán ni habrá retransmisión. Así, en algún punto, una entidad de transporte emisora puede tener varios segmentos pendientes para las cuales no se ha recibido confirmación. Ya que trabajamos con una red fiable, la entidad de transporte emisora puede suponer que los segmentos se han entregado y que la ausencia de confirmaciones es debida a una táctica de control de flujo. Esta táctica no funcionará bien en una red no fiable, ya que la entidad de transporte emisora no sabría si la falta de confirmaciones se debe al control de flujo o a la pérdida de un segmento. No puedo saber quien ocasiona la demora. Flujo o Perdidas?

La cuarta alternativa, un esquema de créditos, proporciona al receptor un mayor grado de control sobre el flujo de datos. Aunque no es estrictamente necesario con un servicio de red fiable, un esquema de créditos debe dar lugar a un flujo de tráfico más fluido. Además, es un esquema más efectivo con un servicio de red no fiable, como veremos.

El esquema de créditos desliga las confirmaciones y el control de flujo. En los protocolos de ventana deslizante fija, como X.25, los dos conceptos son sinónimos. En un esquema de créditos, se puede confirmar un segmento sin la concesión de nuevo crédito y viceversa. En el esquema de créditos se considera que cada octeto individual de datos que se transmite tiene un número de

Puedo confirmar , pero no dar mas crédito, por lo tanto separo el control de flujo de las confirmaciones.
En la ventana deslizante , la confirmación estaba unida al control de flujo.

Cada Octeto transmitido tiene un nro. que lo identifica en la secuancia de octetos transmitidos.

en la cabecera de cada segmento.
SN Número de Secuencia.
AN Número de Confirmación (con esto valido)
W ancho de la Ventana. (con esto controlo el flujo)

secuencia único. Además de los datos, cada segmento transmitido incluye en su cabecera tres campos relacionados con el control de flujo: el número de secuencia (*SN*), el número de confirmación (*AN*) y la ventana (*W*). Cuando una entidad de transporte envía un segmento, incluye el número de secuencia del primer octeto del campo de datos del segmento. Una entidad de transporte confirma un segmento recibido con un segmento de retorno que incluye ($AN = i$, $W = j$), con la siguiente interpretación:

- Todos los octetos cuyos números de secuencia lleguen hasta $SN = i - 1$ se confirman. El siguiente octeto esperado tiene número de secuencia i.
- Se concede permiso para enviar una ventana adicional de $W = j$ octetos de datos. Es decir, los j octetos correspondientes a los números de secuencia desde i hasta $i + j - 1$.

La Figura 20.1 ilustra este mecanismo (compárese con la Figura 7.4). Por simplicidad, se muestra el flujo de datos en un solo sentido y se supone que en cada segmento se envían 200 octetos. Inicialmente, a través del proceso de establecimiento de la conexión, se sincronizan los números de secuencia de emisión y recepción y A obtiene una asignación inicial de 1.400 octetos, comenzando con el octeto número 1.001. Después de enviar 600 octetos en tres segmentos, A ha reducido su ventana a un tamaño de 800 octetos (números del 1.601 al 2.400). Después de que B reciba esos tres segmentos, se contabilizan 600 octetos de los 1.400 originales de crédito, quedando pendientes 800 créditos. Suponga ahora que, llegados a este punto, B es capaz de absorber 1.000 octetos de datos provenientes de esta conexión. De acuerdo a esto, B confirma la recepción de todos los octetos hasta 1.600 y emite un crédito de 1.000 octetos. Esto significa que A puede enviar los octetos comprendidos entre 1.601 y 2.600 (5 segmentos). Sin embargo, cuando el mensaje de B haya llegado a A, A ya habrá enviado dos segmentos, que contienen los octetos del 1.601 al 2.000 (lo cual se permitía según la reserva inicial). Así, el crédito restante de A tras la recepción de la cuota de crédito de B es sólo de 600 octetos (3 segmentos). Conforme el intercambio prosigue, A avanza el borde final de su ventana cada vez que transmite y avanza el borde inicial sólo cuando se le concede crédito.

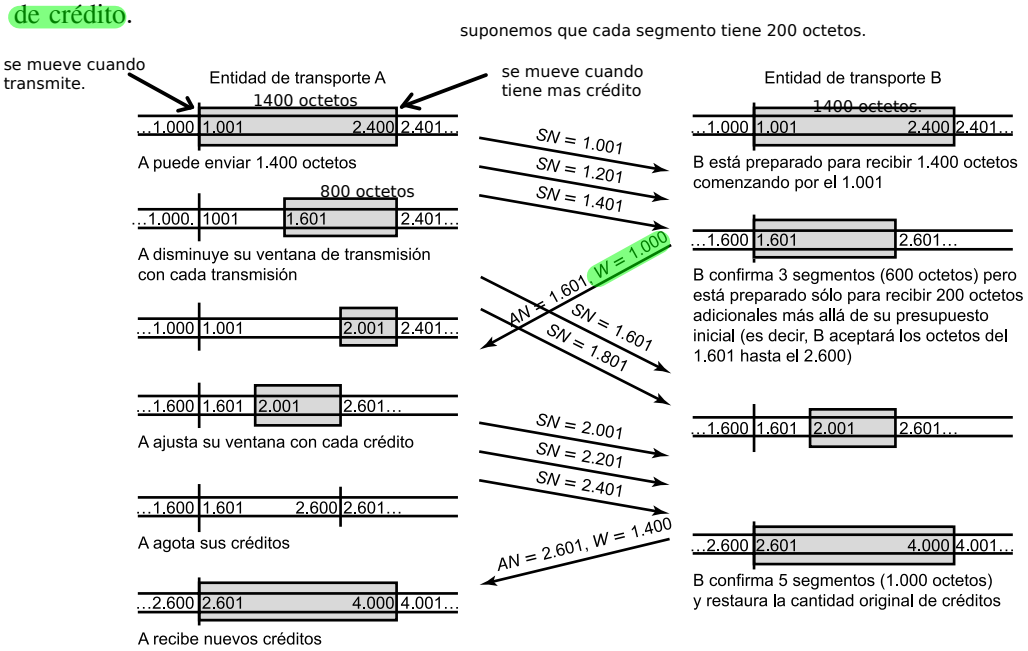


Figura 20.1. Ejemplo del mecanismo de asignación de crédito de TCP.

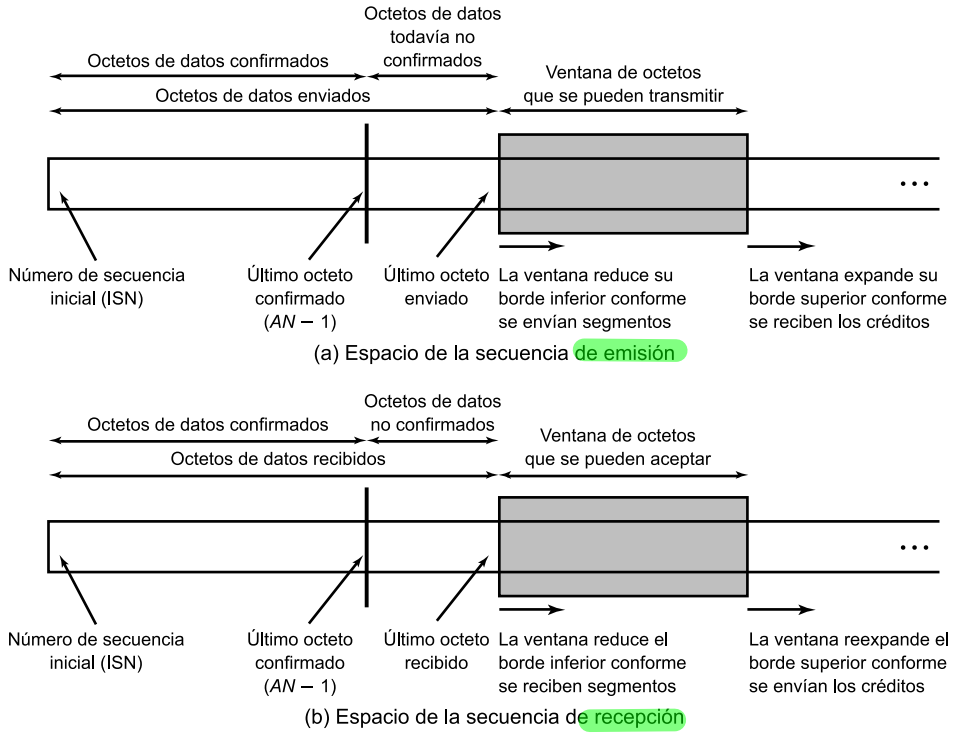


Figura 20.2. Perspectivas del control de flujo en el envío y en la recepción.

La Figura 20.2 muestra una visión de este mecanismo según el emisor y según el receptor (compárese con la Figura 7.3). Normalmente, ambos extremos tienen ambas perspectivas, ya que los datos se pueden intercambiar en ambos sentidos. Observe que no se requiere que el receptor confirme inmediatamente los segmentos recibidos, sino que puede esperar y emitir una confirmación acumulada para varios segmentos.

El receptor necesita adoptar alguna política sobre la cantidad de datos que le va a permitir transmitir al emisor. La opción conservadora consiste en permitir sólo nuevos segmentos hasta el límite del espacio de memoria temporal disponible. Si esta política fuera la utilizada en la Figura 20.1, el primer mensaje de crédito implica que B dispone de 1.000 octetos en su memoria temporal y el segundo mensaje, que B tiene 1.400 octetos disponibles.

Un esquema de control de flujo conservador puede limitar el rendimiento de la conexión de transporte en situaciones de gran retardo. El receptor podría incrementar potencialmente el rendimiento mediante la concesión de forma optimista de créditos de espacio que no tiene. Por ejemplo, si la memoria temporal del receptor está llena, pero anticipa que puede liberar el espacio para 1.000 octetos dentro del tiempo de propagación de ida y vuelta, podría enviar inmediatamente un crédito de 1.000 octetos. Si el receptor puede ir al paso del emisor, este esquema podría incrementar el rendimiento sin causar perjuicio alguno. Sin embargo, si el emisor es más rápido que el receptor, algunos segmentos pueden ser descartados, necesitando una retransmisión. Ya que las retransmisiones no son necesarias en otro caso con un servicio de red fiable (en ausencia de congestión en la interconexión de redes), un esquema optimista de control de flujo complicará el protocolo.

Establecimiento y cierre de la conexión

Incluso con un servicio de red fiable, existe la necesidad de procedimientos de establecimiento y cierre de conexión para ofrecer un servicio orientado a conexión. El establecimiento de la conexión cumple tres objetivos principales:

- Permite a cada extremo asegurarse de que el otro existe.
- Permite el intercambio o negociación de parámetros opcionales (por ejemplo, el tamaño máximo del segmento, el tamaño máximo de la ventana y la calidad de servicio).
- Inicia la reserva de recursos de la entidad de transporte (por ejemplo, espacio de memoria temporal y entradas en la tabla de conexiones).

El establecimiento de la conexión se realiza por mutuo acuerdo, pudiéndose llevar a cabo mediante un conjunto sencillo de órdenes de usuario y segmentos de control, como se muestra en el diagrama de estados de la Figura 20.3. Para comenzar, un usuario TS está en un estado *CLOSED* («CERRADO», es decir, no tiene una conexión de transporte abierta). El usuario TS puede indicar a la entidad TCP local que esperará de forma pasiva una solicitud con una orden de *Passive Open* («apertura pasiva»). Esto es lo que podría hacer un programa servidor, como una aplicación de tiempo compartido o una aplicación de transferencia de ficheros. El usuario TS puede cambiar de idea enviando una orden *Close* («cerrar»). Después de haberse emitido la orden *Passive Open*, la entidad de transporte crea un objeto de conexión de algún tipo (es decir, una entrada en una tabla) que está en el estado *LISTEN* («PREPARADO»).

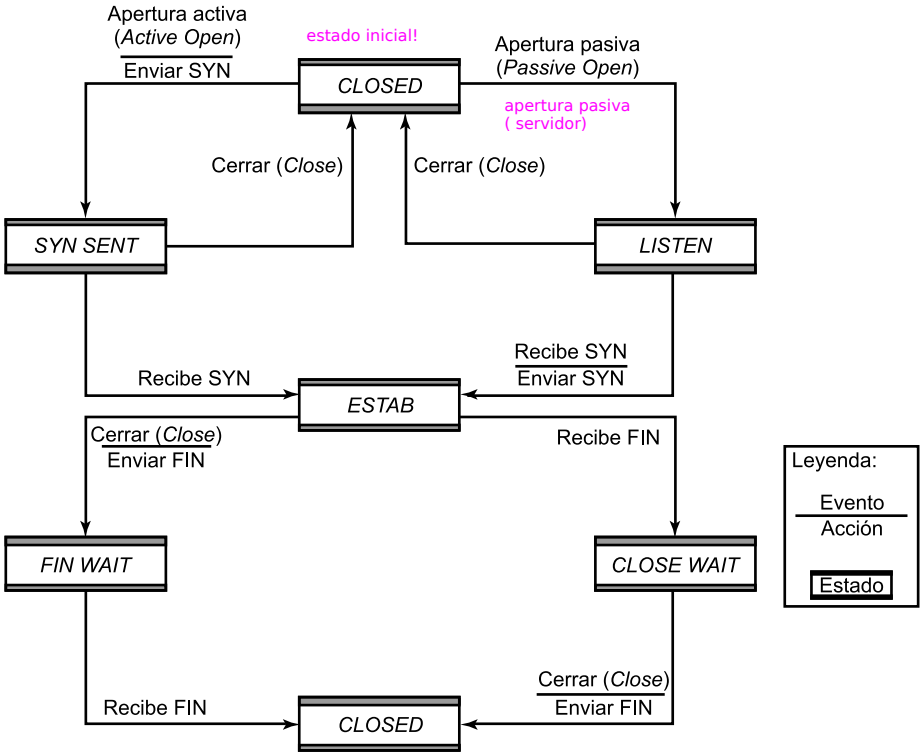


Figura 20.3. Diagrama de estados de la conexión simple.

Desde el estado *CLOSED*, el usuario TS puede abrir una conexión emitiendo una orden *Passive Open*, la cual instruye a la entidad de transporte para que intente establecer una conexión con un usuario TS remoto designado, lo que hace que la entidad de transporte envíe un segmento SYN (para sincronizar). El segmento se entrega a la entidad de transporte receptora, donde se interpreta como una solicitud de conexión a un puerto concreto. Si la entidad de transporte destino está en el estado LISTEN para ese puerto, entonces se establece una conexión por medio de las acciones siguientes, realizadas por la entidad de transporte receptora:

- Informa al usuario TS local que una conexión está abierta.
- Envía un segmento SYN como confirmación a la entidad de transporte remota.
- Sitúa el objeto de conexión en el estado ESTAB (establecida).

Cuando el segmento SYN de respuesta lo recibe la entidad de transporte que inició el proceso, ella también puede pasar la conexión al estado ESTAB. La conexión se interrumpe prematuramente si cualquier usuario TS emite una orden *Close*.

La Figura 20.4 muestra la robustez de este protocolo. Cualquier extremo puede iniciar una conexión. Además, si ambas partes inician la conexión en instantes próximos, ésta se establece sin confusión. Esto es así porque el segmento SYN funciona como petición y como confirmación de la conexión.

El lector se puede preguntar qué ocurre si llega un segmento SYN en un momento en el que el usuario TS solicitado esta inactivo (no atendiendo peticiones). Se pueden seguir tres alternativas:

- La entidad de transporte puede rechazar la petición enviando un segmento RST («reiniciar») a la otra entidad de transporte.
- La solicitud se puede poner en cola hasta que el usuario TS local emita una orden *Open* («abrir») correspondiente.
- La entidad de transporte puede interrumpir al usuario TS local para notificarle una solicitud pendiente.

Observe que si se utiliza este último mecanismo, no es estrictamente necesaria una orden *Passive Open*, sino que se podría sustituir por una orden *Accept* («aceptar»), que es una señal del usuario utilizada para indicarle a la entidad de transporte que acepta la solicitud de conexión.

El cierre de la conexión se trata de manera similar. Pueden iniciar el cierre cualquiera de los extremos, o ambos a la vez. La conexión se cierra por mutuo acuerdo. Esta estrategia permite un

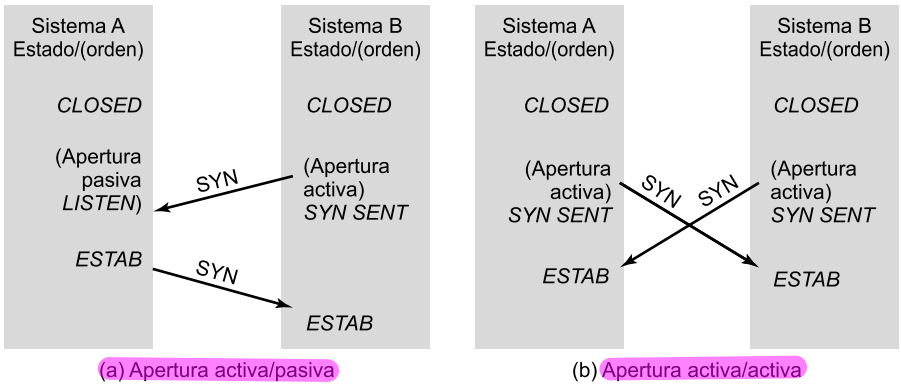


Figura 20.4. Escenarios de establecimiento de la conexión.

caso cliente / servidor

cierre abrupto u ordenado. En un cierre abrupto, los datos en tránsito pueden perderse. Un cierre ordenado impide a ambas partes cerrar la conexión hasta que todos los datos hayan sido entregados. Para conseguir esto último, una conexión que está en el estado *FIN WAIT* («ESPERA CIERRE») debe continuar aceptando segmentos de datos hasta que se reciba un segmento FIN («finalizar»).

La Figura 20.3 muestra el procedimiento para un cierre ordenado. Primero, consideremos el extremo que inicia el proceso de cierre:

1. En respuesta a una primitiva *Close* del usuario TS, la entidad de transporte envía un segmento FIN al otro extremo de la conexión, solicitando el cierre.
2. Habiendo enviado el segmento FIN, la entidad de transporte sitúa la conexión en el estado FIN WAIT. En este estado, la entidad de transporte debe continuar aceptando datos del otro extremo y entregarlos a su usuario.
3. Cuando se recibe como respuesta un segmento FIN, la entidad de transporte informa a su usuario y cierra la conexión.

Desde el punto de vista del extremo que no inicia el cierre:

1. Cuando se recibe un segmento FIN, la entidad de transporte informa a su usuario de la solicitud de cierre y sitúa la conexión en el estado *CLOSE WAIT* («ESPERA CIERRE»). En este estado, la entidad de transporte debe continuar aceptando datos de su usuario y transmitirlos en segmentos de datos al otro extremo.
2. Cuando el usuario emite una primitiva *Close*, la entidad de transporte envía un segmento FIN de respuesta al otro extremo y cierra la conexión.

Este procedimiento asegura que ambos extremos han recibido todos los datos pendientes y que ambos están de acuerdo en terminar la conexión antes del cierre real.

SERVICIO DE RED NO FIABLE

Un caso más difícil para un protocolo de transporte es aquel en que se ofrece un servicio de red no fiable. Ejemplos de tales redes son:

- Una interconexión de redes que utilice IP.
- Una red de retransmisión de tramas que utilice sólo el núcleo del protocolo LAPF.
- Una LAN IEEE 802.3 que use un servicio LLC no orientado a conexión sin confirmaciones.

El problema no consiste sólo en que los segmentos puedan perderse ocasionalmente, sino en que los segmentos pueden no llegar en secuencia debido al retardo variable del tránsito. Como veremos, se necesita un elaborado mecanismo para hacer frente a esas dos deficiencias interrelacionadas de la red. También veremos que se produce un patrón desalentador. La combinación de desorden y ausencia de fiabilidad genera problemas con todos los mecanismos que hemos discutido hasta ahora. Generalmente, la solución a cada problema produce nuevos problemas. Aunque hay problemas que tienen que ser resueltos por los protocolos en todas las capas, parece que existen más dificultades en un protocolo de transporte orientado a conexión fiable que en cualquier otro tipo de protocolo.

En el resto del esta sección, a menos que se indique expresamente, los mecanismos que se tratan son aquellos que utiliza TCP. Siete cuestiones han de ser tratadas:

Problemas de una red real : Perdidas y Fuera de secuencia

- Entrega ordenada.
- Estrategia de retransmisión.
- Detección de duplicados.
- Control de flujo.
- Establecimiento de la conexión.
- Cierre de la conexión.
- Recuperación ante fallos.

Estos puntos están mas allá de contenido de la materia.Continuamos con 20.2

Entrega ordenada

Con un servicio de red no fiable, es posible que los segmentos, incluso en el caso de que lleguen todos, lo hagan de forma desordenada. La solución a este problema consiste en numerar los segmentos secuencialmente. Ya hemos visto que para los protocolos de control del enlace de datos, como HDLC o X.25, cada unidad de datos (trama, paquete) se numera secuencialmente, siendo cada número de secuencia sucesivo mayor en una unidad que el número de secuencia precedente. Este esquema se utiliza en algunos protocolos de transporte, como los protocolos de transporte de ISO. Sin embargo, TCP usa un esquema algo diferente en el que cada octeto de datos que se transmite se numera implícitamente. Así, el primer segmento puede tener el número de secuencia igual a 1. Si ese segmento contiene 200 octetos de datos, entonces el segundo segmento tendría el número de secuencia 201 y así sucesivamente. Por simplicidad, en las discusiones de esta sección supondremos que el número de secuencia de cada segmento sucesivo es 200 más que el del segmento previo, es decir, cada segmento contiene exactamente 200 octetos de datos.

Estrategia de retransmisión

Existen dos eventos que requieren la retransmisión de un segmento. En primer lugar, el segmento se puede dañar en el camino, pero llegar sin embargo a su destino. Si se incluye en el segmento una suma de comprobación, la entidad de transporte receptora puede detectar el error y descartar el segmento. La segunda contingencia consiste en que el segmento no llegue a su destino. En cualquier caso, la entidad de transporte emisora no sabe que la transmisión del segmento no se ha realizado con éxito. Para tratar esta contingencia se utiliza un esquema de confirmaciones positivas: el receptor debe confirmar cada segmento recibido satisfactoriamente devolviendo un segmento que contenga un número de confirmación. Por razones de eficiencia, no se requiere una confirmación por cada segmento. En su lugar, puede utilizarse una confirmación acumulada, como se ha visto ya varias veces en este libro. Así, el receptor puede recibir los segmentos numerados como 1, 201 y 401, pero sólo envía $AN = 601$ de vuelta. El emisor debe interpretar $AN = 601$ como que los segmentos con $SN = 401$ y anteriores se han recibido correctamente.

Si un segmento no llega con éxito, no se enviará una confirmación positiva y se tendrá que efectuar una retransmisión. Para poder hacer frente a esta situación, tiene que haber un temporizador asociado con cada segmento conforme se envía. Si el temporizador expira antes de que se confirme, el emisor debe retransmitir el segmento asociado.

Así pues, la inclusión de temporizadores soluciona ese problema. Siguiente problema: ¿qué valor debe asignarse al temporizador? Existen dos estrategias que surgen por sí mismas. Se podría utilizar un temporizador con un valor fijo, basado en la comprensión del comportamiento típico de la red. Esta estrategia adolece de incapacidad para reaccionar ante los cambios en las condiciones

de la red. Si el valor es demasiado bajo, habrá muchas retransmisiones innecesarias, desperdiciando la capacidad de la red. Si el valor es demasiado alto, el protocolo será muy lento en reaccionar ante la pérdida de un segmento. El temporizador debe fijarse a un valor un poco mayor que el retardo de ida y vuelta (tiempo de enviar un segmento y recibir un ACK). Por supuesto, este retardo es variable incluso para una carga constante de la red. Y lo que es peor, la estadística del retardo variará con las cambiantes condiciones de la red.

La otra estrategia es utilizar un esquema adaptable, el cual tiene sus propios problemas. Supongamos que la entidad de transporte registra el tiempo que se tarda en confirmar los segmentos de datos y fija los temporizadores de retransmisión de acuerdo a la media de los retardos observados. No es posible confiar en este valor por tres razones:

- La entidad de transporte par puede que no confirme inmediatamente un segmento. Recordemos que le hemos dado el privilegio de utilizar confirmaciones acumuladas.
- Si un segmento ha sido retransmitido, el emisor no puede saber si la confirmación positiva recibida es una respuesta a la transmisión inicial o a la retransmisión.
- Las condiciones de la red pueden cambiar de repente.

Cada uno de estos problemas es la causa de alguna complicación adicional del algoritmo de transporte, pero el problema no admite una solución completa. Siempre habrá alguna incertidumbre con respecto al mejor valor para el temporizador de retransmisión.

Por cierto, el temporizador de retransmisión es sólo uno de varios temporizadores necesarios para el correcto funcionamiento del protocolo de transporte. Estos se listan en la Tabla 20.1, junto a una breve explicación.

Tabla 20.1. Temporizadores del protocolo de transporte.

Temporizador de retransmisión	Para retransmitir un segmento no confirmado.
Temporizador de reconexión	Tiempo mínimo entre el cierre de una conexión y el establecimiento de otra con la misma dirección destino.
Temporizador de ventana	Tiempo máximo entre segmentos ACK/CREDIT.
Temporizador de retransmisión de SYN	Intervalo de tiempo entre intentos de establecimiento de una conexión.
Temporizador de persistencia	Utilizado para abortar una conexión cuando no se confirma ningún segmento.
Temporizador de inactividad	Utilizado para abortar una conexión cuando no se recibe ningún segmento.

Detección de duplicados

Si se pierde un segmento y después se retransmite, no se producirá ninguna confusión. Sin embargo, si uno o más segmentos sucesivos se entregan satisfactoriamente pero se pierde el correspondiente ACK, expirará el temporizador de la entidad de transporte emisora y se retransmitirán uno o más segmentos. Si esos segmentos retransmitidos llegan correctamente, se tendrán duplicados de los segmentos anteriormente recibidos. Por ello, el receptor debe ser capaz de reconocer los duplicados. El hecho de que cada segmento lleve un número de secuencia ayuda, pero, de cualquier forma, la detección y gestión de los duplicados no es una tarea fácil. Existen dos casos:

- Se recibe un duplicado antes del cierre de la conexión.
- Se recibe un duplicado después de que se haya cerrado la conexión.

El segundo caso se estudia en la sección acerca del establecimiento de la conexión. Aquí trataremos el primer caso.

Observe que decimos «un» duplicado y no «el» duplicado. Desde el punto de vista del emisor, el segmento retransmitido es el duplicado. Sin embargo, el segmento retransmitido puede llegar antes que el segmento original, en cuyo caso el receptor vería el segmento original como el duplicado. En cualquier caso, se necesitan dos tácticas para tratar el caso de que un duplicado se reciba antes de cerrar una conexión:

- El receptor debe asumir que su confirmación se perdió y, por tanto, debe confirmar el duplicado. Por consiguiente, el emisor debe no confundirse si recibe varias confirmaciones positivas del mismo segmento.
- El espacio de números de secuencia debe ser lo suficientemente amplio para no agotarse antes del tiempo máximo de vida posible de un segmento (tiempo que necesita el segmento para atravesar la red).

La Figura 20.5 ilustra la justificación de este último requisito. En este ejemplo, el espacio de secuencia es de longitud 1.600. Es decir, después de $SN = 1.600$, los números de secuencia vuelven a empezar con $SN = 1$. Por simplicidad, suponemos que la entidad de transporte receptora mantiene una ventana de crédito de tamaño 600. Suponga que A ha transmitido los segmentos de datos con $SN = 1$, 201 y 401. B ha recibido los dos segmentos con $SN = 201$ y 401, pero el segmento con el $SN = 1$ se ha retrasado en el camino. De esta forma, B no envía ninguna confirmación. Eventualmente, el temporizador de A expira y retransmite el segmento $SN = 1$. Cuando llega el duplicado del segmento $SN = 1$, B confirma el 1, el 201 y el 401 con $AN = 601$. Mientras tanto, en A se produce otra expiración y retransmite el $SN = 201$, que confirma B con otro $AN = 601$. Parece que las cosas se han arreglado solas y la transferencia de datos continúa. Cuando el espacio de secuencia se agota, A vuelve a comenzar con el número de secuencia $SN = 1$ y continúa. Desafortunadamente, el antiguo segmento $SN = 1$ hace una aparición tardía y es aceptado por B antes de que el nuevo segmento $SN = 1$ llegue. Cuando el nuevo segmento $SN = 1$ llega, se le trata como un duplicado y se descarta.

Debe quedar claro que la aparición a destiempo de un segmento antiguo no habría causado dificultades si los números de secuencia no se hubieran solapado. El problema se formula así: ¿qué tamaño debe tener el espacio de secuencia? Esto depende, entre otras cosas, de si la red fuerza un tiempo de vida máximo del paquete y de la tasa a la cual los segmentos se transmiten. Afortunadamente, cada incorporación de un único bit al campo de números de secuencia dobla el espacio de secuencias, de forma que es fácil seleccionar un tamaño seguro.

Control de flujo

El mecanismo de control de flujo por medio de la asignación de créditos descrito anteriormente es bastante robusto en presencia de un servicio de red no fiable y requiere pocas mejoras. Como se ha mencionado, un segmento que contenga ($AN = i$, $W = j$) confirma todos los octetos con números de secuencia inferiores a i y concede créditos para j octetos adicionales comenzando por el octeto i. El mecanismo de asignación de créditos es bastante flexible. Por ejemplo, considere que el último octeto de datos recibido por B fue el octeto número $i - 1$ y que el último segmento enviado por B fue ($AN = i$, $W = j$). Entonces:

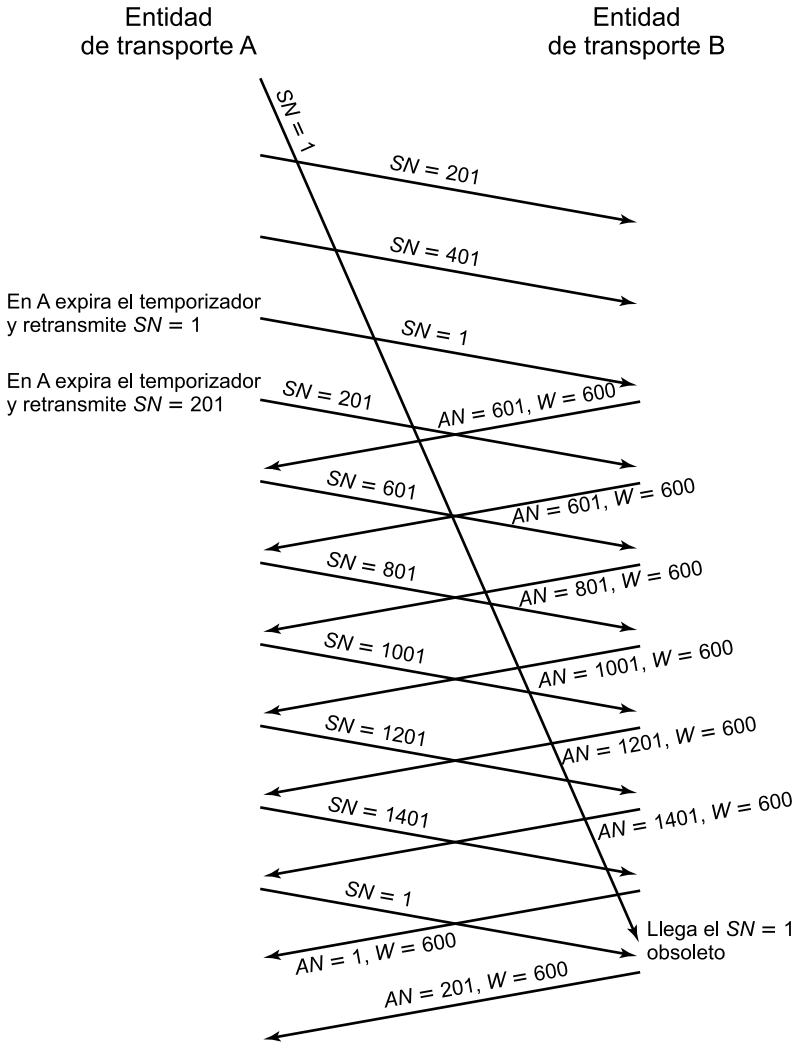


Figura 20.5. Ejemplo de detección incorrecta de duplicado.

- Para incrementar o disminuir los créditos a una cantidad k ($k > j$) cuando no han llegado datos adicionales, B emite ($AN = i$, $W = k$).
- Para confirmar un segmento recibido que contenga m octetos de datos ($m < j$) sin conceder créditos adicionales, B emite ($AN = i + m$, $W = j - m$).

La pérdida de un segmento ACK/CREDIT tiene poco impacto en el funcionamiento del esquema. Confirmaciones posteriores sincronizarán de nuevo el protocolo. Además, si no hay nuevas confirmaciones en camino, en el emisor expirará un temporizador y retransmitirá un segmento de datos, el cual disparará una nueva confirmación. Sin embargo, es aún posible que ocurra un bloqueo mutuo. Considere la situación en la cual B envía ($AN = i$, $W = 0$), cerrando temporalmente la ventana. Con posterioridad, B envía ($AN = i$, $W = j$), pero este segmento se pierde. A está esperando la oportunidad de enviar datos y B piensa que ha concedido esa oportunidad. Para resolver este

problema se puede utilizar un temporizador de ventana. Este temporizador se reinicia cada vez que se envía un segmento (todos los segmentos contienen los campos *AN* y *W*). Si el temporizador expira alguna vez, se le requiere a la entidad de transporte que envíe un segmento, incluso si el nuevo duplica uno anterior. Esto rompe el bloqueo mutuo y le asegura al otro extremo que la entidad de transporte está todavía activo.

Establecimiento de la conexión

Como con otros mecanismos de protocolo, el establecimiento de la conexión debe tener en cuenta la falta de fiabilidad de un servicio de red. Recuérdese que el establecimiento de la conexión requiere el intercambio de SYN, un procedimiento llamado a veces *diálogo en dos pasos*. Supongamos que A emite un SYN a B. Él espera un SYN de vuelta, confirmando la conexión. Dos cosas pueden ir mal: que se pierdan el SYN de A o la respuesta de B. Ambos casos se pueden gestionar mediante el uso de un temporizador de retransmisión de SYN (*véase* Tabla 20.1). A volverá a emitir un SYN cuando el temporizador expire.

Potencialmente, esto puede ocasionar la duplicación de SYN. Si se perdiera el SYN inicial de A, no habría duplicados. Si se perdiera la respuesta de B, entonces B podría recibir dos SYN de A. Es más, si la respuesta de B no se perdiera, sino que simplemente se retrasara, A podría recibir dos SYN de respuesta. Todo esto significa que A y B deben simplemente ignorar los SYN duplicados una vez que la conexión se haya establecido.

Existen otros problemas a los que enfrentarse. Al igual que un SYN retrasado o una respuesta perdida puede producir un SYN duplicado, un segmento de datos retrasado o una confirmación perdida puede dar lugar a la duplicidad de segmentos de datos, como hemos visto en la Figura 20.5. Estos segmentos retrasados o duplicados pueden interferir con la transferencia de datos, tal y como se ilustra en la Figura 20.6. Suponga que con cada nueva conexión, cada entidad del protocolo de transporte inicia la numeración de sus segmentos de datos con el número de secuencia 1. En la figura, una copia duplicada del segmento $SN = 401$ de una antigua conexión llega durante el tiempo de vida de una nueva conexión y se le entrega a la entidad B antes que el segmento de datos legítimo número $SN = 401$. Una forma de abordar este problema es empezar cada nueva conexión con un número de secuencia diferente que difiera lo suficiente del último número de secuencia de la conexión más reciente. Por este motivo, la solicitud de conexión es de la forma SYN *i*, donde *i* es el número de secuencia del primer segmento de datos que será enviado en esta conexión.

Ahora consideremos que un SYN *i* duplicado sobrevive hasta después del cierre de la conexión. La Figura 20.7 representa el problema que puede plantearse. Un SYN *i* obsoleto llega a B después de que la conexión haya terminado. B supone que ésta es una petición nueva y responde con SYN *j*, lo que significa que B acepta la solicitud de conexión y que comenzará a transmitir con $SN = j$. Mientras tanto, A ha decidido abrir una nueva conexión con B y envía SYN *k*. B descarta este último como uno duplicado. Ahora ambos extremos han transmitido y posteriormente recibido un segmento SYN y, por tanto, piensan que existe una conexión válida. Sin embargo, cuando A inicia la transferencia de datos con un segmento numerado con *k*, B rechaza el segmento por no corresponder con la secuencia.

La solución a este problema consiste en que cada lado confirme explícitamente el SYN y número de secuencia del otro. El procedimiento es conocido como *diálogo en tres pasos*. El diagrama de estados de conexión revisado, que es el empleado por TCP, se muestra en la parte superior de la Figura 20.8. Se ha incluido un nuevo estado (*SYN RECEIVED*, «RECIBIDO SYN»). En este

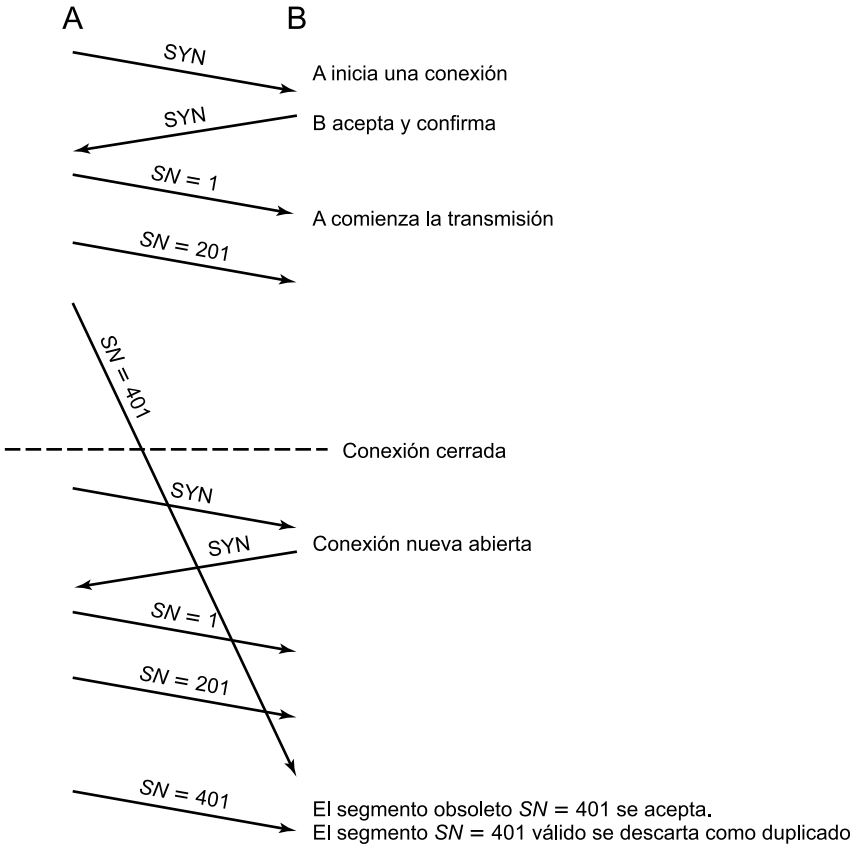


Figura 20.6. Diálogo en dos pasos: problema con un segmento de datos obsoleto.

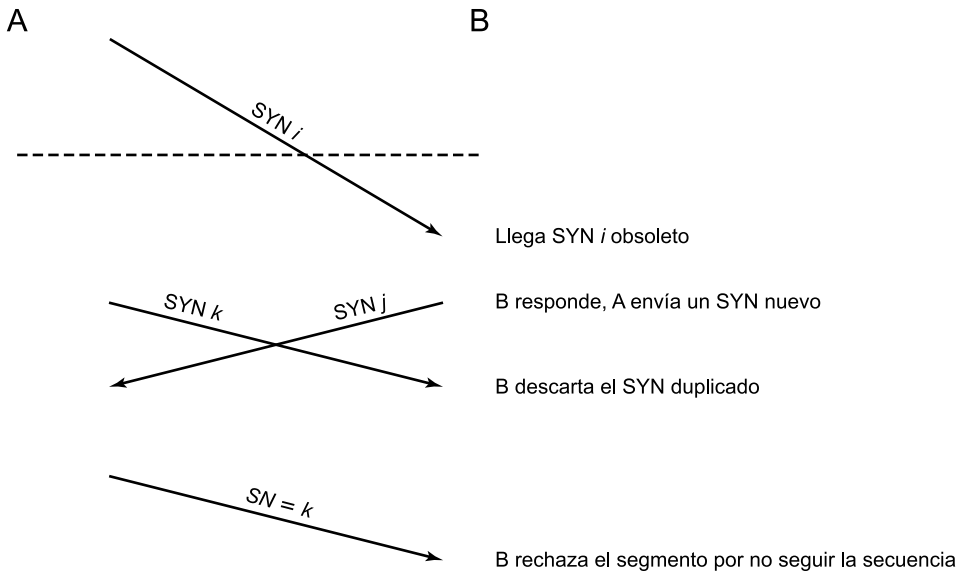


Figura 20.7. Diálogo en dos pasos: problema con segmentos SYN obsoletos.

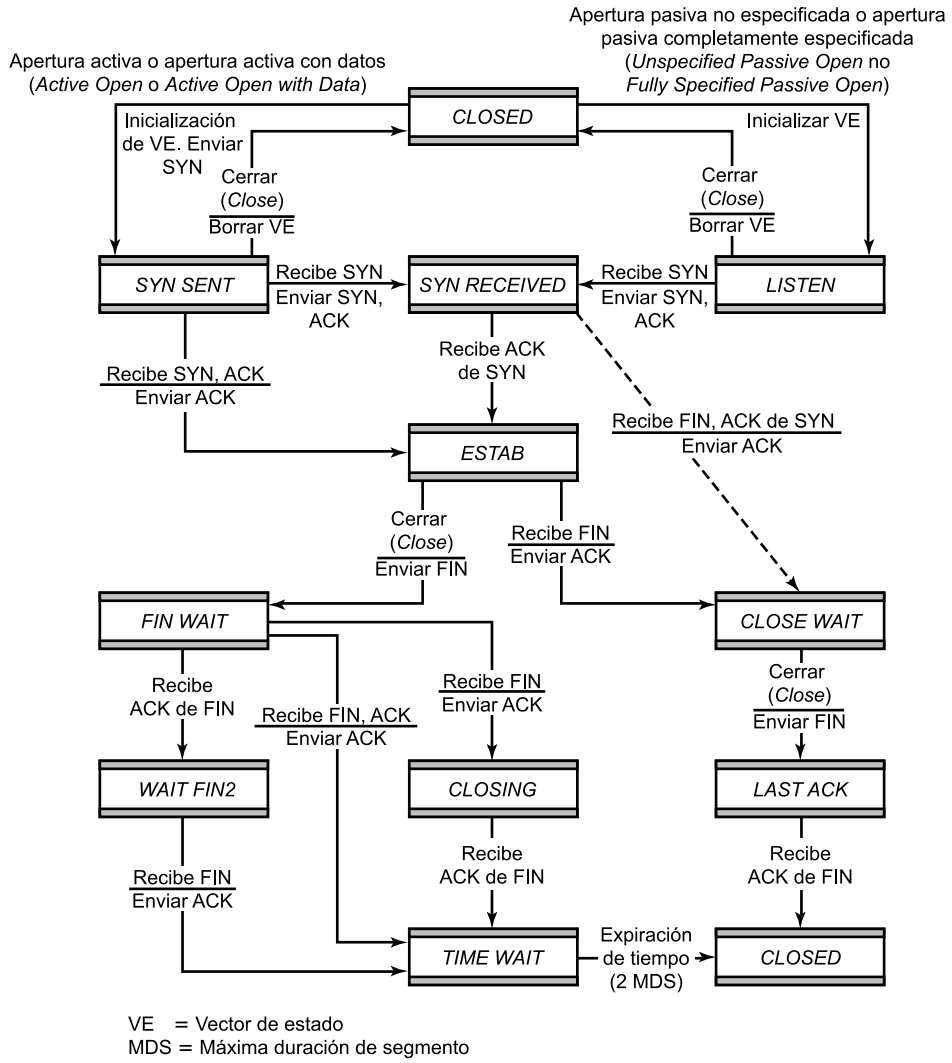


Figura 20.8. Diagrama de estados de la entidad TCP.

estado, la entidad de transporte procede cautelosamente durante el proceso de apertura de la conexión para asegurar que los segmentos SYN enviados por las dos partes han sido confirmados antes de que se declare establecida la conexión. Además del nuevo estado, hay un nuevo segmento de control (RST) para reiniciar el otro lado cuando se detecte un SYN duplicado.

La Figura 20.9 ilustra el funcionamiento típico del diálogo en tres pasos. En la Figura 20.9a, la entidad de transporte A inicia la conexión con un SYN que incluye el número de secuencia de envío, *i*. El valor *i* es el número de secuencia inicial (ISN) y se asocia con el SYN. El primer octeto de datos a transmitir tendrá el número de secuencia $i + 1$. El SYN de respuesta confirma el ISN con ($AN = i + 1$) e incluye su propio ISN. A confirma el SYN/ACK de B en su primer segmento de datos, que comienza con el número de secuencia $i + 1$. La Figura 20.9b muestra una situación en la cual un SYN *i* obsoleto llega a B tras el cierre de la conexión pertinente. B supone que es una solicitud nueva y responde con SYN *j*, $AN = i + 1$. Cuando A recibe este mensaje, se

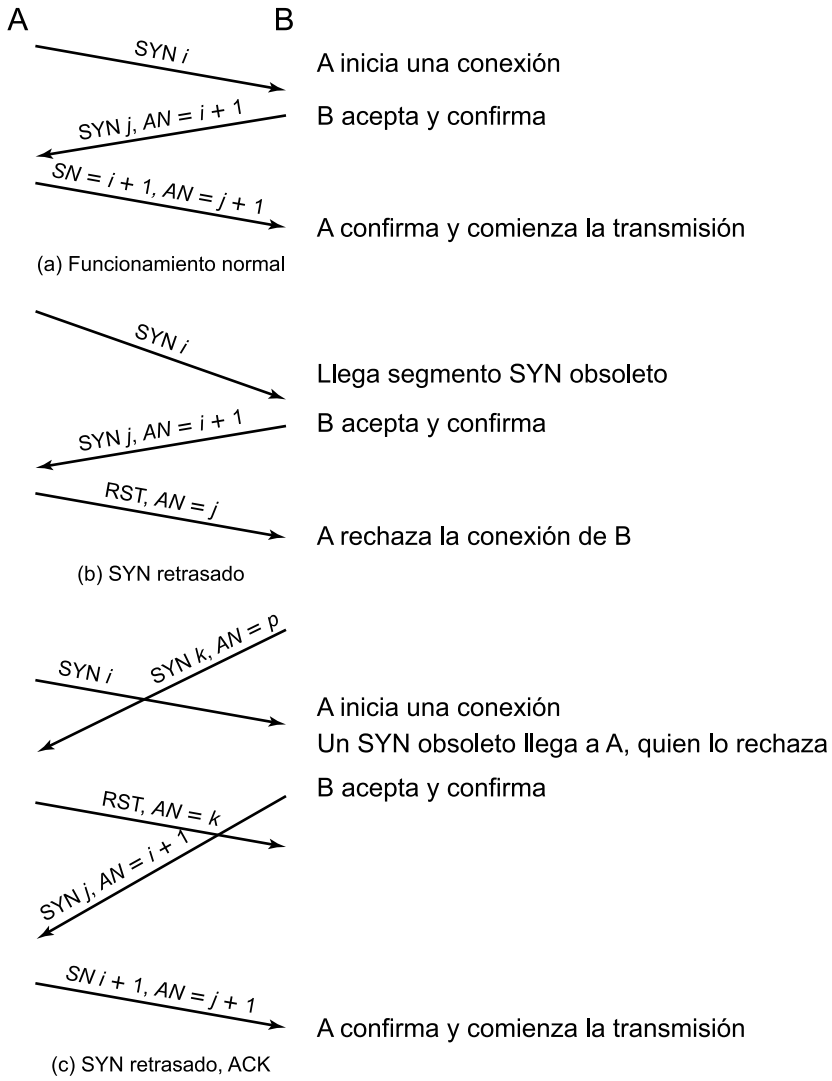


Figura 20.9. Ejemplos de diálogo en tres pasos.

da cuenta de que él no ha solicitado una conexión y, por tanto, envía un RST, $AN = j$. Observe que la porción $AN = j$ del mensaje RST es esencial para que un RST duplicado obsoleto no cancele un establecimiento de conexión legítimo. La Figura 20.9c muestra un caso en que un SYN/ACK antiguo llega en mitad del establecimiento de una nueva conexión. Debido al uso de números de secuencia en las confirmaciones, este evento no causa perjuicio alguno.

Por simplicidad, la parte superior de la Figura 20.8 no incluye transiciones en las que se envíe un segmento RST. La regla básica consiste en enviar un RST si el estado de la conexión no es todavía ESTAB y se recibe un ACK inválido (uno que no referencie a algún segmento que haya sido enviado). El lector debe probar varias combinaciones de eventos para ver que este procedimiento de establecimiento de conexión funciona ante cualquier combinación de segmentos obsoletos o perdidos.

Cierre de la conexión

El diagrama de estados de la Figura 20.3 define el uso de un simple diálogo en dos pasos para el establecimiento de la conexión, que ha resultado ser insatisfactorio para el caso de un servicio de red no fiable. De igual forma, el diálogo en dos pasos definido en ese diagrama para el cierre de la conexión es inadecuado para un servicio de red no fiable. El siguiente escenario podría darse por la llegada de los segmentos en desorden. Una entidad de transporte en el estado *CLOSE WAIT* envía su último segmento de datos, seguido por un segmento FIN, pero el segmento FIN llega al otro extremo antes que el último segmento de datos. La entidad de transporte receptora aceptará ese FIN, cerrará la conexión y perderá el último segmento de datos. Para evitar este problema se puede asociar al segmento FIN un número de secuencia, que puede ser el siguiente número de secuencia tras el último octeto de los datos transmitidos. Con este refinamiento, la entidad de transporte receptora, después de recibir un FIN, antes de cerrar la conexión esperará si es necesario a los datos que lleguen tarde.

Un problema más serio lo constituye la potencial pérdida de segmentos y la posible presencia de segmentos obsoletos. La Figura 20.8 muestra que el procedimiento de cierre adopta una solución similar a la usada para el establecimiento de la conexión. Cada extremo debe explícitamente confirmar el segmento FIN del otro usando un ACK con número de secuencia del FIN a confirmar. Para realizar un cierre ordenado, una entidad de transporte requiere lo siguiente:

- Debe enviar un FIN i y recibir un $AN = i + 1$.
- Debe recibir un FIN j y enviar un $AN = j + 1$.
- Debe esperar un intervalo de tiempo igual a dos veces el máximo tiempo de vida esperado de un segmento.

Recuperación de interrupciones

Cuando el sistema sobre el cual una entidad de transporte se está ejecutando falla y posteriormente se recupera, la información de estado de todas las conexiones activas se pierde. Las conexiones afectadas pasan a estar «semiabiertas» ya que el lado que no se vio afectado por la interrupción no se ha dado cuenta todavía del problema.

El extremo todavía activo de la conexión semiabierta puede cerrar la conexión usando un temporizador de persistencia. Este temporizador mide el tiempo que la máquina de transporte continuará esperando una confirmación (u otra respuesta apropiada) de un segmento transmitido después de que el segmento haya sido retransmitido el máximo número de veces. Cuando el temporizador expira, la entidad de transporte asume que ha fallado la otra entidad o la red intermedia, cierra la conexión e indica al usuario TS que se produjo un cierre anormal.

En el caso en que una entidad de transporte falle y se reinicie rápidamente, la conexión semiabierta se puede terminar más rápidamente mediante el uso del segmento RST. El lado que falla devuelve un RST i por cada segmento i que reciba. Cuando el RST i se recibe en el otro extremo, se debe comprobar su validez basándose en el número de secuencia i, ya que el RST podría ser la respuesta a un segmento obsoleto. Si el reinicio es válido, la entidad de transporte efectúa un cierre anormal.

Estas medidas solucionan la situación en la capa de transporte. La decisión de reabrir la conexión se deja a los usuarios TS. El problema es de sincronización. Cuando ocurrió la interrupción, puede que hubiera uno o más segmentos pendientes en ambos sentidos. El usuario del TS del lado

que no falló sabe cuántos datos ha recibido, pero el otro usuario puede que no, si la información de estado se hubiera perdido. Así, existe el peligro de que algunos datos de usuario se pierdan o se dupliquen.

20.2. TCP

En esta sección examinaremos TCP (RFC 793). En primer lugar, analizaremos el servicio que ofrece al usuario de TS y luego los detalles internos del protocolo.

SERVICIOS TCP

TCP está diseñado para proporcionar una comunicación fiable entre pares de procesos (usuarios TCP) a través de una gran variedad de redes e interconexiones fiables y no fiables. TCP proporciona dos servicios útiles para etiquetar los datos: forzado y urgente:

- **Flujo de datos forzado.** Normalmente, TCP decide cuándo se han acumulado suficientes datos para formar un segmento para su transmisión. El usuario TCP puede requerir que TCP transmita todos los datos pendientes, a los que incluye una etiqueta con un indicador de forzado. En el extremo receptor, TCP entregará los datos al usuario en la misma forma. Un usuario podría requerir esto si se detecta a una interrupción lógica en los datos.
- **Señalización de datos urgentes.** Proporciona un medio para informar al usuario TCP destino que en el flujo de datos que recibe existen datos significativos o «urgentes». Es responsabilidad del usuario destino determinar la acción apropiada.

Como en IP, los servicios proporcionados por TCP se definen en términos de primitivas y parámetros. Los servicios proporcionados por TCP son considerablemente más ricos que los proporcionados por IP y, por tanto, el conjunto de primitivas y parámetros es más complejo. La Tabla 20.2 enumera las primitivas de solicitud de servicio TCP, que son emitidas por un usuario TCP a TCP, y la Tabla 20.3 enumera las primitivas de respuesta de servicio TCP, emitidas por TCP a un usuario TCP local. La Tabla 20.4 proporciona una breve definición de los parámetros involucrados. Las dos órdenes de apertura pasiva indican el deseo del usuario TCP de aceptar una petición de conexión. La apertura activa con datos permite al usuario comenzar transmitiendo datos en la apertura de la conexión.

FORMATO DE LA CABECERA TCP

TCP utiliza un único tipo de unidad de datos de protocolo, llamado segmento TCP. La cabecera se muestra en la Figura 20.10. Ya que una cabecera debe servir para llevar a cabo todos los mecanismos del protocolo, ésta es más bien grande, con una longitud mínima de 20 octetos. Sus campos son los siguientes:

- **Puerto origen (16 bits):** usuario TCP origen.
- **Puerto destino (16 bits):** usuario TCP destino.
- **Número de secuencia (32 bits):** número de secuencia del primer octeto de datos en este segmento, excepto cuando está presente el indicador SYN. Si el indicador SYN está activo, se trata del número de secuencia inicial (ISN) y el primer octeto de datos es el ISN + 1.

nro.de secuencia de octetos.

Tabla 20.2. Primitivas de solicitud de servicio TCP.

Primitiva	Parámetros	Descripción
Apertura pasiva no especificada *(Unspecified Passive Open)*	puerto origen, [tiempo de expiración], [acción tras expiración], [precedencia], [rango de seguridad]	Preparado para intentos de conexión desde cualquier destino remoto, con una seguridad y precedencia especificadas.
Apertura pasiva completamente especificada *(Fully Specified Passive Open)*	puerto origen, puerto destino, dirección-destino, [tiempo de expiración], [acción tras expiración], [precedencia], [rango de seguridad]	Preparado para intentos de conexión desde destino remoto especificado con una seguridad y precedencia especificadas.
Apertura activa *(Active Open)*	puerto origen, puerto destino, dirección destino, [tiempo de expiración], [acción tras expiración], [precedencia], [seguridad]	Solicita una conexión a un destino especificado, con una seguridad y precedencia particulares.
Apertura activa con datos *(Active Open with Data)*	puerto origen, puerto destino, dirección destino, [tiempo de expiración], [acción tras expiración], [precedencia], [seguridad], datos, longitud de datos, indicador FORZADO, indicador URGENTE	Solicita una conexión a un destino especificado, con una seguridad y precedencia particulares, transmitiendo datos con la solicitud.
Enviar *(Send)*	nombre de conexión local, datos, longitud de datos, indicador FORZADO, indicador URGENTE, [tiempo de expiración], [acción tras expiración]	Transfiere datos a través de la conexión indicada.
Asignar *(Allocate)*	nombre de conexión local, longitud de datos	Expide un incremento en la asignación de créditos para la recepción de datos en TCP.
Cerrar *(Close)*	nombre de conexión local	Efectúa un cierre ordenado de la conexión
Abortar *(Abort)*	nombre de conexión local	Efectúa un cierre abrupto de la conexión
Estado *(Status)*	nombre de conexión local	Consulta el estado de la conexión

Nota: los corchetes indican parámetros opcionales.

- **Número de confirmación (32 bits):** contiene el número de secuencia del siguiente octeto que la entidad TCP espera recibir. confirman octetos.
- **Longitud de la cabecera (4 bits):** número de palabras de 32 bits de la cabecera.
- **Reservado (6 bits):** bits reservados para uso futuro. El RFC 3168 usa dos de esos bits para la función de notificación explícita de congestión. Una discusión sobre esta función está fuera de nuestro alcance.
- **Indicadores (6 bits):**

  URG: el campo de puntero urgente es válido.

  ACK: el campo de confirmación es válido.

  PSH: función de forzado.

Tabla 20.3. Primitivas de respuesta del servicio TCP.

Primitiva	Parámetros	Descripción
Identificador de apertura *(Open ID)*	nombre de conexión local, puerto origen, puerto destino*, dirección destino*	Informa al usuario TCP del nombre de conexión asignado a la conexión pendiente solicitada mediante una primitiva de apertura.
Apertura fallida *(Open Failure)*	nombre de conexión local	Informa sobre un fallo de una solicitud de apertura activa.
Apertura correcta *(Open Success)*	nombre de conexión local	Informa sobre la conclusión de una solicitud apertura pendiente.
Entrega *(Deliver)*	nombre de conexión local, datos, longitud de datos, indicador URGENTE	Informa sobre la llegada de datos.
Cierre *(Closing)*	nombre de conexión local	Informa que el usuario TCP remoto ha emitido una orden «cerrar» y que todos los datos enviados por el mismo han sido entregados.
Terminación *(Terminate)*	nombre de conexión local, descripción	Informa que la conexión se ha terminado. Se proporciona una descripción de la razón por la que ha finalizado.
Respuesta de estado *(Status Response)*	nombre de conexión local, puerto origen, puerto destino, dirección origen, dirección destino, ventana de recepción, ventana de envío, cantidad que espera ACK, cantidad por recibir, estado urgente, precedencia, seguridad, tiempo de expiración	Informa del estado actual de la conexión.
Error *(Error)*	nombre de conexión local, descripción	Notifica errores internos o referentes a la solicitud de un servicio.

* = No empleado en la apertura pasiva no especificada.

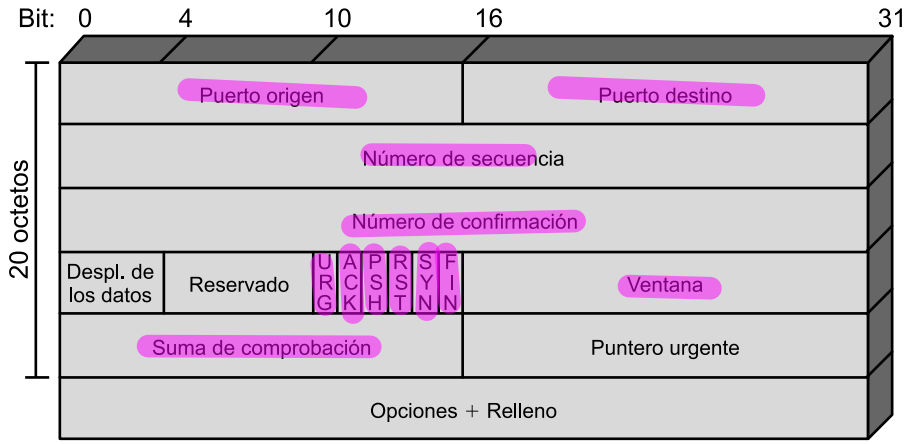


Figura 20.10. Cabecera de TCP.

Tabla 20.4. Parámetros de servicio TCP.

Puerto origen	Usuario TCP local.
Tiempo expiración	El mayor retardo permitido para la entrega de datos antes de efectuar un cierre automático de la conexión o de generar un informe de error. Especificado por el usuario.
Acción tras expiración	Indica qué hacer en caso de expiración de tiempo: terminar la conexión o notificar un error al usuario TCP.
Precedencia	Nivel de precedencia para una conexión. Toma valores de cero (el más bajo) a siete (más alto). Es el mismo parámetro que en IP.
Rango de seguridad	Rangos permitidos en compartimiento, restricciones en la gestión, códigos de control de transmisión y niveles de seguridad.
Puerto destino	Usuario TCP remoto.
Dirección destino	Dirección Internet del computador remoto.
Seguridad	Información de seguridad de una conexión, incluyendo el nivel de seguridad, compartimiento, restricciones en la gestión y códigos de control de transmisión. Son los mismos parámetros que en IP.
Datos	Bloque de datos enviado por el usuario TCP o entregado a un usuario TCP.
Longitud datos	Longitud de los datos enviados o entregados.
Indicador FORZADO (PSH)	Si está activado indica que a los datos asociados se les debe proporcionar el servicio de flujo de datos forzado.
Indicador URGENTE (URG)	Si está activado indica que a los datos asociados se les debe proporcionar el servicio de señalización de datos urgentes.
Nombre de conexión local	Identificador de una conexión definida por un par del tipo (socket local, socket remoto). Lo proporciona TCP.
Descripción	Información suplementaria en una primitiva *Terminate* o *Error.*
Dirección fuente	Dirección Internet del computador local.
Estado de la conexión	Estado de la conexión referenciada (CLOSED, ACTIVE OPEN, PASSIVE OPEN, ESTAB, CLOSING).
Ventana de recepción	Cantidad de datos, en octetos, que la entidad TCP local está dispuesta a recibir.
Ventana de envío	Cantidad de datos, en octetos, que se permite enviar a la entidad TCP remota.
Cantidad que espera ACK	Cantidad de datos previamente transmitidos que esperan confirmación.
Cantidad por recibir	Cantidad de datos, en octetos, almacenados temporalmente en la entidad TCP local, pendientes de ser recibidos por el usuario TCP local.
Estado urgente	Informa al usuario TCP que recibe datos de si hay datos urgentes disponibles o de si todos los datos urgentes, en caso de que hubieran, han sido entregados al usuario.

RST: reiniciar la conexión.

SYN: sincronizar los números de secuencia.

FIN: el emisor no enviará más datos.

- **Ventana (16 bits):** asignación de créditos para el control de flujo, en octetos. Contiene el número de octetos de datos, comenzando con el número de secuencia que se indica en el campo de confirmación que el emisor está dispuesto a aceptar.

- **Suma de comprobación (16 bits):** el complemento a uno de la suma modular complemento a uno de todas las palabras de 16 bits del segmento más una pseudocabecera, descrita más adelante[2].
- **Puntero urgente (16 bits):** este valor, cuando se suma al número de secuencia del segmento, contiene el número de secuencia del último octeto de la secuencia de datos urgentes. Esto permite al receptor conocer la cantidad de datos urgentes que llegan.
- **Opciones (Variable):** un ejemplo lo constituye la opción que especifica la longitud máxima de segmento que será aceptada.

El *número de secuencia* y el *número de confirmación* hacen referencia a octetos en lugar de a segmentos completos. Por ejemplo, si un segmento contiene el número de secuencia 1001 e incluye 600 octetos de datos, el número de secuencia se refiere al primer octeto del campo de datos. El segmento siguiente en orden lógico tendrá el número de secuencia 1601. De esta manera, TCP está lógicamente orientado a flujo: acepta un flujo de octetos del usuario, los agrupa en segmentos según vea apropiado y numera cada octeto del flujo.

se proteje contra errores de IP!

El campo *suma de comprobación* se aplica a todo el segmento más una pseudocabecera incorporada en el momento del cálculo (tanto en la transmisión como en la recepción). La pseudocabecera incluye los siguientes campos de la cabecera IP: dirección red origen y destino, el protocolo y un campo de longitud del segmento. Con la inclusión de la pseudocabecera, TCP se protege ante un reparto erróneo de IP. Es decir, si IP entrega un segmento a una estación errónea, aunque el segmento esté libre de errores de bits, la entidad TCP receptora detectará el error del entrega.

Comparando la cabecera TCP con la interfaz de usuario TCP definida en las Tablas 20.2 y 20.3, el lector podría pensar que faltan algunos campos en la cabecera TCP. Éste es efectivamente el caso. TCP está diseñado específicamente para trabajar con IP. Por tanto, algunos parámetros de usuario se pasan a través de TCP a IP para su inclusión en la cabecera IP. Los más relevantes son:

- Precedencia: un campo de 3 bits.
- Retardo-normal/bajo-retardo.
- Rendimiento-normal/alto-rendimiento.
- Fiabilidad-normal/alta-fiabilidad.
- Seguridad: un campo de 11 bits.

Merece la pena observar que este vínculo TCP/IP significa que la sobrecarga mínima requerida para cada unidad de datos es, en realidad, de 40 octetos.

MECANISMOS TCP

Podemos agrupar los mecanismos de TCP en las categorías de establecimiento de la conexión, transferencia de datos y cierre de la conexión.

Establecimiento de la conexión

El establecimiento de la conexión en TCP siempre utiliza un diálogo en tres pasos. Cuando el indicador SYN está activado, el segmento es esencialmente una solicitud de conexión y funciona tal y

[2] Se puede encontrar una discusión sobre esta suma de comprobación en un documento de apoyo en la página web de este libro.

como se explicó en la Sección 20.1. Para iniciar una conexión, una entidad envía un SYN, $SN = X$, donde X es el número de secuencia inicial. El receptor responde con SYN, $SN = Y$, $AN = X + 1$ mediante la activación de los indicadores SYN y ACK. Observe que la confirmación indica que el receptor está ahora esperando recibir un segmento que comience con el octeto de datos $X + 1$, confirmando el SYN que ocupaba $SN = X$. Finalmente, el que inicia la conexión responde con $AN = Y + 1$. Si los dos extremos emiten SYN cruzados, no se produce ningún problema: ambos lados responden con SYN/ACK (*véase* Figura 20.4).

Una conexión está únivocamente determinada por los sockets (estación, puerto) origen y destino. Así, en cualquier instante de tiempo, sólo puede haber una única conexión TCP entre un único par de puertos. Sin embargo, un puerto dado puede admitir múltiples conexiones, cada una con un puerto diferente.

Transferencia de datos

Flujo de octetos..

Aunque los datos se transmiten en segmentos sobre una conexión de transporte, la transferencia de datos se ve desde un punto de vista lógico como un flujo de octetos. Por tanto, cada octeto es numerado módulo 2^{32}. Cada segmento contiene el número de secuencia del primer octeto del campo de datos. El control de flujo se ejerce utilizando un esquema de asignación de créditos, en el cual el crédito es un número de octetos en lugar de un número de segmentos, tal y como se explicó en la Sección 20.1.

La entidad de transporte almacena temporalmente los datos tanto en la transmisión como en la recepción. TCP normalmente aplica su propio criterio para decidir cuándo construir un segmento para transmitirlo y cuándo entregar los datos recibidos al usuario. El indicador *PUSH* («FORZADO») se usa para obligar a que los datos acumulados sean enviados por el transmisor y entregados al usuario por el receptor. Esto sirve como una función de fin de bloque.

El usuario puede especificar que un bloque de datos es urgente. TCP designará el fin de ese bloque con un puntero de urgente y lo enviará en el flujo de datos ordinario. El usuario receptor es alertado de que se están recibiendo datos urgentes.

Si durante el intercambio de datos llega un segmento que aparentemente no va dirigido a la conexión actual, se envía un segmento con el valor del indicador RST activado. Los SYN duplicados retrasados y las confirmaciones de datos todavía no enviados constituyen ejemplos de esta situación.

Cierre de la conexión

El procedimiento normal de finalización de una conexión es un cierre ordenado. Cada usuario TCP debe emitir una primitiva *Close*. La entidad de transporte establece el bit FIN en el último segmento que envía y que contiene los últimos datos a enviar sobre esa conexión.

Si el usuario emite una primitiva *Abort* («abortar») se produce un cierre abrupto. En este caso, la entidad de transporte abandona todos los intentos de enviar o recibir datos y descarta los datos de sus memorias temporales de transmisión y recepción. Se envía un segmento RST al otro extremo.

Esto va mas allá del alcance de la Materia.

OPCIONES EN LOS CRITERIOS DE IMPLEMENTACIÓN DE TCP

El estándar TCP proporciona una especificación precisa del protocolo que se va a utilizar entre entidades TCP. Sin embargo, ciertos aspectos del protocolo admiten varias opciones de implementación posibles. Aunque dos implementaciones que escojan opciones alternativas pueden interoperar, puede haber consecuencias en el rendimiento. Las áreas de diseño para las que se especifican opciones son las siguientes:

- Política de envío.
- Política de entrega.
- Política de aceptación.
- Política de retransmisión.
- Política de confirmación.

Política de envío

En ausencia de datos marcados con el indicador de forzado y de una ventana de transmisión cerrada (*véase* Figura 20.2a), una entidad TCP emisora es libre de enviar los datos tan pronto como le sea posible, dentro de su asignación actual de crédito. Conforme los datos son emitidos por el usuario se almacenan en la memoria temporal de transmisión. TCP puede construir un segmento por cada lote de datos proporcionado por su usuario o puede esperar a que se acumule una cierta cantidad de datos antes de construir y enviar el segmento. La política concreta dependerá de consideraciones sobre el rendimiento. Si las transmisiones son largas e infrecuentes, hay poca sobrecarga en términos de generación y procesamiento de segmentos. Por otro lado, si las transmisiones son frecuentes y pequeñas, entonces el sistema está proporcionando una respuesta rápida.

Política de entrega

En ausencia del indicador de forzado, una entidad TCP receptora es libre de entregar los datos al usuario tan pronto como le sea posible. Puede entregar los datos conforme se reciben los segmentos en orden, o puede almacenar los datos de varios segmentos en las memorias temporales de recepción antes de efectuar la entrega. La política concreta dependerá de consideraciones sobre el rendimiento. Si las entregas son infrecuentes y voluminosas, el usuario no recibe los datos tan pronto como puede ser deseable. Por otro lado, si las entregas son frecuentes y pequeñas, puede haber un procesamiento innecesario en TCP y en el software del usuario, así como un número innecesario de interrupciones del sistema operativo.

Política de aceptación

Cuando todos los segmentos de datos llegan en orden sobre una conexión TCP, TCP coloca los datos en una memoria temporal de recepción para entregarlos al usuario. Es posible, sin embargo, que los segmentos no lleguen en secuencia. En este caso, la entidad TCP receptora tiene dos opciones:

- Aceptación **ordenada:** acepta sólo segmentos que llegan en orden. Todos los segmentos que no lleguen en secuencia se descartan.
- Aceptación **en ventana:** acepta todos los segmentos que estén dentro de la ventana de recepción (*véase* Figura 20.2b).

La política de aceptación ordenada da lugar a una implementación sencilla, pero sitúa una carga adicional sobre el servicio de red, ya que la entidad TCP que envía debe retransmitir, tras la expiración de los temporizadores correspondientes, aquellos segmentos que se recibieron correctamente pero que fueron descartados por su recepción desordenada. Además, si se pierde un único segmento en el camino, entonces deben retransmitirse todos los segmentos siguientes una vez que expire en el TCP emisor el temporizador del segmento perdido.

La política de aceptación en ventana puede reducir las transmisiones, pero requiere una comprobación de aceptación más compleja y un esquema de almacenamiento de datos más sofisticado para almacenar y llevar el registro de los datos desordenados aceptados.

Política de retransmisión

TCP mantiene una cola de los segmentos que han sido enviados pero que todavía no han sido confirmados. La especificación de TCP establece que TCP retransmite un segmento si no recibe una confirmación dentro de un tiempo determinado. Una implementación de TCP puede emplear una de estas tres estrategias de retransmisión:

- **Sólo el primero:** mantiene un temporizador de retransmisión para toda la cola. Si se recibe una confirmación, elimina de la cola el segmento o segmentos correspondientes y reinicia el temporizador. Si el temporizador expira, retransmite el primer segmento de la cola y reinicia el temporizador.
- **Por lotes:** mantiene un temporizador de retransmisión para toda la cola. Si se recibe una confirmación, elimina de la cola el segmento o segmentos correspondientes y reinicia el temporizador. Si el temporizador expira, retransmite todos los segmentos de la cola y reinicia el temporizador.
- **Individual:** mantiene un temporizador de retransmisión por cada segmento en cola. Si se recibe una confirmación, elimina de la cola el segmento o segmentos apropiados y destruye el temporizador o los temporizadores asociados. Si algún temporizador expira, retransmite el segmento correspondiente y reinicia temporizador.

La política de sólo el primero es eficiente en términos de tráfico generado, ya que solamente se retransmiten los segmentos perdidos (o segmentos cuyo ACK se perdió). Ya que el temporizador para el segundo segmento en la cola no se establece hasta que el primer segmento se confirma, pueden producirse retardos considerables. La política de retransmisión individual soluciona este problema a expensas de una implementación más compleja. La política de retransmisión por lotes también reduce la posibilidad de largos retardos, pero puede producir retransmisiones innecesarias. La efectividad real de una política de retransmisión depende en parte de la política de aceptación del receptor. Si el receptor está empleando una política de aceptación ordenada, entonces descartará los segmentos recibidos tras un segmento perdido. Esta política encaja mejor con una retransmisión por lotes. Si el receptor emplea una política de aceptación en ventana, entonces es mejor la política de retransmisión del primero solamente o de retransmisión individual. Por supuesto, en una red mixta de computadores, se pueden usar ambas políticas de aceptación.

Política de confirmación

Cuando llega un segmento en orden, la entidad TCP receptora tiene dos opciones en cuanto a la generación de las confirmaciones:

- **Inmediata:** cuando los datos se aceptan, se transmite inmediatamente un segmento vacío (sin datos) que contiene el número de confirmación apropiado.
- **Acumulada:** cuando se aceptan los datos, se registra la necesidad de una confirmación, pero espera un segmento de datos de salida con datos e incorpora la confirmación. Para evitar grandes retardos, establece un temporizador de ventana (*véase* Tabla 20.1). Si el temporizador expira antes de que se envíe una confirmación, transmite un segmento vacío que contiene el número de confirmación apropiado.

La política de confirmación inmediata es sencilla y mantiene a la entidad TCP remota completamente informada, lo que evita retransmisiones innecesarias. Sin embargo, esta política da lugar a retransmisiones de segmentos extra, a saber, segmentos vacíos usados sólo para confirmar. Además, esta política puede ocasionar una mayor carga en la red. Considere que una entidad TCP recibe un segmento e inmediatamente envía un ACK. Entonces, los datos se pasan a la aplicación, lo cual expande la ventana de recepción, emitiendo otro segmento TCP vacío para proporcionar crédito adicional a la entidad TCP emisora.

A causa de la potencial sobrecarga causada por la política de confirmación inmediata, normalmente se emplea la política de confirmación acumulada. Reconozcamos, sin embargo, que el uso de esta política requiere más procesamiento en el extremo receptor y complica en la entidad TCP emisora la tarea de estimar el retardo de ida y vuelta.

20.3. CONTROL DE CONGESTIÓN DE TCP

El mecanismo de control de flujo basado en créditos de TCP se diseñó para permitir que el destino restrinja el flujo de segmentos de una fuente y evitar así la saturación de la memoria temporal del destino. Este mismo mecanismo de control de flujo se utiliza ahora de varias formas ingeniosas para proporcionar control de congestión sobre Internet entre la fuente y el destino. La congestión, como ya se ha visto varias veces en este libro, tiene dos efectos principales. En primer lugar, cuando la congestión empieza a producirse, el tiempo de transmisión a través de la red o interconexión de redes aumenta. En segundo lugar, conforme la congestión se hace más severa, la red o los nodos de la interconexión descartan paquetes. El mecanismo de control de flujo de TCP se puede utilizar para identificar el comienzo de la congestión (identificando el incremento de los tiempos de retardo y de los segmentos descartados) y reaccionar mediante la reducción del flujo de datos. Si muchas de las entidades TCP que operan a lo largo de una red practican este tipo de control, la congestión de la red se puede aliviar.

Desde la publicación del RFC 793, se han implementado varias técnicas que pretenden mejorar las características de control de congestión de TCP. Ninguna de estas técnicas extienden o violan el estándar TCP original. Más bien representan criterios de implementación que están dentro del ámbito de la especificación de TCP. Muchas de estas técnicas son de uso obligatorio en TCP, como se refleja en el RFC 1122 («Requisitos para las estaciones en Internet»), mientras otras se especifican en el RFC 2581. Las técnicas se pueden agrupar, en un sentido amplio, en dos categorías: gestión de temporizadores de retransmisión y gestión de la ventana. En esta sección se examinan algunas de las técnicas más importantes y más utilizadas.

Esto va mas allá del alcance de la Materia.

GESTIÓN DE TEMPORIZADORES DE RETRANSMISIÓN

Conforme cambian las condiciones de red o interconexión de redes, un temporizador de retransmisión estático puede expirar demasiado tarde o demasiado pronto. De acuerdo a esto, virtualmente

todas las implementaciones de TCP intentan estimar el retardo de ida y vuelta actual mediante la observación del patrón del retardo de los segmentos más recientes, para establecer el temporizador a un valor un poco mayor que el retardo de ida y vuelta estimado.

Promediado simple

Una posible opción consistiría en tomar simplemente la media de los tiempos de ida y vuelta observados sobre un determinado número de segmentos. Si la media predice con precisión los retardos de ida y vuelta futuros, entonces el temporizador de retransmisión realizará su función apropiadamente. El método del promediado simple se puede expresar como:

$$\text{ARTT}(K+1) = \frac{1}{K+1}\sum_{i=1}^{K+1}\text{RTT}(i) \tag{20.1}$$

donde RTT(i) es el tiempo de ida y vuelta observado para el segmento i-ésimo transmitido y ARTT(K) es el tiempo de ida y vuelta medio de los K primeros segmentos.

Esta expresión se puede reescribir como:

$$\text{ARTT}(K+1) = \frac{K}{K+1}\text{ARTT}(K) + \frac{1}{K+1}\text{RTT}(K+1) \tag{20.2}$$

Con esta formulación, no es necesario recalcular la sumatoria completa cada vez.

Promediado exponencial

Observe que a cada término de la sumatoria se le da el mismo peso. Es decir, cada término se multiplica por la misma constante $1/(K+1)$. Normalmente, nos interesaría dar mayor peso a los retardos más recientes, ya que es más probable que reflejen el comportamiento futuro. Una técnica común para predecir los valores siguientes a partir de una serie temporal de valores pasados, y que es el especificado en el RFC 793, es el promediado exponencial:

$$\text{SRTT}(K+1) = \alpha \times \text{SRTT}(K) + (1-\alpha) \times \text{RTT}(K+1) \tag{20.3}$$

donde SRTT(K) se denomina estimación del tiempo de ida y vuelta suavizado y donde se define SRTT(0) = 0. Compárese esta ecuación con la Ecuación (20.2). Mediante el uso de un valor constante de $\alpha(0 < \alpha < 1)$, independientemente del número de observaciones pasadas, tenemos una circunstancia en la cual se consideran todas los valores pasados, pero con menor peso las más distantes. Para ver esto más claramente, consideremos el desarrollo de la Ecuación (20.3):

$$\begin{aligned}\text{SRTT}(K+1) = {} & (1-\alpha)\text{RTT}(K+1) + \alpha(1-\alpha)\text{RTT}(K) + \\ & + \alpha^2(1-\alpha)\text{RTT}(K-1) + \cdots + \alpha^K(1-\alpha)\text{RTT}(1)\end{aligned}$$

Ya que α y $(1-\alpha)$ son menores que uno, cada término sucesivo en la ecuación precedente es menor. Por ejemplo, para $\alpha = 0{,}8$, el desarrollo es el siguiente:

$$\text{SRTT}(K+1) = (0{,}2)\text{RTT}(K+1) + (0{,}16)\text{RTT}(K) + (0{,}218)\text{RTT}(K-1) + \cdots$$

Cuanto más antigua es la observación, menos cuenta en el promedio.

Cuanto más pequeño es el valor de α, mayor es el peso dado a las observaciones más recientes. Para $\alpha = 0{,}5$, prácticamente todo el peso se le da a las cuatro o cinco observaciones más recientes, mientras que si $\alpha = 0{,}875$, el promediado se extiende alrededor de las diez observaciones más recientes. La ventaja de utilizar valores pequeños de α es que el promedio reflejará rápidamente un cambio rápido en las cantidades observadas. La desventaja radica en que, si se produce una subida breve en las cantidades observadas y después se vuelve a algún valor relativamente constante, el uso de valores pequeños de α producirá cambios fluctuantes en el promedio.

La Figura 20.11 compara el promediado simple con el promediado exponencial (para dos valores diferentes de α). En la parte (a) de la figura, el valor observado empieza con 1, crece gradualmente hasta el valor 10 y luego permanece ahí. En la parte (b) de la figura, el valor observado empieza en 20, decrece gradualmente hasta 10 y luego permanece ahí. Observe que el promediado exponencial sigue los cambios en el comportamiento del proceso más rápidamente que el promediado simple y que el valor más pequeño de α tiene como resultado reacciones más rápidas frente al cambio del valor observado.

La Ecuación (20.3) se utiliza en el RFC 793 para estimar el tiempo actual de ida y vuelta. Como se mencionó, el valor del temporizador debe establecerse a un valor algo mayor que el tiempo estimado de ida y vuelta. Una posibilidad consiste en utilizar un valor constante:

$$\text{RTO}(K + 1) = \text{SRTT}(K + 1) + \Delta$$

donde RTO es el temporizador de retransmisión (también conocido como el valor de expiración de retransmisión) y Δ es una constante. La desventaja de esto es que Δ no es proporcional a SRTT. Para valores altos de SRTT, Δ es relativamente pequeño y las fluctuaciones en el valor real de RTT producirán retransmisiones innecesarias. Para valores bajos de SRTT, Δ es relativamente grande y produce retardos innecesarios en la retransmisión de segmentos perdidos. De acuerdo a esto, el RFC 793 especifica la utilización de un temporizador cuyo valor es proporcional a SRTT, dentro de unos límites:

$$\text{RTO}(K + 1) = \text{MIN}(\text{UBOUND}, \text{MAX}(\text{LBOUND}, \beta \times \text{SRTT}(K + 1))) \qquad (20.4)$$

donde UBOUND y LBOUND son unos límites superior e inferior fijos preseleccionados para el valor del temporizador y β es una constante. El RFC 793 no recomienda valores específicos, pero da como valores de ejemplo los siguientes: α entre 0,8 y 0,9 y β entre 1,3 y 2,0.

Estimación de la varianza del RTT (algoritmo de Jacobson)

La técnica especificada en el estándar TCP, y descrita en las Ecuaciones (20.3) y (20.4), habilita a una entidad TCP a adaptarse a los cambios del tiempo de ida y vuelta. Sin embargo, no trata bien una situación en la cual el tiempo de ida y vuelta exhiba una varianza relativamente elevada. [ZHAN86] señala tres fuentes de esta varianza:

1. Si la velocidad de transferencia de datos en una conexión TCP es relativamente baja, entonces el retardo de transmisión será relativamente alto comparado con el tiempo de propagación y la varianza en el RTT debida a la varianza en el tamaño de los datagramas IP será significativa. De esta forma, el estimador de SRTT está fuertemente influenciado por las características propias de los datos y no de la red.
2. La carga de tráfico y las condiciones en Internet pueden cambiar abruptamente debido al tráfico de otras fuentes, causando cambios bruscos en el RTT.

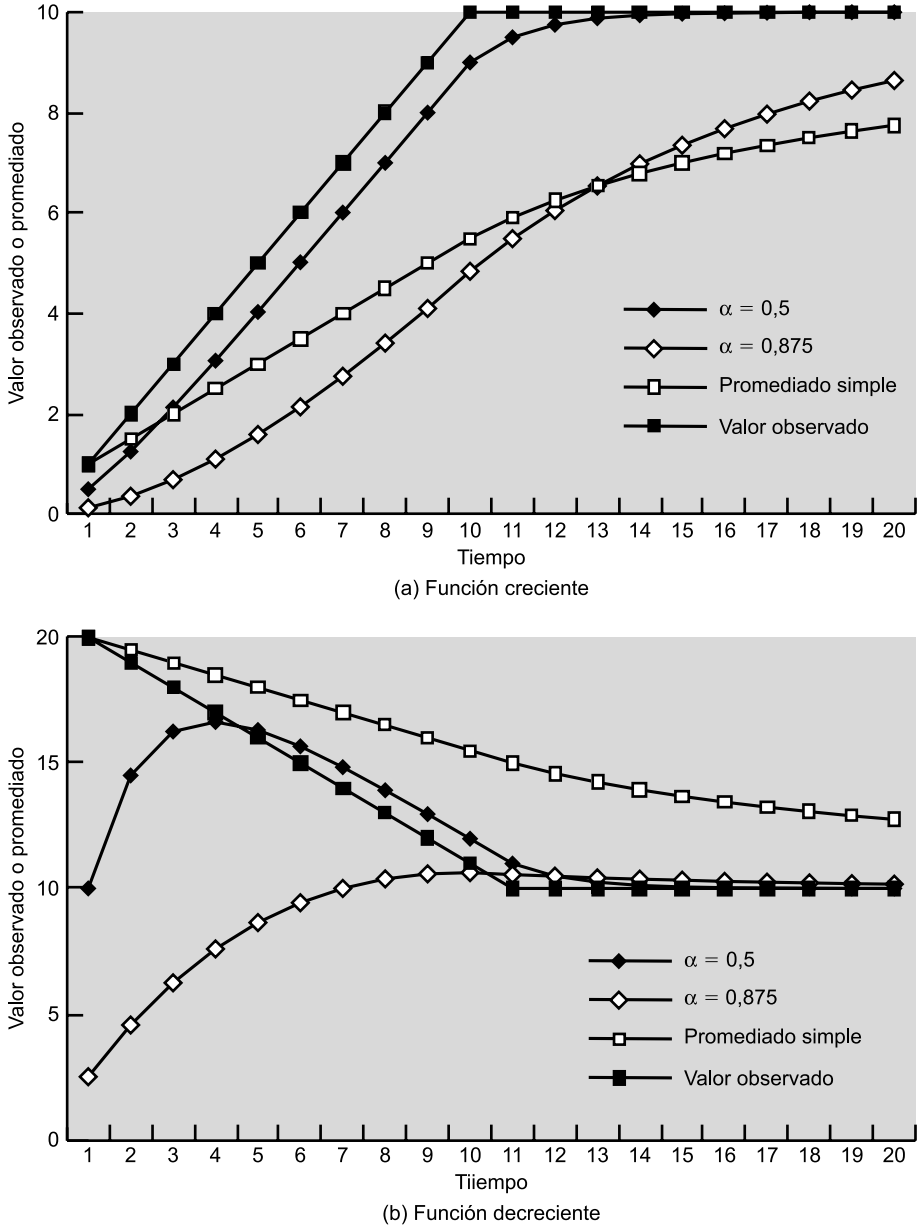


Figura 20.11. Uso del promediado exponencial.

3. La entidad TCP par puede no confirmar cada segmento inmediatamente debido a su propio retardo de procesamiento o debido a que ejerce su privilegio de utilizar confirmaciones acumuladas.

La especificación original de TCP intenta considerar esta variabilidad multiplicando la estimación de RTT por un factor constante, como se muestra en la Ecuación (20.4). En un entorno estable, con una varianza baja de RTT, esta formulación tiene como resultado un valor innecesariamente alto

de RTO y en un entorno inestable un valor de $\beta = 2$ podría ser inadecuado para proteger contra retransmisiones innecesarias.

Una propuesta más efectiva consiste en estimar la variabilidad en los valores de RTT y utilizarla como entrada en el cálculo de una RTO. Una medida de variabilidad fácil de estimar es la desviación media, definida como

$$\text{MDEV}(X) = \text{E}[|X - \text{E}[X]|]$$

donde E[X] es el valor esperado de X.

Como se hizo con la estimación de RTT, se puede utilizar un promediado simple para estimar MDEV:

$$\begin{aligned}
\text{AERR}(K+1) &= \text{RTT}(K+1) - \text{ARTT}(K) \\
\text{ADEV}(K+1) &= \frac{1}{K+1}\sum_{i=1}^{K+1}|\text{AERR}(i)| \\
&= \frac{K}{K+1}\,\text{ADEV}(K) + \frac{1}{K+1}\,|\text{AERR}(K+1)|
\end{aligned}$$

donde ARTT(K) es la media simple definida en la Ecuación (20.1) y AERR(K) es la desviación media medida en el instante K.

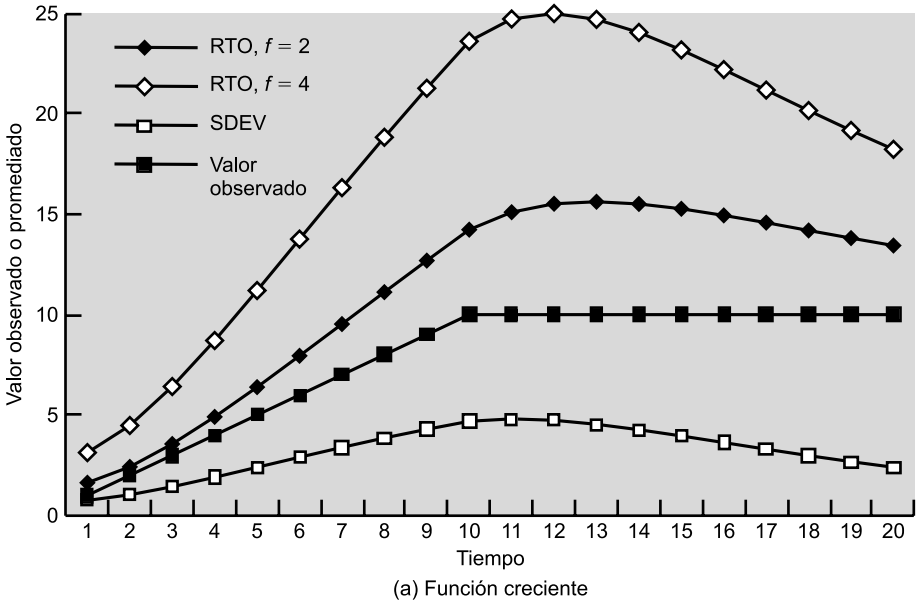
Como con la definición de ARRT, cada término de la sumatoria de ADEV tiene el mismo peso. Es decir, cada término se multiplica por la misma constante $1/(K + 1)$. De nuevo, querríamos dar un peso mayor a las medidas más recientes, ya que es más probable que reflejen el comportamiento futuro. Jacobson, que propuso la utilización de una estimación dinámica de la variabilidad en la estimación de RTT [JACO88], sugiere utilizar la misma técnica de suavizado exponencial que en el cálculo de SRTT. El algoritmo completo propuesto por Jacobson se puede expresar como sigue:

$$\begin{aligned}
\text{SRTT}(K+1) &= (1-g) \times \text{SRTT}(K) + g \times \text{RTT}(K+1) \\
\text{SERR}(K+1) &= \text{RTT}(K+1) - \text{SRTT}(K) \\
\text{SEV}(K+1) &= (1-h) \times \text{SDEV}(K) + h \times |\text{SERR}(K+1)| \\
\text{RTO}(K+1) &= \text{SRTT}(K+1) + f \times \text{SDEV}(K+1)
\end{aligned} \tag{20.5}$$

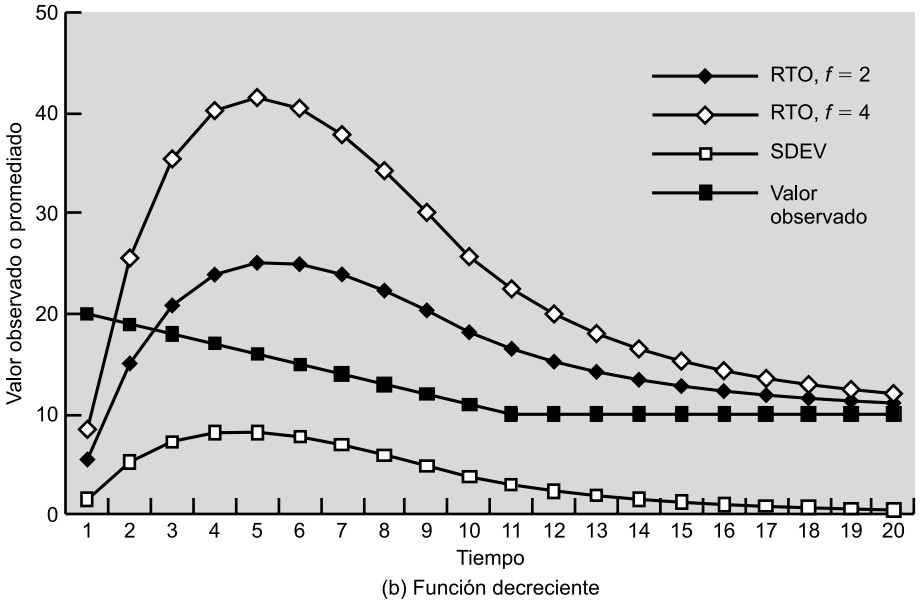
Como en la definición del RFC 793 (Ecuación (20.3)), SRTT es una estimación exponencial suavizada de RTT, con $(1 - g)$ equivalente a α. Ahora, sin embargo, en lugar de multiplicar la estimación SRTT por una constante (Ecuación (20.4)), se suma a SRTT un múltiplo de la desviación media estimada para formar el temporizador de retransmisión. Basándose en sus experimentos de temporización, Jacobson propuso en su artículo original [JACO88] los siguientes valores para las constantes:

$$\begin{aligned}
g &= 1/8 = 0{,}125 \\
h &= 1/4 = 0{,}25 \\
f &= 2
\end{aligned}$$

Después de investigaciones posteriores [JACO90], recomendó cambiar el valor de f a 4, siendo éste el valor estándar utilizado en las implementaciones actuales.



(a) Función creciente



(b) Función decreciente

Figura 20.12. Cálculo del RTO de Jacobson.

La Figura 20.12 muestra el uso de la Ecuación (20.5) sobre el mismo conjunto de datos que se utilizó en la Figura 20.11. Una vez que el tiempo de llegada se estabiliza, la estimación de la variación SDEV decae. Los valores de RTO para ambos casos ($f = 2$ y $f = 4$) son bastante conservadores mientras que RTT esté cambiando, pero comienzan a converger a RTT cuando se estabiliza.

La experiencia ha demostrado que el algoritmo de Jacobson puede mejorar significativamente el rendimiento de TCP. Sin embargo, no basta por sí solo. Se deben considerar otros dos factores:

1. ¿Qué valor de RTO se debe utilizar para un segmento retransmitido? Para este caso se utiliza el algoritmo de decaimiento exponencial de RTO.
2. ¿Qué muestras se deben utilizar como entrada al algoritmo de Jacobson? El algoritmo de Karn determina qué muestras se han de utilizar.

Decaimiento exponencial de RTO

Cuando expira un temporizador en el emisor TCP, éste debe retransmitir el segmento correspondiente. El RFC 793 supone que se va a utilizar el mismo RTO para este segmento retransmitido. Sin embargo, ya que el que expire el temporizador se debe probablemente a la congestión de la red, manifestada como el descarte de un paquete o un largo retardo en el tiempo de ida y vuelta, mantener el mismo valor de RTO no es aconsejable.

Considere el siguiente escenario. Existen varias conexiones TCP activas de varias fuentes enviando tráfico a una interconexión de redes. Aparece la congestión en una región, de forma que los segmentos en muchas de esas conexiones se pierden o retrasan por encima del tiempo RTO de las conexiones. Por tanto, casi al mismo tiempo, muchos segmentos serán retransmitidos por la interconexión de redes, manteniendo o incluso incrementando la congestión. Todas las fuentes esperan un tiempo RTO (local a cada conexión) y retransmiten de nuevo. Este patrón de comportamiento podría causar una condición de congestión continua.

Una política más sensible dicta que una fuente TCP incremente su RTO cada vez que se retransmita el mismo segmento. Esto se conoce como proceso de *decaimiento*. En el escenario del párrafo anterior, después de la primera retransmisión de un segmento de cada conexión afectada, todas las fuentes TCP esperarán un tiempo mayor antes de intentar la segunda retransmisión. Esto puede darle tiempo a la interconexión de redes para despejar la congestión actual. Si se necesita una segunda retransmisión, cada fuente TCP esperará un tiempo todavía mayor antes de que expire el temporizador correspondiente para una tercera retransmisión, dando a la interconexión un periodo aún mayor para recuperarse.

Una técnica simple para implementar el decaimiento de RTO consiste en multiplicar el RTO de un segmento por un valor constante para cada retransmisión:

$$\text{RTO} = q \times \text{RTO} \tag{20.6}$$

La Ecuación (20.6) hace que el RTO crezca exponencialmente con cada retransmisión. El valor de q más comúnmente utilizado es 2. Con este valor, la técnica se conoce como *decaimiento exponencial binario*. Ésta es la misma técnica que se utiliza en el protocolo CSMA/CD de Ethernet (*véase* Capítulo 16).

Algoritmo de Karn

Si no se retransmite ningún segmento, el proceso de muestreo para el algoritmo de Jacobson es directo. El RTT de cada segmento se puede incluir en los cálculos. Sin embargo, suponga que expira el temporizador de un segmento, por lo que debe retransmitirse. Si se recibe una confirmación posterior, existen dos posibilidades:

1. Se trata del ACK de la primera transmisión del segmento. En este caso, el RTT es simplemente mayor que el esperado, pero es un reflejo preciso de las condiciones de la red.
2. Se trata del ACK de la segunda transmisión.

La entidad emisora TCP no puede distinguir entre estos dos casos. Si se trata del segundo caso y la entidad TCP mide simplemente el RTT desde la primera transmisión hasta la recepción del ACK, el tiempo medido será demasiado alto. El RTT medido será del orden del RTT actual más el RTO. Utilizar este falso RTT con el algoritmo de Jacobson producirá un valor alto innecesario de SRTT y, por tanto, de RTO. Es más, este efecto se propaga varias iteraciones, ya que el valor de SRTT de una iteración es un valor de entrada en la siguiente iteración.

Un enfoque incluso peor consistiría en medir el RTT de la segunda transmisión hasta la recepción del ACK. Si éste es en efecto el ACK de la primera transmisión, entonces el RTT medido sería demasiado pequeño, produciendo un valor demasiado bajo de SRTT y de RTO. Esto es probable que tenga un efecto de realimentación positiva, causando retransmisiones adicionales y medidas falsas.

El algoritmo de Karn [KARN91] resuelve este problema mediante las siguientes reglas:

1. No utilizar el RTT medido para un segmento retransmitido para actualizar SRTT y SDEV [Ecuación (20.5)].
2. Calcular el RTO de decaimiento utilizando la Ecuación (20.6) cuando se produzca una retransmisión.
3. Utilizar el valor del RTO de decaimiento para segmentos sucesivos hasta que llegue una confirmación para un segmento que no se haya retransmitido.

Cuando se recibe una confirmación para un segmento que no se ha retransmitido, se activa de nuevo el algoritmo de Jacobson para calcular valores futuros de RTO.

GESTIÓN DE VENTANA

Además de las técnicas para mejorar la efectividad del temporizador de retransmisión, se han examinado varias aproximaciones para gestionar la ventana de emisión. El tamaño de la ventana de emisión de TCP puede tener un efecto decisivo para que TCP pueda ser utilizado eficientemente sin causar congestión. Discutiremos dos técnicas que se encuentran virtualmente en todas las implementaciones recientes de TCP: el arranque lento y el ajuste dinámico de la ventana en caso de congestión[3].

Arranque lento

Cuanto mayor es la ventana de emisión de TCP, más segmentos puede enviar la fuente TCP antes de que deba esperar una confirmación. Esto puede crear un problema cuando se establece por primera vez una conexión TCP, ya que la entidad TCP es libre de vaciar la ventana de datos completa en la red.

Una estrategia que se podría seguir consiste en que el emisor TCP empezara a enviar con una ventana relativamente grande pero no con su máximo tamaño, esperando aproximarse al tamaño máximo que sería proporcionado por la conexión finalmente. Este esquema es arriesgado, ya que el emisor podría inundar la interconexión de redes con muchos segmentos antes de darse cuenta por los temporizadores de que el flujo era excesivo. En lugar de eso, se necesita algún medio para

[3] Estos algoritmos fueron desarrollados por Van Jacobson [JACO88] y son descritos además en el RFC 2581. Van Jacobson utiliza unidades de segmentos TCP, mientras que en el RFC 2581 se trabaja principalmente en unidades de octetos de datos TCP, con alguna referencia a cálculos en unidades de segmentos. Nosotros seguimos el desarrollo de [JACO88].

expandir gradualmente la ventana hasta que se reciban las confirmaciones. Éste es el propósito del mecanismo de arranque lento.

En el arranque lento, la transmisión TCP está restringida por la siguiente relación:

$$awnd = \text{MIN}[crédito, cwnd] \tag{20.7}$$

donde

awnd = ventana permitida, en segmentos. Éste es el número de segmentos que TCP tiene actualmente permitido enviar sin recibir confirmaciones adicionales.

cwnd = ventana de congestión, en segmentos. Ventana utilizada por TCP durante el inicio y para reducir el flujo durante los periodos de congestión.

crédito = la cantidad de créditos concedidos y no utilizados en la confirmación más reciente, en segmentos. Cuando se recibe una confirmación, este valor se calcula como *ventana*/*tamaño_de_segmento*, donde *ventana* es un campo del segmento TCP recibido (la cantidad de datos que está dispuesta a aceptar la entidad TCP par).

La entidad TCP inicializa *cwnd* = 1 cuando se abre una conexión. Es decir, a TCP sólo se le permite enviar un segmento y después debe esperar una confirmación antes de enviar un segundo segmento. Cada vez que reciba una confirmación para datos nuevos, se incrementa el valor de *cwnd* en una unidad, hasta algún valor máximo.

En efecto, el mecanismo de arranque lento sondea la interconexión de redes para asegurarse de que la entidad TCP no esté enviando demasiados segmentos en un entorno ya congestionado. Conforme van llegando las confirmaciones, a TCP se le permite abrir su ventana hasta que el flujo se controle mediante las confirmaciones que se reciben, en lugar de mediante *cwnd*.

El término arranque lento es un nombre poco apropiado, ya que *cwnd* crece en realidad exponencialmente. Cuando llega el primer ACK, TCP abre *cwnd* hasta 2 y puede enviar dos segmentos. Cuando estos dos segmentos se confirman, TCP puede desplazar la ventana 1 segmento e incrementar *cwnd* en uno por cada ACK que llega. Por tanto, en este punto TCP puede enviar cuatro segmentos. Cuando se confirmen estos cuatro segmentos, TCP podrá enviar ocho segmentos.

Ajuste dinámico de la ventana en caso de congestión

Se ha comprobado que el algoritmo de arranque lento funciona de forma efectiva para inicializar una conexión. Permite a TCP determinar rápidamente un tamaño de ventana razonable para la conexión. ¿No sería útil la misma técnica cuando haya un incremento de la congestión? En particular, suponga que una entidad TCP inicia una conexión y ejecuta el procedimiento de arranque lento. En algún punto, antes o después de que *cwnd* alcance el tamaño de créditos asignados por el otro extremo, se pierde un segmento (expira un temporizador). Esto es una señal de que se está produciendo congestión. Pero no está claro cómo de severa es la congestión. Por tanto, un procedimiento prudente sería inicializar *cwnd* a 1 y comenzar el procedimiento de arranque lento de nuevo.

Éste parece un procedimiento conservador razonable, pero en realidad no es lo bastante conservador. Jacobson [JACO88] señala que «es fácil llevar una red a la saturación, pero es difícil para la red recuperarse». En otras palabras, una vez que la congestión se produce, puede transcurrir un largo intervalo de tiempo hasta que ésta desaparezca[4]. De esta forma, el crecimiento exponencial

[4] Kleinrock se refiere a este fenómeno como el efecto de larga cola en periodos punta. *Véanse* las Secciones 2.7 y 2.10 de [KLEI76] para un estudio detallado.

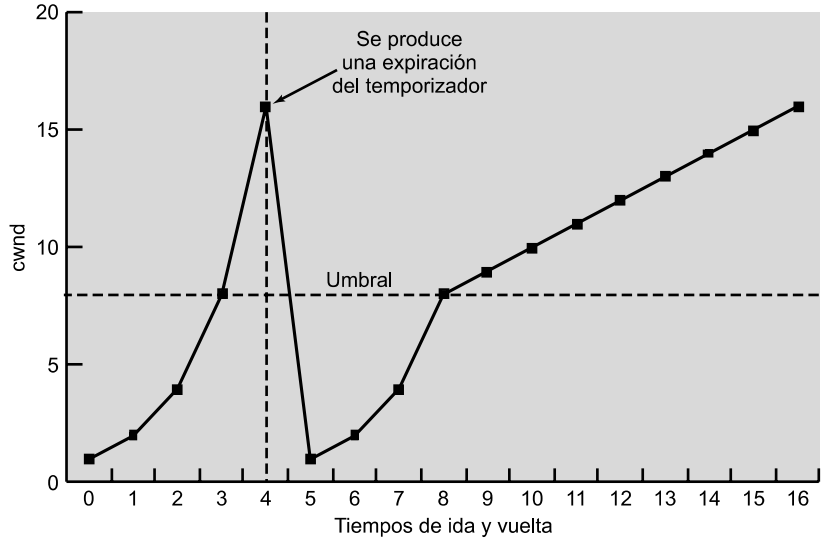


Figura 20.13. Ilustración del arranque lento y la supresión de congestión.

de *cwnd* bajo el arranque lento puede ser demasiado agresivo y empeorar la congestión. En lugar de esto, Jacobson propuso el uso del arranque lento para comenzar, seguido de un crecimiento lineal de *cwnd*. Cuando expira un temporizador las reglas son las siguientes:

1. Establecer un umbral de arranque lento igual a la mitad de la ventana de congestión actual. Es decir, hacer *ssthresh* = *cwnd*/2.
2. Establecer *cwnd* = 1 y ejecutar el procedimiento de arranque lento hasta que *cwnd* = *ssthresh*. En esta fase, *cwnd* se incrementa en 1 por cada ACK recibido.
3. Para *cwnd* $\geqslant$ *ssthresh*, incrementar *cwnd* en uno por cada tiempo de ida y vuelta.

La Figura 20.13 muestra este comportamiento. Observe que le lleva 11 veces el tiempo de ida y vuelta recuperar el nivel *cwnd* que inicialmente se consiguió con 4 veces el tiempo de ida y vuelta.

20.4. UDP

Además de TCP, existe otro protocolo en la capa de transporte que se usa comúnmente como parte del conjunto de protocolos TCP/IP: el protocolo de datagrama de usuario (UDP, *User Datagram Protocol*), especificado en el RFC 768. UDP proporciona un servicio no orientado a conexión para los procedimientos de la capa de aplicación. Así, UDP es básicamente un servicio no fiable: no se garantizan la entrega y la protección contra duplicados. En contrapartida, se reduce la sobrecarga del protocolo, lo que puede ser adecuado en muchos casos. Un ejemplo de uso de UDP se tiene en el contexto de la gestión de red, como se describe en el Capítulo 22.

La fortaleza del enfoque orientado a conexión es clara. Permite características relacionadas con la conexión, como son el control de flujo, el control de errores y la entrega ordenada. Sin embargo, un servicio no orientado a conexión es más apropiado para algunos contextos. En capas inferiores (interconexión y red) es más robusto (por ejemplo, *véase* la discusión de la Sección 10.6). Además,

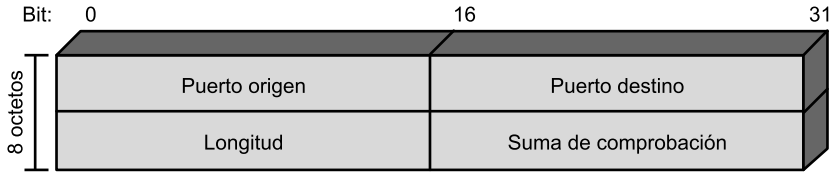


Figura 20.14. Cabecera UDP.

representa un «menor común denominador» del servicio que esperan las capas superiores. Pero, incluso a nivel de transporte y superiores, existe una justificación para un servicio no orientado a conexión. Existen casos en los que la sobrecarga del establecimiento y cierre de la conexión no están justificados o son incluso contraproducentes. Algunos ejemplos son:

- **Recolección de datos internos:** implica un muestreo activo o pasivo de fuentes de datos, como los procedentes de sensores e informes automáticos de autocomprobación de seguridad de equipos o componentes de red. En una situación de monitorización en tiempo real, la pérdida ocasional de una unidad de datos no causaría ningún desastre, ya que el siguiente informe debe llegar en breve.
- **Diseminación de datos externos:** incluye la difusión de mensajes a los usuarios de la red, el anuncio de un nuevo nodo o el cambio de la dirección de un servicio y la distribución de los valores de reloj de tiempo real.
- **Petición/respuesta:** aplicaciones en las cuales un servidor común proporciona un servicio de transacción a varios usuarios TS distribuidos y para el cual usar una secuencia del tipo petición/respuesta es usual. El uso del servicio se regula en la capa de aplicación y las conexiones de capas inferiores son frecuentemente innecesarias y molestas.
- **Aplicaciones en tiempo real:** como aplicaciones de voz y de telemedida, que llevan consigo el requisito de utilizar cierto grado de redundancia y/o transmisión en tiempo real. Estos requisitos no pueden tener funciones orientadas a conexión como la retransmisión.

Así, tienen cabida en la capa de transporte tanto servicios orientados a conexión como servicios no orientados a conexión.

UDP se sitúa encima de IP. Ya que es no orientado a conexión, UDP tiene muy pocas tareas que llevar a cabo. Esencialmente, incorpora a IP la capacidad de un direccionamiento de puerto. Esto se ve mejor examinando la cabecera UDP mostrada en la Figura 20.14. La cabecera incluye un puerto origen y un puerto destino. El campo de longitud contiene la longitud de todo el segmento UDP, incluyendo la cabecera y los datos. La suma de comprobación es el mismo algoritmo usado en TCP e IP. Para UDP, la suma de comprobación se aplica al segmento UDP entero más una pseudocabecera incorporada a la cabecera UDP en el momento del cálculo, que es la misma que la utilizada en TCP. Si se detecta un error, el segmento se descarta sin tomar ninguna medida adicional.

El campo de suma de comprobación en UDP es opcional. Si no se utiliza, se le asigna el valor cero. Sin embargo, hay que indicar que la suma de comprobación de IP se aplica sólo a la cabecera IP y no al campo de datos, que en este caso está compuesto por la cabecera UDP y los datos de usuario. Así, si UDP no efectúa ningún cálculo de suma de comprobación, los datos de usuario no se verifican.

20.5. LECTURAS RECOMENDADAS

Quizá el mejor estudio de las distintas estrategias de TCP para el control de flujo y de la congestión se encuentre en [STEV94]. Un artículo fundamental para comprender las cuestiones implicadas es el clásico [JACO88].

JACO88 Jacobson, V. «Congestion Avoidance and Control». *Proceedings, SIGCOMM'88, Computer Communication Review,* agosto de 1988; reimpreso en *Computer Communication Review*, enero de 1995; una versión revisada puede encontrarse en ftp.ee.lbl.gov/papers/congavoid.ps.Z.

STEV94 Stevens, W. *TCP/IP Illustrated, Volume 1: The Protocols.* Reading, MA: Addison-Wesley, 1994.

20.6. TÉRMINOS CLAVE, CUESTIONES DE REPASO Y EJERCICIOS

TÉRMINOS CLAVE

algoritmo de Karn
arranque lento
control de congestión TCP
control de flujo
crédito
detección de duplicados
estrategia de retransmisión
flujo de datos forzado
multiplexación
número de secuencia
opciones en los criterios de implementación TCP
promediado exponencial
protocolo de control de transmisión (TCP)
protocolo de datagrama de usuario (UDP)
protocolo de transporte
puerto
señalización de datos urgentes
socket
suma de comprobación

CUESTIONES DE REPASO

20.1. ¿Qué elementos de direccionamiento son necesarios para especificar un usuario de servicio de transporte (TS) destino?

20.2. Describa cuatro estrategias por las que un usuario TS emisor pueda averiguar la dirección de un usuario TS receptor.

20.3. Explique el uso de la multiplexación en el contexto de un protocolo de transporte.

20.4. Describa brevemente el esquema de créditos utilizado por TCP para el control de flujo.

20.5. ¿Cuál es la diferencia principal entre el esquema de créditos de TCP y el esquema de control de flujo de ventana deslizante utilizada por muchos otros protocolos, como por ejemplo HDLC?

20.6. Explique los mecanismos de diálogo en dos y tres pasos.

20.7. ¿Cuál es el beneficio del mecanismo de diálogo en tres pasos?

20.8. Defina las características de urgencia y forzado de TCP.

20.9. ¿Qué es una opción en los criterios de implementación de TCP?

20.10. ¿Cómo puede utilizarse TCP para tratar la congestión de red o de interconexión de red?

20.11. ¿Qué proporciona UDP que no ofrezca IP?

EJERCICIOS

20.1. Es una práctica común en la mayoría de los protocolos de transporte (en realidad, en la mayoría de los protocolos de todas las capas) que los datos y la señalización de control se multiplexen sobre el mismo canal lógico en cada conexión por usuario. Una alternativa consiste en establecer una única conexión de control de transporte entre cada par de entidades de transporte que se comuniquen. Esta conexión se usaría para transmitir las señales de control de todas las conexiones de los usuarios de transporte entre las dos entidades. Discuta las implicaciones de esta estrategia.

20.2. La discusión sobre control de flujo con un servicio de red fiable, referido como mecanismo de contrapresión, utiliza un protocolo de control de flujo de una capa inferior. Discuta las desventajas de esta estrategia.

20.3. Dos entidades de transporte de comunican a través de una red fiable. Supongamos que el tiempo normalizado para transmitir un segmento es igual a 1. Supongamos que el retardo de propagación extremo a extremo vale 3 y que la entrega de un segmento recibido al usuario de transporte requiere un tiempo de 2. El emisor tiene inicialmente concedido un crédito de siete segmentos. El receptor utiliza un criterio de control de flujo conservador y actualiza su asignación de créditos en cuanto puede. ¿Cuál es el máximo rendimiento alcanzable?

20.4. Dibuje un diagrama similar al de la Figura 20.4 para los siguientes casos (suponga un servicio de red fiable ordenado):

a) Cierre de la conexión: activo/pasivo.

b) Cierre de la conexión: activo/activo.

c) Rechazo de la conexión.

d) Cancelación de la conexión: un usuario emite un *Open* a un usuario que está preparado y entonces emite un *Close* antes de que se intercambie ningún dato.

20.5. Con un servicio de red fiable y ordenado, ¿son estrictamente necesarios los números de secuencia de los segmentos? ¿Qué capacidad se pierde sin ellos?

20.6. Considere un servicio de red orientado a conexión que sufre un reinicio. ¿Cómo podría ser tratado por un protocolo de transporte que suponga que el servicio de red es fiable excepto en el caso de un reinicio?

20.7. La discusión de la política de retransmisión hizo referencia a tres problemas asociados con el cálculo dinámico del valor del temporizador. ¿Qué modificaciones sobre la política ayudarían a aliviar estos problemas?

20.8. Considere un protocolo de transporte que usa un servicio de red orientado a conexión. Suponga que ese protocolo de transporte utiliza un esquema de asignación de créditos para el control de flujo y que el protocolo de red usa un esquema de ventana deslizante. ¿Qué relación, si existe, debería haber entre la ventana dinámica del protocolo de transporte y la ventana fija del protocolo de red?

20.9. En una red que tiene un tamaño máximo de paquete de 128 bytes, un tiempo de vida máximo de 30 s y un número de secuencia de paquetes de 8 bits, ¿cuál es la máxima tasa de transmisión de datos por conexión?

20.10. ¿Es posible que se produzca un bloqueo mutuo utilizando un diálogo en dos pasos en lugar de un diálogo en tres pasos? Dé un ejemplo o demuéstrelo en caso contrario.

20.11. A continuación, se enumeran cuatro estrategias que se pueden utilizar para proporcionar a un usuario de transporte las direcciones de un usuario de transporte destino. Para cada una, describa una analogía con el usuario del servicio de correo postal.

a) Conocer la dirección de antemano.

b) Hacer uso de una dirección «bien conocida».

c) Utilizar un servidor de nombres.

d) El destinatario se genera al realizar la solicitud.

20.12. En un esquema de créditos para control de flujo como el de TCP, ¿qué provisión de créditos se podría hacer para la asignación de créditos que se pierdan o se desordenen durante la transmisión?

20.13. ¿Qué ocurre en la Figura 20.3 si llega un SYN mientras el usuario solicitado está en el estado *CLOSED*? ¿Hay alguna forma de llamar la atención del usuario cuando no esté preparado?

20.14. En la discusión sobre el cierre de la conexión con referencia a la Figura 20.8, se estableció que además de recibir una confirmación de su segmento FIN y enviar una confirmación del segmento FIN recibido, una entidad TCP debe esperar un intervalo de tiempo igual al doble del máximo tiempo de vida esperado de un segmento (el estado *TIME WAIT*). La recepción de un ACK de su segmento FIN le asegura que todos los segmentos que ha enviado han sido recibidos por el otro extremo. El envío de un ACK del segmento FIN del otro extremo asegura a la otra entidad que todos sus segmentos han sido recibidos. Dé una razón por la que se necesite aún esperar antes de cerrar la conexión.

20.15. Normalmente, el campo «ventana» de la cabecera TCP da una asignación de créditos en octetos. Cuando se utiliza la opción de «escalado de ventana», el valor del campo «ventana» se multiplica por 2^F, donde F es el valor de la opción de escalado de ventana. El valor máximo de F que acepta TCP es 14. ¿Por qué se limita esta opción a 14?

20.16. La elección de un valor inicial del estimador original de SRTT de TCP constituye un problema. En ausencia de alguna información especial sobre las condiciones de la red, la opción habitual es la de elegir un valor arbitrario, como 3 segundos, y esperar que converja rápidamente a un valor preciso. Si la estimación es demasiado baja, TCP llevará a cabo retransmisiones innecesarias. Si es demasiado alta, TCP esperará demasiado tiempo antes de retransmitir en caso de que el primer segmento se pierda. Es más, la convergencia puede ser lenta, como indica este problema.

a) Elija $\alpha = 0{,}85$ y SRTT(0) = 3 segundos, suponga que todos lo valores de RTT medidos son iguales a 1 segundo y que no se producen pérdidas de paquetes. ¿Cuál es el valor de SRTT(19)? *Sugerencia:* la Ecuación (20.3) se puede reescribir para simplificar los cálculos, utilizando la expresión $(1 - \alpha^n)/(1 - \alpha)$.

b) Sea ahora SRTT(0) = 1 segundo y suponga que los valores medidos de RTT son 3 segundos y que no se produce pérdida de paquetes. ¿Cuál es el valor de SRTT(19)?

20.17. Una mala implementación del esquema de ventana deslizante de TCP puede llevar a un rendimiento extremadamente malo. Existe un fenómeno conocido como el «síndrome de la ventana absurda» (SWS, *Silly Window Sindrome*), que puede fácilmente causar una degradación del rendimiento en varios factores de 10. Como ejemplo de SWS, considere una aplicación que está ocupada en la transferencia de un fichero largo y que TCP está transfiriendo el fichero en segmentos de 200 octetos. El receptor inicialmente asigna un crédito

de 1.000. El emisor agota esta ventana con 5 segmentos de 200 octetos. Ahora suponga que el receptor devuelve una confirmación por cada segmento y proporciona un crédito adicional de 200 octetos por cada segmento recibido. Desde el punto de vista del receptor, esto abre la ventana de nuevo a 1.000 octetos. Sin embargo, desde el punto de vista del emisor, si la primera confirmación llega tras haber enviado cinco segmentos, se dispone de una ventana de sólo 200 octetos. Suponga que en algún momento el receptor calcula una ventana de 200 octetos pero tiene sólo 50 octetos para enviar hasta llegar a un punto de forzado. Por tanto, envía 50 octetos en un segmento, seguido de 150 octetos en el siguiente segmento, y reanuda la transmisión de segmentos de 200 octetos. ¿Qué podría ahora ocurrir para dar lugar a un problema de rendimiento? Plantee el SWS en términos más generales.

20.18. TCP impone que tanto el receptor como el emisor incorporen mecanismos para hacer frente al SWS.

a) Sugiera una estrategia para el receptor. *Sugerencia:* permita al receptor «mentir» sobre la capacidad de memoria temporal de que dispone bajo ciertas circunstancias. Plantee una regla razonable experimental para esto.

b) Sugiera una estrategia para el emisor. *Sugerencia:* considere la relación entre la ventana máxima posible de envío y lo que hay disponible para enviar.

20.19. En la Ecuación (20.5), reescriba la definición de SRTT($K+1$) en función de SERR($K+1$). Interprete el resultado.

20.20. Una entidad TCP abre una conexión y utiliza el arranque lento. Aproximadamente, ¿cuántos tiempos de ida y vuelta se necesitan antes de que TCP pueda enviar N segmentos?

20.21. Aunque el arranque lento con supresión de congestión es una técnica efectiva para hacer frente a la congestión, puede traducirse en largos tiempos de recuperación en redes de alta velocidad, como demuestra este problema:

a) Suponga un retardo de ida y vuelta de 60 ms (lo que podría ocurrir a través de un continente), un enlace con un ancho de banda disponible de 1 Gbps y un tamaño de segmento de 576 octetos. Determine el tamaño de ventana necesario para mantener lleno el cauce y el tiempo que tardaría en alcanzar el tamaño de ventana después de la expiración del temporizador utilizando el criterio de Jacobson.

b) Repita (a) para un tamaño de ventana de 16 kbytes.